



Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester)

Lecture 2: Camera Model

IMAGE PROCESSING

Technical University of Cluj Napoca
Computer Science Department



Introduction

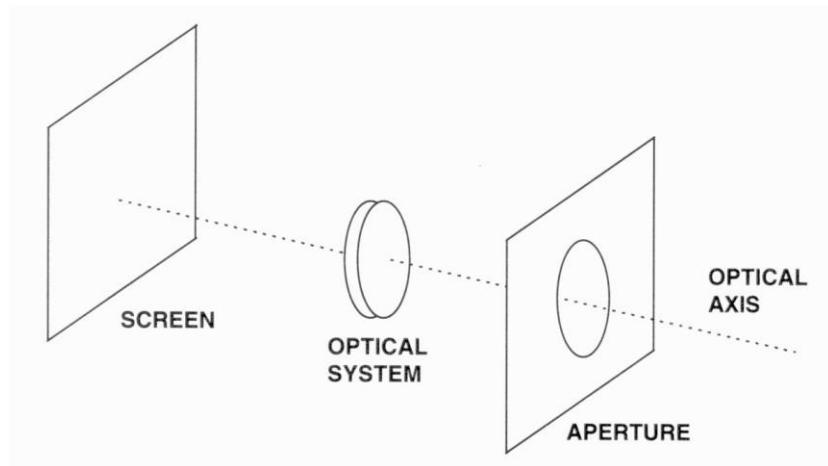
Purpose

Principles of digital image formation

Sensors

Video signal

The basic elements of an imaging device

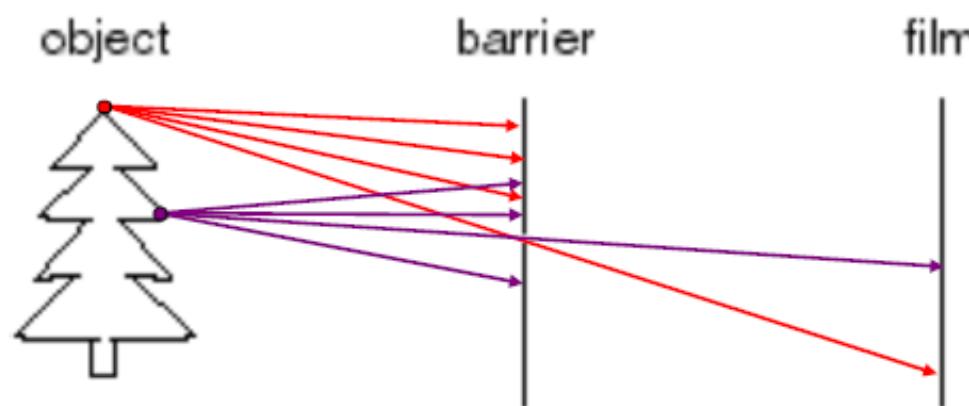
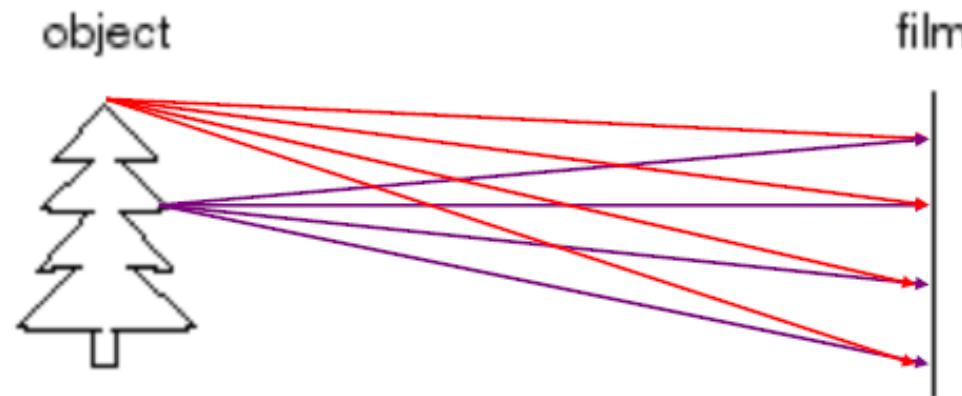


The aperture is a hole or an opening through which light is admitted. It is implemented through a device, called diaphragm, allowing for different size openings.



Camera model

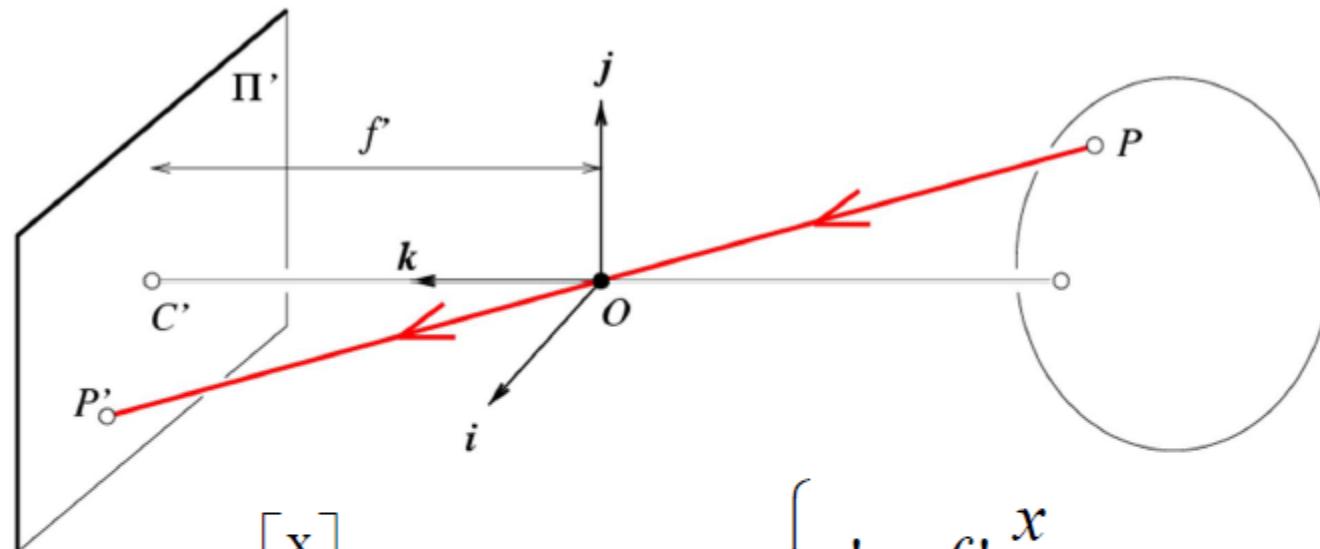
Image formation process





Camera model

Pinhole or perspective camera model



$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad \left\{ \begin{array}{l} x' = f' \frac{x}{z} \\ y' = f' \frac{y}{z} \end{array} \right.$$

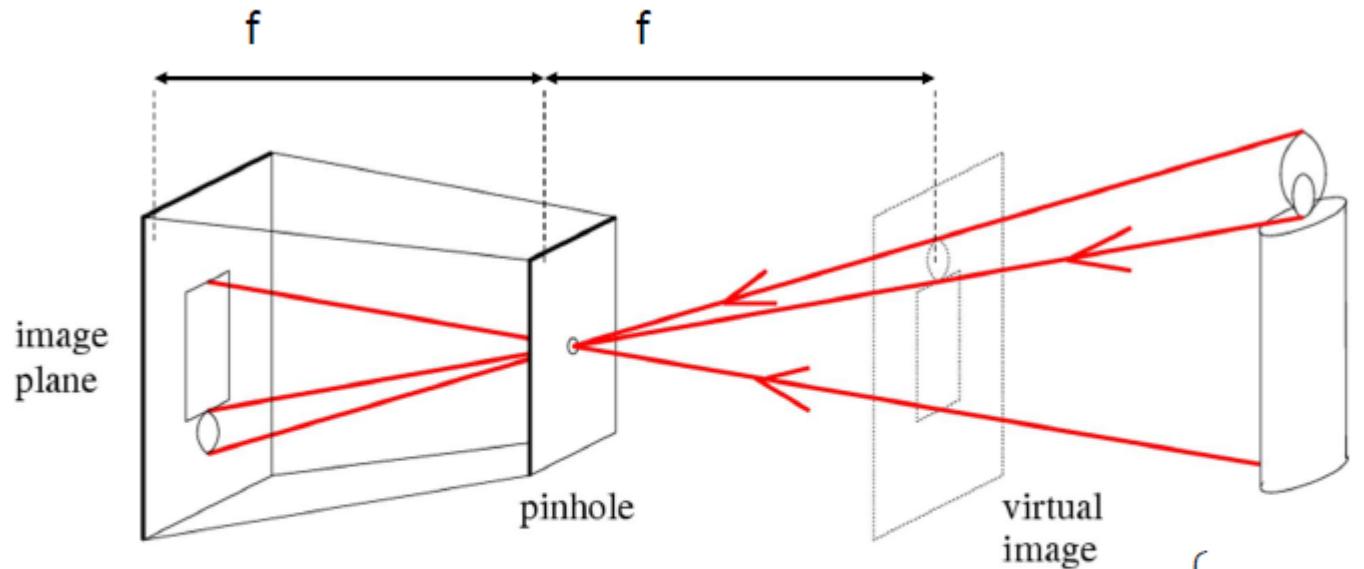
Note: z is always negative.

Derived using similar triangles



Camera model

Pinhole or perspective camera model



- Common to draw image plane *in front* of the focal point
- Moving the image plane merely scales the image.

$$\begin{cases} x' = f \frac{x}{z} \\ y' = f \frac{y}{z} \end{cases}$$



Camera model

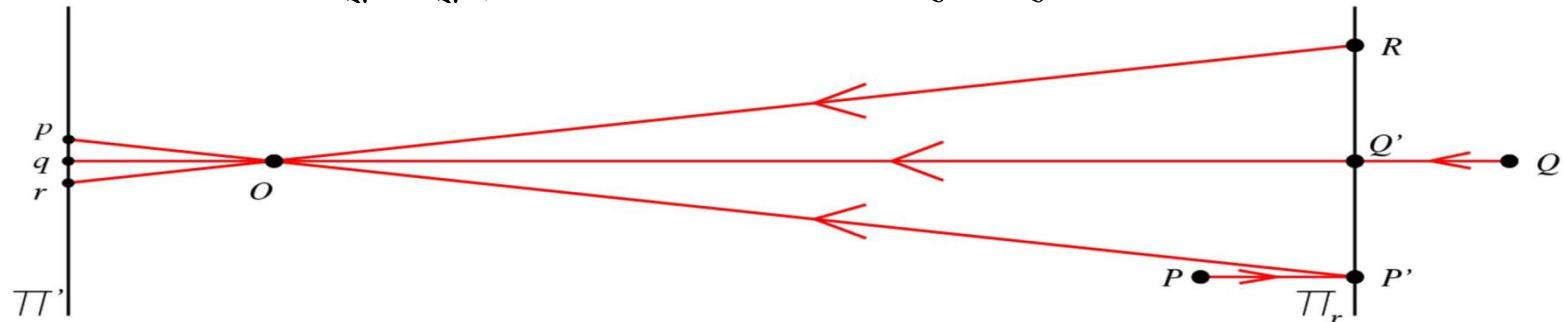
Weak perspective camera model

The relative distance along the optical axis, between two scene points δz is much smaller than the average distance z_{AV} between the camera and these points.

$$\delta z < z_{AV}/20 \quad (1)$$

The fundamental equation of the weak perspective camera are:

$$x' = f \frac{x}{z} \approx \frac{f}{z_{AV}} x \quad y' = f \frac{y}{z} \approx \frac{f}{z_{AV}} y \quad (2)$$



$$x' = -m^* x$$

$$y' = -m^* y$$

$$m = -f/z_{AV} = \text{magnification}$$

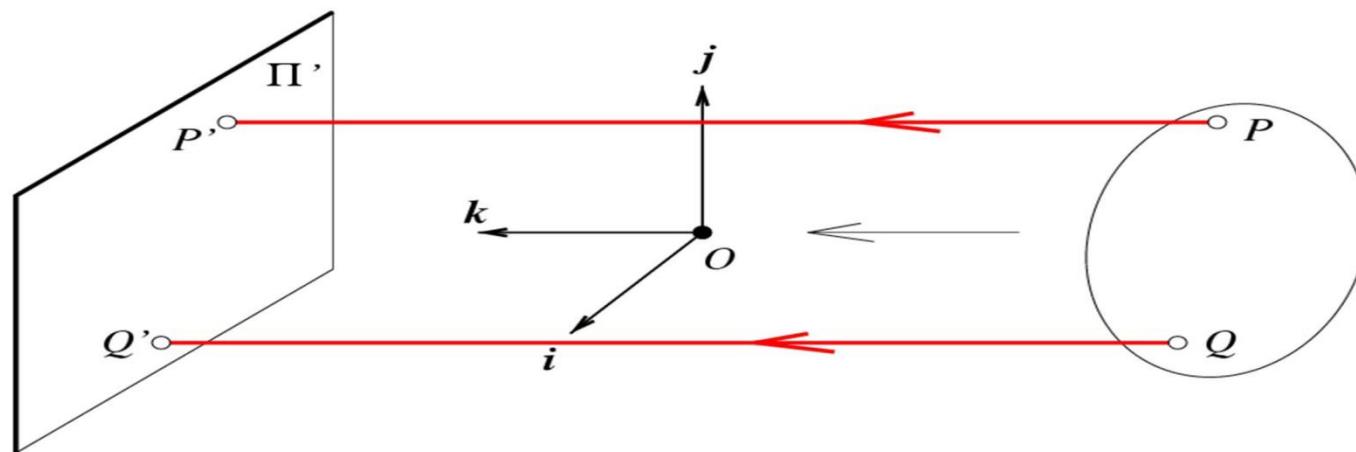


Camera model

Weak perspective camera model

Equations describe a sequence of two transformations:

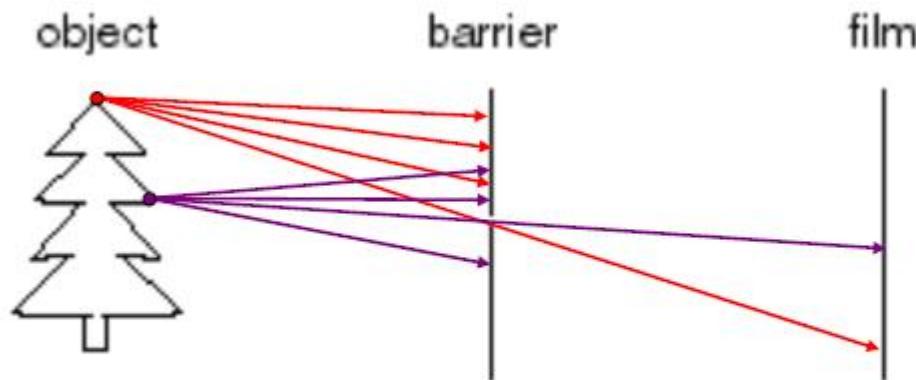
- an **orthographic projection**, in which points are projected along rays parallel to the optical axis ($x'=x$; $y'=y$);
- an **isotropic scaling** by the factor f/z_{AV}





Camera model

Aperture size problem



Kate Iazuka ©

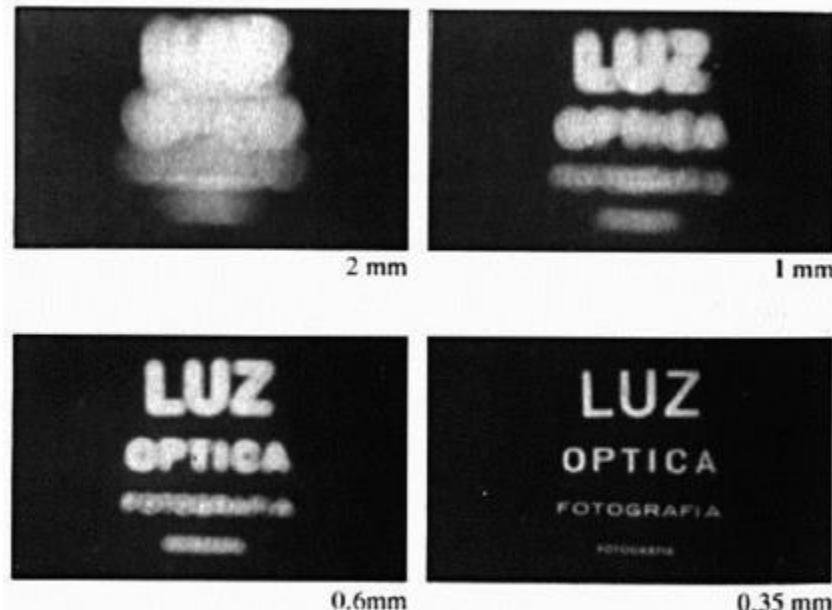
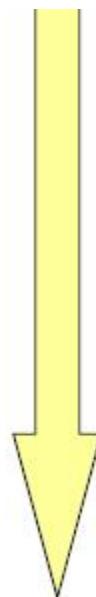


Camera model

Aperture size problem

Shrinking
aperture
size

- Rays are mixed up



-Why the aperture cannot be too small?

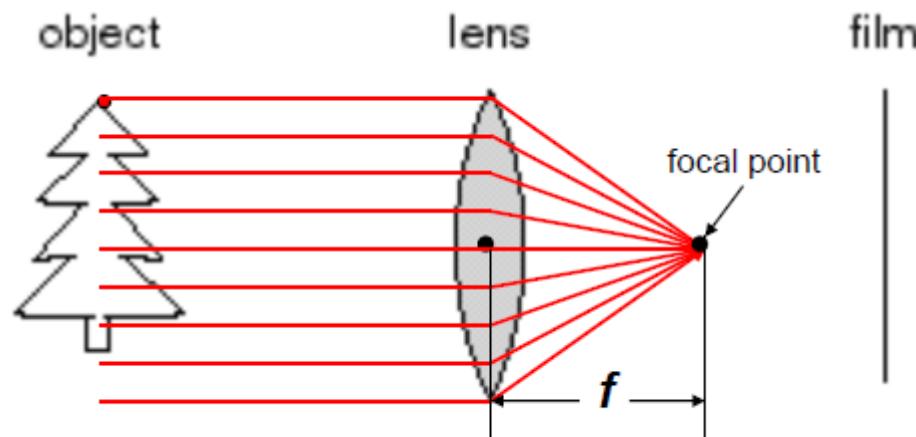
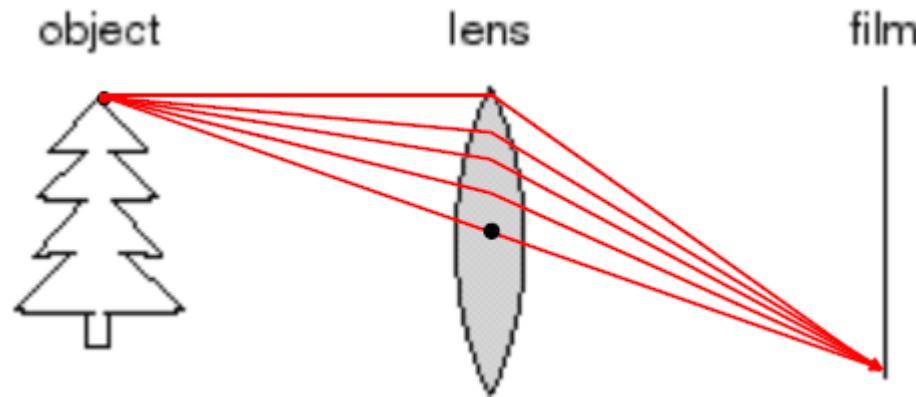
- Less light passes through
- Diffraction effect

Adding lenses!



Camera model

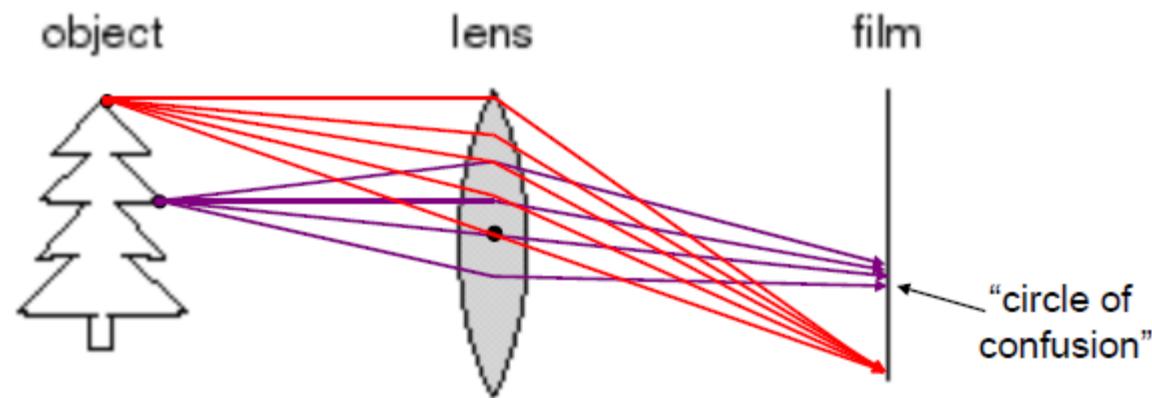
Camera and lenses





Camera model

Camera and lenses





Camera model

Laws of geometric optics

- Light travels in straight lines in homogeneous medium
- Reflection upon a surface: incoming ray, surface normal, and reflection are co-planar
- Refraction: when a ray passes from one medium to another

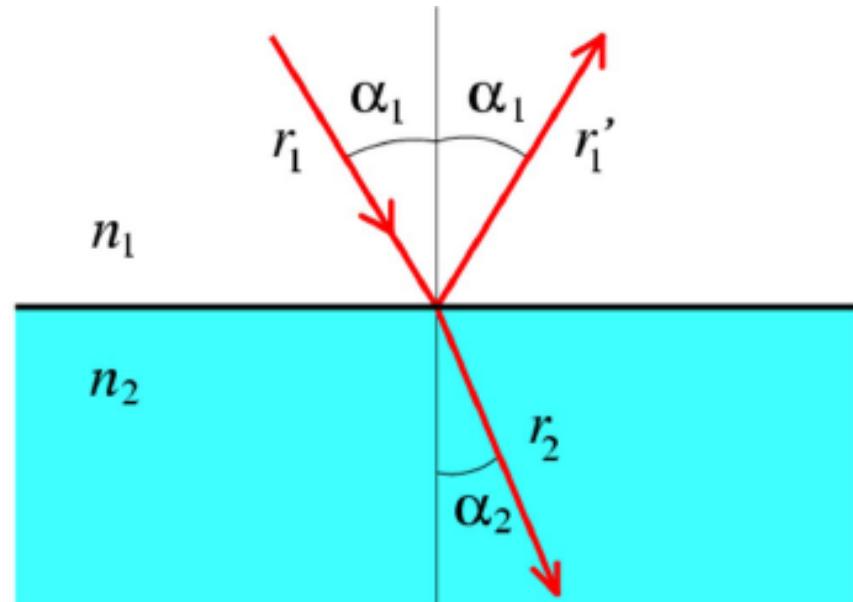
Snell's law

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

α_1 = incident angle

α_2 = refraction angle

n_i = index of refraction





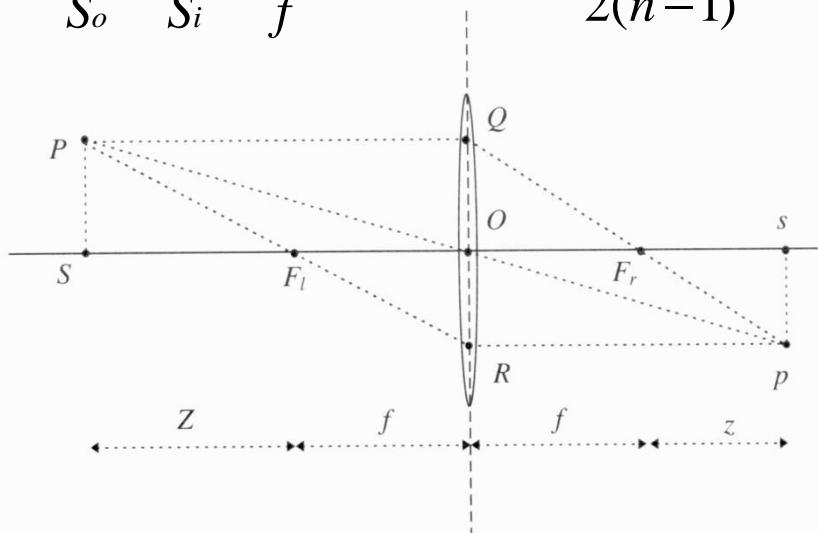
Camera model

The “thin lens” camera model

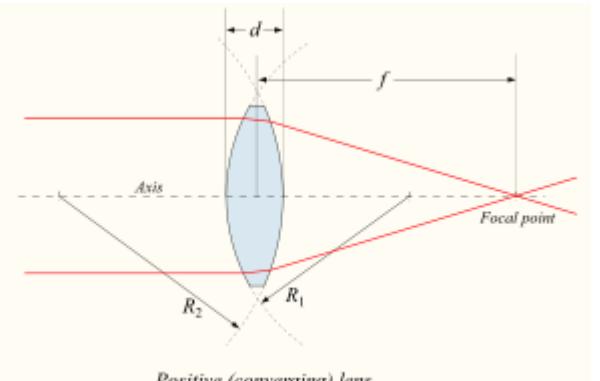
1. Any ray entering the lens parallel to the optical axis on one side goes through the focus on the other side.
2. Any ray entering the lens from the focus on one side emerges parallel to the optical axis on the other size.
3. The ray going through the lens center, O, named *principal ray*, goes through point p un-deflected.

The fundamental equations of thin lenses: $Z \cdot z = f^2$ (1). Setting $S_o = Z + f$ and $S_i = z + f$ equation (1) can be reduced to the fundamental equations of thin lenses:

$$\frac{1}{S_o} + \frac{1}{S_i} = \frac{1}{f} \quad (2) \quad f = \frac{R}{2(n-1)} \quad (3)$$



n - index of refraction of the lens material
R - radii of the curvature of the lens surfaces



A lens can be considered a **thin lens** if $d \ll R_1$ or $d \ll R_2$.
Technical University of Cluj Napoca
Computer Science Department



Camera model

Image focusing

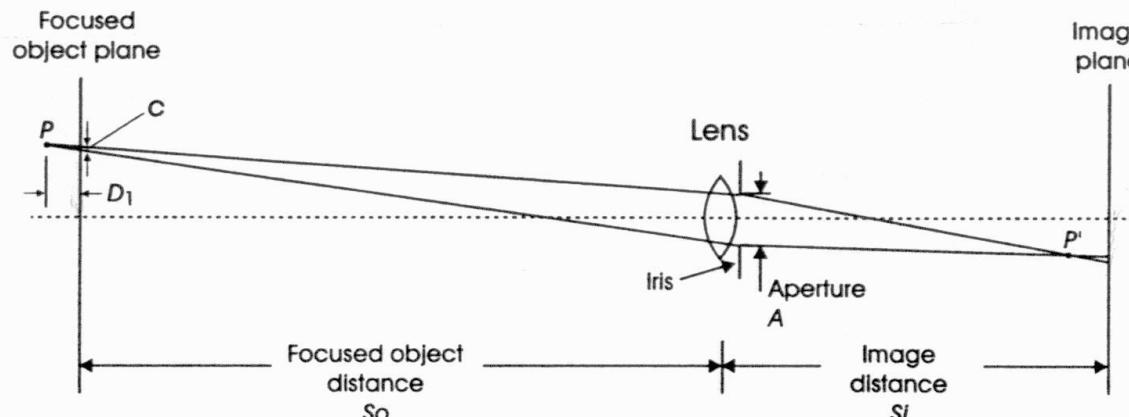


Image is in focus: all rays coming from a single scene point P must converge onto a single point on the image plain p -> sharp image
Image is not in focus: the image of P is spread over a circle -> blurred image

Obtaining a focused image

- pinhole camera/aperture
- optical system (lens)

Measures

- Circle-of-confusion (c) – its projection on the image plane < 1 pixel (focused image)
- Depth of field – distance (D_1) around the FOP within the diameter of the projection of the circle of confusion(c) on the image remains less than 1 pixel



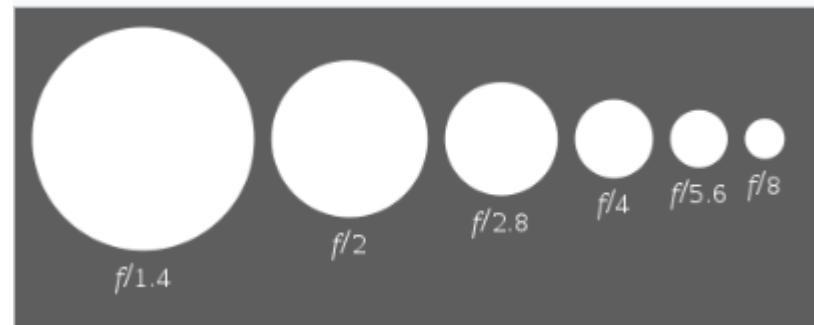
Camera model

Apertures

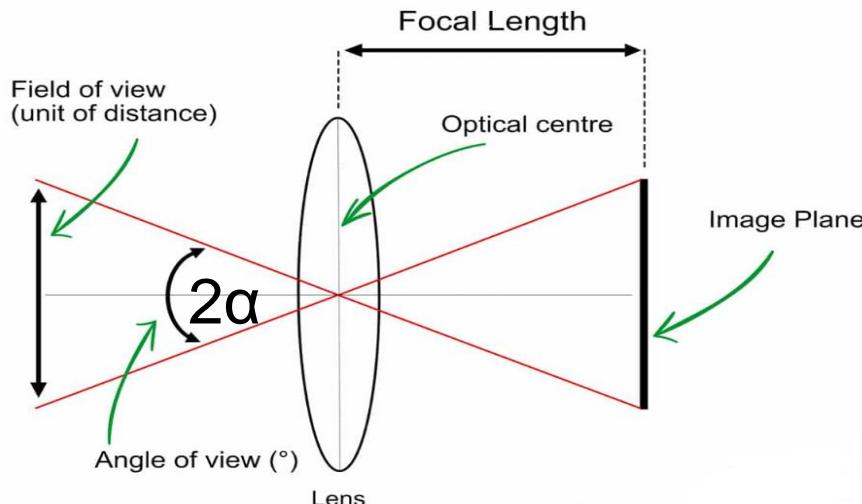


The **f_number** of an optical system is the ratio of the system's focal length to the diameter of the entrance pupil

$$f_number = \frac{\text{Focal Length}}{\text{Diameter}}$$



Field of view (FOV)



Focal Length/ f_number pairs

$$\text{FOV} = 2 * \alpha = 2 * \arctan\left(\frac{\text{Imager size}}{2 * \text{Focal Length}}\right)$$

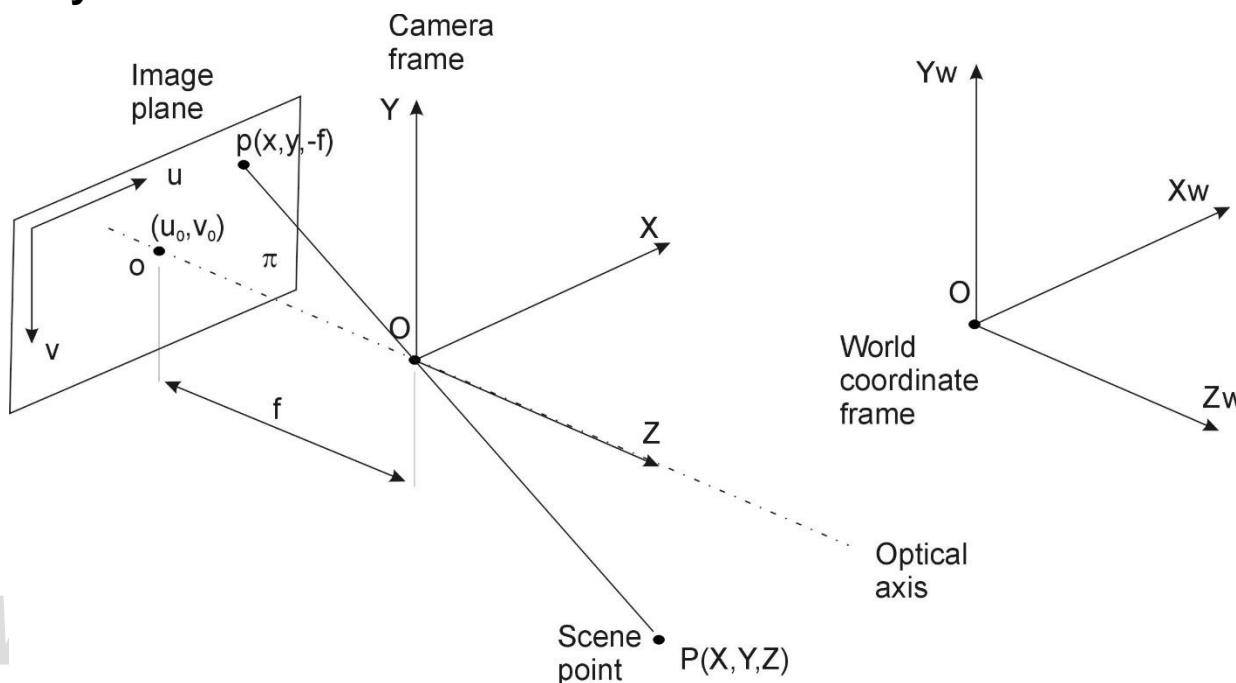


Camera model

The perspective camera model (pinhole)

- The most common geometric model of an imaging camera.
- Each point in the object space is projected by a straight line through the projection center (pinhole/lens center) into the image plane.

The **fundamental equations of the perspective camera model** are:
[X_C, Y_C, Z_C] are the coordinates of point P in the camera coordinate system



$$\begin{cases} x = f \cdot \frac{X_C}{Z_C} \\ y = f \cdot \frac{Y_C}{Z_C} \end{cases}$$



Camera model

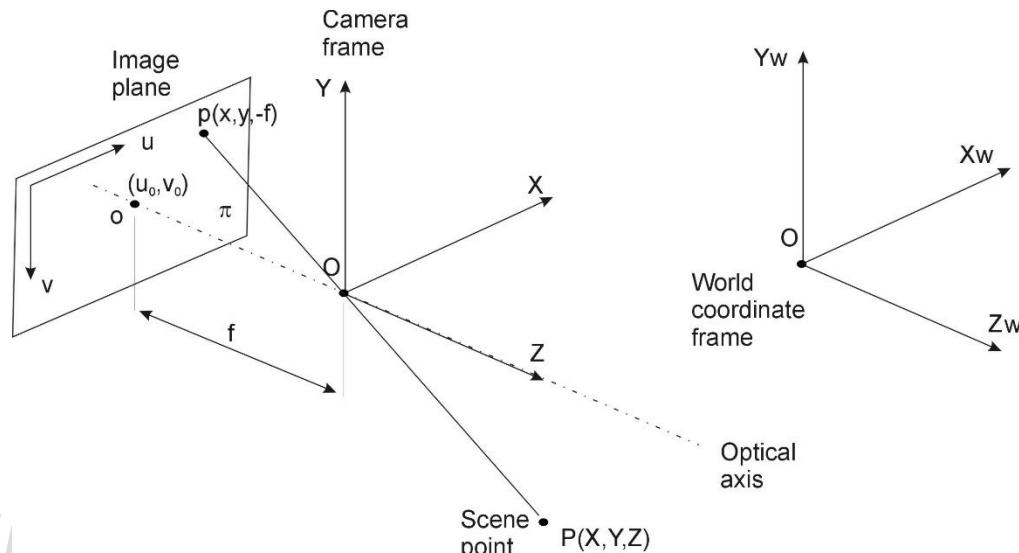
Physical camera parameters

Intrinsic parameters := internal camera geometrical and optical characteristics (those that specify the camera itself).

- **Focal length** := the distance between the optical center of the lens and the image plane: f [mm] or [pixels].
- **Effective pixel size** (dpx, dpy) [mm];
- **Principal point** := location of the image center in pixel coordinates: (u_0, v_0)
- **Distortion coefficients** of the lens: radial (k_1, k_2) and tangential (p_1, p_2).

Extrinsic parameters := the 3-D position and orientation of the camera frame relative to a certain world coordinate system:

- **Rotation vector** $r = [Rx, Ry, Rz]^T$ or its equivalent **rotation matrix** R
- **Translation vector** $T = [Tx, Ty, Tz]^T$;



In multi-camera (stereo) systems, the extrinsic parameters also describe the relationship between the cameras



Camera model

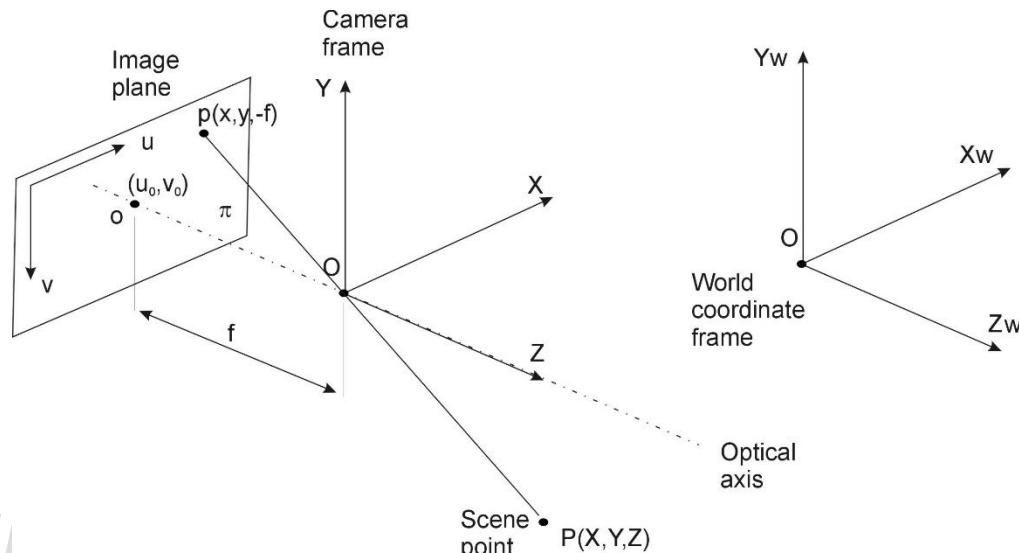
Physical camera parameters

Intrinsic parameters := internal camera geometrical and optical characteristics (those that specify the camera itself).

- **Focal length** := the distance between the optical center of the lens and the image plane: f [mm] or [pixels].
- **Effective pixel size** (dpx, dpy) [mm];
- **Principal point** := location of the image center in pixel coordinates: (u_0, v_0)
- **Distortion coefficients** of the lens: radial (k_1, k_2) and tangential (p_1, p_2).

Extrinsic parameters := the 3-D position and orientation of the camera frame relative to a certain world coordinate system:

- **Rotation vector** $r = [Rx, Ry, Rz]^T$ or its equivalent **rotation matrix** R
- **Translation vector** $T = [Tx, Ty, Tz]^T$;



In multi-camera (stereo) systems, the extrinsic parameters also describe the relationship between the cameras



Camera frame \leftrightarrow image plane transformation

Camera frame \Rightarrow image plane transformation

(projection / normalization) : $P = [X_C, Y_C, Z_C]^T$ [metric units] $\Rightarrow p = [u, v]^T$ [pixels]

1. Transform $P = [X_C, Y_C, Z_C]^T \Rightarrow p = [x, y, -f]^T$

Fundamental equations of the *perspective camera model* normalized with $1/Z$:

$$\begin{bmatrix} x \\ y \end{bmatrix} = f \begin{bmatrix} X_c / Z_c \\ Y_c / Z_c \end{bmatrix} = f \begin{bmatrix} x_N \\ y_N \end{bmatrix} \quad f - \text{focal distance [metric units]}$$

2. Transform p [x, y] T [metric units] \Rightarrow image coordinates [u, v] T [pixels]

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} D_u \cdot x \\ D_v \cdot y \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \quad D_u, D_v - \text{coefficients needed to transform metric units to pixels: } D_u = 1 / dpx; D_v = 1 / dpy$$

1 + 2 \Rightarrow projection equation: $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot \begin{bmatrix} x_N \\ y_N \\ 1 \end{bmatrix}$

$$A = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

A – is the camera matrix:

f_x – is the focal distance expressed in units of horizontal pixels:

f_y – is the focal distance expressed in units of vertical pixels:

$$f_x = f \cdot D_u = \frac{f}{dpx}$$

$$f_y = f \cdot D_v = \frac{f}{dpy}$$

Technical University of Cluj Napoca

Computer Science Department



Camera frame \leftrightarrow image plane transformation

Image plane transformation \Rightarrow camera frame

(reconstruction) : $p = [u, v]^T$ [pixels] $\Rightarrow P = [X_C, Y_C, Z_C]^T$ [metric units]

$$\begin{bmatrix} x_N \\ y_N \\ 1 \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Notes:

With one camera we cannot measure depth (Z). We can determine only the projection equation / normalized coordinates:

$$\begin{bmatrix} x_N \\ y_N \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}$$

To measure the depth (Z) a stereo system (2 cameras) is needed



Camera model

Modeling the lens distortions

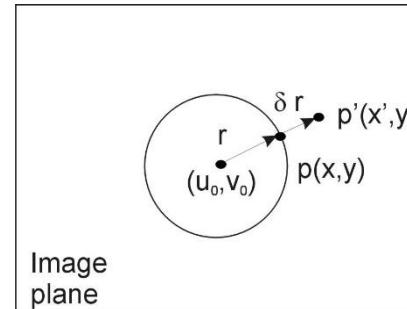
Radial lens distortion

Causes the actual image point to be displaced radially in the image plane

$$\begin{bmatrix} \partial x^r \\ \partial y^r \end{bmatrix} = \begin{bmatrix} x \cdot (k_1 \cdot r^2 + k_2 \cdot r^4 + \dots) \\ y \cdot (k_1 \cdot r^2 + k_2 \cdot r^4 + \dots) \end{bmatrix}$$

$$r^2 = x^2 + y^2;$$

k_1, k_2, \dots - radial distortion coefficients



Tangential distortion

Appears if the centers of curvature of the lenses' surfaces are not strictly collinear

$$\begin{bmatrix} \partial x^t \\ \partial y^t \end{bmatrix} = \begin{bmatrix} 2p_1 \cdot xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 \cdot xy \end{bmatrix}$$

p_1, p_2 – tangential distortion coefficients

Transform $\mathbf{p} [x, y]^T$ [metric units] \Rightarrow image coordinates $[u, v]^T$ [pixels]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} D_u \cdot (x + \partial x^r + \partial x^t) \\ D_v \cdot (y + \partial y^r + \partial y^t) \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

→The projection equations become non-linear

Solution: perform distortion correction on image and afterwards linear projection



Camera model

Distortion correction algorithm

Evaluate the distortion coefficients: k_1, k_2, p_1, p_2

Use calibration tools: http://www.vision.caltech.edu/bouguetj/calib_doc/

- Camera Calibration Toolbox for Matlab

The principle behind the distortion correction algorithm is **the existence of a bivalent correspondence between the distorted image pixels (x', y') and the undistorted image pixels (x, y)**

The correction algorithm

For each pixel (u, v) from the corrected image D:

- compute the (x, y) coordinates (1)
- compute the (x', y') coordinates in the distorted image S (2)
- compute the (u', v') coordinates in the distorted image S (3)
- $D(u, v) = S(u', v')$



Camera model

Distortion correction algorithm

$$\begin{cases} x = \frac{u - u_0}{f_x} \\ y = \frac{v - v_0}{f_y} \end{cases} \quad (1)$$

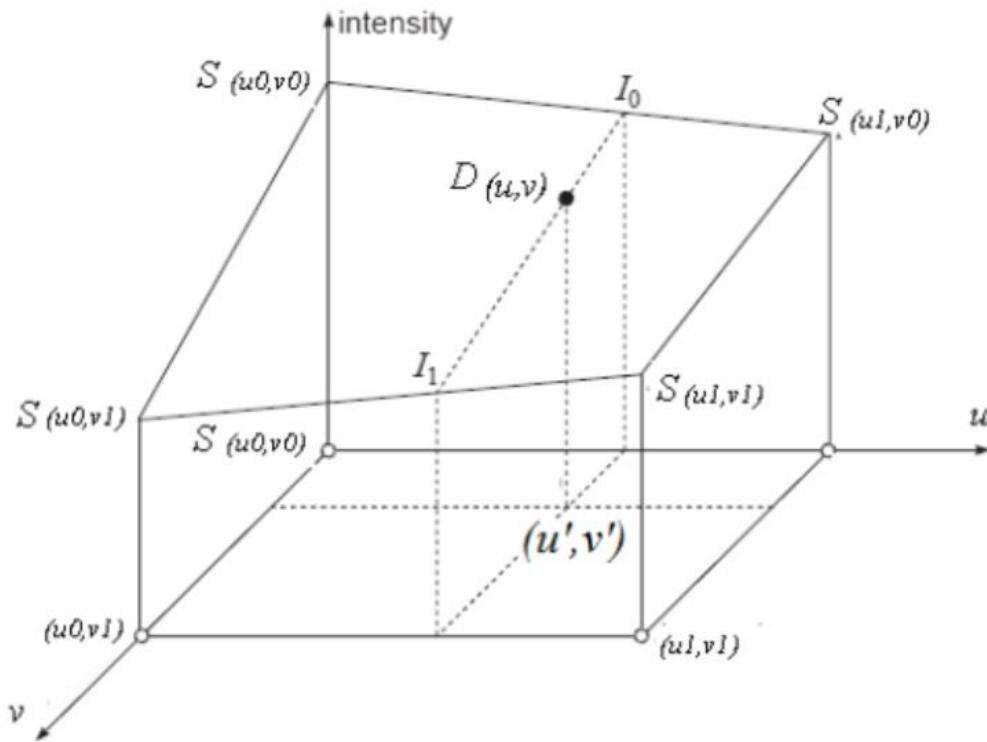
$$(x', y') = (x + \delta x, y + \delta y) \quad (2)$$

$$\begin{cases} u' = u_0 + x' f_x \\ v' = v_0 + y' f_y \end{cases} \quad (3)$$



Camera model

Distortion correction algorithm



$$u_0 = \text{int}(u'); \\ v_0 = \text{int}(v');$$

$$u_1 = u_0 + 1; \\ v_1 = v_0 + 1;$$

$$I_0 = S(u_0, v_0) \cdot (u_1 - u') + S(u_1, v_0) \cdot (u' - u_0); \\ I_1 = S(u_0, v_1) \cdot (u_1 - u') + S(u_1, v_1) \cdot (u' - u_0);$$

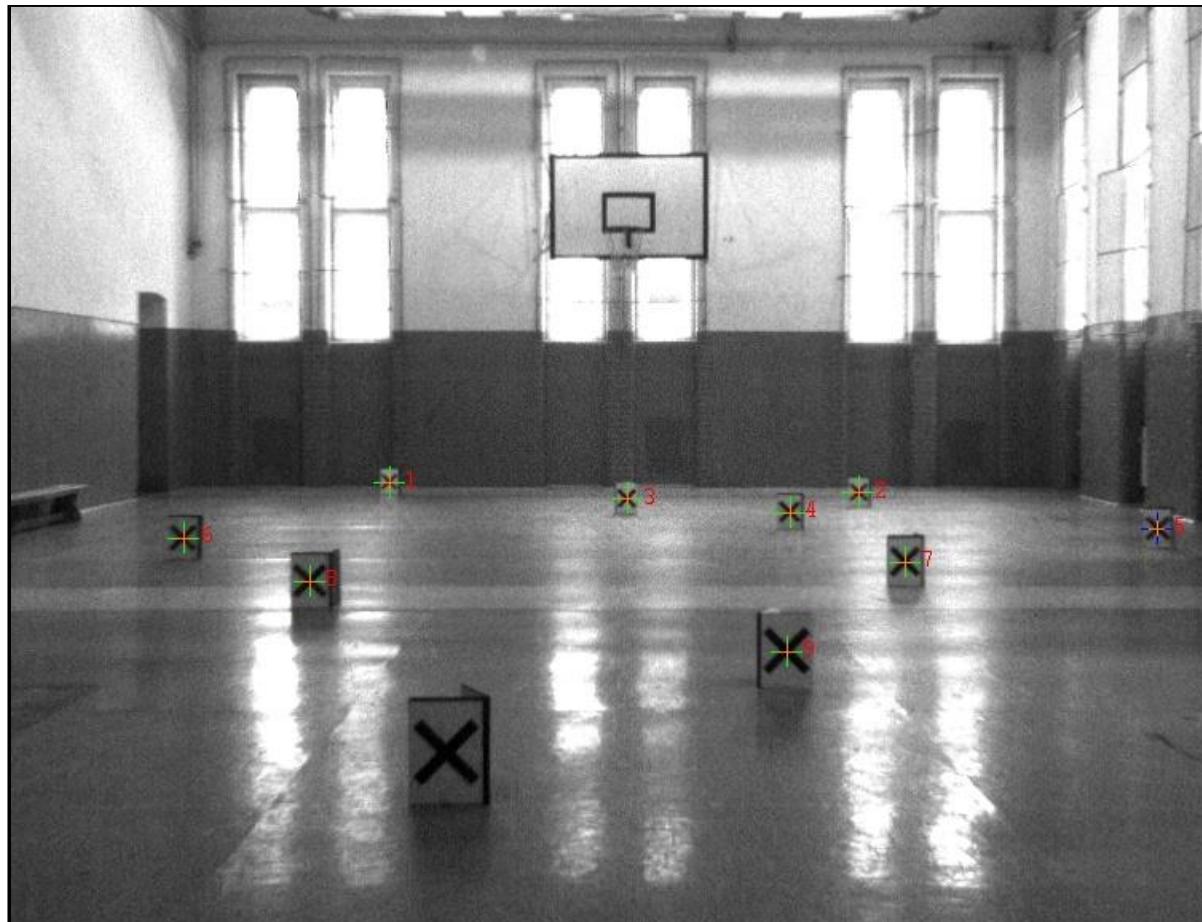
$$D(u, v) = I_0 \cdot (v_1 - v') + I_1 \cdot (v' - v_0);$$

Bilinear interpolation of the destination pixel Intensity $D(u, v)$ starting from the floating point coordinates of the source pixel (u', v')

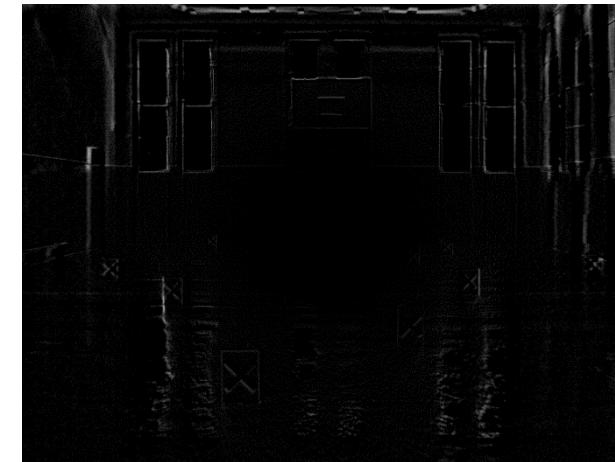


Lenses distortion correction

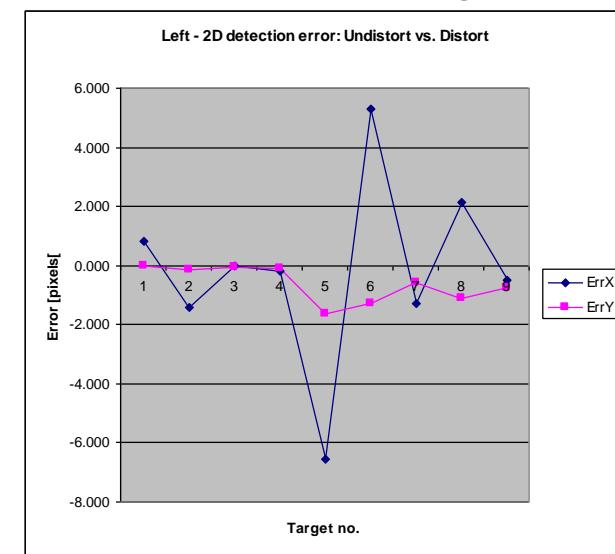
8.5 mm lens, CCD camera



Undistorted image



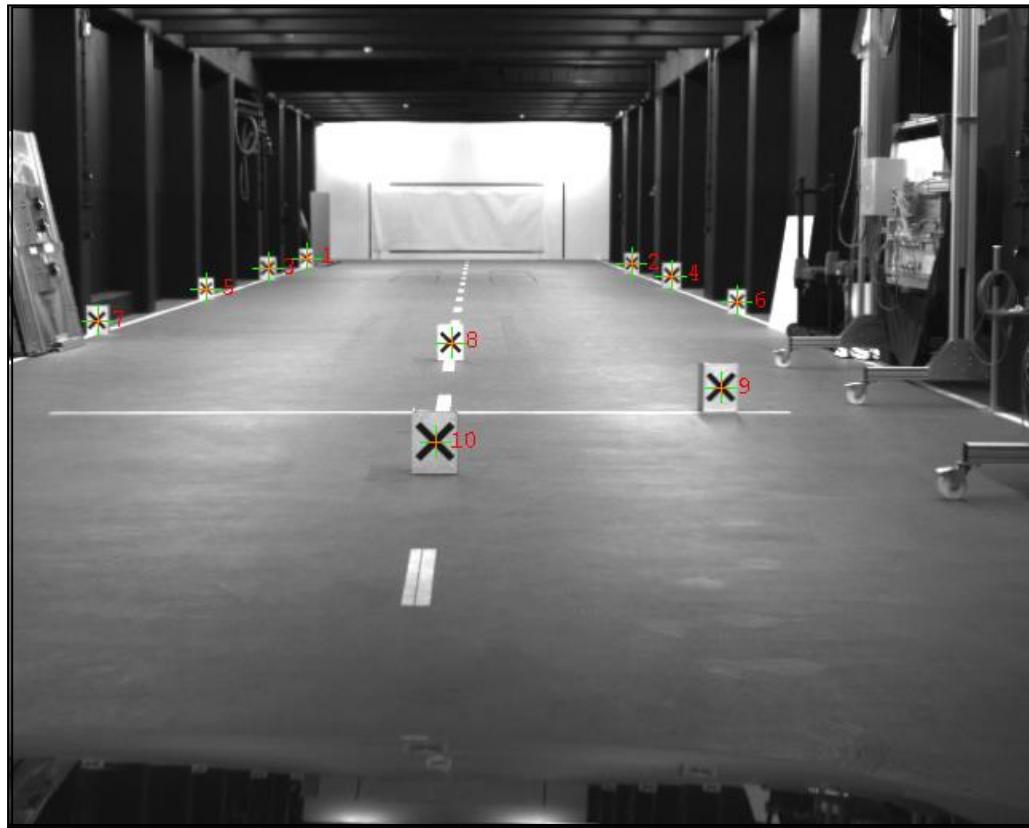
Difference image



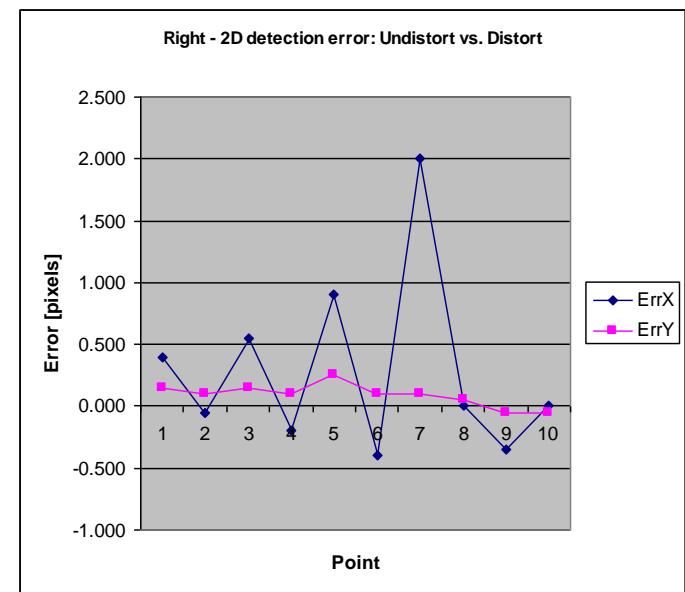


Lenses distortion correction

16 mm lens, CCD camera



Undistorted image

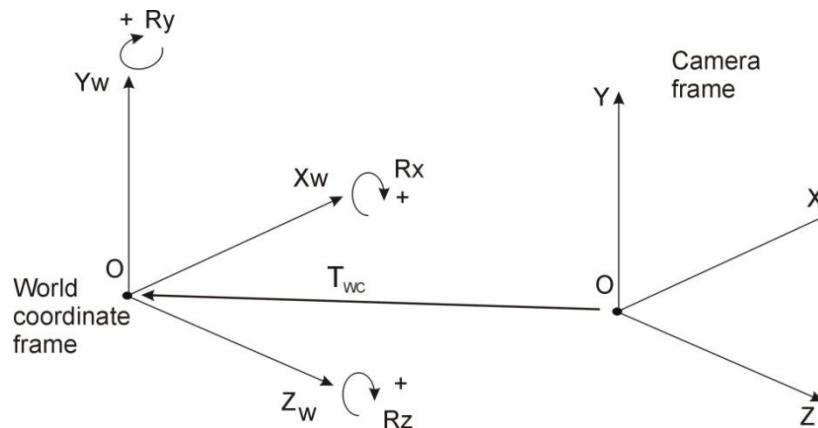




Camera frame \leftrightarrow world reference frame transformation

Direct mapping (world \Rightarrow camera)

$\mathbf{XX}_W = [X_W, Y_W, Z_W]^T$ (world coordinate system - WRF) $\Rightarrow \mathbf{XX}_C = [X_C, Y_C, Z_C]^T$ (camera coordinate system – CRF)



$$\mathbf{XX}_C = \mathbf{R}_{WC} \cdot \mathbf{XX}_W + \mathbf{T}_{WC}$$

where:

$\mathbf{T}_{WC} = [Tx, Ty, Tz]^T$ – world to camera translation vector;
 \mathbf{R}_{WC} – world to camera rotation matrix:



Camera frame \leftrightarrow world reference frame transformation

Inverse mapping (camera \Rightarrow world)

$\mathbf{XX}_C = [X_C, Y_C, Z_C]^T$ (camera coordinate system – CRF) $\Rightarrow \mathbf{XX}_W = [X_W, Y_W, Z_W]^T$ (world coordinate system - WRF)

$$\mathbf{XX}_W = \mathbf{R}_{WC}^{-1} \cdot (\mathbf{XX}_C - \mathbf{T}_{WC})$$

Rotation matrix is orthogonal [Trucco1998]:

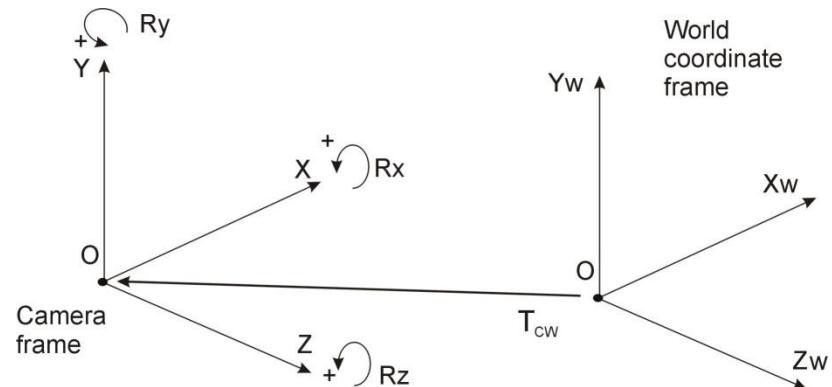
$$\mathbf{R} \cdot \mathbf{R}^T = \mathbf{R}^T \cdot \mathbf{R} = 1 \Rightarrow \mathbf{R}^T = \mathbf{R}^{-1}$$

$$\mathbf{XX}_W = \mathbf{R}_{WC}^T \cdot (\mathbf{XX}_C - \mathbf{T}_{WC}) = \mathbf{R}_{CW} \cdot (\mathbf{XX}_C + \mathbf{T}_{CW})$$

where:

$$\mathbf{T}_{CW} = [T_X \ T_Y \ T_Z]^T - \text{camera to world translation vector} \quad T_{CW} = -T_{WC}$$

$$\mathbf{R}_{CW} - \text{camera to world translation vector} \quad R_{CW} = R_{WC}^T$$





Rotation Matrix

World-to-camera

$$\mathbf{R}_{WC} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{n}^{XW} & \mathbf{n}^{YW} & \mathbf{n}^{ZW} \end{bmatrix} = \begin{bmatrix} n_X^{XW} & n_X^{YW} & n_X^{ZW} \\ n_Y^{XW} & n_Y^{YW} & n_Y^{ZW} \\ n_Z^{XW} & n_Z^{YW} & n_Z^{ZW} \end{bmatrix}$$

$\mathbf{n}^{XW} = [n_X^{XW} \quad n_Y^{XW} \quad n_Z^{XW}]^T$ – normal vector of \mathbf{OX}_W axis in the CRF

$\mathbf{n}^{YW} = [n_X^{YW} \quad n_Y^{YW} \quad n_Z^{YW}]^T$ – normal vector of \mathbf{OY}_W axis in the CRF

$\mathbf{n}^{ZW} = [n_X^{ZW} \quad n_Y^{ZW} \quad n_Z^{ZW}]^T$ – normal vector of \mathbf{OZ}_W axis in the CRF

Camera-to-world

$$\mathbf{R}_{CW} = \mathbf{R}_{WC}^T = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{n}^{XC} & \mathbf{n}^{YC} & \mathbf{n}^{ZC} \end{bmatrix} = \begin{bmatrix} n_X^{XC} & n_X^{YC} & n_X^{ZC} \\ n_Y^{XC} & n_Y^{YC} & n_Y^{ZC} \\ n_Z^{XC} & n_Z^{YC} & n_Z^{ZC} \end{bmatrix}$$

$\mathbf{n}^{XC} = [n_X^{XC} \quad n_Y^{XC} \quad n_Z^{XC}]^T$ – normal vector of \mathbf{OX}_C axis in the WRF

$\mathbf{n}^{YC} = [n_X^{YC} \quad n_Y^{YC} \quad n_Z^{YC}]^T$ – normal vector of \mathbf{OY}_C axis in the WRF

$\mathbf{n}^{ZC} = [n_X^{ZC} \quad n_Y^{ZC} \quad n_Z^{ZC}]^T$ – normal vector of \mathbf{OZ}_C axis in the WRF



Rotation Matrix \leftrightarrow Rotation Vector

Rotation vector – Rotation matrix

$$\mathbf{r} = [\theta, \psi, \gamma]^T$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\mathbf{Rx} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{pmatrix} \quad \mathbf{Ry} = \begin{pmatrix} \cos \psi & 0 & \sin \psi \\ 0 & 1 & 0 \\ \sin \psi & 0 & \cos \psi \end{pmatrix} \quad \mathbf{Rz} = \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R} = \mathbf{Rx}\mathbf{Ry}\mathbf{Rz}$$



Rotation Matrix \leftrightarrow Rotation Vector

Rotation vector

$$\mathbf{r}_{WC} = [R_X \ R_Y \ R_Z]^T \quad (R_X - \text{pitch}, R_Y - \text{yaw}, R_Z - \text{tilt / roll})$$

$\mathbf{r}_{WC} \Rightarrow \mathbf{R}_{WC}$ transform:

$$r_{11} = \cos(R_Y)\cos(R_Z)$$

$$r_{12} = \sin(R_X)\sin(R_Y)\cos(R_Z) - \cos(R_X)\sin(R_Z)$$

$$r_{13} = \cos(R_X)\sin(R_Y)\cos(R_Z) + \sin(R_X)\sin(R_Z)$$

$$r_{21} = \cos(R_Y)\sin(R_Z)$$

$$r_{22} = \sin(R_X)\sin(R_Y)\sin(R_Z) + \cos(R_X)\cos(R_Z)$$

$$r_{23} = \cos(R_X)\sin(R_Y)\sin(R_Z) - \sin(R_X)\cos(R_Z)$$

$$r_{31} = -\sin(R_Y)$$

$$r_{32} = \sin(R_X)\cos(R_Y)$$

$$r_{33} = \cos(R_X)\cos(R_Y)$$

$\mathbf{R}_{WC} \Rightarrow \mathbf{r}_{WC}$ transform:

$$R_Y = \arcsin(r_{31})$$

If $\cos(R_Y) \neq 0$:

$$R_X = \text{atan2}\left(-\frac{r_{32}}{\cos(R_Y)}, \frac{r_{33}}{\cos(R_Y)}\right)$$

$$R_Z = -\text{atan2}\left(-\frac{r_{21}}{\cos(R_Y)}, \frac{r_{11}}{\cos(R_Y)}\right)$$

If $\cos(R_Y) = 0$:

$$R_X = \text{atan2}(r_{12}, r_{22})$$

$$R_Z = 0$$



3D (world) \Rightarrow 2D (image) mapping using the Projection Matrix

Projection matrix

$$\mathbf{P} = \mathbf{A} \cdot [\mathbf{R}_{WC} \mid \mathbf{T}_{WC}]$$

The projection equation of a 3D world point $[X_W, Y_W, Z_W]$ expressed in normalized coordinates :

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

$s=z_s$ – scaling factor

Obtaining the 2D image coordinates from normalized coordinate

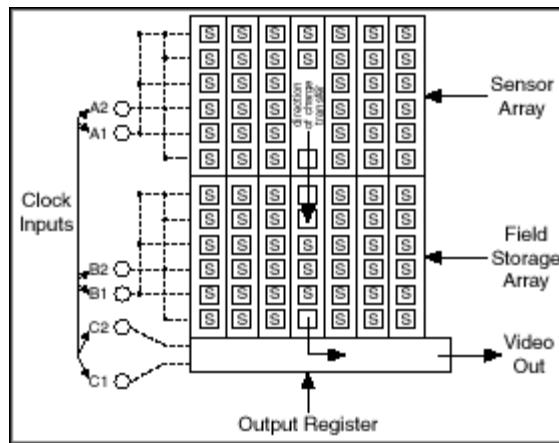
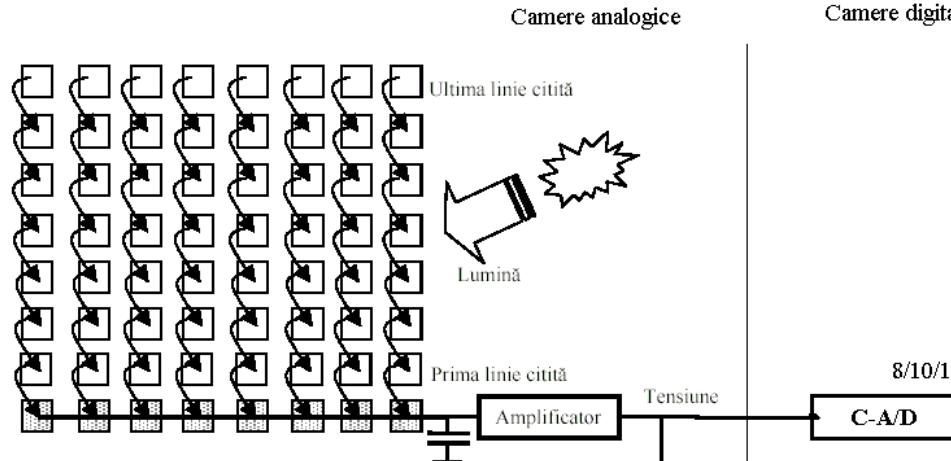
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x_s / z_s \\ y_s / z_s \end{bmatrix}$$



Imaging sensors

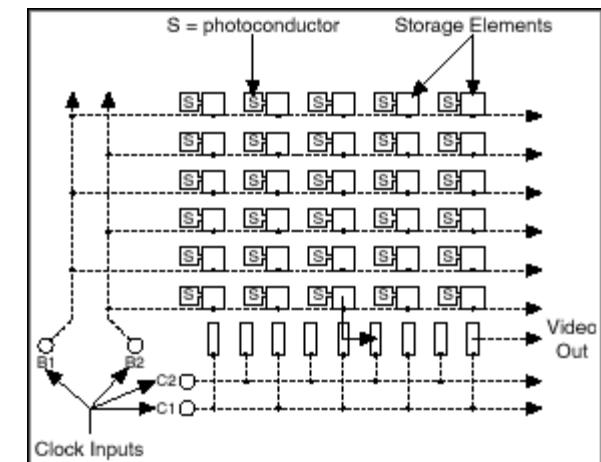
Sensor types

CCD (Charged
Coupled
Device)



Frame Transfer Architecture

Senzor video (analogic)
Pal 752 x 582
Frame grabber
analogic



Interline Transfer Architecture

Technical University of Cluj Napoca

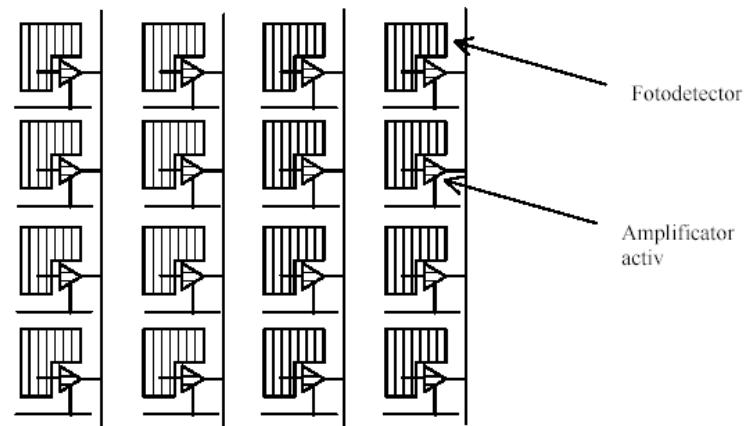
Computer Science Department



Imaging sensors

Sensor types

CMOS





Imaging sensors

CMOS vs. CCD

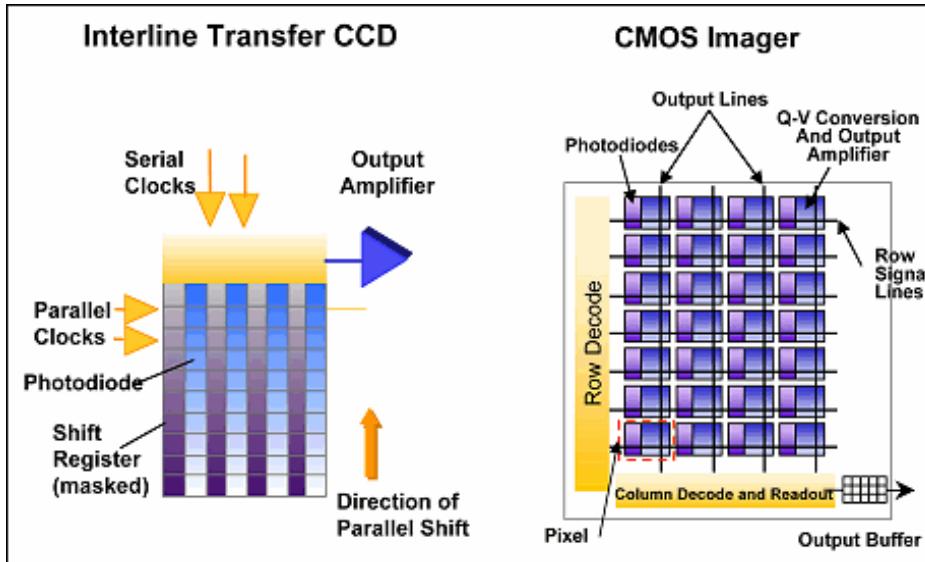


TABLE 1

Comparison of CCD and CMOS Image Sensor Features

<u>CCD</u>	<u>CMOS</u>
Smallest pixel size	Single power supply
Lowest noise	Single master clock
Lowest dark current	Low power consumption
~100% fill factor for full-frame CCD	X, Y addressing and subsampling
Established technology market base	Smallest system size
Highest sensitivity	Easy integration of circuitry
Electronic shutter without artifacts	



Imaging sensors

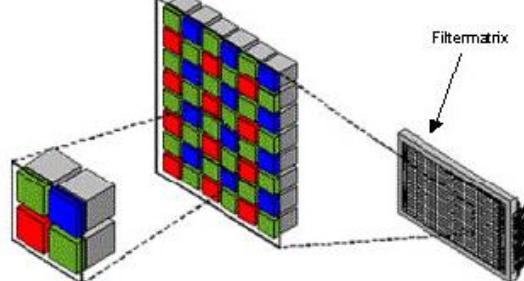
Color imagers

<http://www.siliconimaging.com/RGB%20Bayer.htm>

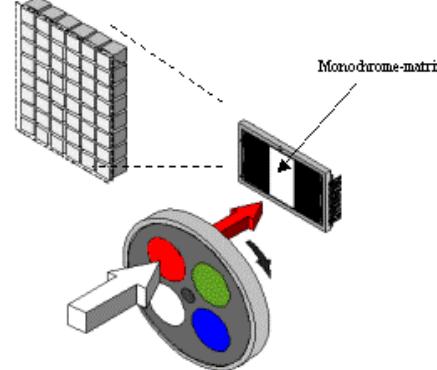
<http://www.zeiss.de/c1256b5e0047ff3f/Contents-Frame/c89621c93e2600cac125706800463c66>

a) Bayer mask

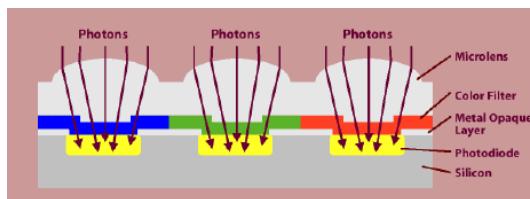
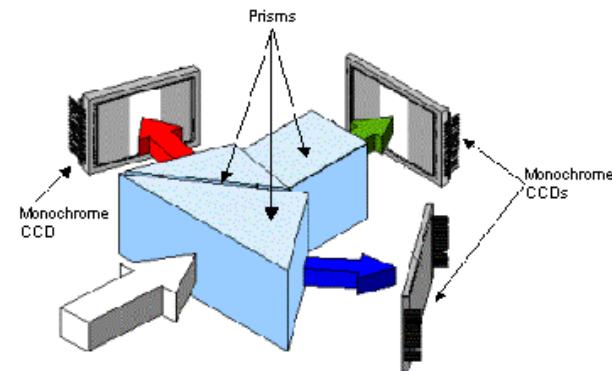
For color photos, the majority of commercial digital color cameras use pixels covered with special color filters in the three primary colors red, green and blue.



b) Filter wheel



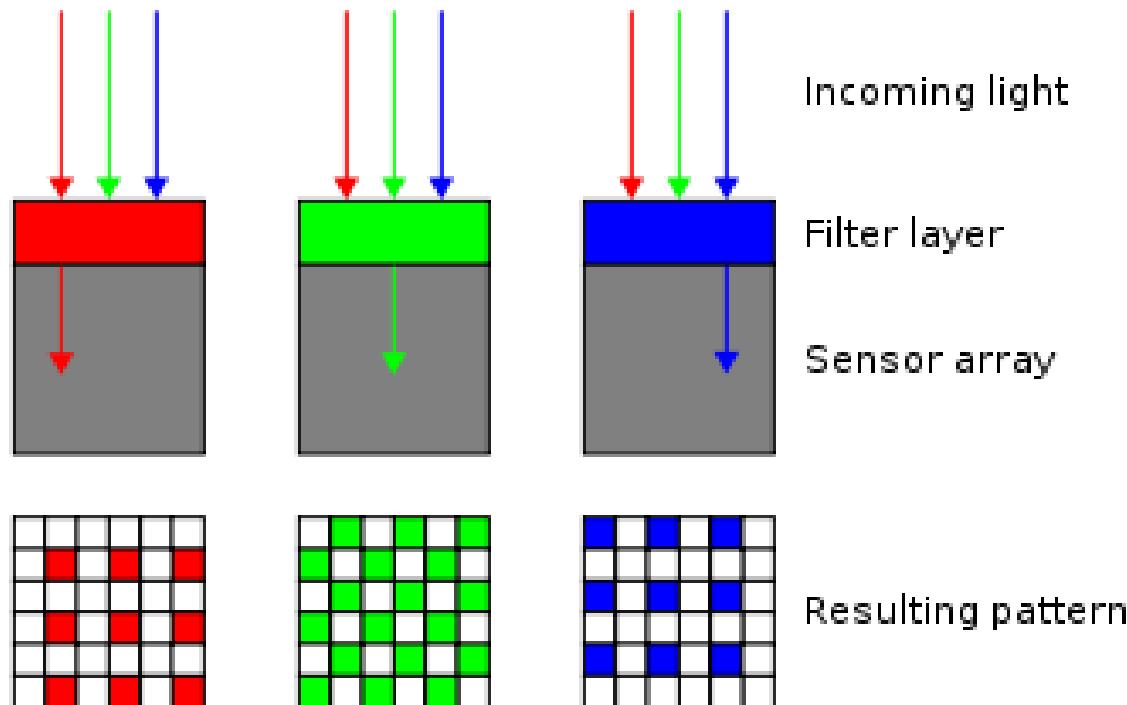
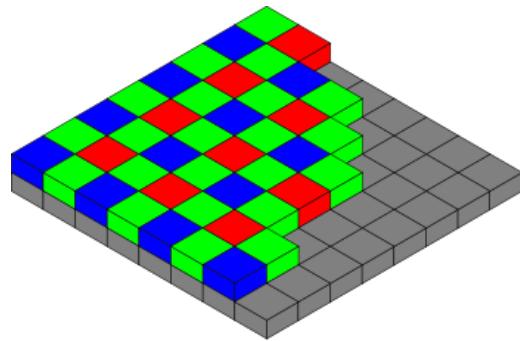
c) 3-CCD camera





Imaging sensors

Demosaicing Bayer pattern:
Bilinear interpolation



$$G = (G_n + G_w + G_e + G_s)/4$$

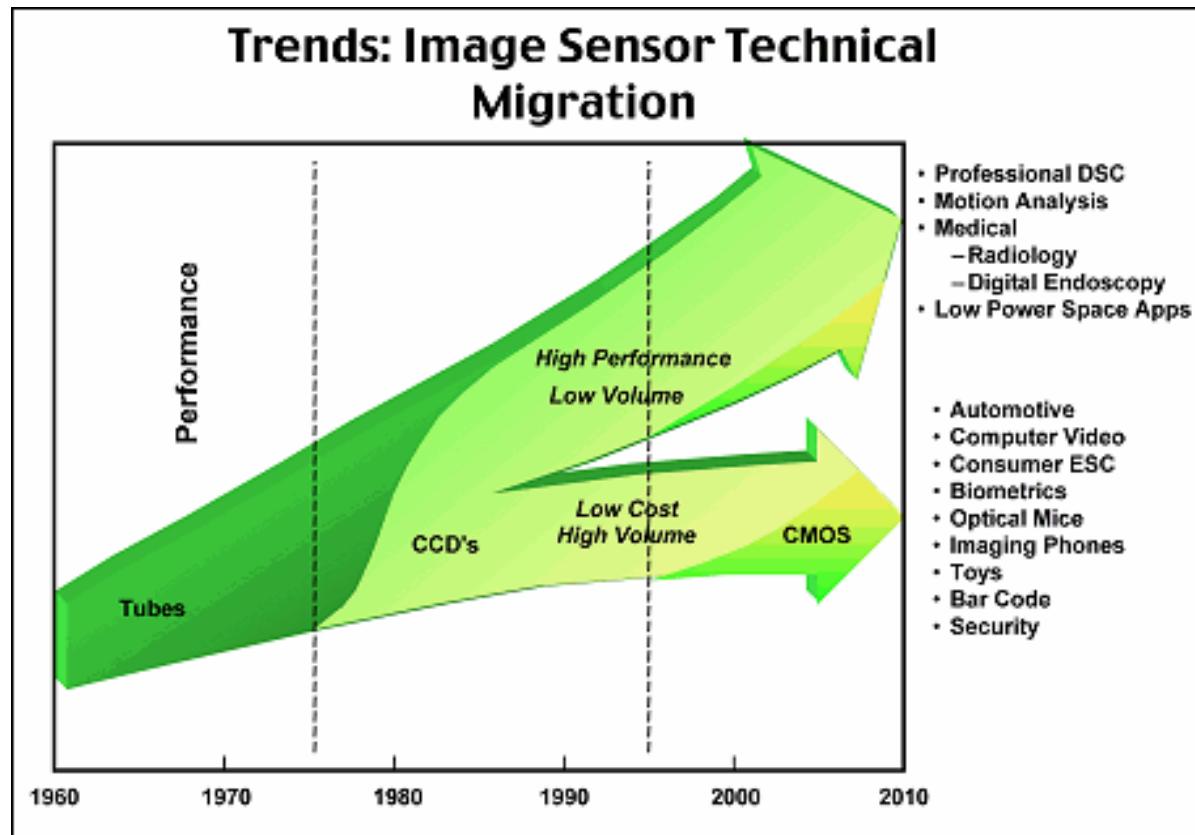
$$R_4 = (R_{nw} + R_{ne} + R_{se} + R_{sw})/4$$

$$R_{2c} = (R_n + R_s)/2$$

$$R_{2l} = (R_w + R_e)/2$$



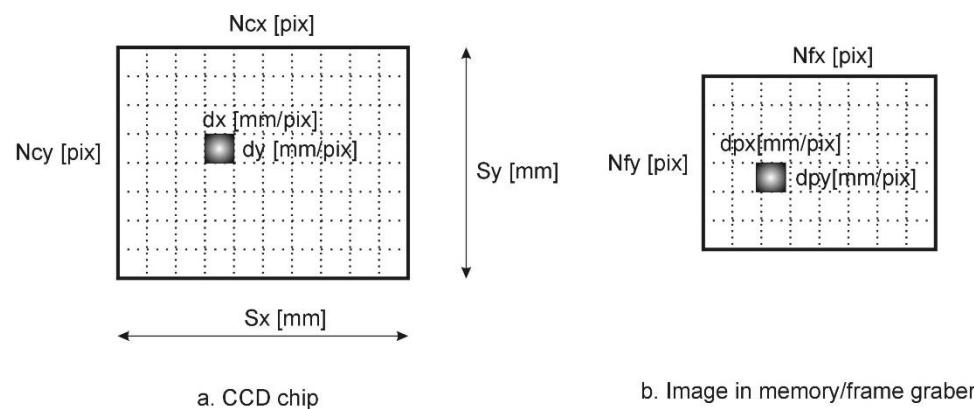
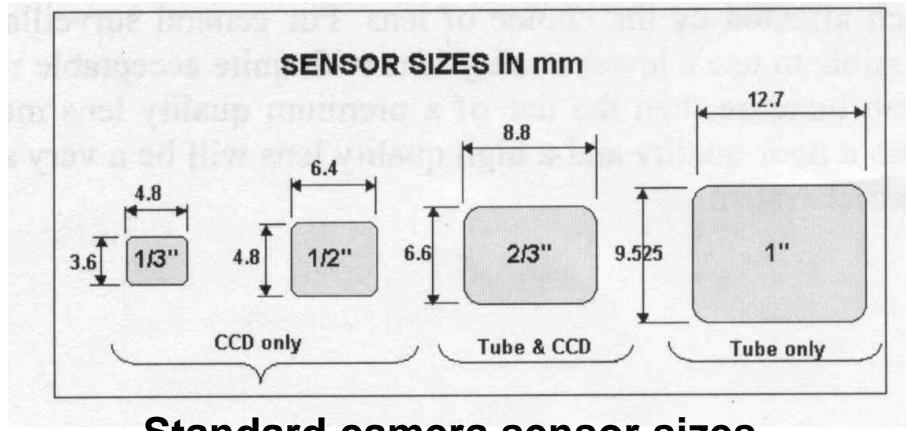
Imaging Sensors





Imager parameters

Imager (sensor) parameters



Parameters of the imager and image in memory



Imager /image parameters

Sensor parameters:

Sx – width of the sensor chip [mm]

Sy – height of the sensor chip [mm]

Ncx – number of sensor elements in camera's x direction;

Ncy – number of sensor elements in camera's y direction;

dx – center to center distance between adjacent sensor elements in X (scan line) direction;

$$dx = Sx/Ncx;$$

dy - center to center distance between adjacent CCD sensor in the Y direction;

$$dy = Sy/Ncy;$$

Image parameters (related to the image in memory/framegrabber):

Nfx – number of pixels in x direction as sampled by the computer;

Nfy – number of pixels in frame grabber's y direction

dpx – effective X dimension of pixel in frame grabber, $dpx = dx * Ncx / Nfx$;

dpy – effective Y dimension of pixel in frame grabber, $dpy = dy * Ncy / Nfy$;

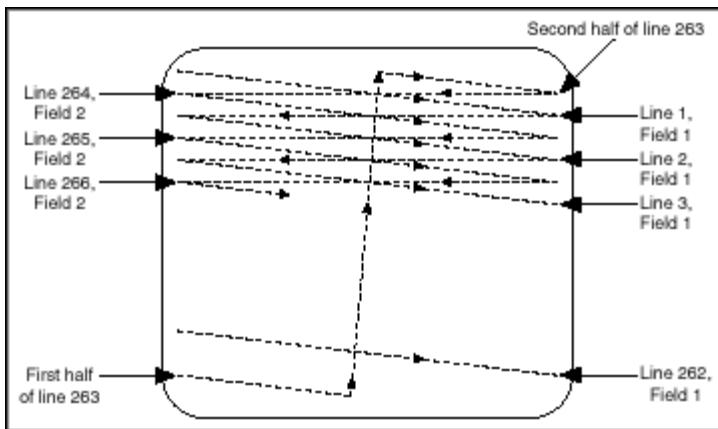
Ncx / Nfx – uncertainty factor for scaling horizontal scanlines;



Image scanning

Scanning Techniques

- Determined by application area
- TV imposed the *interlacing* of two successive images read from a standard camera



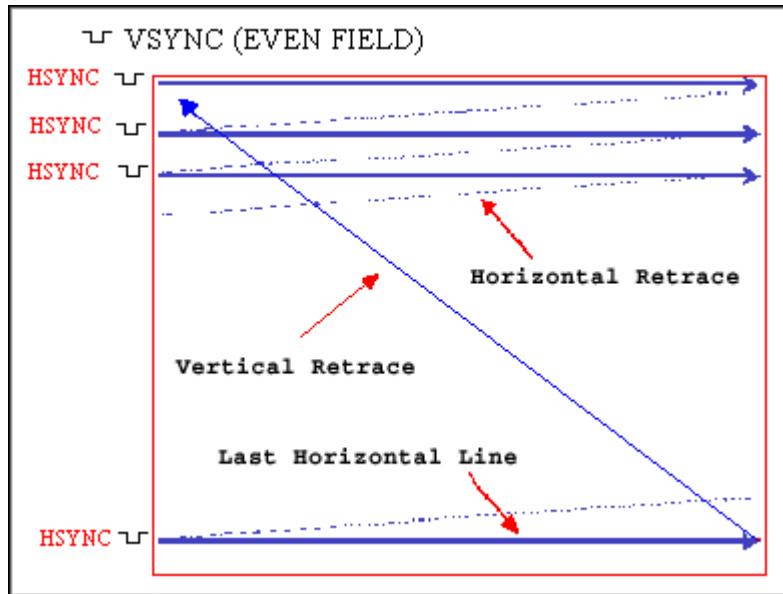
2:1 Interlaced Scanning, shown here for the NTSC Video Format. PAL and SECAM Interlacing is similar, with the difference being the number of lines in each field

- Read/display all even-numbered lines (even field, half-size)
 - Restart
 - Read/display all odd-numbered lines (odd field, half-size)
 - Stitch the even and odd fields together and form a single, full-size frame
 - Output the full-size frame
-
- even and odd frames, full frames
 - NTSC Video format: 30 frames/s (525 lines/frame)
 - PAL Video format: 25 frames/s (625 lines/frame)

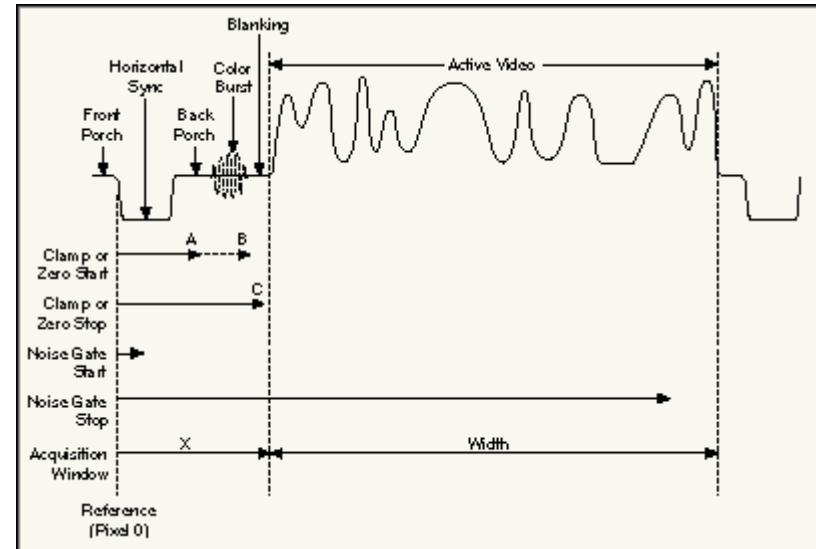


Video signals

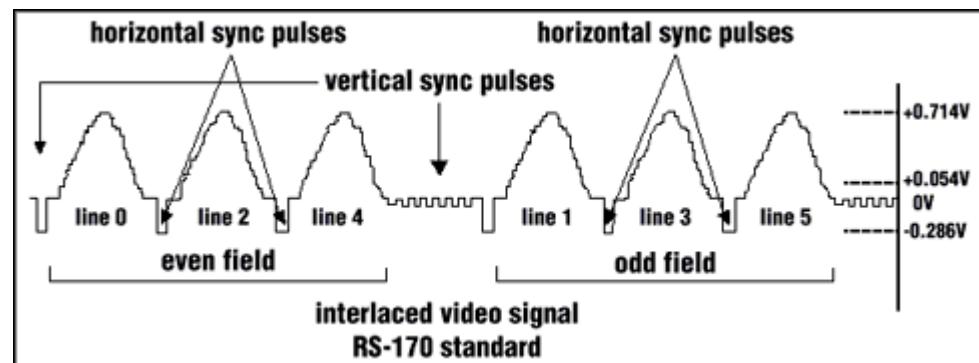
- Analog video signals



The incoming video signal conversion in individual pixel values for displaying



Analog video signal components



Standard monochrome signal



Analog standards

Format	Country	Mode	Signal Name	Frame Rate (frame/sec)	Vertical Line Resolution	Line Rate (lines/sec)	Image Size (WxH) pixels
NTSC	US, Japan	Mono	RS-170	30	525	15,750	640x480
		Color	NTSC Color	29.97	525	15,734	
PAL	Europe (except France)	Mono	CCIR	25	405	10,125	768x576
		Color	PAL Color	25	625	15,625	
SECAM	France, Eastern Europe	Mono		25	819	20,475	N/A
		Color		25	625	15,625	

Parameters of interest:

- # of lines/frame: 525 (this includes 485 lines for display; the rest are VSYNC lines for each of the two fields)
- line frequency: 15.734 kHz
- line duration: 63.556 microsec.
- active horizontal duration: 52.66 microsec.
- # active pixels/line: 640



Digital video signals

Diagram 9: Equivalence between analog composite and digital video

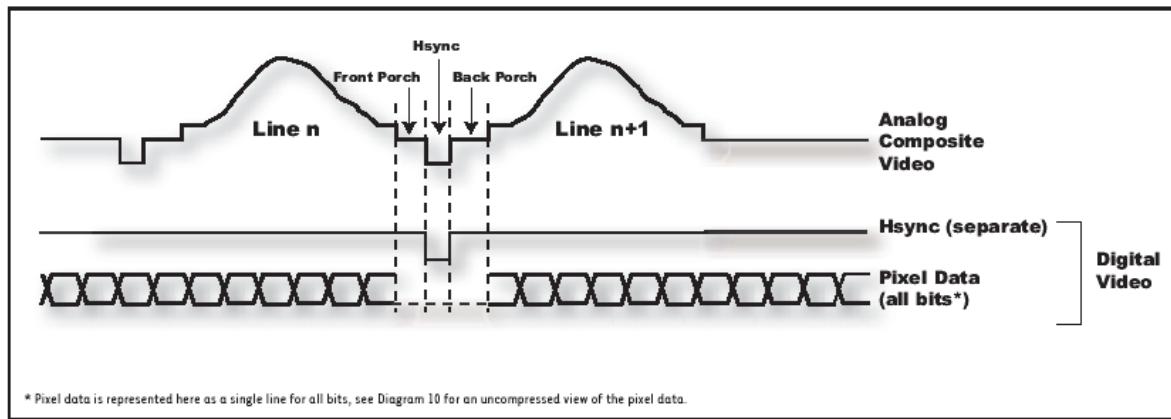
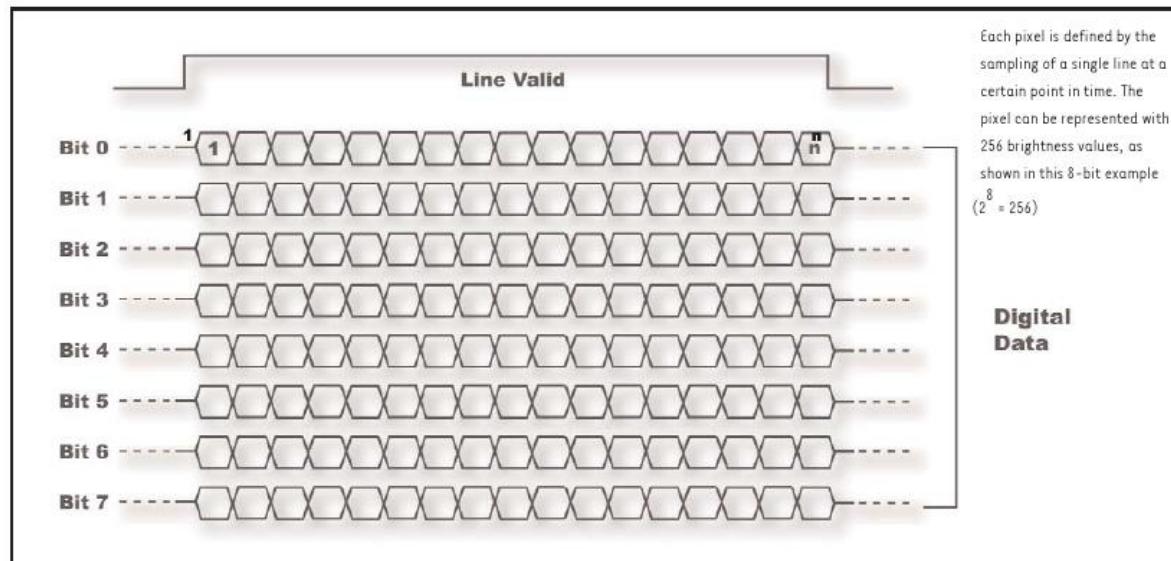


Diagram 10: 8-bit digital video.



Digital transmission standards:

- Camera Link: 1.2Gbps (base) ... 3.6Gbps (full)
- RS 422 / EIA-644 (LVDS): 655Mbps
- USB 2.0: 480 Mbps
- IEEE 1394: 400 Mbps
- USB 1.1: 12 Mbps



Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester)

Binary Images: Tresholding (I)

Binary Images: Simple Geometrical Properties(II)



Presentation outline

- **Introduction**
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image
- Binary Images: geometric properties
- Projections
- Run-Length Coding



Binary Image Processing

INTRODUCTION

The simplest type of image which is used widely in a variety of industrial and medical applications is **binary**, i.e. a black-and-white or silhouette image.

Advantages

- Easy to acquire: simple digital cameras; thresholding may be applied to grey-level images.
- Low storage: no more than 1 bit/pixel, often this can be reduced by compression (e.g. run-length coding).
- Simple and fast processing: the algorithms are in most cases much simpler than those applied to grey-level images.

Disadvantages

- Limited application: application is restricted to tasks where internal detail is not required as a distinguishing characteristic.
- Does not extend to 3D: the 3D nature of objects can rarely be represented by silhouettes.
- Specialized lighting is required to obtain reliable binary images without restricting the environment.



Binary Image Processing

THRESHOLDING

In the simplest case, an image may consist of a single object or several separated objects of relatively high intensity, viewed against a background of relatively low intensity. This allows figure/ground separation by thresholding.



The goal of thresholding is to segment an image into regions of interest and to remove all other regions considered inessential. The simplest thresholding methods use a single threshold in order to isolate objects of interest. In many cases, however, no single threshold provides a good segmentation result over an entire image. In such cases variable and multilevel threshold techniques based on various statistical measures are used.



Binary Image Processing

Thresholding –individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value and as “background” pixels otherwise (assuming an object to be brighter than the background)

Segmentation -partitioning of an image into regions



Thresholding



Segmentation



Binary Image Processing

Segmentation -partitioning of an image into regions

$$f(i,j) \rightarrow P_1, P_2, \dots, P_k$$

Def. 1.1 A region is a subset of an image

Def. 1.2 Segmentation is grouping pixels into regions, such that:

$$\sum_{i=1}^k P_i = \text{entire image (exhaustive partitioning)}$$

$$P_i \cap P_j = 0 \text{ if } i \neq j \quad (\text{exclusive partitioning})$$

Each region P_i satisfies a predicate; that is all points of the partition have same common property.

Pixels belonging to adjacent regions, when taken jointly, do not satisfy the predicate.



Strategies of threshold selection

- Global Thresholding
- Semithresholding
- Multilevel thresholding
- Variable thresholding
 - Threshold selection using mean and standard deviation
 - Threshold selection by Maximizing Between-Class Variance (Otsu method)
 - Threshold selection by searching the best approximation of the gray level image with a two leveled image



Global Thresholding

Let $f(x,y)$ be the source image
and $b(x,y)$ the binary image.

$$b(x, y) = \begin{cases} 1, & f(x, y) \leq T \\ 0, & \text{otherwise} \end{cases}$$

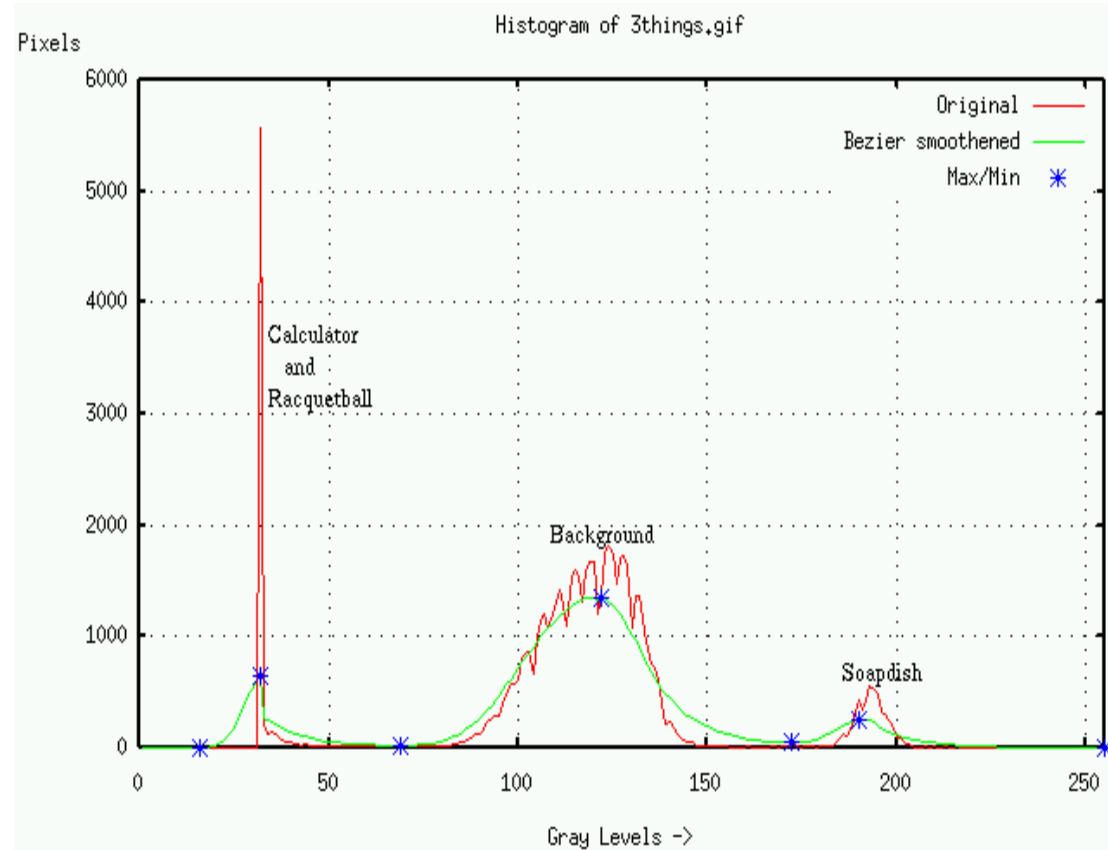
Object $\in [T_1, T_2]$

$$b(x, y) = \begin{cases} 1, & T_1 \leq f(x, y) \leq T_2 \\ 0, & \text{otherwise} \end{cases}$$

Object $\in Z$

where Z is a set of intensity values

$$b(x, y) = \begin{cases} 1, & f(x, y) \in Z \\ 0, & \text{otherwise} \end{cases}$$



Histogram analysis



Semithresholding

- Pixels whose values lie within a given threshold range retain their original values. Pixels with values lying outside of the threshold range are set to 0.

$$b(x, y) = \begin{cases} f(x, y) & \text{if } h \leq f(x, y) \leq k \\ 0 & \text{otherwise} \end{cases}$$

- Regions of high values can be isolated using:

$$b(x, y) = \begin{cases} f(x, y) & \text{if } f(x, y) \geq k \\ 0 & \text{otherwise} \end{cases}$$

- Regions of low values can be isolated using:

$$b(x, y) = \begin{cases} f(x, y) & \text{if } f(x, y) \leq k \\ 0 & \text{otherwise} \end{cases}$$



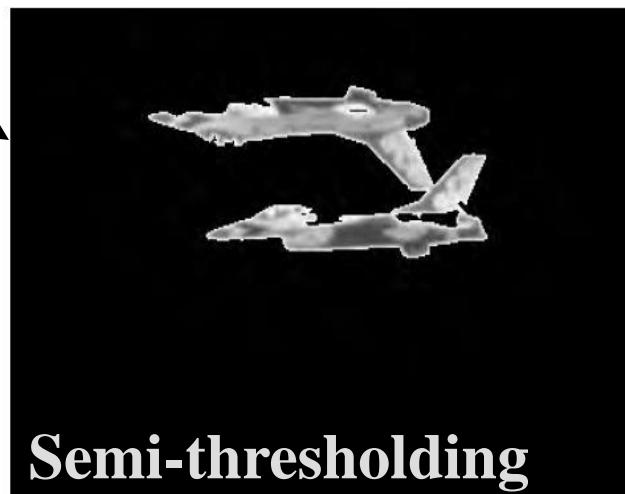
Example



Original image



Global Thresholding



Semi-thresholding



Multilevel thresholding

- Multilevel thresholding allows for segmentation of pixels into multiple classes.
- For example, if the image histogram contains three peaks, then it is possible to segment the image using two thresholds. These thresholds divide the value set into three non-overlapping ranges, each of which can be associated with a unique value in the resulting image.
- Let $f(x,y)$ be the source image and let k_1, \dots, k_n be threshold values satisfying $k_1 > k_2 > \dots > k_n$. These values partition \mathbb{R} into $n+1$ intervals which are associated with values v_1, \dots, v_{n+1} in the thresholded result image.
- The threshold image b is defined by:

$$b(x, y) = \begin{cases} v_1 & \text{if } f(x, y) < k_1 \\ v_i & \text{if } k_{i-1} \leq f(x, y) < k_i \\ v_{n+1} & \text{if } k_n \leq f(x, y) \end{cases}$$



Variable Thresholding

- Variable thresholding allows different threshold levels to be applied to different regions of an image.
- Let $f(x,y)$ be the source image and let $d(x,y)$ denote the local (region) threshold value associated with each point in the image, that is $d(x,y)$ is the threshold value associated with the region in which point (x,y) lies.
- The thresholded image $b(x,y)$ is defined by:

$$b(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq d(x, y) \\ 0 & \text{if } f(x, y) < d(x, y) \end{cases}$$



Threshold selection using mean and standard deviation

- Let $f(x,y)$, be an image
- The mean of $f(x,y)$ is: $\mu = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n f(i, j)$
- The standard deviation of the image a is:

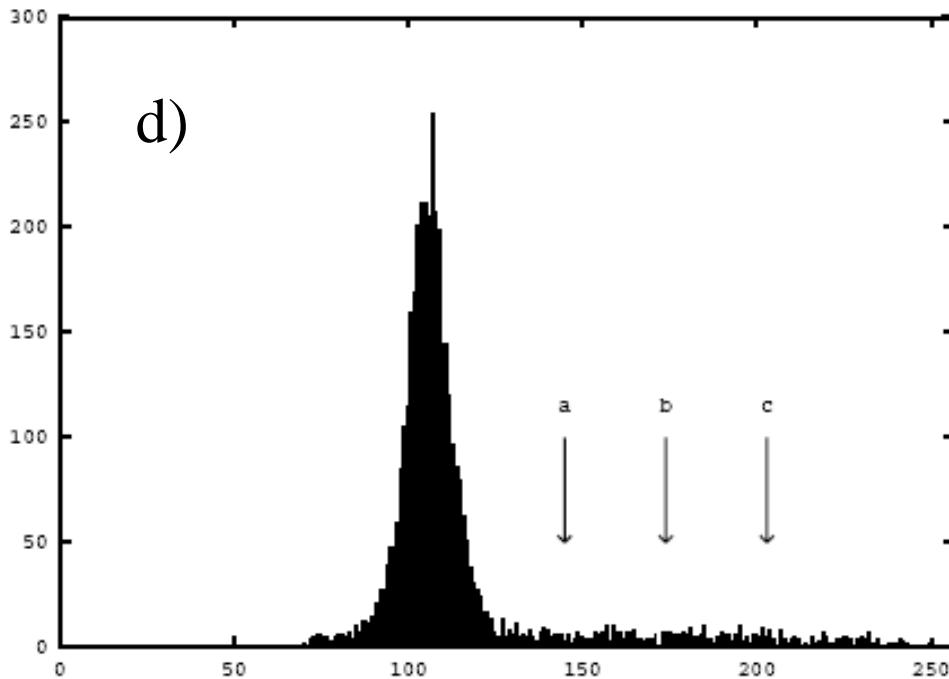
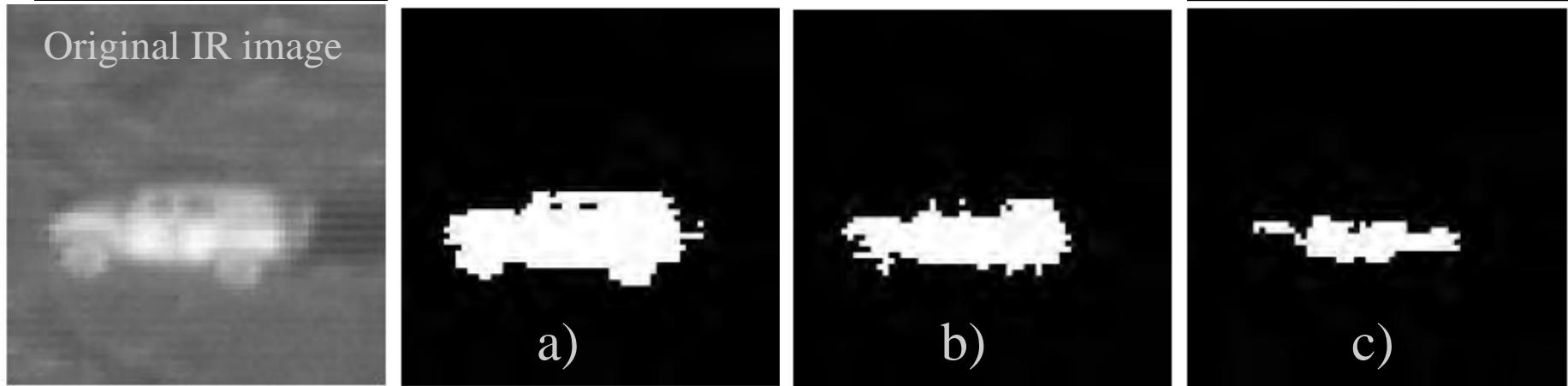
$$\sigma = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n [f(i, j) - \mu]^2}$$

- The threshold level T is set at: $T = k_1\mu + k_2\sigma$ where the constants k_1 and k_2 are image type dependent.
- For typical low-resolution IR images $k_1 = k_2 = 1$ seems to work fairly well.
- For higher resolutions $k_1 = 1$ or $k_1 = 1.5$ and $k_2 = 2$ may yield better results.



Example

Original IR image

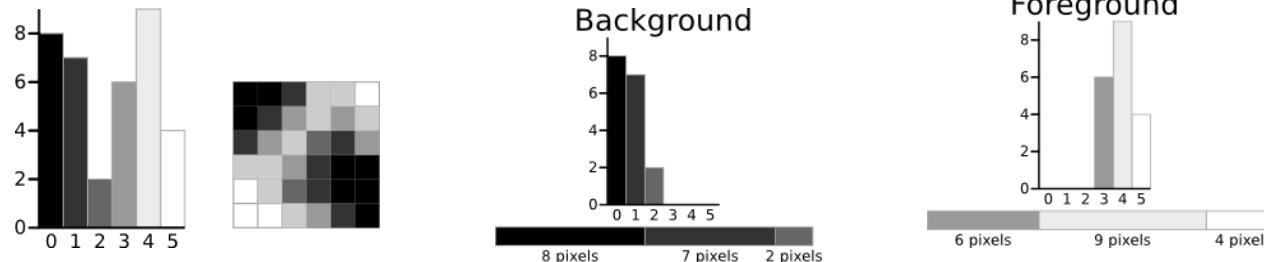


- a) $k_1=k_2=1, T=145$
- b) $k_1=1, k_2=2, T=174$
- c) $k_1=1.5, k_2=1, T=203$
- d) histogram of the original image and the threshold levels corresponding to images a), b) and c)



The Otsu Method – one threshold

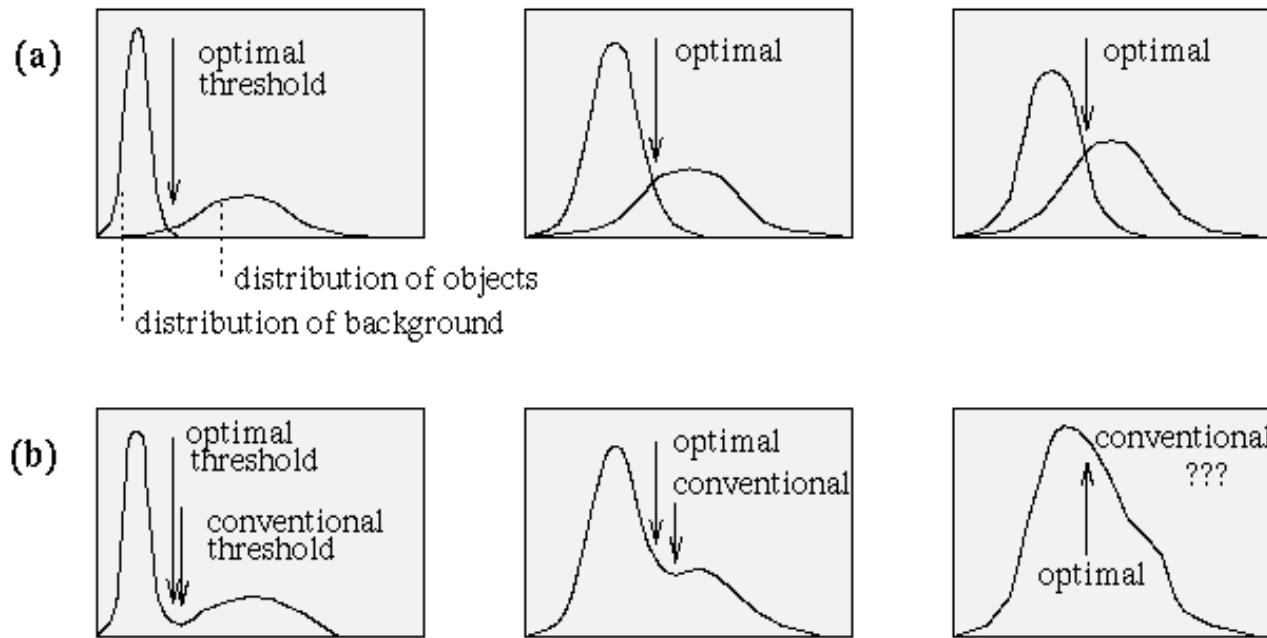
- Problem statement: we have two groups of pixels, one with one range of values and one with another. Thresholding is difficult because these ranges usually overlap.
- Idea: minimize the error of classifying a background pixel as a foreground one or vice versa.



- Minimize the area under the histogram for one region that lies on the other region's side of the threshold. Consider the values in the two regions as two *clusters*.
- Set the threshold so as to try to make each cluster as tight as possible, thus (hopefully!) minimizing their overlap.
- **A measure of group homogeneity is variance.** A group with high homogeneity will have low variance. A group with low homogeneity will have high variance.



Adaptive thresholding



Gray level histograms approximated by two normal distributions; the threshold is set to give minimum probability of segmentation error.

(a) Probability distributions of background and objects

(b) Corresponding histograms and optimal threshold



Otsu's Thresholding Method (1979)

- Find the threshold that *minimizes the weighted within-class variance* which turns out to be the same as *maximizing the between-class variance*.
- Operates directly on the gray level histogram [e.g. 256 numbers, $P(i)$], so it's fast (once the histogram is computed).
- Histogram (and the image) are *bimodal*.
- Assumes uniform illumination (implicitly), so the bimodal brightness behavior arises from object appearance differences only.



Otsu's Thresholding Method (2)

- The **weighted within-class variance** is: $\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$
- The **class probabilities are estimated** as: $q_1(t) = \sum_{i=1}^t P(i) \quad q_2(t) = \sum_{i=t+1}^I P(i)$
 - Where $P(i) = \frac{H(i)}{I}$, H(i) the i-th entry in the histogram
 $q_2 = 1 - q_1$
- And **the class means are given** by: $\mu_1(t) = \sum_{i=1}^t \frac{iP(i)}{q_1(t)} \quad \mu_2(t) = \sum_{i=t+1}^I \frac{iP(i)}{q_2(t)}$
- Finally, **the individual class variances** are:

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^I [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

- Now, we could actually stop here. All we need to do is just run through the full range of t values [1, 256] and pick the value that **minimizes the within class variance**.
- But the relationship between the within-class and between-class variances can be exploited to generate a recursion relation that permits a much faster calculation.



Otsu's Thresholding Method (3)

- The relationship between the total variance and the within group variance can make the calculation of the best threshold less computationally complex.

$$\sigma^2 = \sum_{i=1}^{I-1} [(i - \mu)^2 P(i)] \quad \mu = \sum_{i=1}^I i P(i)$$

$$\begin{aligned}\sigma^2 &= \sum_{i=1}^t [i - \mu_1(t) + \mu_1(t) - \mu]^2 P(i) + \sum_{i=t+1}^I [i - \mu_2(t) + \mu_2(t) - \mu]^2 P(i) \\ &= \sum_{i=1}^t \{[i - \mu_1(t)]^2 + 2[i - \mu_1(t)][\mu_1(t) - \mu] + [\mu_1(t) - \mu]^2\} P(i) \\ &\quad + \sum_{i=t+1}^I \{[i - \mu_2(t)]^2 + 2[i - \mu_2(t)][\mu_2(t) - \mu] + [\mu_2(t) - \mu]^2\} P(i)\end{aligned}$$

But: $\sum_{i=1}^t [i - \mu_1(t)][\mu_1(t) - \mu] P(i) = 0 \quad \sum_{i=t+1}^I [i - \mu_2(t)][\mu_2(t) - \mu] P(i) = 0$

$$\begin{aligned}\rightarrow \sigma^2 &= \sum_{i=1}^t [i - \mu_1(t)]^2 P(i) + [\mu_1(t) - \mu]^2 q_1(t) + \sum_{i=t+1}^I [i - \mu_2(t)]^2 P(i) + [\mu_2(t) - \mu]^2 q_2(t) \\ \sigma^2 &= [q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)] + \{q_1(t)[\mu_1(t) - \mu]^2 + q_2(t)[\mu_2(t) - \mu]^2\}\end{aligned}$$

$$\mu = q_1(t)\mu_1(t) + q_2(t)\mu_2(t) \quad 1 - q_1(t) = q_2(t)$$

$$\rightarrow \sigma^2 = \sigma_w^2(t) + q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$$



Otsu's Thresholding Method (4)

- **Between/Within/Total Variance:** For any given threshold, the *total variance* is the sum of the *within-class variances* (weighted) and the *between class variance*, which is the sum of weighted squared distances between the class means.

- As shown in the previous slide, we can express the total variance:

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class}} + \underbrace{q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2}_{\text{Between-class}},$$

- Since the total is constant and independent of t , the effect of changing the threshold is simply to move the contributions of the two terms back and forth.
- So, *minimizing the within-class variance is the same as maximizing the between-class variance.*



Otsu's Thresholding Method (5)

- For each potential threshold, T:
 1. Separate the pixels into two clusters according to the threshold
 2. Find the mean of each cluster
 3. Square the difference between the means
 4. Multiply by the number of pixels in one cluster times the number in the other.
- The optimal threshold is the one that maximizes the between-class variance (or, conversely, minimizes the within-class variance).
- The nice thing about this is that we can compute the quantities *using a recursion relation* as we run through the range of t values.

Initialization... $q_1(1) = P(1)$; $\mu_1(0) = 0$

Recursion... $q_1(t+1) = q_1(t) + P(t+1)$

$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)}$$

$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}$$



Example

Original image

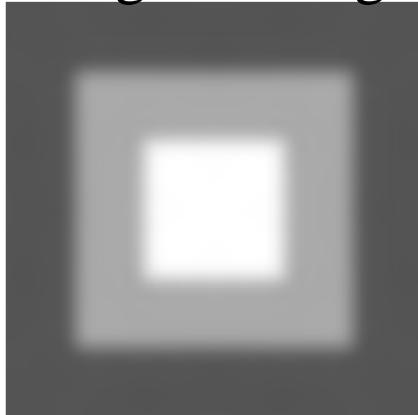
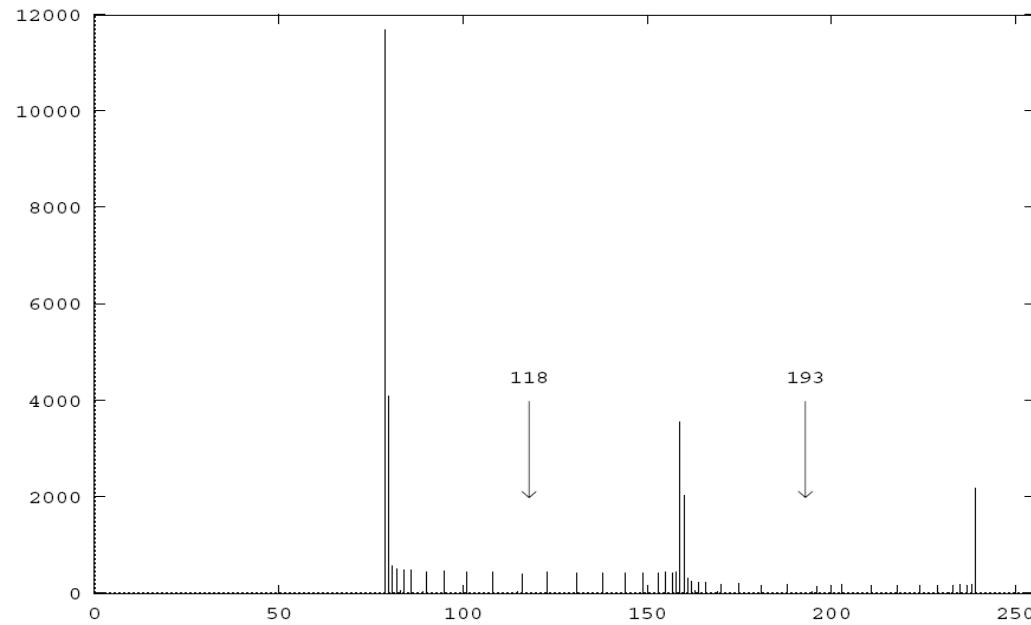
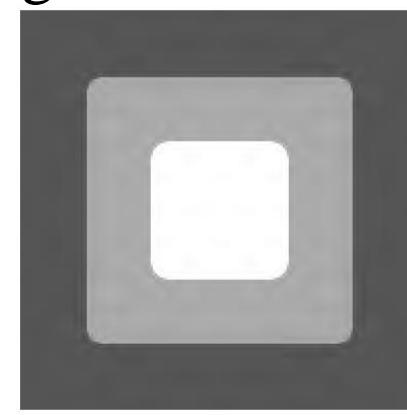


Image after thresholding



Histogram of the original image with threshold values marked.

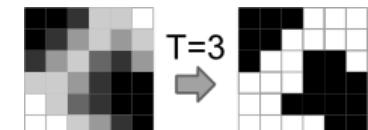


Threshold selection by best approximation with a two level image

- Given an image f_{ij} the binary image is $g_{ij} = \begin{cases} a & \text{if } f_{ij} < t \\ b & \text{if } f_{ij} \geq t \end{cases}$
 - Where t is the threshold
 - a and b are constants chosen to minimize the distance on the corresponding intervals.
 - The image has P gray levels

- The Euclidian distance on the interval $[0, P-1]$ is

$$D = \sum_{i=1}^D \sum_{j=1}^D (f_{ij} - g_{ij})^2 = \sum_{i=1}^D \sum_{j=1}^D [(f_{ij})^2 - 2f_{ij}g_{ij} + (g_{ij})^2] =$$
$$= \underbrace{\sum_{k=0}^{t-1} k^2 H_k - 2a \underbrace{\sum_{k=0}^{t-1} k H_k}_{F(a)} + a^2 \underbrace{\sum_{k=0}^{t-1} H_k}_{F(b)}}_{\text{F}(a)} + \underbrace{\sum_{k=t}^{P-1} k^2 H_k - 2b \underbrace{\sum_{k=t}^{P-1} k H_k}_{F(b)} + b^2 \underbrace{\sum_{k=t}^{P-1} H_k}_{F(b)}}$$



- The minimum of $F(a)$ and $F(b)$ are the mean values:

$$a = \text{Min}(F(a)) = \mu_i = \frac{\sum_{k=0}^{t-1} k H_k}{\sum_{k=0}^{t-1} H_k}$$

$$b = \text{Min}(F(b)) = \mu_s = \frac{\sum_{k=t}^{P-1} k H_k}{\sum_{k=t}^{P-1} H_k}$$



Threshold selection by best approximation with a two level image(2)

- The distance on the entire image is:

$$D = \sum_{i=1}^D \sum_{j=1}^D (f_{ij} - g_{ij})^2 = \underbrace{\sum_{k=0}^{P-1} k^2 H_k}_{\text{const}} - \left[\frac{\left(\sum_{k=0}^{t-1} k H_k \right)^2}{\sum_{k=0}^{t-1} H_k} + \frac{\left(\sum_{k=t}^{P-1} k H_k \right)^2}{\sum_{k=t}^{P-1} H_k} \right]$$

F – must be maximum for an optimal threshold

- $F = \mu_i^2 A_i + \mu_s^2 A_s$ where
 - A_i and A_s are the areas below and above the threshold
- F has to be evaluated for all t and the t maximizing F is selected.
- The algorithm can be applied for several levels of segmentation t_1, t_2, \dots, t_n by dividing the intervals. At the first step the threshold will be $t_{n/2}$, next the algorithm is applied on the intervals $[0, t_{n/2}]$ and $[t_{n/2}, P-1]$ and so on.



Presentation outline

- Introduction
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image
- **Binary Images: geometric properties**
- Projections
- Run-Length Coding



Binary Image Processing

Simple Geometrical Properties

Area

$$A = \iint b(x, y) dx dy$$

$$A = \sum_{i=1}^n \sum_{j=1}^m b(i, j)$$

Position

The usual practice is to chose the center of area. The center of area is the center of mass of a figure of the same shape with constant mass per unit area.

The center of the mass is that point where all the mass of the object could be concentrated without changing the first moment of the object about any axis.

$$\bar{x} \iint b(x, y) dx dy = \iint x b(x, y) dx dy$$

$$\bar{i} \sum_{i=1}^n \sum_{j=1}^m b(i, j) = \sum_{i=1}^n \sum_{j=1}^m i b(i, j)$$

The moment about the x-axis

$$\bar{y} \iint b(x, y) dx dy = \iint y b(x, y) dx dy$$

$$\bar{j} \sum_{i=1}^n \sum_{j=1}^m b(i, j) = \sum_{i=1}^n \sum_{j=1}^m j b(i, j)$$

The moment about the y-axis



Binary Image Processing

Position

(\bar{x}, \bar{y}) And (\bar{i}, \bar{j}) represent the position of the center of area

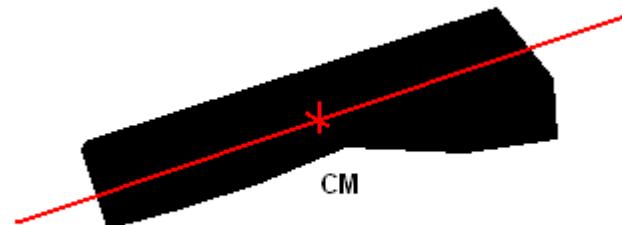
$$\bar{i} = \frac{\sum_{i=1}^n \sum_{j=1}^m i b(i,j)}{A}$$

$$\bar{j} = \frac{\sum_{i=1}^n \sum_{j=1}^m j b(i,j)}{A}$$

Orientation

Let assume that the object is somewhat elongated; then the orientation of the axis of elongation can be used to define the orientation of the object.

The axis of elongation is equivalent with the axis of least second order moment. It gives the line for which the integral of the square of the distance to points in the object is a minimum.





Binary Image Processing

Orientation

Least inertia axis: $E = \iint r^2 b(x, y) dx dy$

Where r is the perpendicular distance from the point (x, y) to the line.

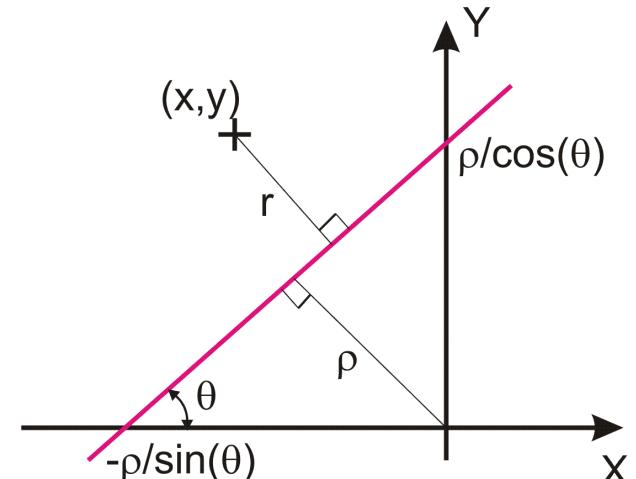
To choose a particular line in the plane, we need to specify two parameters. A convenient pair consists of the distance ρ from the origin to the closest point on the line and the angle θ between the x-axis and the line, measured counter clockwise.

The line equation is:

$$x \sin\theta - y \cos\theta + \rho = 0$$

The line intersects the x-axis at $-\rho/\sin\theta$ and the y-axis at $+\rho/\cos\theta$.

The closest point on the line to the origin is at $(-\rho\sin\theta, +\rho\cos\theta)$.

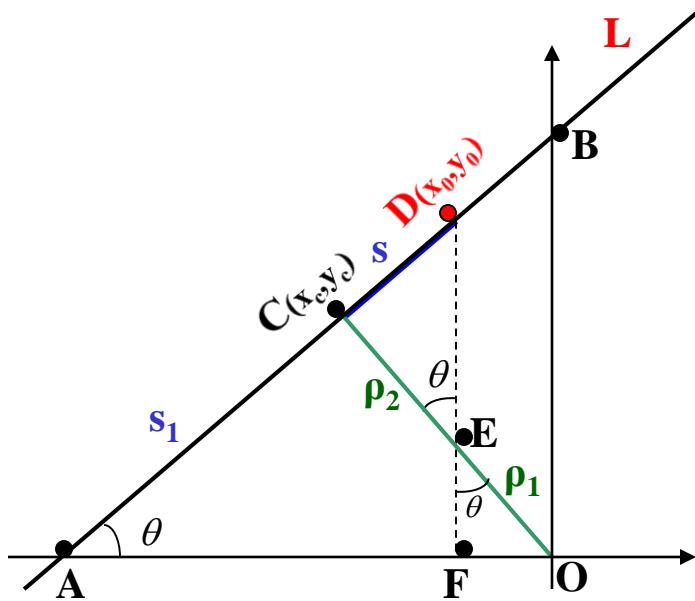


Given a point (x, y) on the object, we need to find the closest point (x_0, y_0) on the line, so that we can compute the distance r .

$$r^2 = (x \sin\theta - y \cos\theta + \rho)^2$$



Binary Image Processing



$$\rho = \rho_1 + \rho_2$$

$\bullet C(x_c, y_c)$ is the closest point on the line to the origin, where $x_c = -\rho \sin \theta$; $y_c = \rho \cos \theta$

\bullet Try to write the parametric equation for the points on the line: $(x_0, y_0) \in L$

\bullet Consider s the distance along the line from the point closest to the origin: $CD = s$

$$\begin{aligned} \Delta FEO: \sin \theta &= \frac{FO}{EO} = \frac{FO}{\rho_1} \Rightarrow FO = \rho_1 \cdot \sin \theta = (\rho - \rho_2) \cdot \sin \theta \\ \Delta ECD: \frac{\sin \theta}{\cos \theta} &= \frac{CD}{CE} = \frac{s}{\rho_2} \Rightarrow \rho_2 = \frac{s \cos \theta}{\sin \theta} \end{aligned}$$

$$\begin{aligned} \Delta AFD: \sin \theta &= \frac{FD}{AD} = \frac{FD}{s + s_1} \Rightarrow FD = (s + s_1) \cdot \sin \theta \\ \Delta AOC: \frac{\sin \theta}{\cos \theta} &= \frac{CO}{AC} = \frac{\rho}{s_1} \Rightarrow s_1 = \frac{\rho \cos \theta}{\sin \theta} \end{aligned}$$

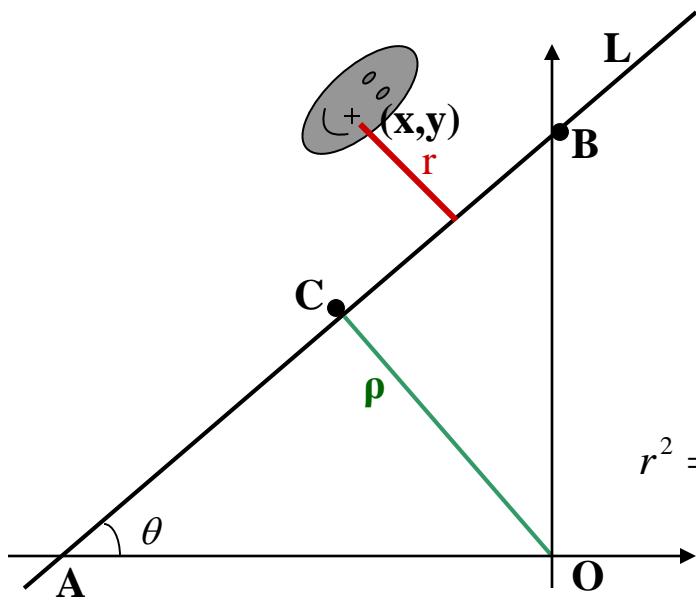
$$x_0 = -\rho \sin \theta + s \cos \theta$$

Parametric equation for the points on the line

$$y_0 = \rho \cos \theta + s \sin \theta$$



Binary Image Processing



- Given a point (x, y) on the object, we need to find the closest point (x_0, y_0) on the line L so that we can compute the distance r between the point and the line.

$$r^2 = (x - x_0)^2 + (y - y_0)^2$$

- Substitute the parametric equations for x_0 and y_0 :

$$r^2 = (x^2 + y^2) + \rho^2 + 2\rho(x \sin \theta - y \cos \theta) - 2s(x \cos \theta + y \sin \theta) + s^2$$

- Differentiate with respect to s and set the result equal to 0:

$$s = x \cos \theta + y \sin \theta$$

- This result can now be substituted back into the parametric equations for x_0 and y_0 :

$$\left. \begin{array}{l} x - x_0 = +\sin \theta(x \sin \theta - y \cos \theta + \rho) \\ y - y_0 = -\cos \theta(x \sin \theta - y \cos \theta + \rho) \end{array} \right\} \implies r^2 = (x \sin \theta - y \cos \theta + \rho)^2$$

- Comparing this result with the equation for the line, we see that the line is the locus of points for which $r=0$!

$$E = \iint_I r^2 b(x, y) dxdy = \iint_I (x \sin \theta - y \cos \theta + \rho)^2 b(x, y) dxdy$$



Binary Image Processing

Orientation

$$E = \iint r^2 b(x, y) dx dy = \iint (x \sin \theta - y \cos \theta + \rho)^2 b(x, y) dx dy$$

Differentiating with respect to ρ and setting the result to zero lead to
 $E'_\rho(\theta, \rho) = 0$

$$\begin{aligned} E'_\rho &= \iint 2(x \sin \theta - y \cos \theta + \rho) b(x, y) dx dy = \\ &= 2 \sin \theta \iint x b(x, y) dx dy - 2 \cos \theta \iint y b(x, y) dx dy + \rho \iint b(x, y) dx dy \\ E'_\rho &= 2A(\bar{x} \sin \theta - \bar{y} \cos \theta + \rho) = 0 \quad \text{where } \bar{x} \text{ and } \bar{y} \text{ is the center of area defined above.} \end{aligned}$$

Thus the axis of least second moment passes through the center of area. This suggests a change of coordinates to $x' = x - \bar{x}$ and $y' = y - \bar{y}$

$$x \sin \theta - y \cos \theta + \rho = x' \sin \theta - y' \cos \theta$$

$$a = \iint (x')^2 b(x, y) dx' dy'$$

$$E = a \sin^2 \theta - b \sin \theta \cos \theta + c \cos^2 \theta$$

$$b = 2 \iint (x' y') b(x, y) dx' dy'$$

where a , b , c , are the second order moments given by:

$$c = \iint (y')^2 b(x, y) dx' dy'$$



Binary Image Processing

Orientation

Now we can write the formula of E in the form

$$E = \frac{1}{2}(a+c) - \frac{1}{2}(a-c)\cos 2\theta - \frac{1}{2}b\sin 2\theta$$

Differentiating with respect to θ and setting the result to zero, we have

$$E'_\theta = (a-c)\sin 2\theta - b\cos 2\theta = 0$$

$$\tan 2\theta = \frac{b}{a-c} \quad \text{unless } b=0 \text{ and } a=c$$

Consequently

$$\sin 2\theta = \pm \frac{b}{\sqrt{b^2 + (a-c)^2}} \quad \text{and} \quad \cos 2\theta = \frac{a-c}{\sqrt{b^2 + (a-c)^2}}$$

Of the two solutions, the one with the plus signs in the expressions for $\sin 2\theta$ and $\cos 2\theta$ leads to the desired **minimum for E (E_{\min})**.

Consequently, the solution with minus signs in the two expressions corresponds to the **maximum value for E (E_{\max})**.



Binary Image Processing

$$A_i = \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} I_i(r, c)$$

$$\bar{r}_i = \frac{1}{A_i} \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} r I_i(r, c) \quad \bar{c}_i = \frac{1}{A_i} \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} c I_i(r, c)$$

$$\tan(2\theta_i) = 2 \frac{\sum_{r=0}^{N-1} \sum_{c=0}^{N-1} (r - \bar{r}_i)(c - \bar{c}_i) I_i(r, c)}{\sum_{r=0}^{N-1} \sum_{c=0}^{N-1} (c - \bar{c}_i)^2 I_i(r, c) - \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} (r - \bar{r}_i)^2 I_i(r, c)}$$

$$T = 4\pi \left(\frac{A}{P^2} \right) \quad \text{thinness ratio}$$

$$F = \frac{E_{min}}{E_{max}} \quad \text{roundness (F=0 straight line, F=1 circle)}$$



Presentation outline

- Introduction
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image
- Binary Images: geometric properties
- **Projections**
- Run-Length Coding



Projections

Projections

The integral of $b(x,y)$ along the line L gives one value of the projection.

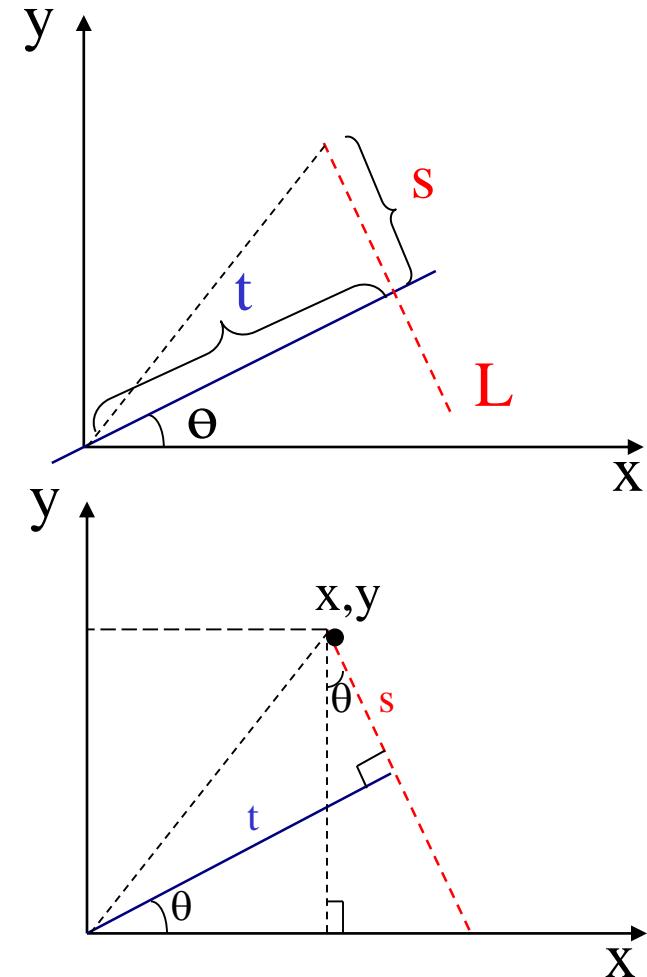
$$p_\theta(t) = \int b(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)ds$$

Vertical projection ($\theta= 0$)

$$v(x) = \int b(x, y)dy$$

Horizontal projection ($\theta= \pi/2$)

$$h(y) = \int b(x, y)dx$$





Projections(2)

Projections

Area $A = \iint b(x, y) dx dy$ $A = \int v(x) dx = \int h(y) dy$

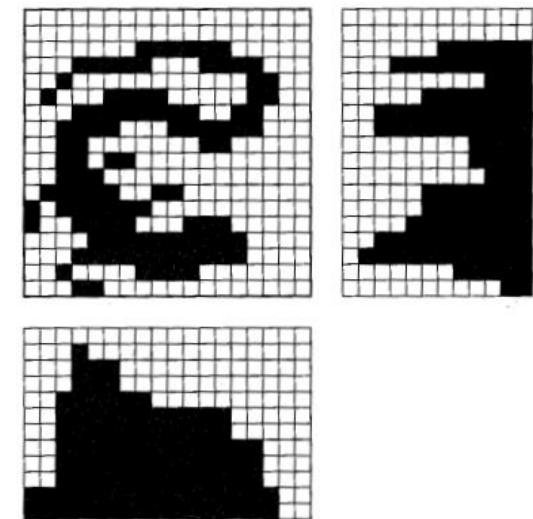
Coordinates of the center of the area

$$\bar{x}A = \iint xb(x, y) dx dy = \int xv(x) dx$$

$$\bar{y}A = \iint yb(x, y) dx dy = \int yh(y) dy$$



The first moments of the projections are equal to the first moments of the original image.





Projections(3)

Projections

Orientation

To compute orientation we need the second moments, too. Two of these are easy to compute from the projections:

$$\iint_I x^2 b(x, y) dx dy = \int x^2 v(x) dx$$

$$\iint_I y^2 b(x, y) dx dy = \int y^2 h(y) dy$$

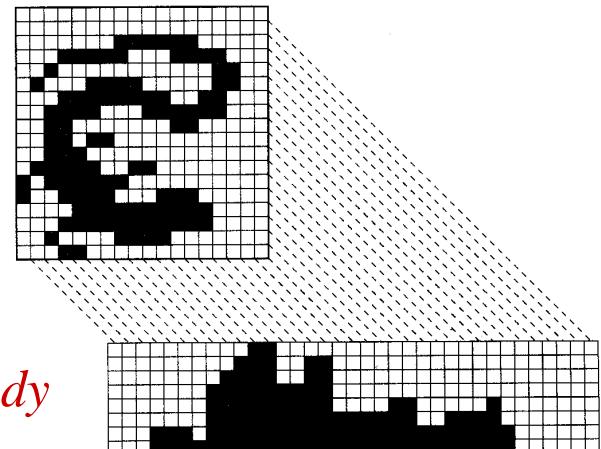
We cannot compute the integral of the product xy from the two projections introduced so far.

We can add the diagonal projection ($\theta = \pi/4$) $d(t) = \int b\left(\frac{1}{\sqrt{2}}(t-s), \frac{1}{\sqrt{2}}(t+s)\right) ds$

Now consider that:

$$\iint_I \frac{1}{2}(x+y)^2 b(x, y) dx dy = \iint_I t^2 b(x, y) ds dt = \int t^2 d(t) dt$$

$$\iint_I \frac{1}{2}(x+y)^2 b(x, y) dx dy = \iint_I \left(\frac{1}{2}x^2 + xy + \frac{1}{2}y^2\right) b(x, y) dx dy$$



So:

$$\iint_I xy b(x, y) dx dy = \int t^2 d(t) dt - \frac{1}{2} \int x^2 v(x) dx - \frac{1}{2} \int y^2 h(y) dy$$



Presentation outline

- Introduction
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image
- Binary Images: geometric properties
- Projections
- **Run-Length Coding**



Run-Length Coding

Run-Length Encoding

This method exploits the fact that along any particular scan line there will usually be long runs of zeros or ones.

Two approaches are commonly used in run-length encoding.

- the start positions and length of runs of 1s for each row are used.
- lengths of runs, starting with the length of the 0 run.

1	1	1	0	0	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Start and length of 1 runs: (1,3) (7,2) (12,4) (17,2) (20,3)
(5,13) (19,4)
(1,3) (17,6)

Length of 0 and 1 runs: 0,3,3,2,3,4,1,2,1,3
4,13,1,4
0,3,13,6



Run-Length coding (2)

Computation of the geometric properties from run-length

Use the convention that r_{ik} is the k-th run of the i-th line and that the first run in each row is a run of zeros (thus all the even runs will correspond to ones in the image). There are m_i runs on the i-th line

Area

$$A = \sum_{i=1}^n \sum_{k=1}^{m_i/2} r_{i,2k}$$

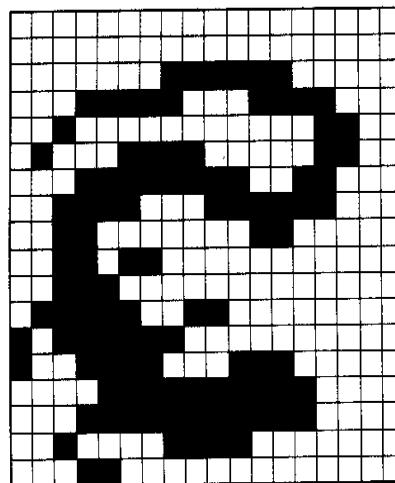
Position

First of all, we obtain the horizontal projection as follows:

$$h_i = \sum_{k=1}^{m_i/2} r_{i,2k}$$

From this we can easily compute the vertical position of the center of area using:

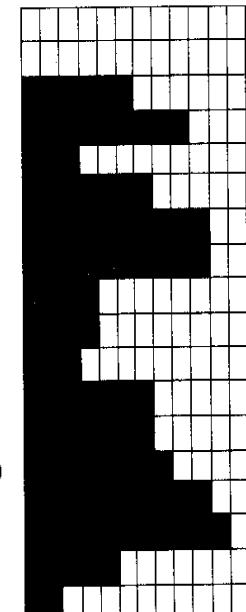
$$\bar{i}A = \sum_{i=1}^n ih_i$$



Run-length code

(0,0)
(0,0)
(8,6)
(4,5) (12,4)
(3,1) (15,2)
(2,1) (6,4) (15,2)
(4,8) (14,2)
(3,4) (10,6)
(3,2) (12,2)
(3,2) (6,2)
(3,3)
(2,5) (9,2)
(1,1) (3,6)
(1,1) (4,4) (11,3)
(5,10)
(4,11)
(3,1) (8,4)
(4,2)

Horizontal projection





Run-Length coding (3)

The **vertical projection** is obtained by adding up all picture cell values in one column of the image.

It is difficult to compute vertical projection directly from the run lengths. Consider instead the **first difference of the vertical projection**:

$$\bar{v}_j = v_j - v_{j-1}, \quad \bar{v}_1 = v_1$$

The **first difference of the vertical projection** can be obtained by projecting not the image data, but the **first horizontal differences of the image data**:

$$\bar{b}_{i,j} = b_{i,j} - b_{i,j-1}$$

The first difference $\bar{b}_{i,j}$ has the advantage over $b_{i,j}$ itself that is nonzero only at the beginning of each run. It equals +1 where the data change from 0 to a 1, and -1 where the data change from a 1 to a 0.



Run-Length coding (4)

										0	
					0	1	1	0		2	
				0	1	1	1	0		3	
			0	1	1	1	1	1	0	5	
	0	1	0	0	1	1	1	1	0	6	
0	1	1	1	1	1	1	1	1	0	8	
0	1	1	1	1	1	1	1	0		8	
0	1	1	1	1	1	1	1	0		7	
	0	1	1	1	0					3	
									h_i		

$r_{i,k}$

8,2,3

7,3,3

6,5,2

3,1,2,5,2

2,8,3

2,8,4

1,7,5

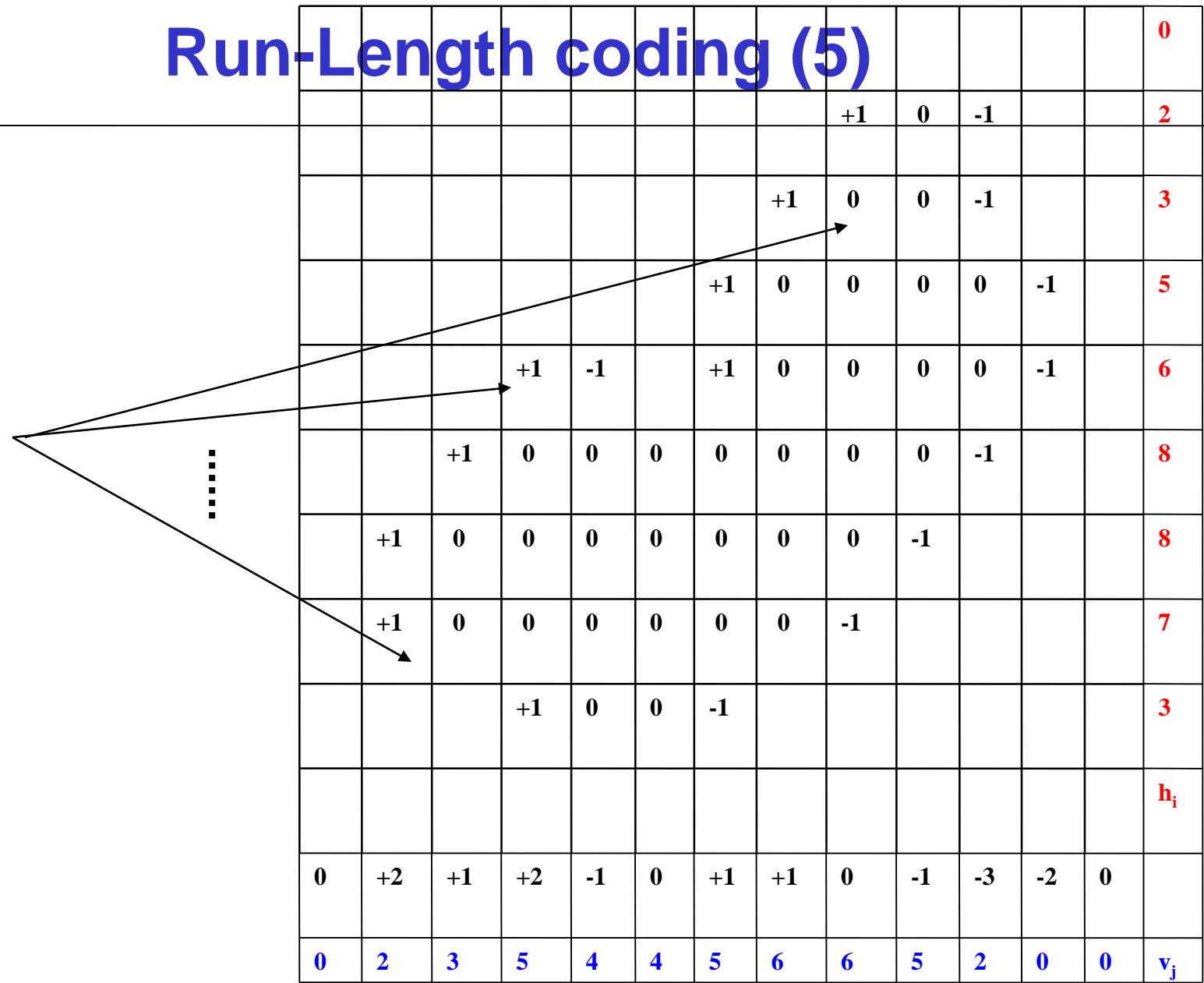
3,3,7

Original
image



Run-Length coding (5)

$$\bar{b}_{i,j}$$





Run-Length coding (6)

We can locate these places by computing the summed run-length code

$$\tilde{r}_{ik} = 1 + \sum_{p=1}^k r_{i,p}$$

recursively: $\tilde{r}_{i,0} = 1$ and $\tilde{r}_{i,j+1} = \tilde{r}_{i,j} + r_{i,j+1}$

Then, for a transition at $j = \tilde{r}_{ik}$ we add $(-1)^{k+1}$ to the accumulated total for the first difference of the vertical projection



Run-Length coding (7)

														0
						0	1	1	0					2
						0	1	1	1	0				3
						0	1	1	1	1	1	0		5
		0	1	0	0	1	1	1	1	1	0			6
0	1	0				8								
0	1	0					8							
0	1	0					7							
		0	1	1	1	0								3
														h_i
0	+2	+1	+2	-1	0	+1	+1	0	-1	-3	-2	0		
0	2	3	5	4	4	5	6	6	5	2	0	0		v_j

$r_{i,k}$	$\tilde{r}_{ik} = 1 + \sum_{p=1}^k r_{i,p}$
8,2,3	1,9,11,14
7,3,3	1,8,11,14
6,5,2	1,7,12,14
3,1,2,5,2	1,4,5,7,12,14
2,8,3	1,3,11,14
1,8,4	1,2,10,14
1,7,5	1,2,9,14
3,3,7	1,4,7,14



Run-Length coding (4)

- From this difference we can compute the vertical projection itself using the simple summation

$$v_j = \sum_{p=1}^j \bar{v}_p$$

- Or recursively: $v_0 = 0$ and $v_{j+1} = v_j + \bar{v}_{j+1}$
- Given the vertical projection, we can easily compute the horizontal projection of the center of area $A\bar{j} = \sum_{j=1}^m jv_j$
- The diagonal projection can be obtained in a way similar to that used to obtain the vertical projection.



References:

- Gerhard X. Ritter, Joseph N. Wilson *Handbook of Computer Vision Algorithms in Image Algebra*
- Linda Shapiro: *Computer Vision*
- Bryan S. Morse, Brigham Young University, Thresholding lecture
- Berthold Klaus, Paul Horn, Robot Vision, The MIT Press, 1986



Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester)

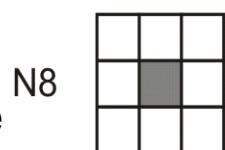
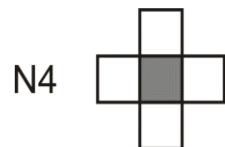
Binary Images: Object Labeling. Contour Tracing (III)



Binary Algorithms - Definitions

1. Neighbors

- two pixels are **4-neighbors** or **d-neighbors** if they share a common side
- two pixels are **i-neighbors** if they share a common corner
- the **neighbour** concept includes both **d-neighbour** and **i-neighbour**
- two pixels are neighbors or **8-neighbors** if they share at least a corner
- the N-neighbour is the neighbour in direction N, where N is a direction code and $0 \leq N \leq 3$ or $0 \leq N \leq 7$



A pixel is said to be 4-connected to its 4-neighbors and 8-connected to its 8-neighbors.

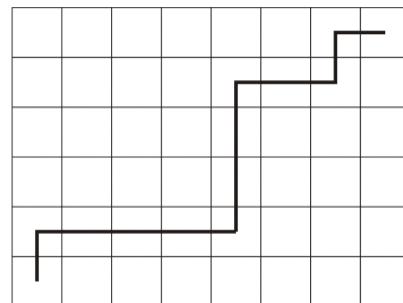
2. Path

Path ($p[i_0, j_0] \Rightarrow p[i_n, j_n]$) := $\{[i_0, j_0], [i_1, j_1], \dots, [i_n, j_n] \mid [i_k, j_k] N_{4/8} [i_{k+1}, j_{k+1}] \forall k = 0 .. n-1\}$

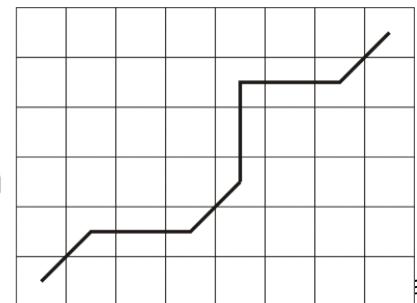
$N4 \Rightarrow$ 4-path or d-path

$N8 \Rightarrow$ 8-path or i-path

4-path



8-path





Binary Algorithms - Definitions

3. Foreground $S := \{ p[i,j] \mid p[i,j] = 1 \}$

4. Connectivity $p \leftrightarrow q$ (connected) if \exists Path $(p \Rightarrow q) \subset S$, $p \in S$, $q \in S$.

5. Connected points $\{p_i \in S, i = 1 \dots n \mid p_k \leftrightarrow p_j, \forall (p_k, p_j) \in S, k, j = 1 \dots n\}$

6. Boundary Boundary $(S) := S' = \{ p \in S \mid \exists q \in N_{4/8}(p), q \in C(S) \}$
 $C(S)$ – complement of S

7. Interior Interior $(S) = S - S'$

8. Connected component

Maximal set of connected points $\{p_i \in S, i = 1 \dots n \mid p_k \leftrightarrow p_j, \forall (p_k, p_j) \in S, k, j = 1 \dots n\}$

9. Background set of all connected components belonging to $C(S)$ that have points on the border of an image. All other components of the image belonging to $C(S)$ are called holes.



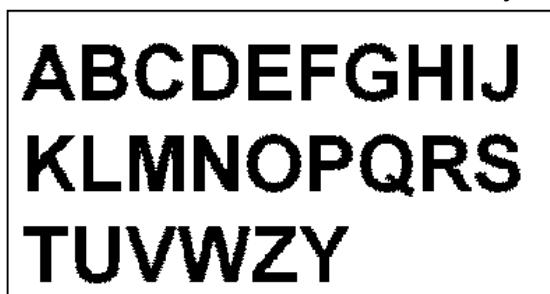
Labeling Connected Components

Connected component : maximal set of connected points

$$\{p_i \in S, i = 1 \dots n \mid p_k \leftrightarrow p_j, \forall (p_k, p_j) \in S, k, j = 1 \dots n\}$$

One way to label the objects in a discrete binary image is to choose a point where $b_{ij} = 1$ and assign a label to this point and its neighbors. Next, label all the neighbors of these neighbors, and so on.

- When this recursive procedure terminates, one component will have been labeled completely, and we can continue by choosing another start point.
- To find a new start point, we can simply scan through the image in a systematic way, starting a labeling operation whenever an unlabeled point is found where $b_{ij} = 1$.



⇒
Labeling





Sequential Labeling

Iterative Algorithm (Haralick 1981)

- no auxiliary storage to produce the labeled image from the binary image.
- useful in environments whose storage is severely limited.

1. initialization step
2. repeat

top-down & left-right label propagation

bottom-up & right-left label propagation

until no changes occur

procedure Iterate;

“Initialization of each 1-pixel to a unique label”

for L:=1 to NLINES **do**

for P:=1 to NCOLUMNS **do**

if I(L,P) =1

then LABEL(L,P):=NEWLABEL()

else LABEL(L,P):=0

end for

end for;



Sequential Labeling

a	b	c
d	e	

	e	d
c	b	a

		e

“Top-down passes;

Bottom up passes;

8-connected neighborhood”

“procedure Iterate – page 2”

“Iteration of top-down followed by bottom-up passes”

repeat

“Top-down passes”

CHANGE:=false;

for L:=1 to NLINES **do**

for P:=1 to NCOLUMNS **do**

if LABEL(L,P)<>0 **then**

begin

 M:=MIN(LABELS(NEIGHBORS(L,P)U(L,P)));

if M<> LABEL(L,P)

then CHANGE:=true;

 LABEL(L,P):=M

end

end for

end for;

IMAGE PROCESSING



Sequential Labeling

“procedure Iterate – page 3”

“Bottom-up pass”

```
for L:= NLINES to 1 by -1 do
    for P:= NCOLUMNS to 1 by -1 do
        if LABEL(L,P)<>0 then
            begin
                M:=MIN(LABELS(NEIGHBORS(L,P)U(L,P)));
                if M<> LABEL(L,P)
                then CHANGE:=true;
                LABEL(L,P):=M
            end
        end for
    end for;

until CHANGE:=false

end Iterate
```



Sequential Labeling

Example (N4)

	1	1		1	1	
	1	1		1	1	
	1	1	1	1	1	

1. Initial image

	1	2		3	4	
	5	6		7	8	
	9	10	11	12	13	

2. Initialization

	1	1		3	3	
	1	1		3	3	
	1	1	1	1	1	

3. Top-down & left-right
label propagation

	1	1		1	1	
	1	1		1	1	
	1	1	1	1	1	

4. Bottom-up & right-left
label propagation



Classical Algorithm

- Based on the classical connected components algorithm for graphs.
- 2 passes through the image but requires a large global table for recording equivalences.

1. First pass: performs label propagation, much as described above.

- Whenever a situation arises in which two different labels can propagate to the same pixel, the smaller label propagates and each such equivalence found is entered in an equivalence table (e.g. (1,2) → EqTable).
- Each entry in the equivalence table consists of an ordered pair, the values of its components being the labels found to be equivalent.
- After the first pass, the equivalence classes are found.
- Each equivalence class is assigned a unique label, usually the minimum (or oldest) label in the class.

1	1
1	1
1	1
1	1
1	1
2	2
2	2
2	2
2	2
2	2
X	

2. A second pass through the image performs a translation, assigning to each pixel the label of the equivalence class of its 1-st pass label.



Classical Algorithm

procedure Classical

“Initialize global equivalence table”

EQTABLE:=CREATE();

“Top-down pass 1”

for L:= 1 to NLINES **do**

 “Initialize all labels on line L to zero”

for P:= 1 to NCOLUMNS **do**

 LABEL(L,P):=0

end for

 “Process the line”

for P:=1 to NCOLUMNS **do**

if I(L,P):= 1 **then**

begin

 A:= NEIGHBORS((L,P));

if ISEMPY(A)

then M:=NEWLABEL()

else M:= MIN(LABELS(A));

 LABEL(L,P):=M;

for X in LABELS(A) and X<>M

 ADD(X, M, EQTABLE)

end for;

end

end for

end for;



Classical Algorithm

“Find the equivalence classes”

```
EQCLASSES:=Resolve(EQTABLE);
```

“Find the equivalence label of an equivalence class“

```
for E in EQCLASSES
    EQLABEL(E):= min(LABELS(E))
end for;
```

“Top-down pass 2”

```
for L:= 1 to NLines do
    for P:= 1 to NCOLUMNS do
        if I(L,P) = 1
            then LABEL(L,P):=EQLABEL(CLASS(LABEL(L,P)))
    end for
end for
end Classical
```

- **RESOLVE** - algorithm for finding the connected components of the graph structure, defined by the set of equivalences (**EQTABLE**) defined in pass 1.
- The main problem with the classical algorithm is the global equivalence table (large images with many regions, the equivalence table can become very large)



Classical Algorithm

Example (N4)

1					1	1
		1	1			1
		1				1
		1				1
1	1	1				1
	1	1		1		1
	1	1	1	1		1
				1	1	1

1. Initial image

1					2	2
		3	3			2
		3				2
		3				2
4	4	3				2
	4	3		5		2
	4	3	3	3		2
				3	3	2

2. Top down (pass 1)

EQCLASSES:

1: {4, 3, 5, 2}

2: {6,8,9, ...}

....

n: {....}

EQLABEL:

1, 2

2, 6

...

n, x



Classical Algorithm (improvement)

A Space-Efficient Two-Pass Algorithm That Uses a Local Equivalence Table

⇒ use of a small local equivalence table that stores only the equivalences detected from the current and previous lines

Maximum number of equivalences = number of pixels / line.

1. First pass:

the equivalences from one line are used in the propagation step to the next line.

1. Second pass is required:

The new equivalence classes and labels finding followed by assigning the final labels.

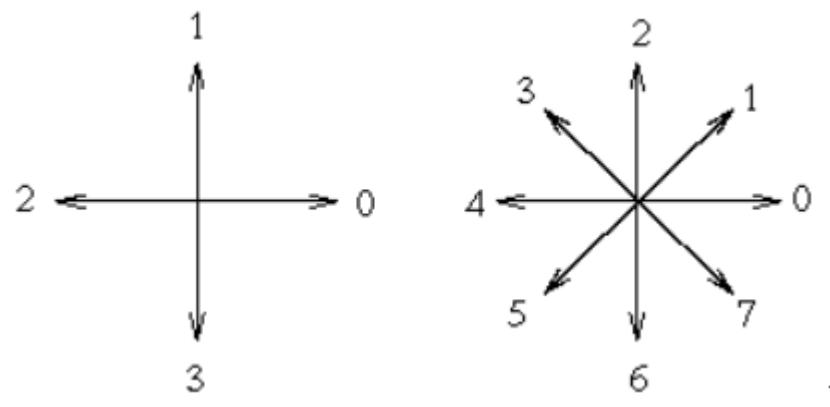


Contour Tracing

Boundary/contour

$$\text{Contour}(R) = \{ p \in R \mid \exists q \in N_4/8(p), q \in C(R) \}$$

- The **contour** or **i-contour** of R (where R is a connected set of pixels) is defined as the set of all pixels in R which have at least one **d-neighbor** not in R .
- The **d-contour** of R is the set of all pixels in R , which have at least one neighbor not in R .
- **N-neighbor (chain-code / direction codes):** c
(numerical operations on c are assumed to be modulo 4 or 8)



4-neighbour

8-neighbour



Contour Tracing

Single contour tracing

- The algorithm can be described in terms of an observer who walks counterclockwise along pixels belonging to the set and selects the rightmost pixel available.
- The tracing terminates when the current pixel is the same as the initial pixel.
- The TRACER algorithm must be applied once for each hole of a region, in addition to one application for the external contour.
- Therefore, it must be combined with a search algorithm for locating holes in the interior of the region.
- Description of the contours (output): { A(x_0, y_0, c_0), C_i(x_i, y_i, c_i), i=1 ... n }

TRACER Procedure

Notations:

- A: the starting point of the contour of the set R (can be found in a number of ways, including a top-to-bottom, left-to-right scan);
C: the current point whose neighborhood is examined;
S : the search direction in the terms of the direction codes;
first: is a flag that is true only when the tracing starts;
found: is a flag that is true when a next point on the contour is found;

```
if ((A:=NEXT_START_ELEMENT())!=NULL)
{Q:=CREATE_LIST();
 TRACER(A,Q);
}
```



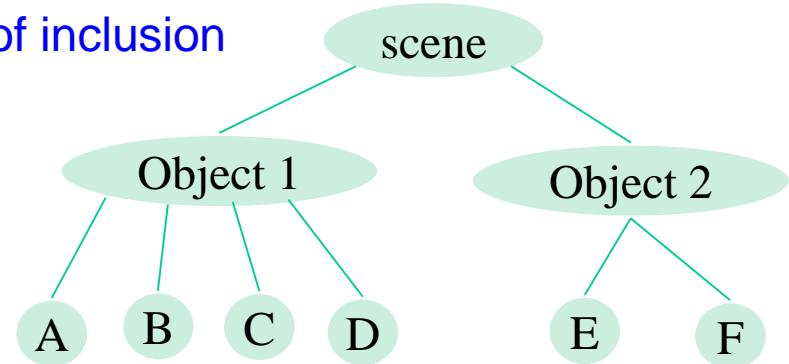
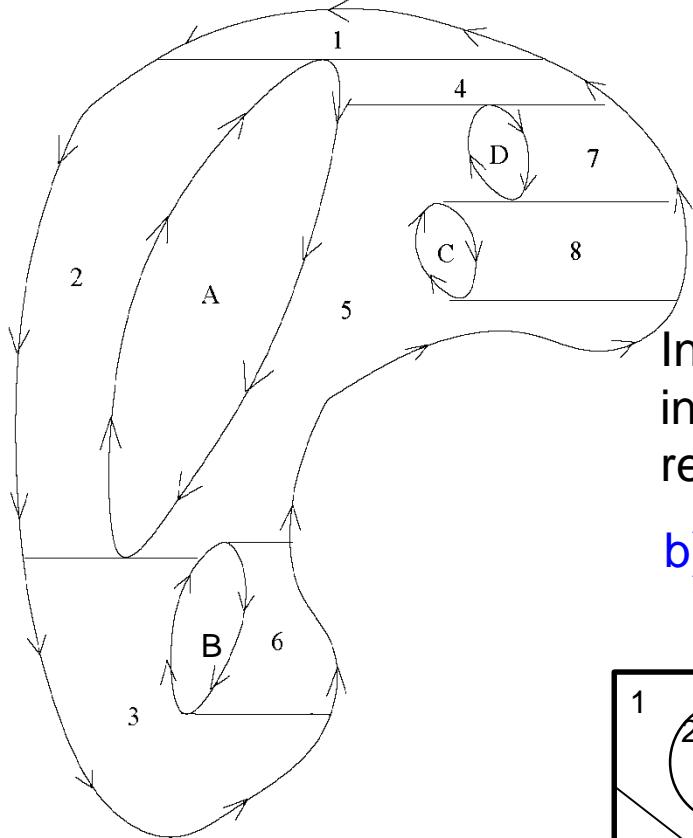
Contour Tracing

```
procedure TRACER(A,Q)
begin
    first:=TRUE;
    C:=A;
    S:=6;
    while((C!=A) or (first=TRUE))
    {
        found:=FALSE;
        count:=0;
        while((found=FALSE) and (count=<3))
        {
            if( B, the (S-1)-neighbor of C, is in R) then
            { APPEND((C,S-1),Q), C:=B, S:=S-2, found:=TRUE; }
            else if (B, the S-neighbor of C, is in R) then
            { APPEND((C,S),Q), C:=B, found:=TRUE; }
            else if (B, the (S+1)-neighbor of C, is in R) then
            {APPEND((C,S+1),Q), C:=B, found:=TRUE; }
            else
            {S:=S+2, count:=count+1; }
            endif
        endif
    }
    first:=FALSE;
}
end
```



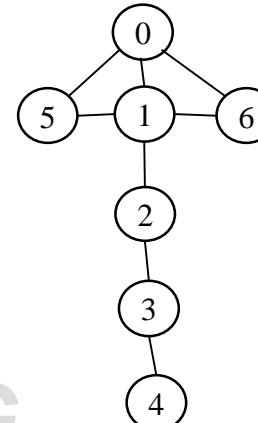
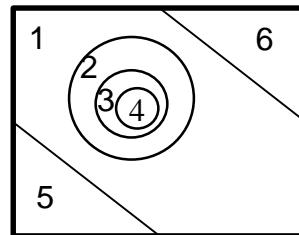
Traversal of all the contours of an image

Structural description of the scene: a) Tree of inclusion



In this tree the nodes represent the connected regions in the image and the edges represent the inclusion relations. The root models the whole scene.

b) Adjacency graph



The nodes represent the connected regions in the image and the edges link adjacent regions.



Traversal of all the contours of an image

Traversal of All the Contours of a Region

⇒ closed i-paths and follows external contours counterclockwise (TRACER()) and contours of hole clockwise (TRACER1()).

Changes:

- When a pixel is marked as the current point, its value is incremented by 1. Then, at the end of the tracing, pixels of the contour will have values 2 or greater. These values are used when the interior is searched for holes.
- After the external contour has been found and placed in the queue Q, we start examining the contents of the latter. If we find a point located on a downward arc, we start a search to the right. Such pixels can be characterized easily by the requirement that the previous element of the chain code must have values 4 to 7 while the next element should be in the range 5 to 7.
- While scanning along the horizontal direction one must search for either the start of a hole or the other side of the outside contour.
- The TRACER1() procedure is called when an unmarked start of a hole point is found.
- The extracted contours of the holes are inserted into the Q list.
- The content of the list is examined until its end.



Traversal of all the contours of an image

MULTI_TRACE Procedure

Notations:

- P: current pixel with x, y coordinate;
- c: direction code; value 8 is used to specify the beginning of a new contour;
- c0: initial direction code;
- Q: $\{(P, c)\}$

```
If ((A:=NEXT_START_ELEMENT())!=NULL)
{
    Q:=CREATE_LIST();
    MULTI_TRACE(A,Q);
}
```



Traversal of all the contours of an image

```
procedure MULTI_TRACE(A,Q)
begin
```

```
    TRACER(A,Q);
    while (Q !=NULL)
    {
```

```
        (P,c):=Remove(Q);
```

```
        if (c=8)
```

```
        {
```

```
            c0:=c;
```

```
            (P,c):=Remove(Q);
```

```
        }
```

```
        if ((c0 ∈{4 : 7}) and (c∈{5 : 7}))
```

```
        {
```

Starting from P search in the x-direction and examine triplets of successive pixels, A, B, C.

```
        if((A!=0) and (B=1) and (C=0))
```

```
        {
```

```
            TRACER1(B,Q);
```

```
            goto LABEL1 ;
```

```
        }
```

```
        if ((A=1) and (B=2) and (C=0))
```

```
            goto LABEL1;
```

```
}
```

```
LABEL1: c0=c;
```

```
end
```

IMAGE PROCESSING



Polygonal Approximation

Polygonal approximation of contours

Curve C: $f(x,y)=0 \Rightarrow$ polygon that closely approximates C with an error smaller than ε and having a number of vertices as small as possible:



- Any polygonal fitting algorithm requires that the data points be subdivided into groups, each one of them to be approximated by a side of the polygon.
- The first simplification of the polygon fit problem is to draw a line between the endpoints of each group rather than search for the optimal solution.
- If the approximation error is too big the group could be split in two and so on until the error becomes acceptable.
- Let Q a contour consisting of $P_i (x_i, y_i)$ where $i=1,2,\dots,n$, and ε the error threshold.



Polygonal Approximation

Procedure POLIGONAL_APROX(Q)

begin

A:=Create_List();

B:=Create_List();

i=Index_of_first_point(Q);

j=Index_of_the_most_far_point(Q);

Insert(j,A); Insert(j,B);

Insert(i,A);

while((A!=NULL)

{

Let *k* and *l* the indexes of the last elements of the lists A and B;

Let $P_k P_l$ the segment generated by these two points;

Let *m* the index of the most far point to $P_k P_l$ segment among the contour points starting with P_k and ending with P_l , when the contour is scanned in counterclockwise direction.

if (d=Distance($P_k P_l$, P_m) > ε)

then Insert(m, A)

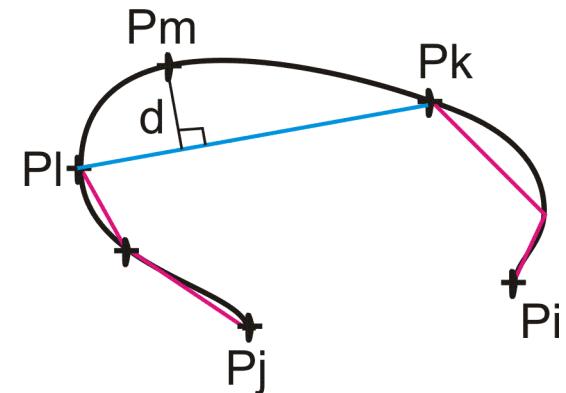
else { Delete(k, A)

Insert(k, B); }

}

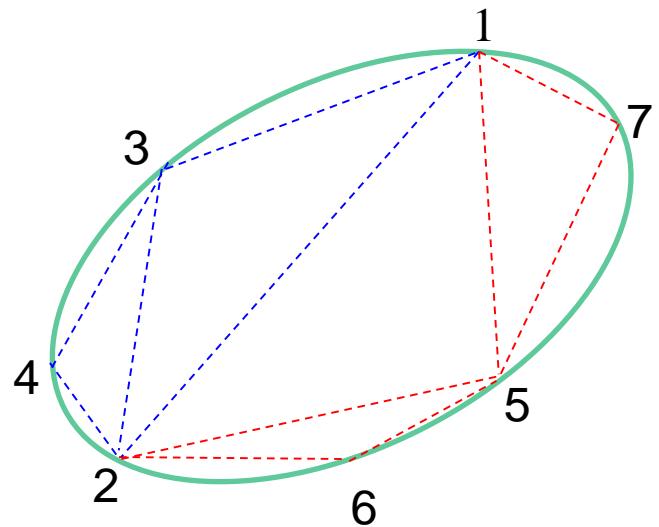
end

$$\text{distance}(P_1, P_2, (x_0, y_0)) = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}.$$





Polygonal Approximation – Example



A	B
2	2
1	
3	
4	
2	2
1	4
3	
2	2
1	4
	3
2	2
	4
	3
	1
2	2
5	4
	3
	1

A	B
2	2
5	4
7	3
	1
2	2
5	4
	3
	1
	7
2	2
	4
	3
	1
	7
	5
2	2
6	4
	3
	1
	7
	5

A	B
2	2
	4
	3
	1
	7
	5
	6
	2
	4
	3
	1
	7
	5
	6
	2



Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester)

Binary Images: Morphological Image Processing (V)



Morphologic Image Processing

1. Introduction

- **Morphology:** denotes a branch of biology that deals with the form and structure of animals and plants
- **Mathematical morphology:**
 - **a tool for extracting image components** that are useful **in the representation and description of region shape**, such as boundaries, skeletons, and the convex hulls
 - **for pre or post-processing**, such as morphological filtering, thinning, and pruning
- **Set theory:** is the language of mathematical morphology

2. Some Basic Concepts from Set Theory

- Let A be a **set** in Z^2 . If $a=(a_1,a_2)$ is an element of A , then we write

$$a \in A$$

- Similarly, if a is not an element of A , we write

$$a \notin A$$

-The set with no elements is called the **null** or empty set and is denoted by the symbol \emptyset .

-A set is specified by the contents of two braces: $\{ \dots \}$. The **elements of the sets** with which we are concerned in this chapter **are the coordinates of pixels** representing objects or other features of interest in an image.



Morphologic Image Processing

Some Basic Concepts from Set Theory (2)

- If every element of a set A is also an element of another set B , then A is said to be a **subset** of B , denoted as: $A \subseteq B$
- The **union** of two sets A and B , is the set of all elements belonging to either A , B , or both: $C = A \cup B$
- Similarly, the **intersection** of two sets A and B , is the set off all elements belonging to both A and B : $D = A \cap B$
- Two sets A and B are said to be **disjoint** or **mutually exclusive** if they have no common elements: $A \cap B = \emptyset$
- The **complement** of a set A is the set of elements not contained in A :
$$A^C = \{w \mid w \notin A\}$$
- The **difference** of two sets A and B , denoted $A - B$, is defined as:
$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^C$$
- The **reflection** of set B , is defined as:
$$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$$

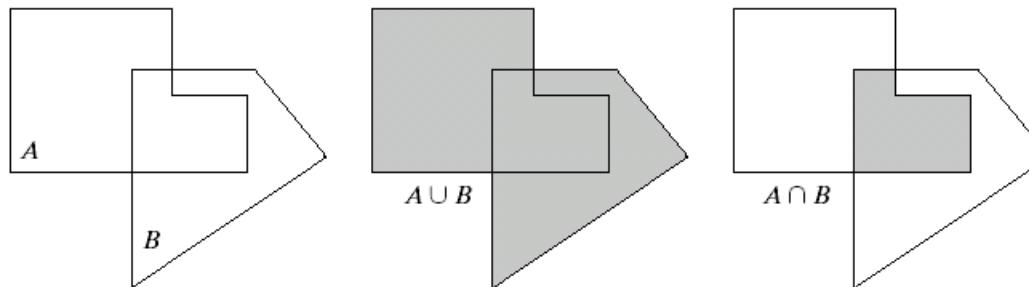
- The **translation** of set A by point $z = (z_1, z_2)$, is defined as:

$$(A)_z = \{c \mid c = a + z, \text{ for } a \in A\}$$



Morphologic Image Processing

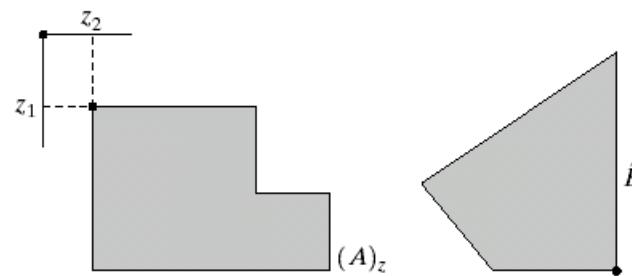
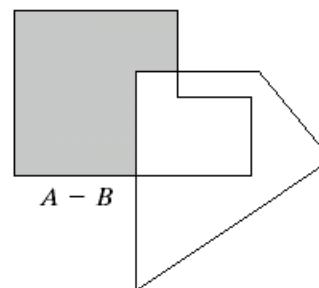
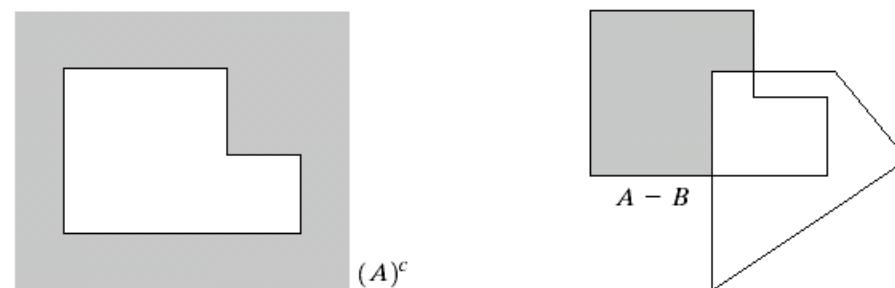
Some Basic Concepts from Set Theory (3)



a	b	c
d	e	

FIGURE 9.1

- (a) Two sets A and B .
- (b) The union of A and B .
- (c) The intersection of A and B .
- (d) The complement of A .
- (e) The difference between A and B .



a	b
---	---

FIGURE 9.2

- (a) Translation of A by z .
- (b) Reflection of B . The sets A and B are from Fig. 9.1.

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods

Technical University of Cluj Napoca

Computer Science Department



Morphologic Image Processing

3. Logic operations involving Binary Images

- The principal logic operations used in image processing are AND, OR, and NOT (COMPLEMENT). These operations can be combined to form any other logic operation.
- Logic operations are performed on a pixel by pixel basis between corresponding pixels of two or more images (except NOT, which operates on the pixels of a single image).
- The logic operations have a one-to-one correspondence with the set operations, with the limitation that logic operations are restricted to binary variables, which is not the case in general for set operations.



Morphologic Image Processing

Logic operations involving Binary Images(2)

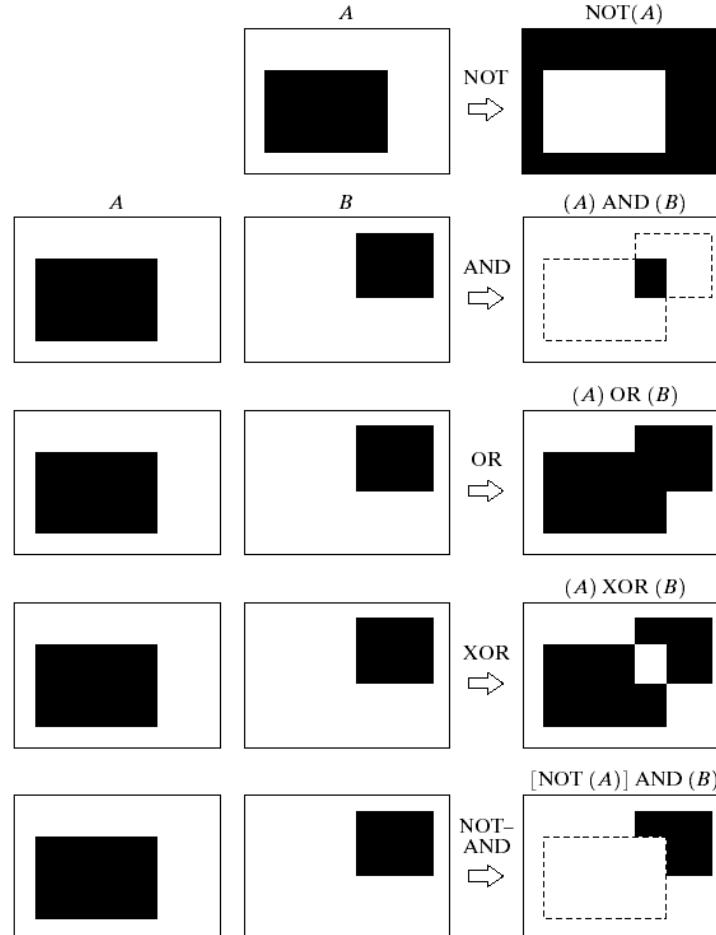


FIGURE 9.3 Some logic operations between binary images. Black represents binary 1s and white binary 0s in this example.

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

4. Dilation and Erosion

Dilation and erosion are two primitive morphological operations.

4.1 Dilation

a) With A and B as sets in Z^2 , the *dilation* of A by B , denoted $A \oplus B$, is defined as:

$$A \oplus B = \left\{ z \mid (\hat{B})_z \cap A \neq \emptyset \right\}$$

This equation is based on obtaining the reflexion of B about its origin and shifting this reflexion by z . The dilation of A by B then is the set off all displacements z , such that $(\hat{B})_z$ and A overlap by at least one element. Based on this interpretation we can write:

$$A \oplus B = \left\{ z \mid [(\hat{B})_z \cap A] \subseteq A \right\}$$

Set B is commonly referred to as **the structuring element**.



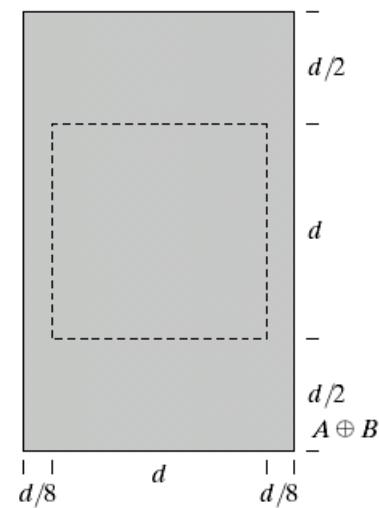
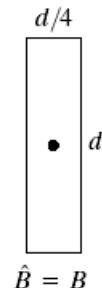
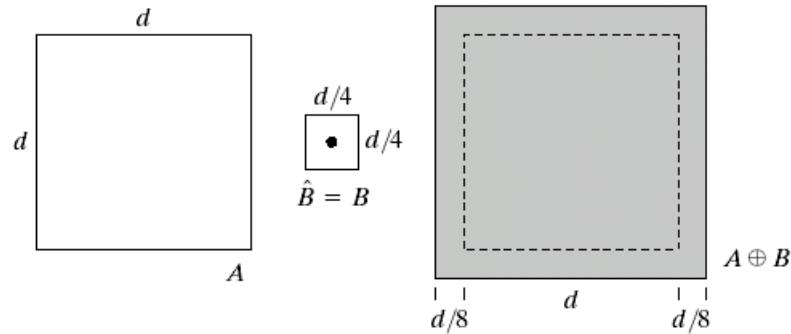
Morphologic Image Processing

Dilation (2)

a b c
d e

FIGURE 9.4

- (a) Set A .
- (b) Square structuring element (dot is the center).
- (c) Dilation of A by B , shown shaded.
- (d) Elongated structuring element.
- (e) Dilation of A using this element.



From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

Dilation (3)

b). Dilation is the morphological transformation that combines two sets using vector addition of set elements. If A and B are sets in Z^2 with elements a and b respectively, $a=(a_1, a_2, \dots, a_N)$ and $b=(b_1, b_2, \dots, b_N)$ being element coordinates, then the dilation of A by B is the set of all possible vector sums of pairs of elements, one coming from A and one coming from B.

$$A \oplus B = \{z \in Z^2 \mid z = a + b, \text{ for some } a \in A \text{ and } b \in B\}$$

Because addition is commutative, dilation is commutative $A \oplus B = B \oplus A$
In practice the sets A and B are not thought of symmetrically.

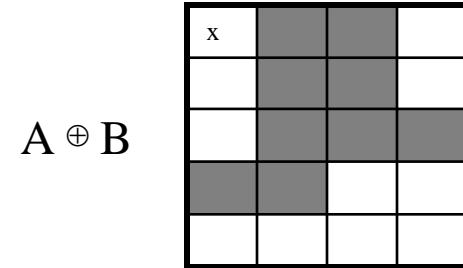
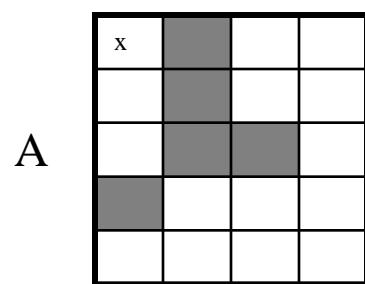


Morphologic Image Processing

Dilation (4)

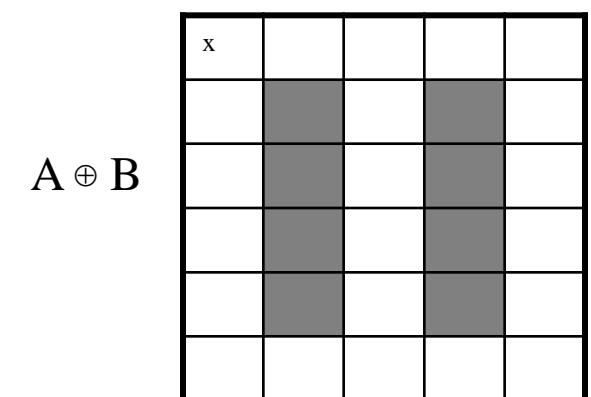
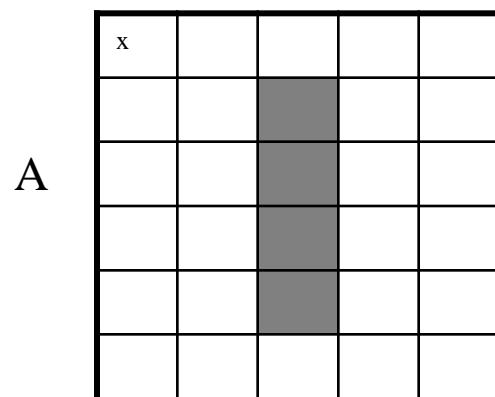
$$A = \{(0,1), (1,1), (2,1), (2,2), (3,0)\}; B = \{(0,0), (0,1)\}$$

$$A \oplus B = \{(0,1), (1,1), (2,1), (2,2), (3,0), (0,2), (1,2), (2,2), (2,3), (3,1)\}$$



$$A = \{(1,2), (2,2), (3,2), (4,2)\}; B = \{(0,-1), (0,1)\}$$

$$A \oplus B = \{(1,1), (2,1), (3,1), (4,1), (1,3), (2,3), (3,3), (4,3)\}$$





Morphologic Image Processing

Dilation (5)

- c) The dilation operation can be represented as a union of translates of the structuring element.

$$A \oplus B = \bigcup_{a \in A} B_a$$

If B has a center on the origin, then the dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A .

The dilation process is performed by laying the structuring element on the image and sliding it across the image in a manner similar to convolution. The difference is in the operation performed. It is best described in a sequence of steps:

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a '1' in the image, perform the OR logic operation on all pixels within the structuring element.



Morphologic Image Processing

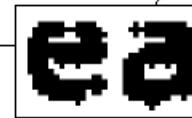
Dilation (6)

One of the simplest applications of dilation is for bridging gaps.

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



a
b
c

FIGURE 9.5
(a) Sample text of poor resolution with broken characters (magnified view).
(b) Structuring element.
(c) Dilation of (a) by (b). Broken segments were joined.

0	1	0
1	1	1
0	1	0

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



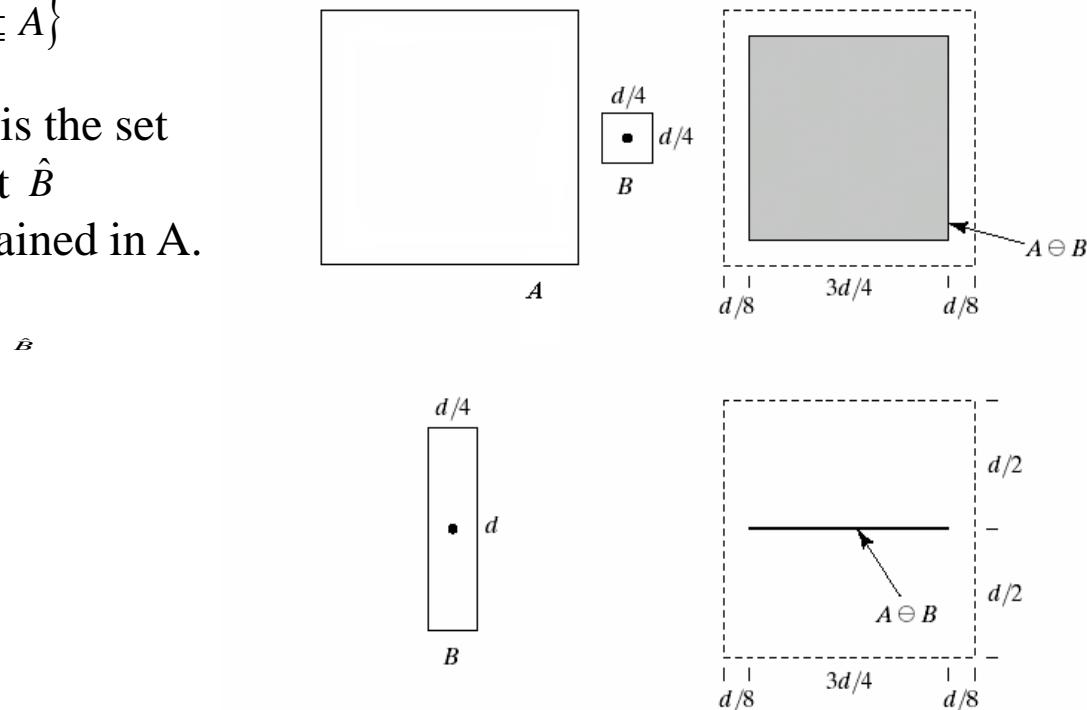
Morphologic Image Processing

4.2 Erosion

a) With A and B as sets in \mathbb{Z}^2 , the *erosion* of A by B , denoted $A \ominus B$, is defined as:

$$A \ominus B = \{z \mid (\hat{B})_z \subseteq A\}$$

The erosion of A by B is the set of all points z such that \hat{B} translated by z , is contained in A .



From “Digital Image Processing”
by R. C. Gonzales and R. E. Woods

a	b	c
d	e	

FIGURE 9.6 (a) Set A . (b) Square structuring element. (c) Erosion of A by B , shown shaded. (d) Elongated structuring element. (e) Erosion of A using this element.



Morphologic Image Processing

Erosion (2)

b) Erosion is the morphological transformation that combines two sets by using containment to its basis set. If A and B are sets in Z^2 , then the erosion of A by B is the set of all elements x for which $x+b \in A$ for every $b \in B$.

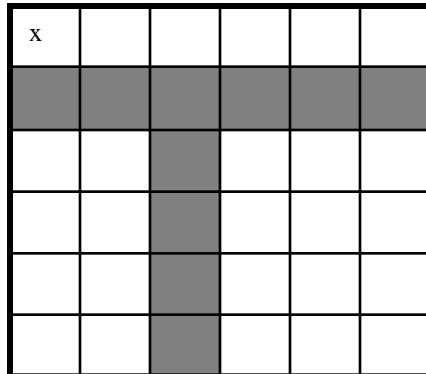
$$A \Theta B = \{x \in Z^2 \mid x+b \in A, \text{for_every_} b \in B\}$$

$$A = \{(1,0), (1,1), (1,2), (1,3), (1,4), (1,5), (2,1), (3,1), (4,1), (5,1)\}$$

$$B = \{(0,0), (0,1)\}$$

$$A \Theta B = \{(1,0), (1,1), (1,2), (1,3), (1,4)\}$$

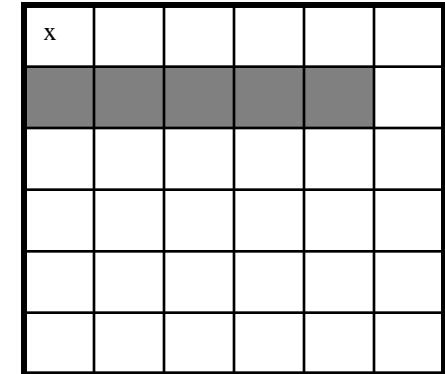
A



B



$A \Theta B$





Morphologic Image Processing

Erosion (3)

c) Erosion can be viewed as a morphological transformation that combines two sets by using vector subtraction of set elements. Expressed as a difference of elements a and b, erosion becomes

$$A \ominus B = \{x \in Z^2 / \text{for every } b \in B, \text{ there exists an } a \in A \text{ such that } x = a - b\}$$

The erosion process is similar to dilation, but we turn pixels to '0', not '1'. As before, slide the structuring element across the image and then follow these steps:

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a '1' in the image, and any of the '1' pixels in the structuring element extend beyond the object ('1' pixels) in the image, then change the '1' pixel in the image to a '0'.

\in



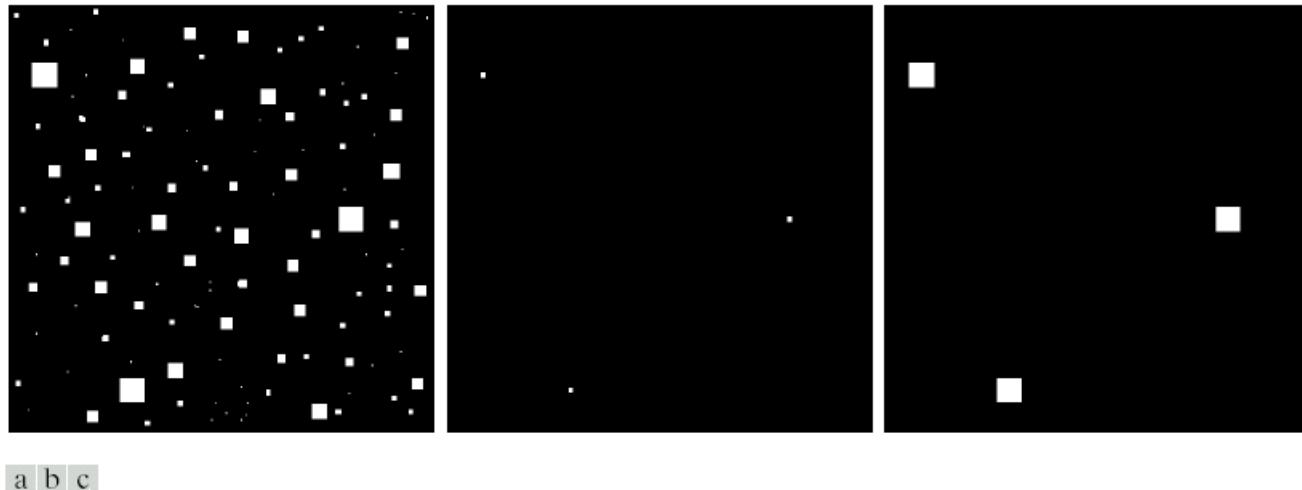
Morphologic Image Processing

Erosion (4)

One of the simplest uses of erosion is for eliminating irrelevant detail from a binary image. It is used in conjunction with dilation.

Dilation and erosion are duals of each other with respect to set complementation and reflection.

$$(A \Theta B)^c = A^c \oplus \hat{B}$$



a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

5. Opening and Closing

As we have seen, dilation expands an image and erosion shrinks it.

The ***opening*** of set A by structuring element B , denoted $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B$$

Thus, the opening A by B is the erosion of A by B , followed by a dilation of the result by B . Opening generally **smoothes the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions**.

The ***closing*** of set A by structuring element B , denoted $A \bullet B$ is defined as

$$A \bullet B = (A \oplus B) \ominus B$$

Thus, the closing of set A by B is simply the dilation of A by B , followed by the erosion of the result by B . Closing also tends to **smooth sections of contours** but, as opposed to opening, it generally **fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour**.



Morphologic Image Processing

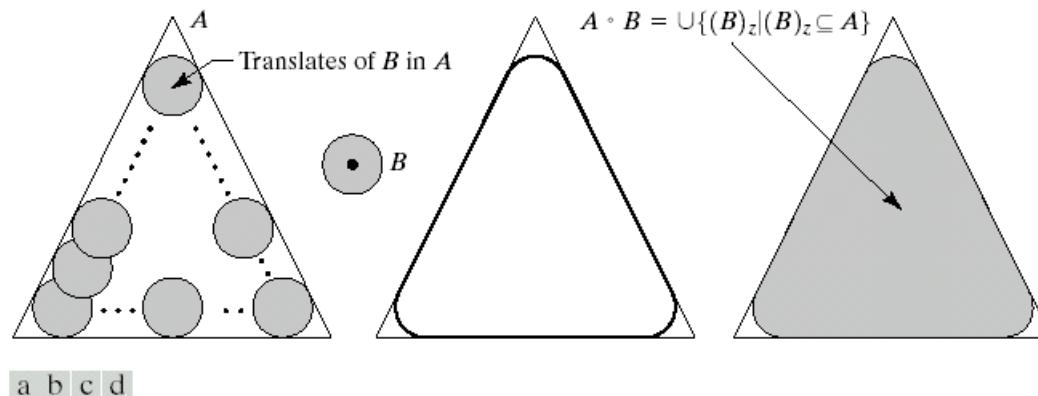


FIGURE 9.8 (a) Structuring element B “rolling” along the inner boundary of A (the dot indicates the origin of B). (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded).

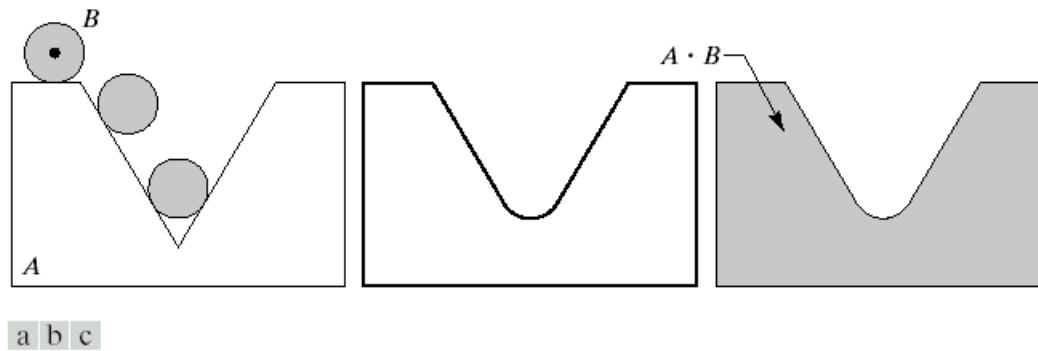


FIGURE 9.9 (a) Structuring element B “rolling” on the outer boundary of set A . (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

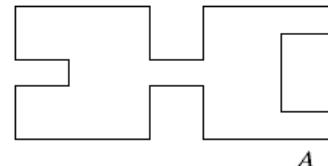
From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



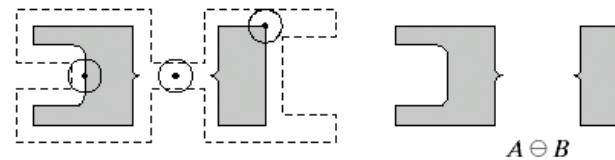
Morphologic Image Processing

a
b c
d e
f g
h i

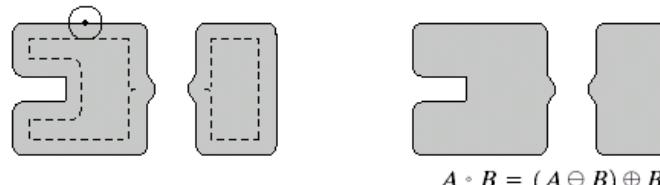
FIGURE 9.10
Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The dark dot is the center of the structuring element.



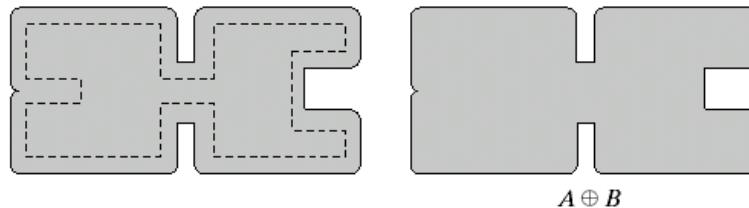
A



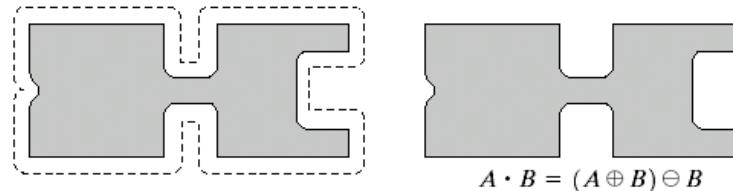
$A \ominus B$



$A \circ B = (A \ominus B) \oplus B$



$A \oplus B$



$A \bullet B = (A \oplus B) \ominus B$

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

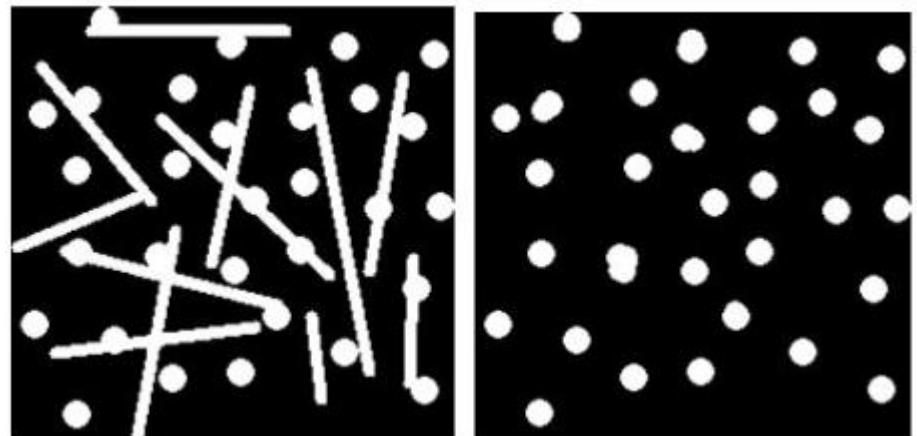
Opening:

- Extracting the horizontal and vertical lines
- The results of an opening with a 3×19 vertically and 19×3 horizontally oriented structuring element



Opening:

- Separate out the circles from the lines so that they can be counted.
- Opening with a disk shaped structuring element 11 pixels in diameter





Morphologic Image Processing

6. Hit-or-Miss Transform

The *hit-or-miss transform* is a natural operation to **select out pixels that have certain geometric properties**, such as **corner points, isolated points, or border points**, and that performs **template matching, thinning, thickening, and centering**.

This transform is accomplished by using **intersections of erosions**.

Let J and K be two structuring elements that satisfy $J \cap K = \emptyset$. Let be $B = \{J, K\}$. The hit-or-miss transformation of a set A by B is denoted by
$$A \circledast B = (A \ominus J) \cap (A^C \ominus K).$$

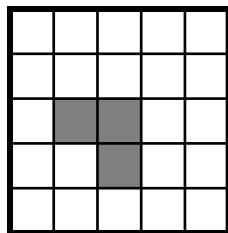
Ex: The hit-or-miss transform used to find **the upper right-hand corner points** of a set A.

The J structuring element locates all pixels of A that have south and est neighbors that are also parts of A. The K structuring element locates all pixels of A^C that have south and est neighbors in A^C . Notice that J and K are displaced from one another. K is J translated by one pixel to the northeast. The pixels that K locates can be thought of as all pixels of A^C that are candidates for being an exterior border pixel to a corner pixel of A.

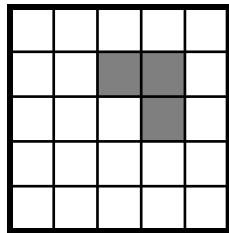


Morphologic Image Processing

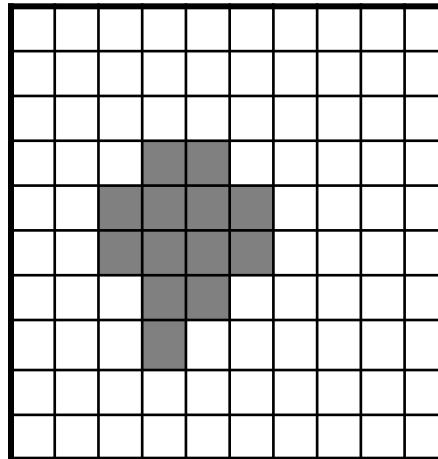
J



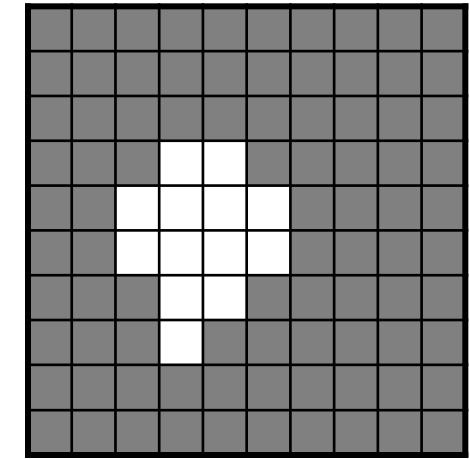
K



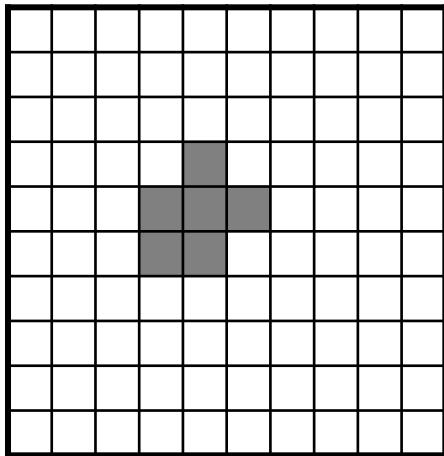
A



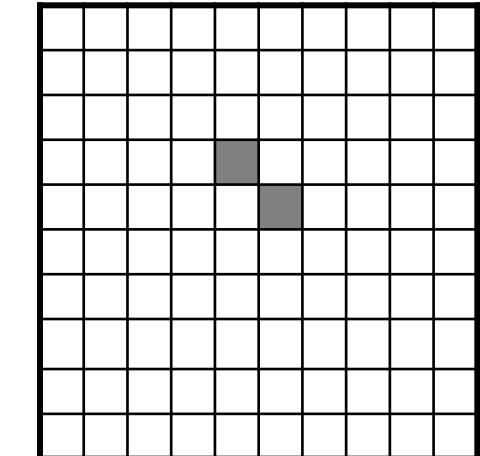
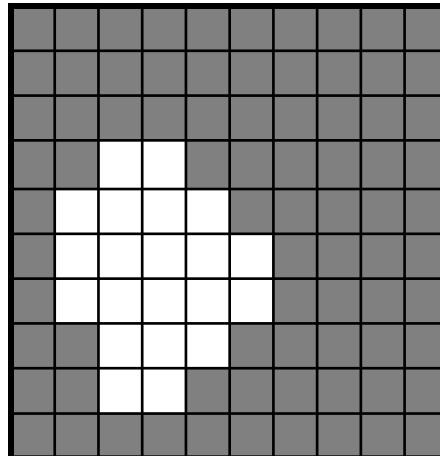
A^C



$A \ominus J$



$A^C \ominus K$



$$A \oplus B = (A \ominus J) \cap (A^C \ominus K)$$



Morphologic Image Processing

7. Some Basic Morphological Algorithms

7.1 Boundary extraction

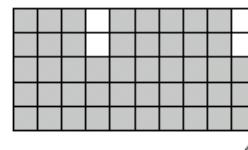
The boundary of a set A , denoted by $\beta(A)$, can be obtained by first eroding A by B and then performing the set differences between A and its erosion. That is,

$$\beta(A) = A - (A \ominus B)$$

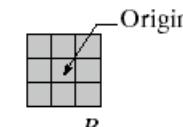
where B is a suitable structuring element.

a b
c d

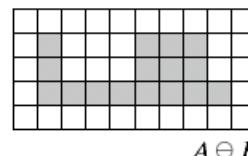
FIGURE 9.13 (a) Set A . (b) Structuring element B . (c) A eroded by B . (d) Boundary, given by the set difference between A and its erosion.



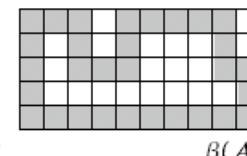
A



B



$A \ominus B$



$\beta(A)$

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

7.2 Region Filling

Next it is presented a simple algorithm for region filling based on set dilations, complementation, and intersections.

Beginning with a point p inside the boundary, the objective is to fill the entire region with 1's. If we adopt the convention that all non-boundary (background) points are labelled 0, then we assign a value of 1 to p to begin. The following procedure fills the region with 1's:

$$X_k = (X_{k-1} \oplus B) \cap A^C, k = 1, 2, 3, \dots$$

where $X_0 = p$, and B is the symmetric structuring element. The algorithm terminates at iteration step k if $X_k = X_{k-1}$. The set union of X_k and A contains the filled set and its boundary.



Morphologic Image Processing

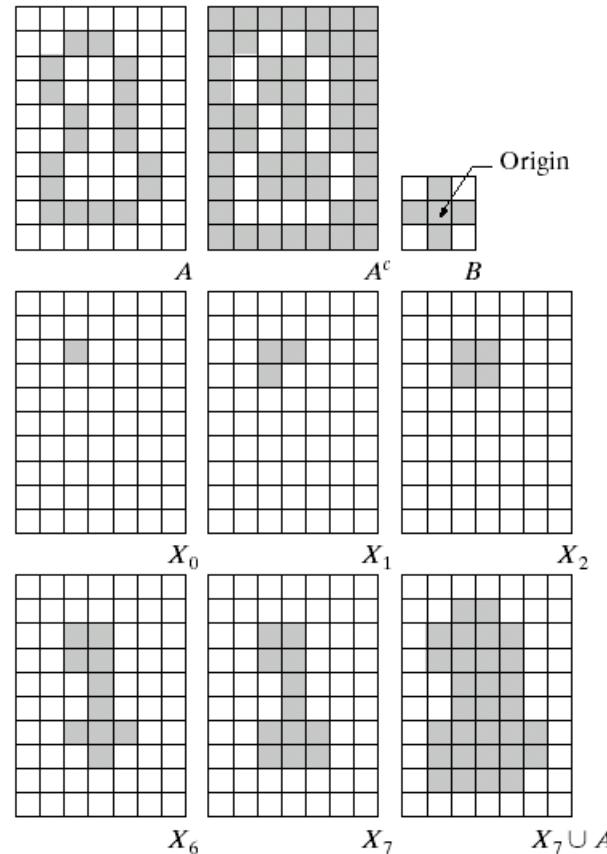
Region Filling (2)

a	b	c
d	e	f
g	h	i

FIGURE 9.15

Region filling.

- (a) Set A .
- (b) Complement of A .
- (c) Structuring element B .
- (d) Initial point inside the boundary.
- (e)–(h) Various steps of Eq. (9.5-2).
- (i) Final result [union of (a) and (h)].



From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



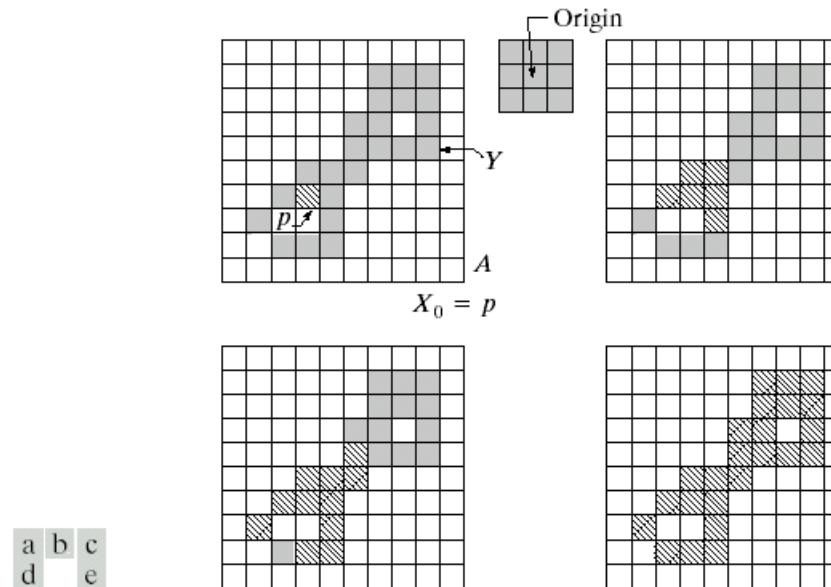
Morphologic Image Processing

7.3 Extraction of Connected Components

Let Y represent a connected component contained in a set A and assume that a point p of Y is known. Then the following iterative expression yields all the elements of Y :

$$X_k = (X_{k-1} \oplus B) \cap A, k = 1, 2, 3, \dots$$

where $X_0 = p$, and B is a suitable structuring element. If $X_k = X_{k-1}$, the algorithm has converged and we let $Y = X_k$.



From “Digital Image Processing”
by R. C. Gonzales and R. E. Woods

FIGURE 9.17 (a) Set A showing initial point p (all shaded points are valued 1, but are shown different from p to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.



Morphologic Image Processing

7.4 Thinning

The thinning of a set A by a structuring element B , denoted

$$A \otimes B = A - (A \circledast B) \\ = A \cap (A \circledast B)^c.$$

For thinning A symmetrically a sequence of structuring elements is necessary:

$$B = \{B^1, B^2, B^3, \dots, B^n\} \\ \text{where } B^i \text{ is a rotated version of } B^{i-1}$$

$$A \otimes \{B\} = ((\dots((A \otimes B^1) \otimes B^2) \dots) \otimes B^n)$$

From “Digital Image Processing”
by R. C. Gonzales and R. E. Woods

a
b
c
d
e
f
g
h
i
j
k
l

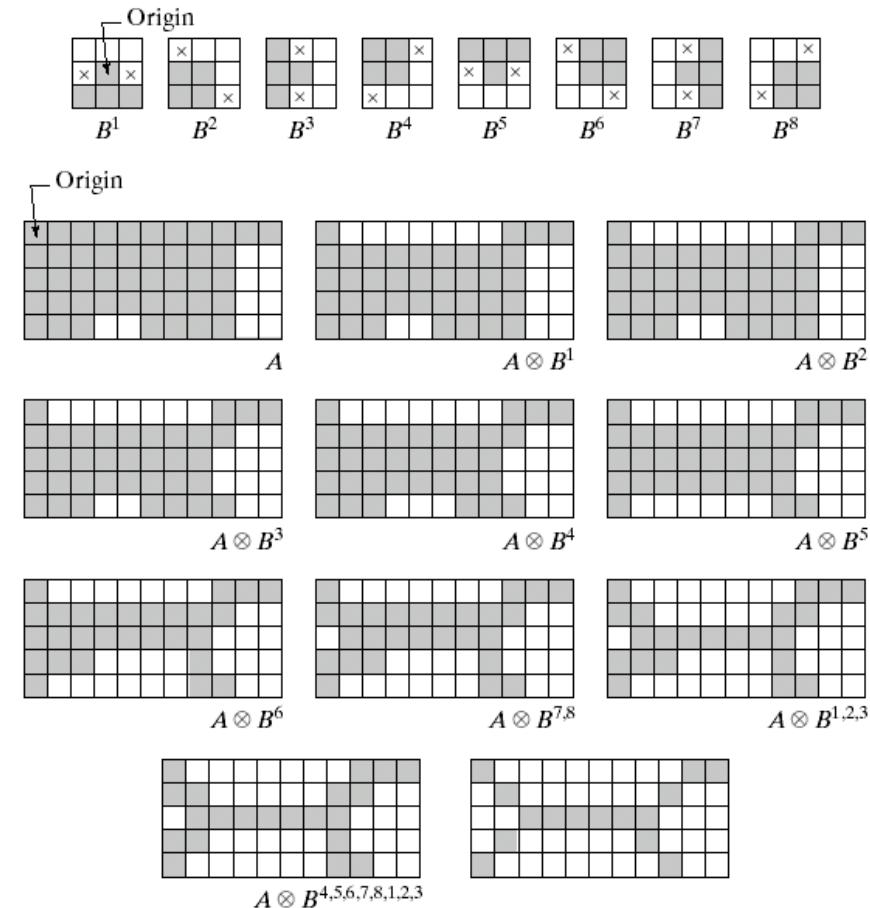


FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A . (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there were no changes for the next two elements). (k) Result after convergence. (l) Conversion to m -connectivity.



Morphologic Image Processing

Other Morphological algorithms:

-Convex Hull detection, Thickening, Skeletons, Pruning

References:

Rafael C. Gonzales, Richard E. Woods, “Digital Image Processing”, *Prentice Hall*, 2002

Robert M. Haralick, Linda G. Shapiro, “Computer and Robot Vision”, *Addison-Wesley Publishing Company*, 1993



Morphologic Image Processing

TABLE 9.2

Summary of morphological operations and their properties.

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Translation	$(A)_z = \{w \mid w = a + z, \text{ for } a \in A\}$	Translates the origin of A to point z .
Reflection	$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$	Reflects all elements of B about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in A .
Difference	$A - B = \{w \mid w \in A, w \notin B\}$ $= A \cap B^c$	Set of points that belong to A but not to B .
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$	“Expands” the boundary of A . (I)
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	“Contracts” the boundary of A . (I)
Opening	$A \circ B = (A \ominus B) \oplus B$	Smoothes contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks. (I)
Closing	$A \bullet B = (A \oplus B) \ominus B$	Smoothes contours, fuses narrow breaks and long thin gulfs, and eliminates small holes. (I)

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

Hit-or-miss transform	$\begin{aligned} A \circledast B &= (A \ominus B_1) \cap (A^c \ominus B_2) \\ &= (A \ominus B_1) - (A \oplus \hat{B}_2) \end{aligned}$	The set of points (coordinates) at which, simultaneously, B_1 found a match (“hit”) in A and B_2 found a match in A^c .
Boundary extraction	$\beta(A) = A - (A \ominus B)$	Set of points on the boundary of set A . (I)
Region filling	$X_k = (X_{k-1} \oplus B) \cap A^c; X_0 = p$ and $k = 1, 2, 3, \dots$	Fills a region in A , given a point p in the region. (II)
Connected components	$X_k = (X_{k-1} \oplus B) \cap A; X_0 = p$ and $k = 1, 2, 3, \dots$	Finds a connected component Y in A , given a point p in Y . (I)
Convex hull	$\begin{aligned} X_k^i &= (X_{k-1}^i \circledast B^i) \cup A; i = 1, 2, 3, 4; \\ k &= 1, 2, 3, \dots; X_0^i = A; \text{ and} \\ D^i &= X_{\text{conv}}^i \end{aligned}$	Finds the convex hull $C(A)$ of set A , where “conv” indicates convergence in the sense that $X_k^i = X_{k-1}^i$. (III)

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Thinning	$A \otimes B = A - (A \oslash B)$ $= A \cap (A \oslash B)^c$ $A \otimes \{B\} =$ $((\dots((A \otimes B^1) \otimes B^2) \dots) \otimes B^n)$ $\{B\} = \{B^1, B^2, B^3, \dots, B^n\}$	Thins set A . The first two equations give the basic definition of thinning. The last two equations denote thinning by a sequence of structuring elements. This method is normally used in practice. (IV)
Thickening	$A \odot B = A \cup (A \oslash B)$ $A \odot \{B\} =$ $((\dots(A \odot B^1) \odot B^2 \dots) \odot B^n)$..	Thickens set A . (See preceding comments on sequences of structuring elements.) Uses IV with 0's and 1's reversed.

TABLE 9.2
Summary of morphological results and their properties.
(continued)

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

Skeletons

$$\begin{aligned} S(A) &= \bigcup_{k=0}^{\infty} S_k(A) \\ S_k(A) &= \bigcup_{k=0}^K \{(A \ominus kB) \\ &\quad - [(A \ominus kB) \circ B]\} \\ \text{Reconstruction of } A: \\ A &= \bigcup_{k=0}^K (S_k(A) \oplus kB) \end{aligned}$$

Finds the skeleton $S(A)$ of set A . The last equation indicates that A can be reconstructed from its skeleton subsets $S_k(A)$. In all three equations, K is the value of the iterative step after which the set A erodes to the empty set. The notation $(A \ominus kB)$ denotes the k th iteration of successive erosion of A by B . (I)

Pruning

$$\begin{aligned} X_1 &= A \otimes \{B\} \\ X_2 &= \bigcup_{k=1}^8 (X_1 \oplus B^k) \\ X_3 &= (X_2 \oplus H) \cap A \\ X_4 &= X_1 \cup X_3 \end{aligned}$$

X_4 is the result of pruning set A . The number of times that the first equation is applied to obtain X_1 must be specified. Structuring elements V are used for the first two equations. In the third equation H denotes structuring element I .

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Morphologic Image Processing

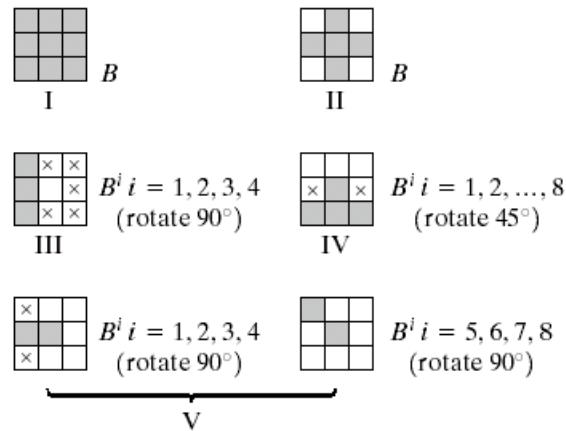


FIGURE 9.26 Five basic types of structuring elements used for binary morphology. The origin of each element is at its center and the \times 's indicate “don't care” values.

From “Digital Image Processing” by R. C. Gonzales and R. E. Woods



Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester)

Lecture 6:

Grayscale Image processing (I)

Statistical image features and applications. Image enhancement



Basic Statistical Properties

Mean and Variance

For the digital image the definition of mean and variance are given by:

$$\mu = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j)$$
$$\sigma^2 = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (f(i, j) - \mu)^2$$

Notation

For a 1-D digital signal the mean or average is defined as:

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} f(i) = \langle f(i) \rangle$$

Similarly in 2-D we have:

$$\mu = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) = \langle f(i, j) \rangle$$

The variance is then written as:

$$\sigma^2 = \langle |f(i, j) - \mu|^2 \rangle$$



Basic Statistical Properties

Calculation of Mean and Variance

Looks like a “double scan” through the image

1. Calculate

$$\mu = \langle f(i, j) \rangle$$

2. Calculate

$$\sigma^2 = \langle |f(i, j) - \mu|^2 \rangle$$

But we can expand

$$\sigma^2 = \langle |f(i, j) - \mu|^2 \rangle = \langle |f(i, j)|^2 \rangle - 2\langle f(i, j) \rangle \mu + \mu^2 = \langle |f(i, j)|^2 \rangle - \langle f(i, j) \rangle^2$$

both of which can be formed in a single pass through the image.

We are able to calculate **both** mean and variance by calculating:

$$\langle |f(i, j)|^2 \rangle \quad \& \quad \langle f(i, j) \rangle$$



Histogram

Take the digital image $f(i, j)$ as a random function f with $0 \leq f \leq 255$.

We can define:

The **Probability Distribution Function** as:

$$P(f) = \text{Prob. Pixel value} \leq f$$

so that

$$0 \leq P(f) \leq 1$$

and

$$P(f_{max})=1$$

The **Probability Density Function (PDF)** as:

$$p(f)=dP(f)/df$$

For a digital image if there are M_0 pixels with values $f_0 \rightarrow f_0 + \Delta f$ then PDF can be estimated by:

$$p(f_0)=M_0/N^2 \Delta f$$

So if $\Delta f = 1$ then the PDF is the **normalized histogram**

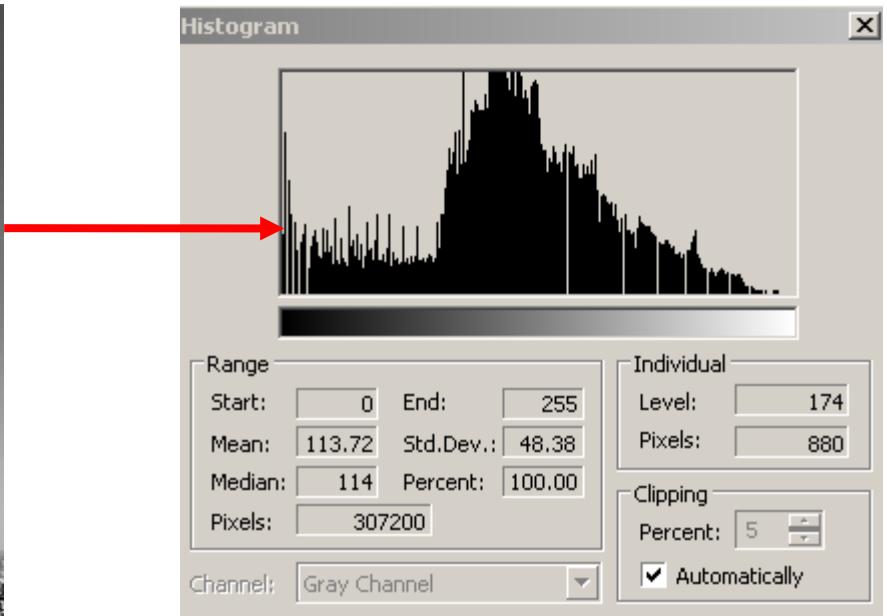
$$p(f)=h(f)/N^2$$

where $h(f)$ is the gray level **histogram of the image f** which shows the distribution of gray-levels over the range of values.



Histogram

```
for i=0 to N-1  
  for j=0 to N-1  
    increment h(f(i, j))  
  endfor  
endfor
```





Basic Statistical Properties

Mean and Variance

The *mean* and *variance* can be expressed in terms of the Probability Density Function, (PDF), being given by:

$$\mu = \int_{-\infty}^{\infty} fp(f)df$$

$$\sigma^2 = \int_{-\infty}^{\infty} (f - \mu)^2 p(f)df$$

So in the discrete case of the histogram $h(f)$:

$$\mu = \frac{1}{N^2} \sum_{f=0}^{f_{\max}} fh(f)$$

$$\sigma^2 = \frac{1}{N^2} \sum_{f=0}^{f_{\max}} (f - \mu)^2 h(f)$$

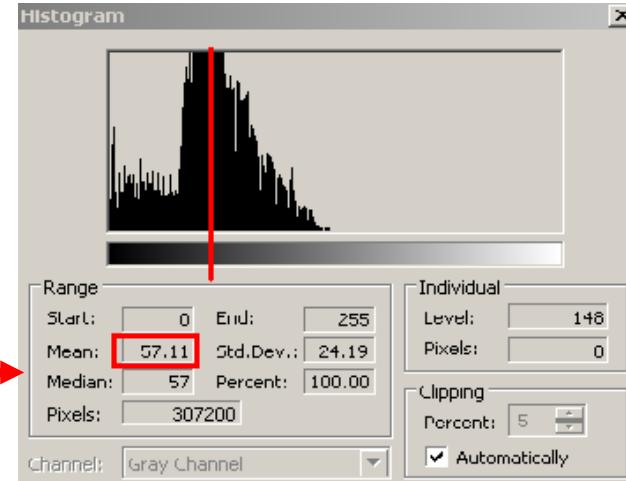


Statistical features

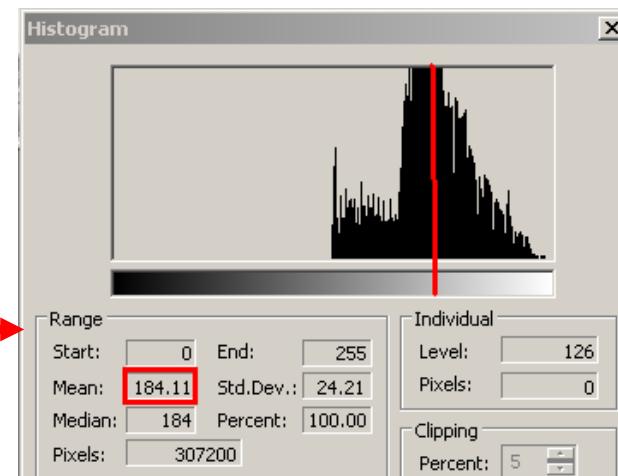
Mean (μ)

⇒ measure of the average brightness of the image/ROI

Dark image



Bright image



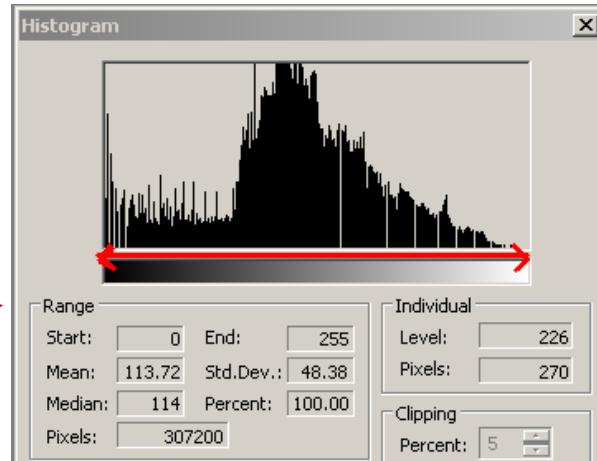


Statistical features

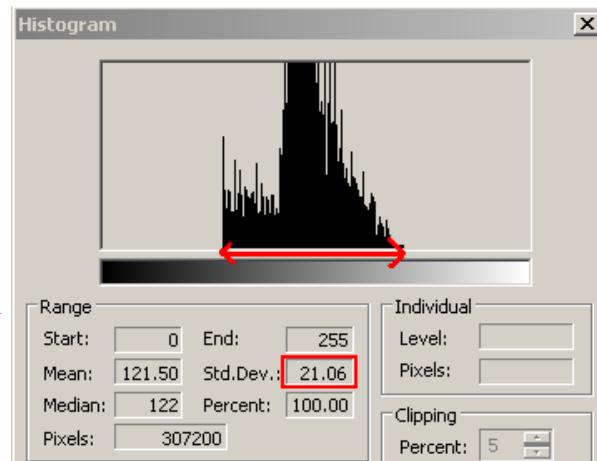
Standard deviation (σ), Variance (σ^2)

⇒ measure of the contrast of the image/ROI

High
contrast



Low
contrast





Application: grayscale image segmentation

Basic global thresholding algorithm

- Computes automatically the threshold (T)
- Can be applied on images with **bimodal** histograms

The algorithm

1. Take an initial value for T :

$$T_0 = \mu \text{ (object area} = \text{background area)}$$

$$T_0 = (f_{MAX} + f_{MIN})/2$$

2. Segment the image after T by dividing the image pixels in 2 groups:

$$G1: f[i,j] \leq T \Rightarrow \mu_{G1}$$

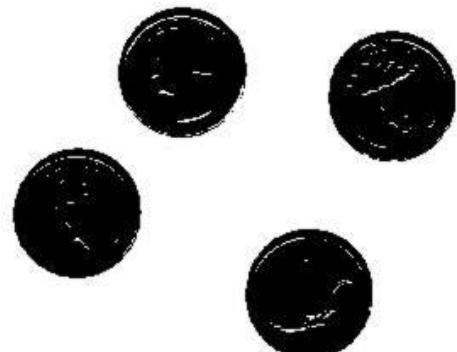
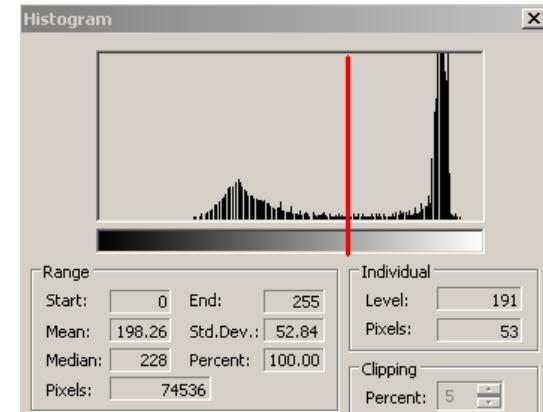
$$G2: f[i,j] > T \Rightarrow \mu_{G2}$$

3. $T = (\mu_{G1} + \mu_{G2})/2$

4. Repeat 2-3 until $T_k - T_{k-1} < e$

Efficient implementation \Rightarrow compute the image

histogram first then perform all computations on the histogram !!!





Application: grayscale image segmentation

$$\mu_{G_1} = \frac{1}{N^2} \sum_{f=0}^{f_t} fh(f) = \sum_{f=0}^{f_t} fp(f)$$

$$\mu_{G_2} = \frac{1}{N^2} \sum_{f=f_{t+1}}^{f_{\max}} fh(f) = \sum_{f=f_{t+1}}^{f_{\max}} fp(f)$$



Image enhancement: histogram slide

$$\text{Slide}(f[i, j]) = f[i, j] + \text{offset}$$

$\text{offset} > 0 \Rightarrow \text{brighter image}$
 $\text{offset} < 0 \Rightarrow \text{darker image}$

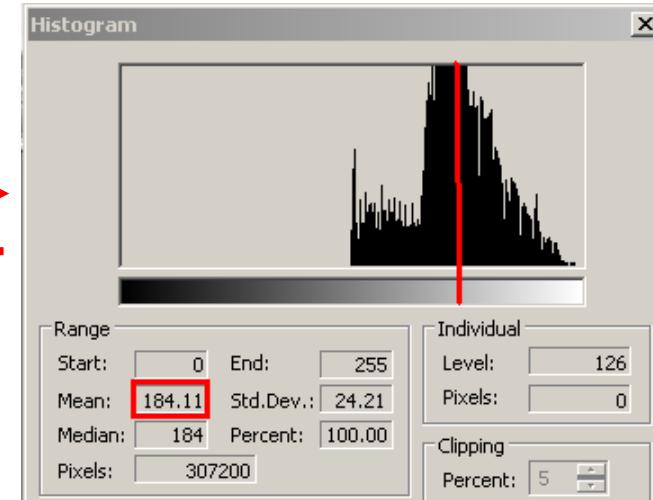
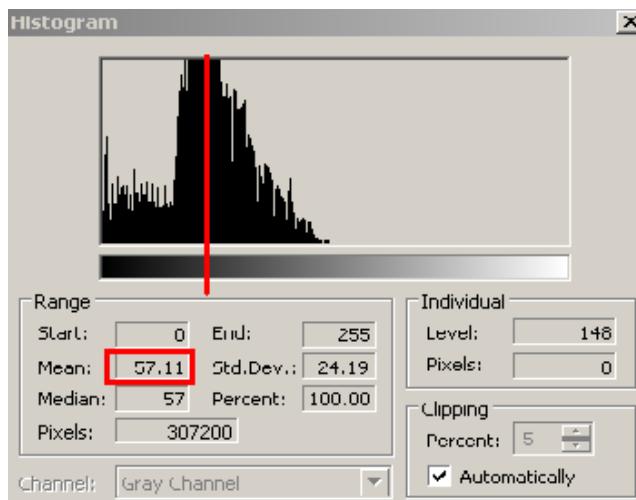
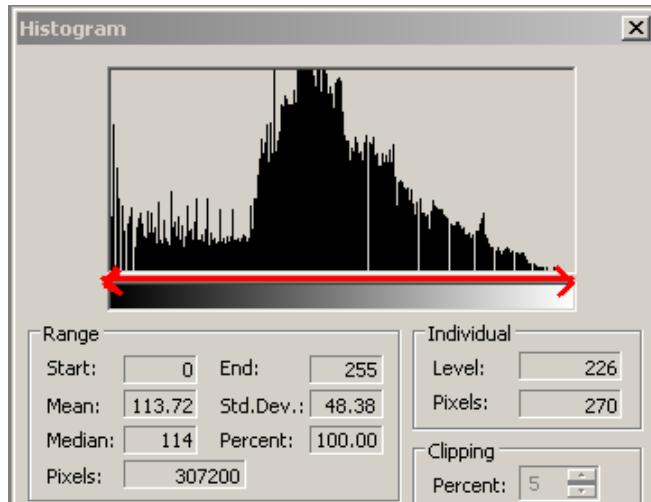




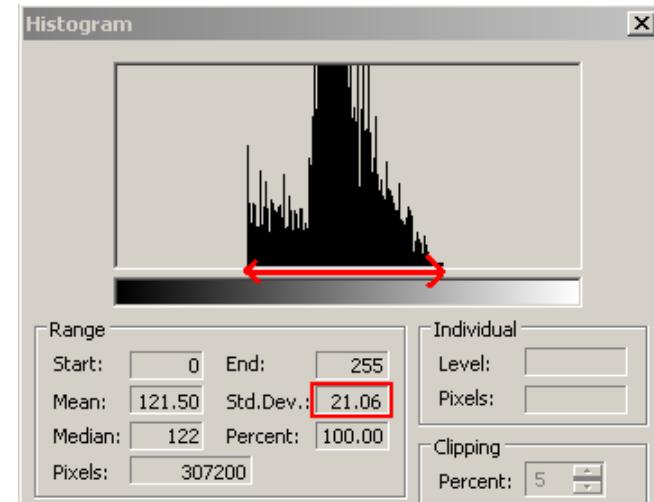
Image enhancement:histogram stretch/shrink

$$\text{Stretch/Shrink}(f[i,j]) = \text{Final}_{\text{MIN}} + (\text{Final}_{\text{MAX}} - \text{Final}_{\text{MIN}}) * (f[i,j] - f_{\text{MIN}}) / (f_{\text{MAX}} - f_{\text{MIN}})$$



shrink

stretch



shrink

stretch





Image enhancement

Gray level mapping using a transformation function

$$g_{output} = T(f_{input})$$

Ex. - gamma correction:

$$g_{out} = c \cdot f_{in}^{\gamma}$$

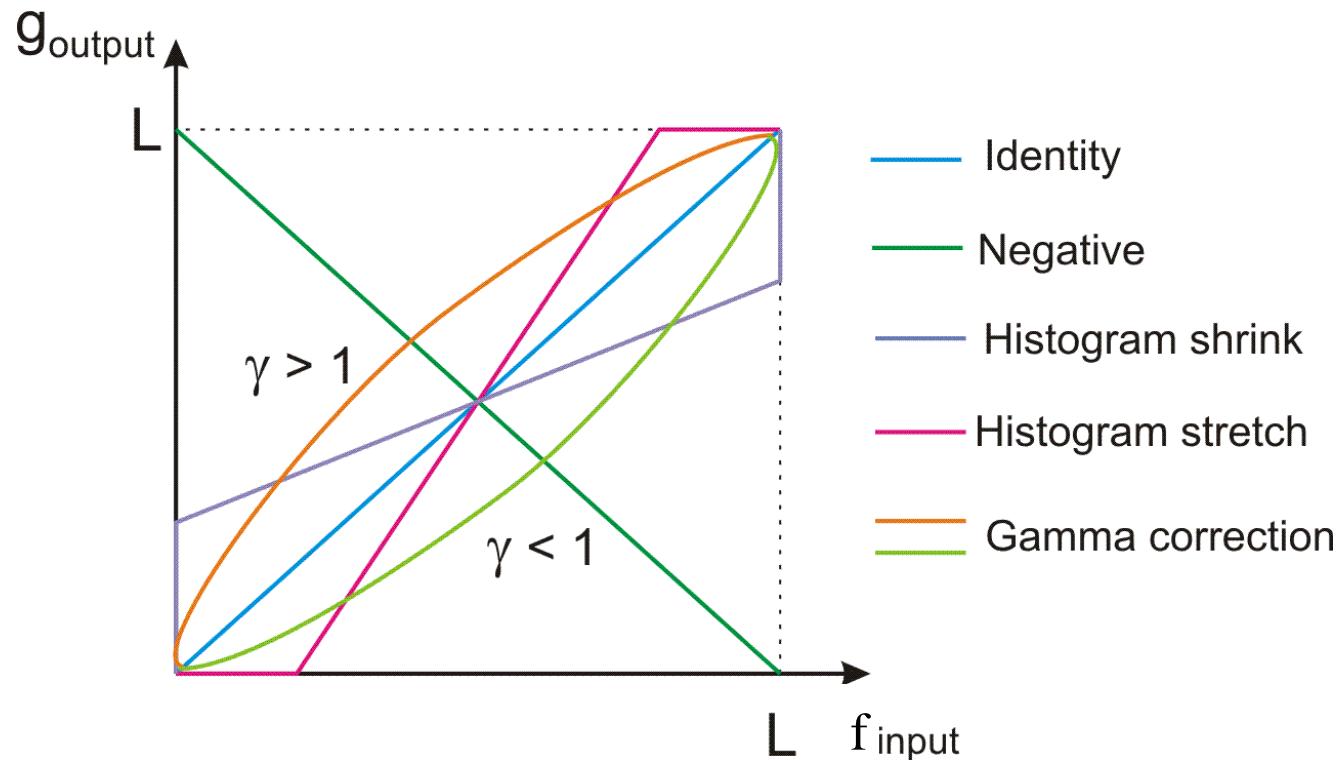
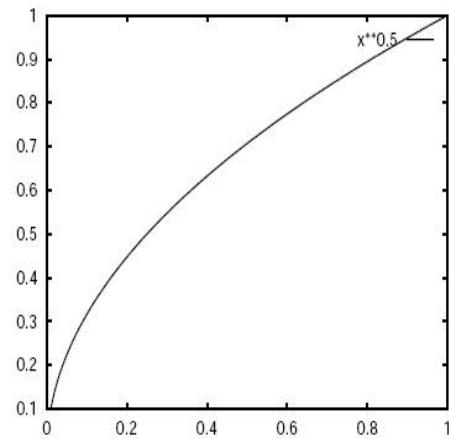


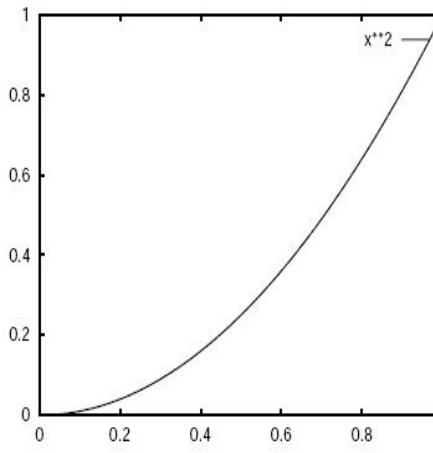
Image enhancement: gamma correction

The photographic process in practice contains non-linearities of the type:

- $g(x,y) = f(x,y)^\gamma$ where $f(x,y)$ is the real intensity, $g(x,y)$ is the recorded intensity and γ is a constant.
- We digitize and display $g(x,y)$. To correct this we need a transformation of the form: $T(g)=g^{1/\gamma}$. Really $T(g) = g_{max} (g/g_{max})^{1/\gamma}$.



Correction of $\gamma=2$

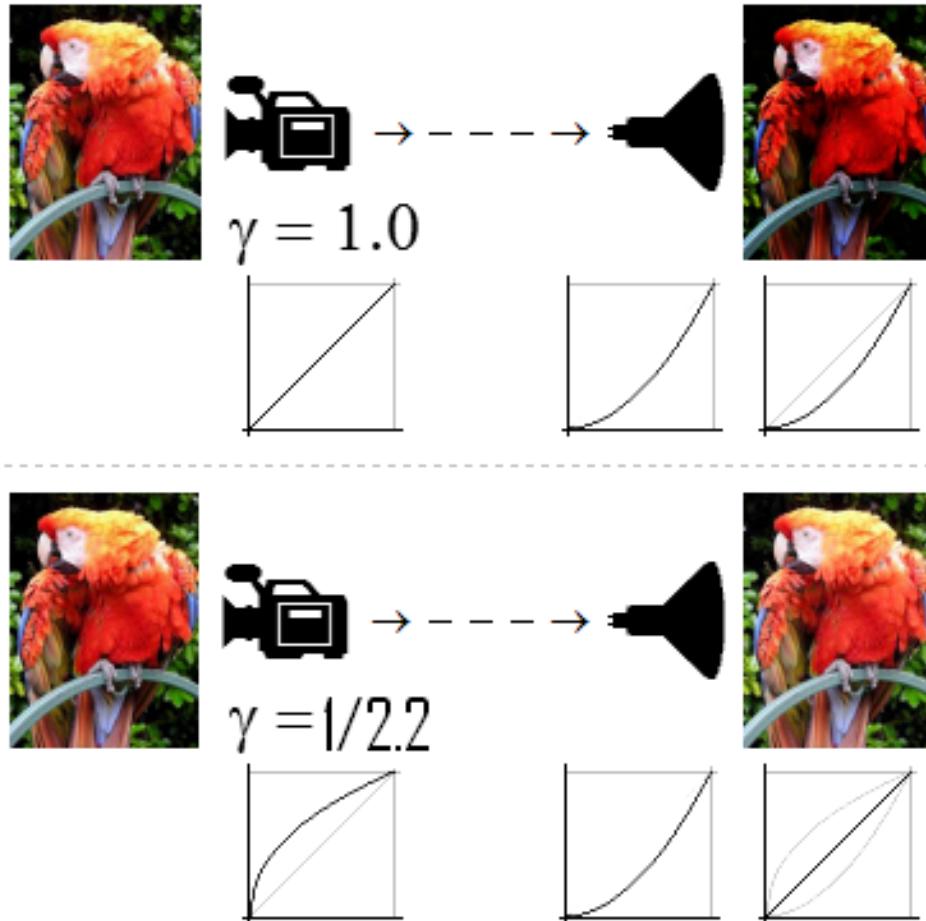


Correction of $\gamma=0.5$





Image enhancement: gamma correction

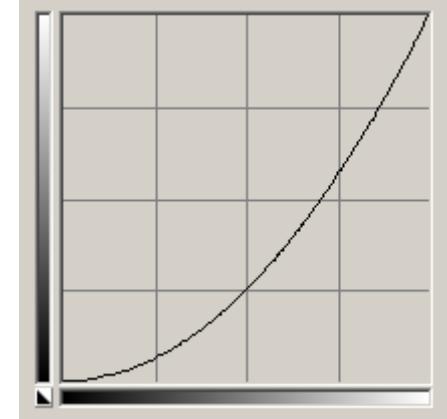
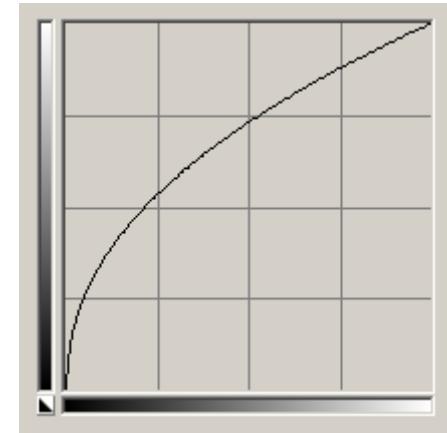


- The three curves represent input–output functions of the camera, the display, and the overall system, respectively.



Image enhancement

Ex. - gamma correction:





More statistical features

- Shannon derived a measure of information content called the **self-information** or "**surprisal**" of a message m :

$$I(m) = \log(1/p(m)) = -\log(p(m))$$

- where **p(m) = Pr(M=m)** is the probability that message m is chosen from all possible choices in the message space. The base of the logarithm only affects a scaling factor and, consequently, the units in which the measured information content is expressed. **If the logarithm is base 2, the measure of information is expressed in units of bits.**
- Information is transferred from a source to a recipient only if the recipient of the information did not already have the information to begin with. Messages that convey information that is certain to happen and already known by the recipient contain no real information.
- Infrequently occurring messages contain more information than more frequently occurring messages. This fact is reflected in the above equation - **a certain message, i.e. of probability 1, has an information measure of zero.**
- A compound message of two (or more) unrelated (or mutually independent) messages would have a quantity of information that is the sum of the measures of information of each message individually.**



More statistical features

Information - the information associated to the gray-level f :

$$I_g = -\log_2 p(f) \quad [bits]$$

⇒ information is large when an unlikely gray-level is generated

Entropy – average information of the image:

$$H = -\sum_{f=0}^L p(f) \cdot \log_2 p(f) \quad [bits]$$

⇒ how many bits we need to code the image data:

H is high – pixel values are distributed among many gray levels

$$H_{max} = -\sum_{f=0}^L \frac{1}{L} \log_2 \frac{1}{L} = \sum_{f=0}^L \frac{1}{L} \log_2 L = \log_2 L \quad [bits] \text{ (uniform PDF)}$$

Energy – how the gray-levels are distributed:

$$E = \sum_{f=0}^L [p(f)]^2$$

E (low) – number of gray-levels of the image is high

$$E_{max} = 1 \quad (\text{only one gray-level in the image})$$



Histogram processing

Histogram equalization

Aim is to distribute pixels equally across available grey level range.

Normalized grayscale levels:

$$f \in [0 \dots L-1] \Rightarrow r \in [0 \dots 1]$$

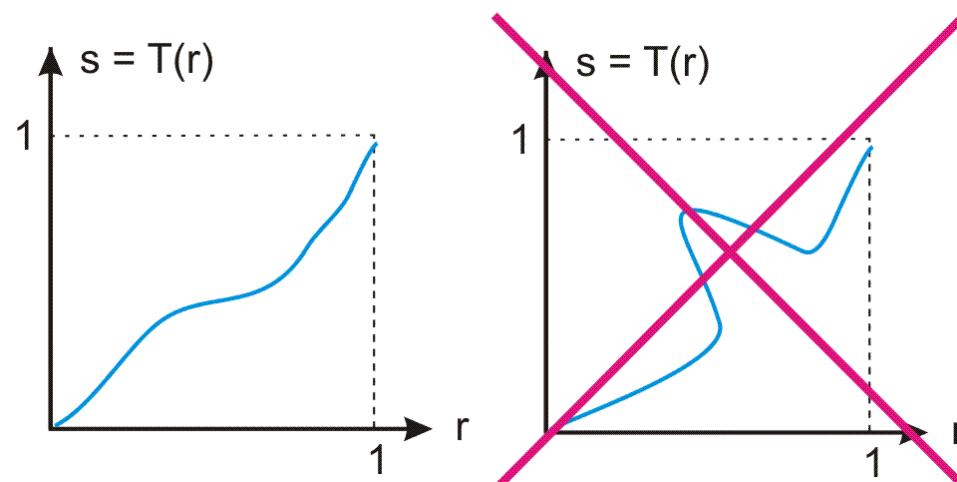
Transformation function:

$$s = T(r) \in [0 \dots 1] \Rightarrow g \in [0 \dots L-1]$$

T features:

(a). single valued and monotonically increasing $\Rightarrow \exists r = T^{-1}(s)$

(b). $0 \leq T(r) \leq 1$





Histogram processing

Histogram equalization

- $p_r(r)$, $p_s(s)$ – probability density functions of the input and output
- $p_r(r)$, $s=T(r)$ are known and T^{-1} satisfies condition (a) $\rightarrow r=T^{-1}(s)$

$$P(s) = \text{Prob}(\text{pixel value} < s) = \int_{-\infty}^s p_s(s) ds$$

$$P(s) = \text{Prob}(T(r) < s)$$

$$T(r) < s \mid T^{-1} \quad T^{-1}(T(r)) < T^{-1}(s) \quad r < T^{-1}(s)$$

$$P(s) = \text{Prob}(r < T^{-1}(s)) = \int_{-\infty}^{T^{-1}(s)} p_r(r) dr$$

$$p(s) = \frac{dP(s)}{ds} = \frac{d(T^{-1}(s))}{ds} p_r(T^{-1}(s)) = \frac{dr}{ds} p_r(r)$$

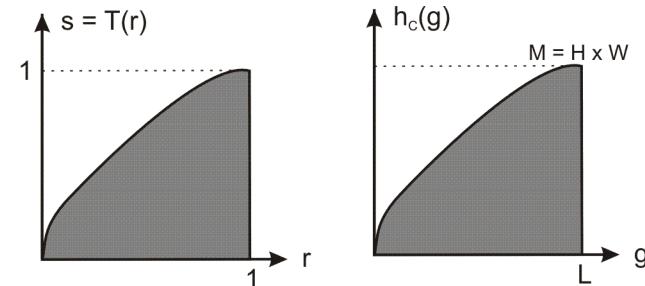
$$p_s(s) = \frac{dP(s)}{ds} = p_r(r) \frac{dr}{ds} \quad (1)$$



Histogram processing

Cumulative histogram / cumulative density function (CDF)

$$s = T(r) = \int_0^r p_r(w) dw \quad (2)$$



T satisfies (a) & (b)

Leibniz rule: the derivative of a definite integral with respect to its upper limit is simply the integrand evaluated at that limit

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d}{dr} \left[\int_0^r p_r(w) dw \right] = p_r(r) \quad (3)$$



Histogram processing

Histogram equalization

$$(1) + (3) \Rightarrow$$

$$p_s(s) = p_r(r) \frac{dr}{ds} = p_r(r) \frac{1}{p_r(r)} = 1 \quad , \quad 0 \leq s \leq 1$$

$p_s(s)$:

- uniform PDF
- independent from $p_r(r)$

Histogram equalization algorithm

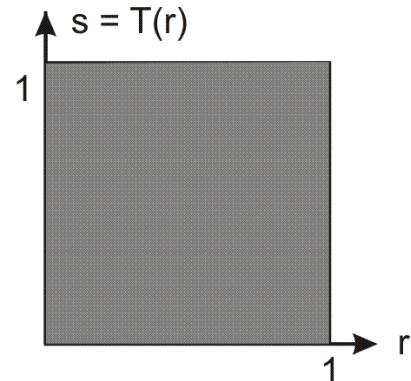
$$p_r(r_k) = \frac{n_k}{n} \quad , \quad k = 0..L$$

$$s_k = T(r_k) = \sum_{j=0}^k p_r(r_j) = \sum_{j=0}^k \frac{n_j}{n} \quad , \quad k = 0..L-1$$

\Rightarrow re-map the gray-scale values of the output image: $r_k \rightarrow s_k$

$$g_k = \text{round}(s_k(L - 1))$$

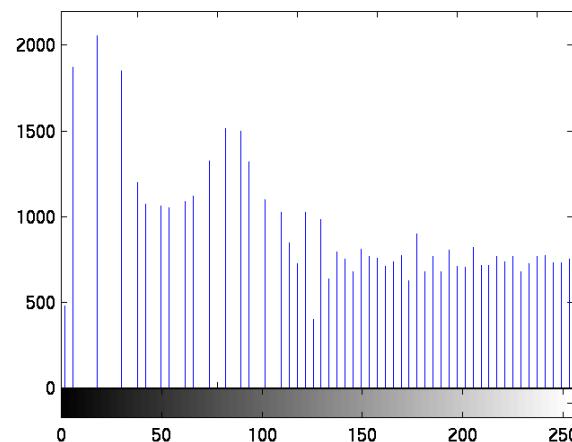
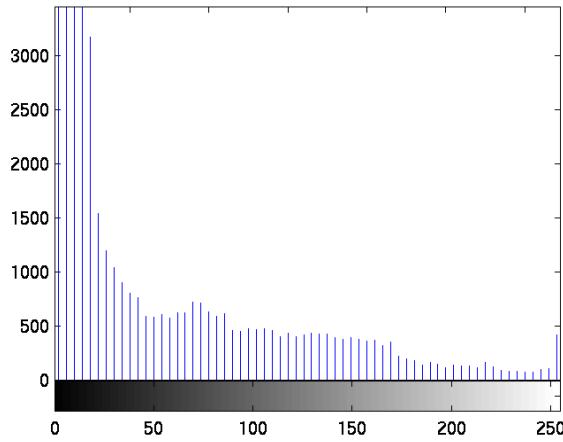
\Rightarrow re-scale the gray-scale values of the output image: $s_k \rightarrow g_k$





Histogram processing

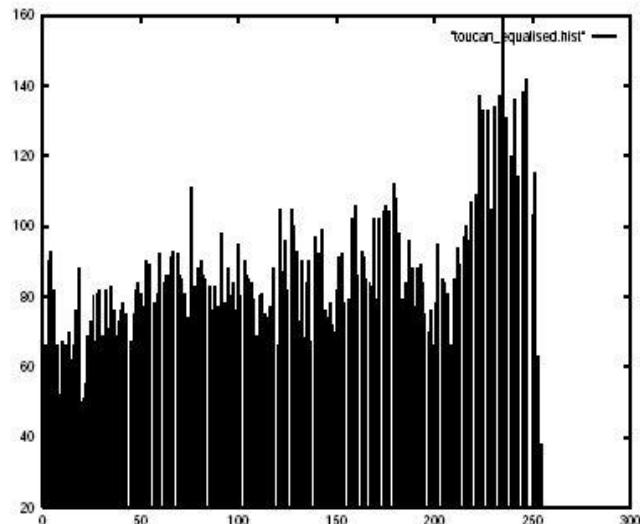
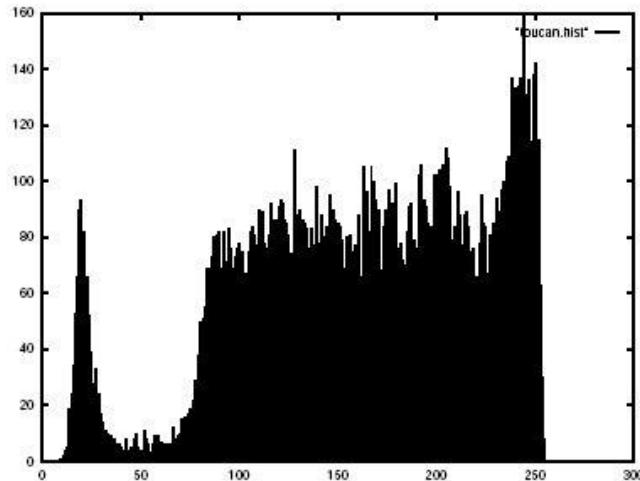
Histogram equalization (results)





Histogram processing

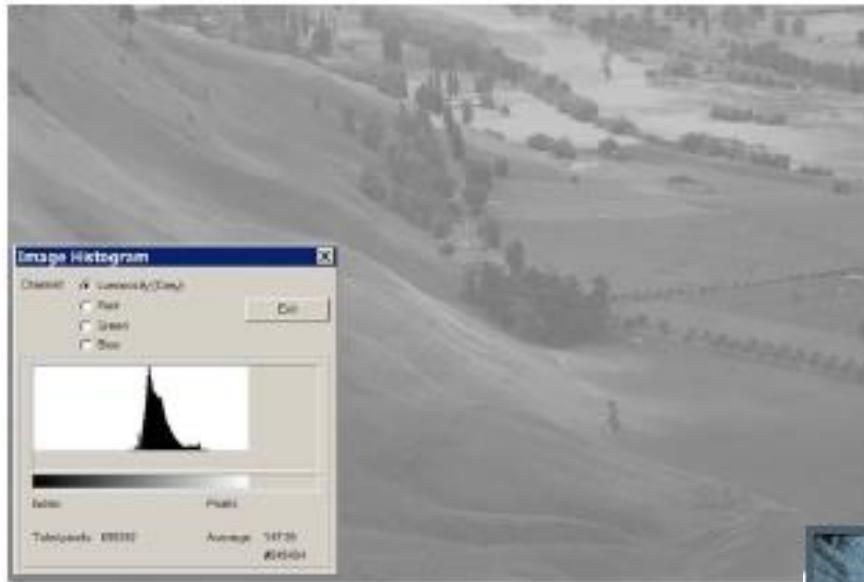
Histogram equalization (results)





Histogram processing

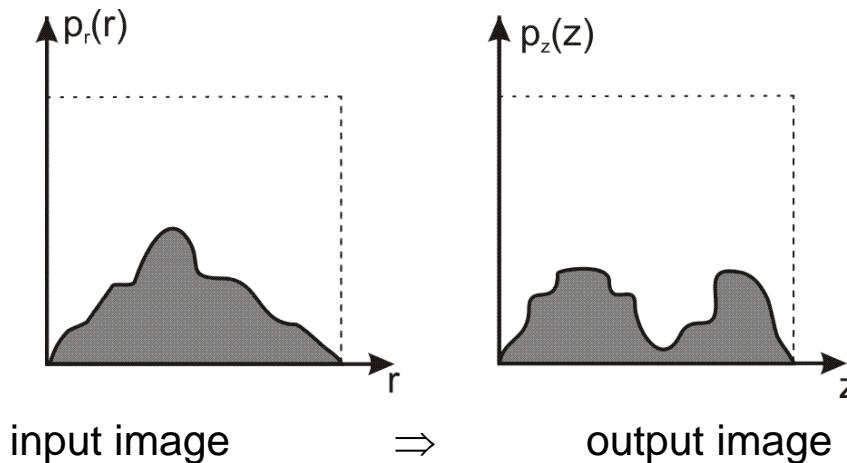
Histogram equalization (results)





Histogram processing

Histogram specification / matching



For the input image we have: $s = T(r) = \int_0^r p_r(w)dw$ (1)

We define a random variable z with the property:

$$G(z) = \int_0^z p_z(t)dt = s \quad (2)$$

$$(1) + (2) \Rightarrow G(z) = T(r)$$

$$z = G^{-1}(s) = G^{-1}[T(r)]$$



Histogram processing

Histogram specification / matching

$$s_k \leftrightarrow z_k$$

No analytical expressions for $T(r)$ and G^{-1} !?

Algorithm:

1. $r_k \leftrightarrow s_k$:

$$s_k = T(r_k) = \sum_{j=0}^k p_r(r_j) = \sum_{j=0}^k \frac{n_j}{n}, \quad k = 0..L$$

2. $s_k \leftrightarrow v_k$:

$$v_k = G(z_k) = \sum_{j=0}^k p_z(z_j) = s_k, \quad k = 0..L$$

3. $s_k \leftrightarrow z_k$:

Let $z' = z_k, \quad k = 0, \dots L$

z_k will be the smallest z' satisfying the condition:

$$(G(z') - s_k) \geq 0$$

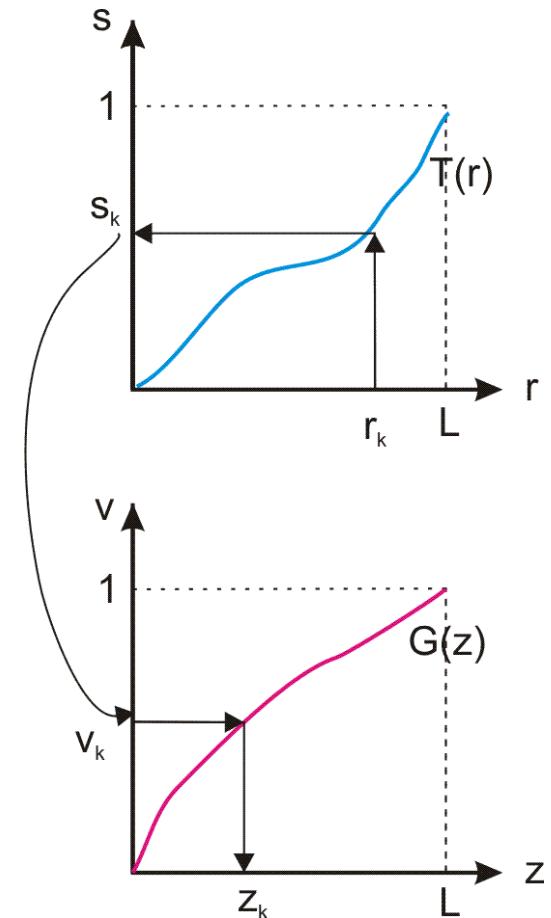




Image Processing

(Year III, 2-nd semester)

Lecture 7:
Grayscale Images:
Convolution, Fourier Transform (II)



Notations and definitions

A continuous image is represented as a function of two independent variables: $f(x,y)$, $u(x,y)$, $v(x,y)$ and so on.

A sampled image is represented as a 2-dimensional sequence of real numbers: $f(i,j)$, $u(k,l)$, $v(m,n)$, and so on.

The symbols i, j, k, l, m, n will be integer indices of arrays and vectors.

The symbol j represents $\sqrt{-1}$

Some well known one-dimensional functions that will be often used are **Dirac** and **Kronecker**. Their two-dimensional versions are functions of the separable form $f(x,y) = f_1(x)f_2(y)$:

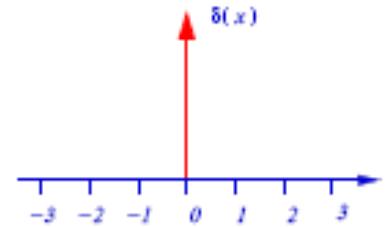
- Two-dimensional continuous **Dirac** delta function is defined by $\delta(x,y) = \delta(x)\delta(y)$.
- Two-dimensional discrete Kronecker delta function is defined by $\delta(m,n) = \delta(m)\delta(n)$.



Dirac Delta and Unit Impulse Function

The Dirac Delta Function can be thought of as a function on the real line which is zero everywhere except at the origin, and which is also constrained to satisfy the identity:

$$\delta(x) = 0 \text{ for } x \neq 0; \quad \int_{-\infty}^{+\infty} \delta(x) dx = 1$$



This can be thought of as a very “**tall-and-thin” spike with unit area located at the origin**, as shown in figure.

The δ -functions should **not be considered to be an infinitely high spike of zero width** since it scales as:

$$\int_{-\infty}^{+\infty} a\delta(x) dx = a$$

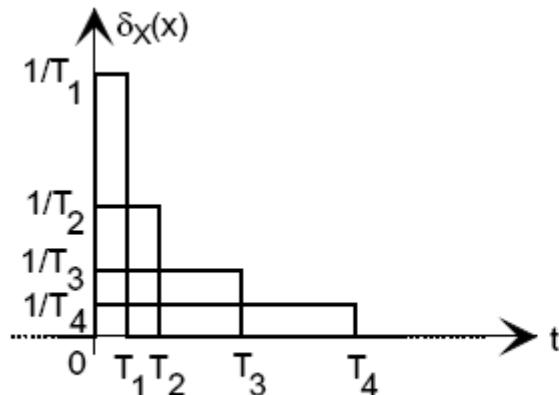
where a is a constant.

The delta function has the fundamental property that

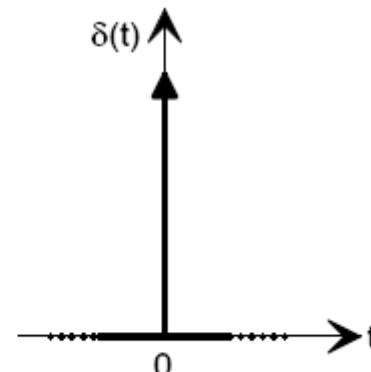
$$\int_{-\infty}^{+\infty} f(x)\delta(x)dx = f(0) \quad \text{and} \quad \int_{a-\varepsilon}^{a+\varepsilon} f(x)\delta(x)dx = f(0) \quad \text{for } a=0, \varepsilon>0.$$

This property can be extended (the shifting property)

$$\int_{-\infty}^{+\infty} f(x)\delta(x-a)dx = f(a) \quad \text{and} \quad \int_{a-\varepsilon}^{a+\varepsilon} f(x)\delta(x-a)dx = f(a) \quad \text{for } \varepsilon>0.$$



a) Unit pulses of different extents



b) The impulse function

Figure shows a *unit pulse function* $\delta_T(t)$, that is a brief rectangular pulse function of duration T , defined to have a constant amplitude $1/T$ over its extent, so that the area $T \times 1/T$ under the pulse is unity:

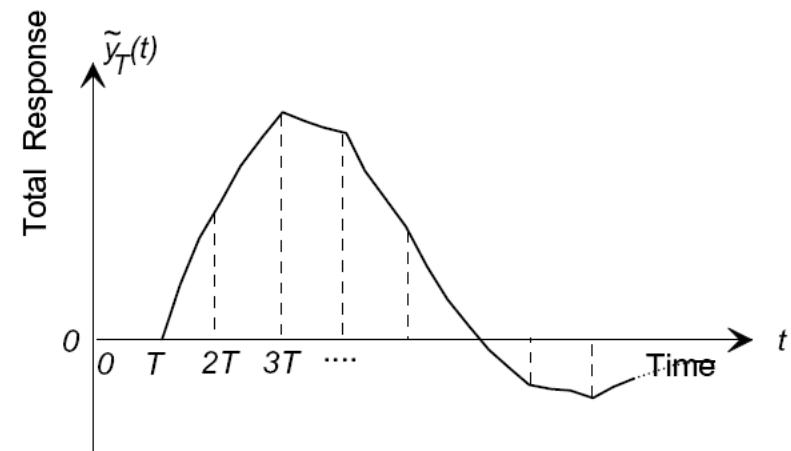
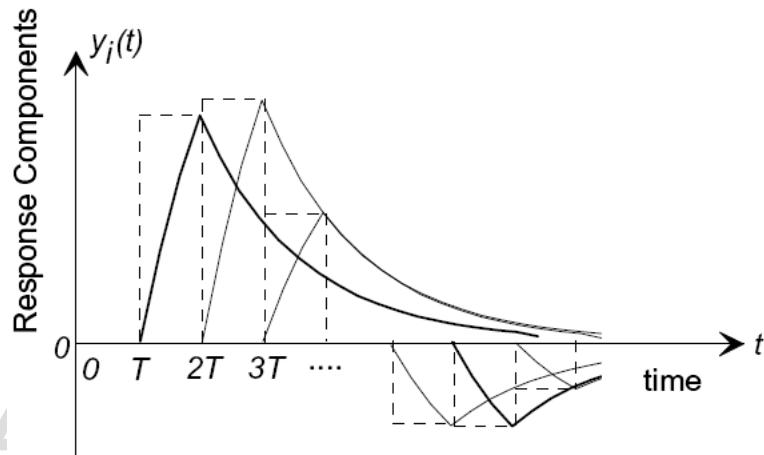
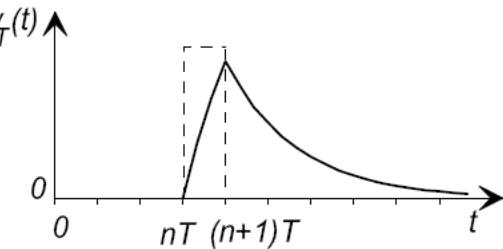
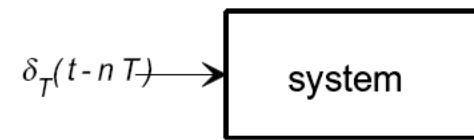
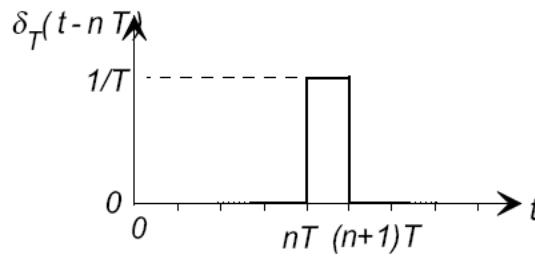
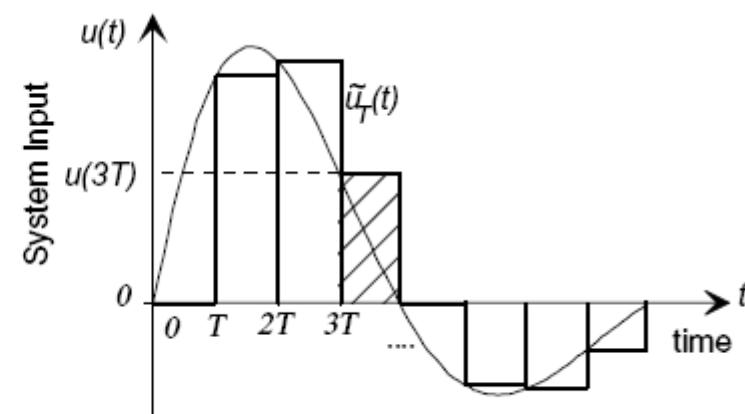
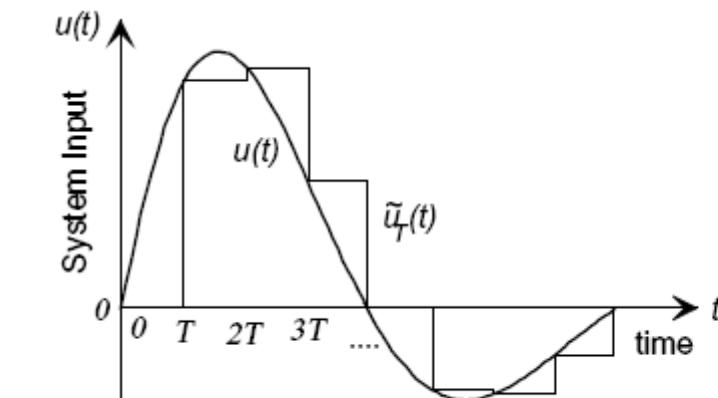
$$\delta_T(t) = \begin{cases} 0 & \text{for } t \leq 0 \\ 1/T & \text{for } 0 < t \leq T \\ 0 & \text{for } t > T \end{cases}$$

The Dirac Delta Function can be defined as the limiting form of the unit pulse $\delta_T(t)$ as the duration T approaches zero. As the duration T of $\delta_T(t)$ decreases, the amplitude of the pulse increases to maintain the requirement of unit area under the function, and

$$\delta(t) = \lim_{T \rightarrow 0} \delta_T(t)$$



System response to individual pulses in the staircase approximation





Linear Systems and Shift Invariance

- Let $f(m,n)$ be the **input sequence** and $g(m,n)$ be the **output sequence** of a two-dimensional system $g(m,n) = H[f(m,n)]$
- The system is **linear** if and only if for arbitrary constants a_1 and a_2 it holds:

$$H[a_1 f_1(m,n) + a_2 f_2(m,n)] =$$

$$= a_1 H[f_1(m,n)] + a_2 H[f_2(m,n)] = a_1 g_1(m,n) + a_2 g_2(m,n)$$

- This is called **linear superposition property**



Linear Systems and Shift Invariance(2)

- The output $g(m,n)$ of any linear system can be obtained as follows:

$$g(m,n) = H[f(m,n)] = H \left[\sum_{m'} \sum_{n'} f(m',n') \delta(m-m',n-n') \right]$$

$$= \sum_{m'} \sum_{n'} f(m',n') H[\delta(m-m',n-n')]$$

$$\Rightarrow g(m,n) = \sum_{m'} \sum_{n'} f(m',n') h(m,n; m',n')$$

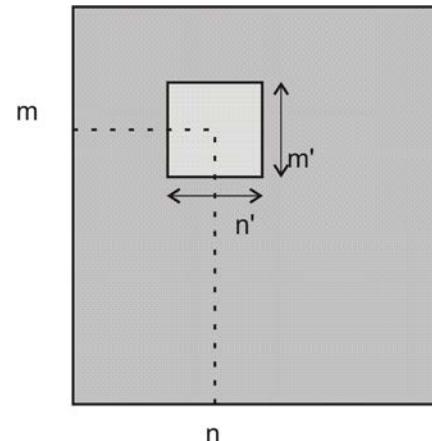
where $h(m,n; m',n')$ is the impulse response of the system.

- The impulse response $h(m,n; m',n')$ is the output of the system $H[\cdot]$ at location (m,n) , when the input is the two-dimensional Kronecker delta function at location (m',n')
- The impulse response is called the **point spread function, PSF**, when the inputs and outputs represent a positive quantity e.g. the intensity of light in imaging systems. The term impulse response is more general, values may be negative and complex.



Linear Systems and Shift Invariance(3)

- The **region of support** of an impulse response is the smallest closed region in the (m,n) plane outside which the impulse response is zero.
- A system is said to be a:
 - **finite impulse response, FIR** system if its impulse response has finite region of support;
 - **infinite impulse response, IIR** system if its impulse response has infinite region of support.





Linear Systems and Shift Invariance(4)

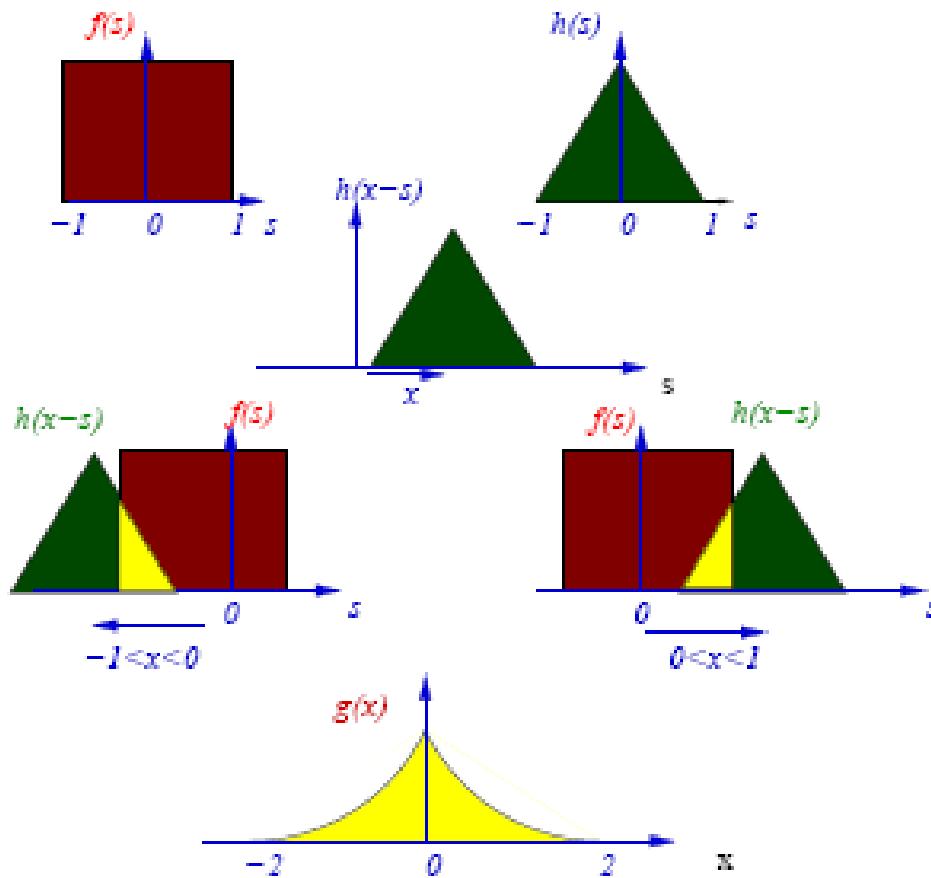
- A system is called **spatially invariant** or **shift invariant** if a translation of the input causes corresponding translation of the output.
- For shift invariant systems it holds: $h(m,n;m',n')=h(m-m', n-n')$
- The impulse response is a function of two variables only. The variables describe displacement.
- The shape of the impulse response does not change with the movements of the impulse in the (m,n) plane.
- For shift invariant systems, the output equals:

$$g(m,n) = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h(m - m', n - n') f(m', n')$$

which is called the **convolution** of the input with the impulse response.



Convolution





Convolution

- We will use the symbol $*$ to denote the convolution:

$$g(x, y) = h(x, y) * f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x', y') f(x - x', y - y') dx' dy'$$

$$g(x, y) = h(x, y) * f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x - x', y - y') f(x', y') dx' dy'$$

$$g(m, n) = h(m, n) * f(m, n) = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h(m', n') f(m - m', n - n')$$

$$g(m, n) = h(m, n) * f(m, n) = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h(m - m', n - n') f(m', n')$$



Convolution

Commutativity: $f * g = g * f$

Associativity: $f * (g * h) = (f * g) * h$

Distributivity: $f * (g + h) = (f * g) + (f * h)$

Identity element: $f * \delta = \delta * f = f$

Associativity with scalar multiplication:

$$a(f * g) = (af) * g = f * (ag)$$

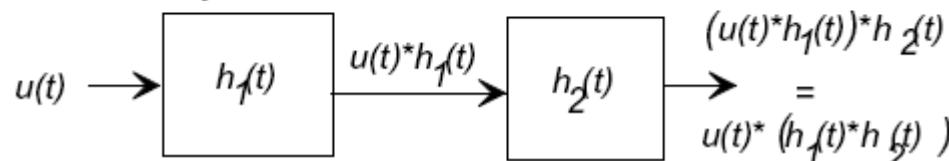
Differentiation rule: $D(f * g) = Df * g = f * Dg$

Convolution theorem: $F(f * g) = kF(f)F(g)$

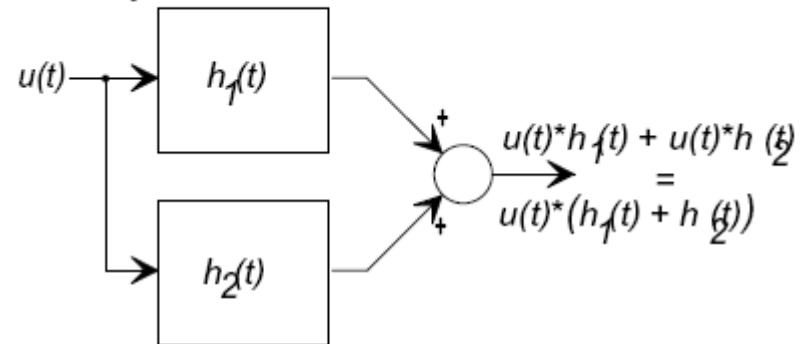


Convolution

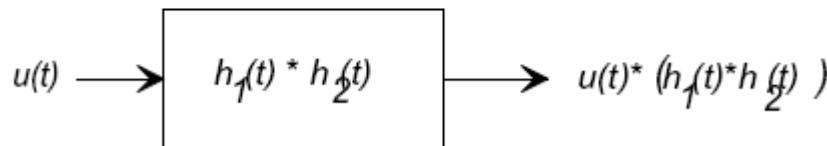
Cascade systems:



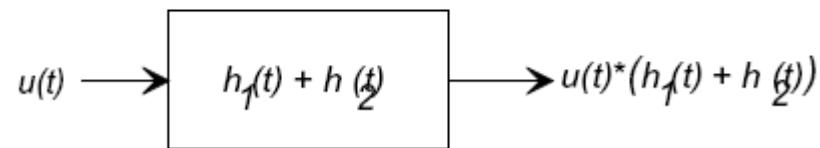
Parallel systems:



Equivalent system:



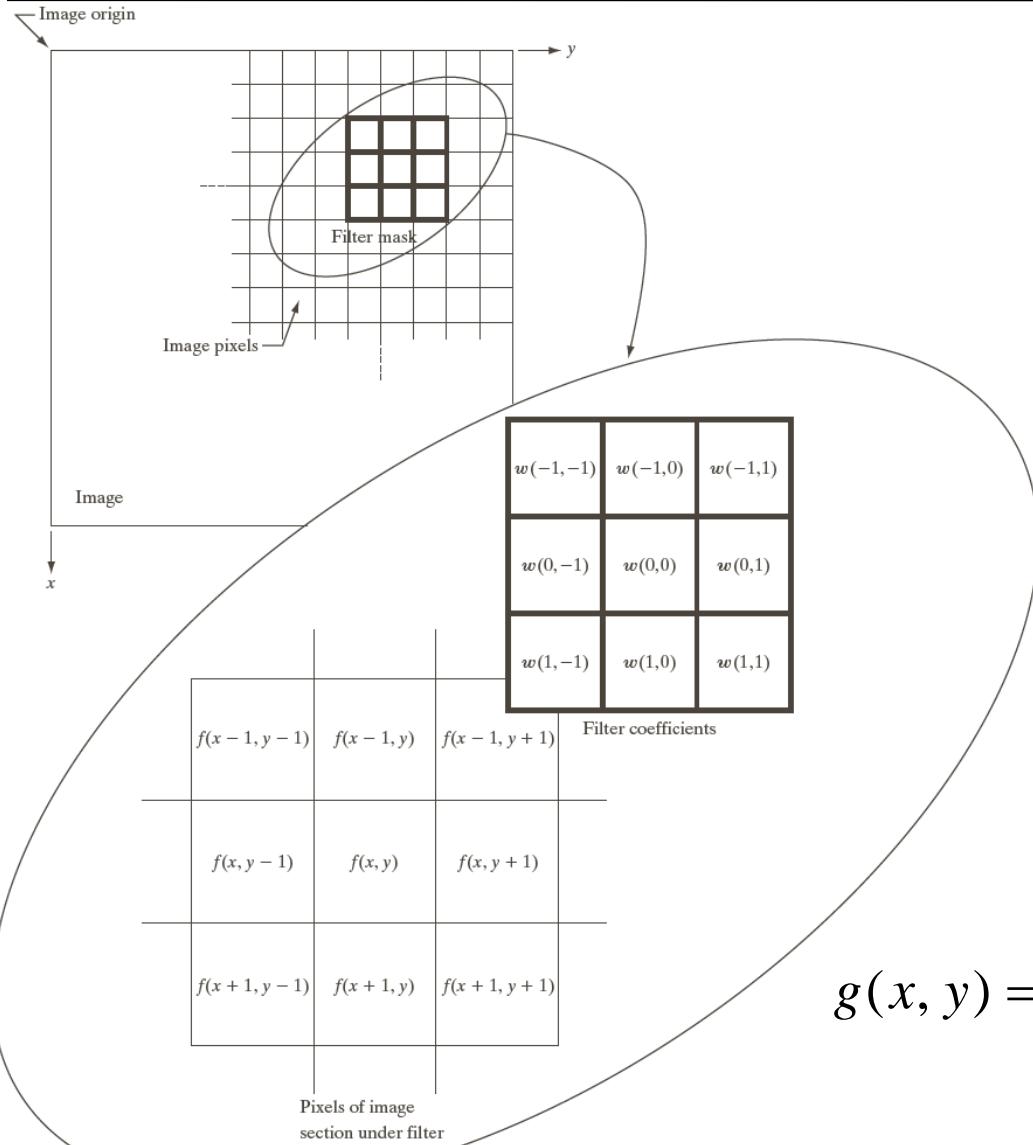
Equivalent system:



Impulse response of serial and parallel connected systems.



Convolution



$$g(x, y) = \sum_{s=-1}^{s=1} \sum_{t=-1}^{t=1} w(s, t) f(x + s, y + t)$$

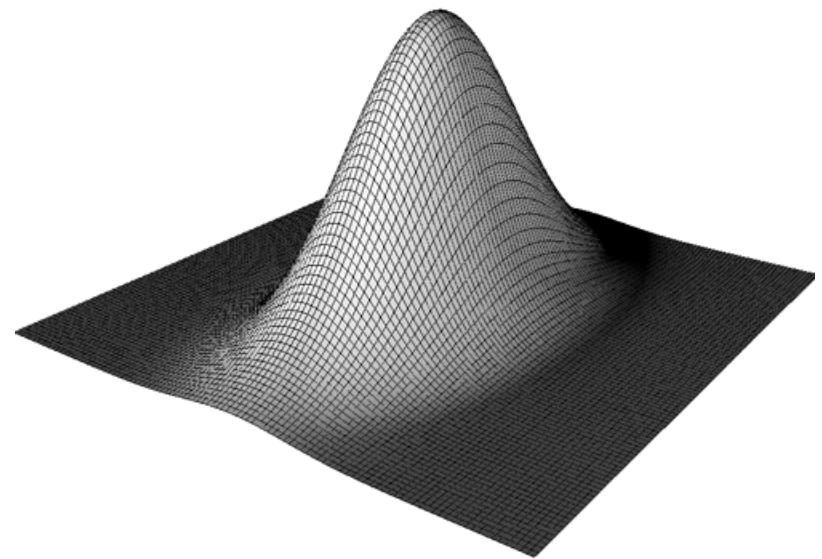


Convolution

$$G_{2D}(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1



Discrete approximation of the Gaussian
function with $\sigma=1.0$



The Fourier Transform

The Fourier Transform converts spatial image data $f(x,y)$ into a frequency representation $F(u,v)$. Both representations contain equivalent information. However, each representation has a set of strengths and weaknesses.

Spatial Domain

- +Intuitive Representation of Image data
- +Kernels applied directly to spatial data.
- Filtering with large Kernels may result in long processing times.

Frequency Domain

- Non-intuitive representation of image;
- +Filtering with large Kernels can be performed very quickly;
- Image and Kernel must first be converted to frequency domain, modified, then reconverted;
- +Engineering of the frequency filters is somewhat easier



The Fourier Transform

Image

- has been viewed as a spatial array of gray values,
- can be thought of as a spatially varying continuous (discrete) function.

The Fourier Transform decomposes the image into a set of orthogonal functions, called **basis functions**

- when basis functions are combined (linearly) the original function will be reconstructed.
- the Fourier basis functions are **sinusoids**
- changes in image intensity → changes in spatial frequency
- we write:
$$f(x,y) = \sum \text{Weighted basis functions}$$



The Fourier Transform

- The **forward Fourier transform** of a complex function $f(x)$ is defined as:

$$F(u) \stackrel{\Delta}{=} \mathcal{F}[f(x)] = \int_{-\infty}^{\infty} f(x) \exp(-j2\pi ux) dx$$

$$F(u) \stackrel{\Delta}{=} \mathcal{F}[f(x)] = \int_{-\infty}^{\infty} f(x) (\cos 2\pi ux - j \sin 2\pi ux) dx$$

- The **inverse Fourier transform** of $F(u)$ is:

$$f(x) \stackrel{\Delta}{=} \mathcal{F}^{-1}[F(u)] = \int_{-\infty}^{\infty} F(u) \exp(j2\pi ux) du$$

$$f(x) \stackrel{\Delta}{=} \mathcal{F}^{-1}[F(u)] = \int_{-\infty}^{\infty} F(u) (\cos 2\pi ux + j \sin 2\pi ux) du$$

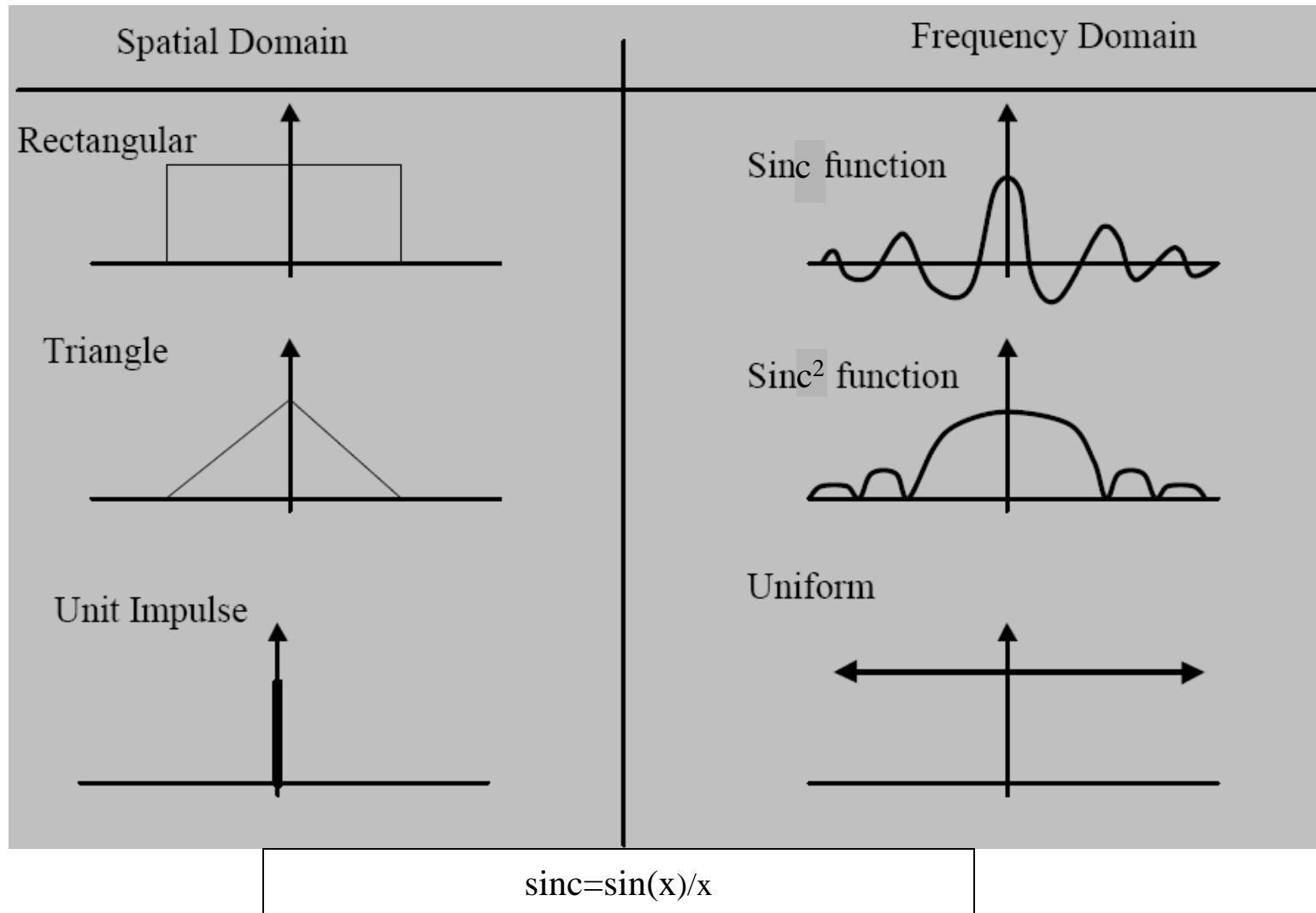
- Two dimensional case:

$$F(u, v) \stackrel{\Delta}{=} \mathcal{F}[f(x, y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp(-j2\pi(ux + vy)) dx dy$$

$$f(x, y) \stackrel{\Delta}{=} \mathcal{F}^{-1}[F(u, v)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp(j2\pi(ux + vy)) du dv$$



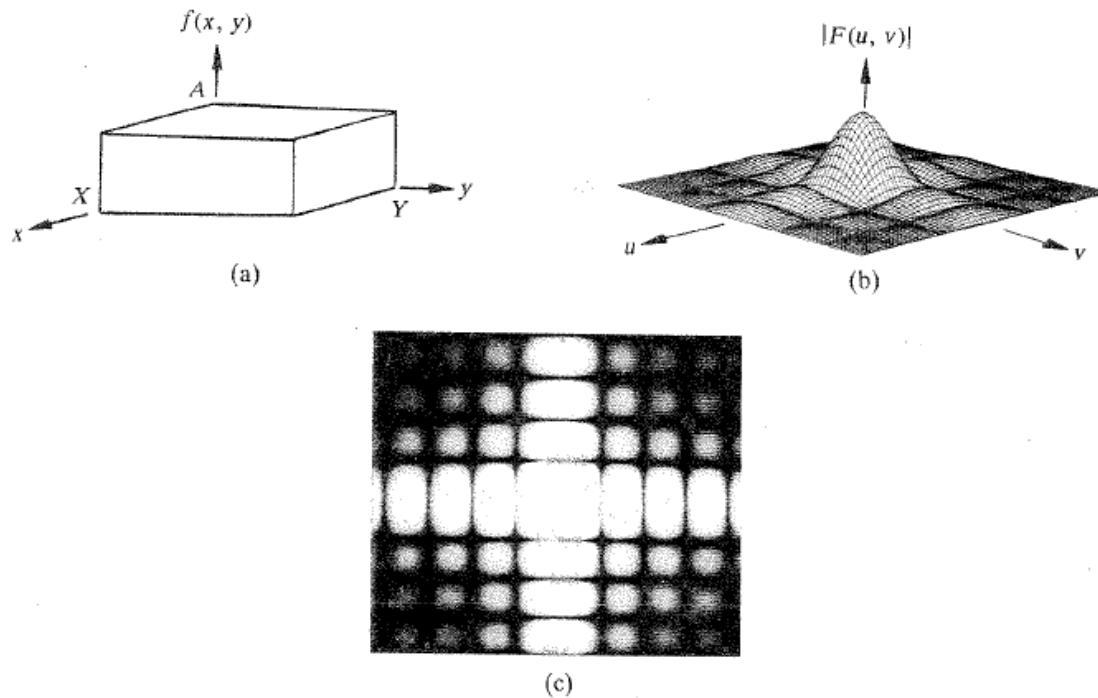
The Fourier Transform





The Fourier Transform

FT of a 2D rectangle function





Discrete Fourier Transform

- The Discrete Fourier Transform (DFT) is defined as:
- Forward DFT

$$F(u, v) \stackrel{\Delta}{=} \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux/M + vy/N)]$$
$$(u = 0, 1, \dots, M-1, v = 0, 1, \dots, N-1)$$

- Inverse DFT

$$f(x, y) \stackrel{\Delta}{=} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi(ux/M + vy/N)]$$
$$(x = 0, 1, \dots, M-1, y = 0, 1, \dots, N-1)$$



Properties of the Fourier Transform

Spatial frequencies: If $f(x,y)$ is luminance and x,y the spatial coordinates, then u, v are spatial frequencies representing luminance changes with respect to spatial distances. The units of u, v are the reciprocals of x and y .

Uniqueness: For continuous functions, $f(x,y)$ and $F(u, v)$ are unique with respect to one another.

Separability: The Fourier transform kernel is separable. Hence two dimensional Fourier transform can be realized by two one-dimensional transforms along the spatial coordinates.



Properties of the Fourier Transform

$$F(u, v) = \frac{1}{N^2} \sum_{x=0}^{N-1} e^{-j2\pi(\frac{ux}{N})} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(\frac{vy}{N})}$$

$$\sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(\frac{vy}{N})} = F(x, v)$$

$$F(u, v) = \frac{1}{N^2} \sum_{x=0}^{N-1} e^{-j2\pi(\frac{ux}{N})} F(x, v)$$



Properties of the Fourier Transform (2)

Linearity: The Fourier transform is a linear operation so that the Fourier transform of the sum of two functions is given by the sum of the individual Fourier transforms.

$$F \{a f(x,y) + b g(x,y)\} = a F(u,v) + b G(u,v)$$

Complex Conjugate: The Fourier transform of the Complex Conjugate of a function is given by

$$F \{f^*(x,y)\} = F^*(-u,-v)$$

where $F(u,v)$ is the Fourier transform of $f(x,y)$.

Forward and Inverse: We have that

$$F \{F(u,v)\} = f(-x,-y)$$

so that if we apply the Fourier transform twice to a function, we get a spatially reversed version of the function.

Similarly with the inverse Fourier transform we have that,

$$F^{-1} \{f(x,y)\} = F(-u,-v)$$

so that the Fourier and inverse Fourier transforms differ only by a sign.



Properties of the Fourier Transform (3)

Scaling: $f(a^*x) \longleftrightarrow (1/|a|) * F(u/a)$

This means that if you make the function wider in the x-direction, its spectrum will become smaller in the u-direction, and vice versa. The amplitude will also be changed.

Time Shifting (Translation in the spatial domain):

$$f(x - x_0, y - y_0) \longleftrightarrow F(u, v) e^{-j2\pi(\frac{ux_0+vy_0}{N})}$$

The only thing that happens if you shift the time, is a multiplication of the Fourier Transform with the exponential of an imaginary number, you won't see the difference of a time shift in the amplitude of the spectrum, only in the phase.

Frequency Shifting (Translation in the frequency domain):

$$f(x, y) \exp[-j2\pi(u_0x/N + v_0y/N)] \longleftrightarrow F(u-u_0, v-v_0)$$

This is the dual of the time shifting.



Properties of the Fourier Transform (4)

Differentials: The Fourier transform of the derivative of a function is given by: $\mathcal{F}\{d f(x)/dx\} = j2\pi u F(u)$

and the second derivative is given by:

$$\mathcal{F}\{d^2 f(x)/dx^2\} = -(2\pi u)^2 F(u)$$

$$\mathcal{F}\{df(x, y)/dx\} = j2\pi u F(u, v)$$

$$\mathcal{F}\{df(x, y)/dy\} = j2\pi v F(u, v)$$

$$\mathcal{F}\{\Delta f(x, y)\} = -(2\pi w)^2 F(u, v) \quad \text{where } w^2 = u^2 + v^2$$

Convolution theorem: $g(x, y) = h(x, y) * f(x, y)$ $G(u, v) = H(u, v)F(u, v)$

Inner product preservation: The inner product of two functions is equal to the inner product of their Fourier transforms. From this we obtain the **Parseval energy conservation formula**:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(x, y)|^2 dx dy = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |F(u, v)|^2 du dv$$



Fourier Transform

Fourier coefficients are generally complex numbers expressed as two components:

$$F(u,v) = R(u,v) + jI(u,v)$$

Typically these are converted into equivalent vectors represented by magnitude and phase angle:

$$F(u,v) = |F(u,v)| e^{j\theta(u,v)}$$

$$|F(u,v)| = \sqrt{R(u,v)^2 + I(u,v)^2} \quad - \text{Fourier spectrum}$$

$$\Theta(u,v) = \tan^{-1}(I(u,v)/R(u,v)) \quad - \text{Phase angle}$$

$$|F(u,v)|^2 = R(u,v)^2 + I(u,v)^2 = P(u,v) \quad - \text{Power spectrum of } f(x,y)$$



Fourier Transform

The $N \times N$ DFT is a cyclic of period N , in both u and v directions,

$$F(u + nN, v + mN) = F(u, v)$$

Most algorithms locate $F(0, 0)$ at top/left. So we can shift the $F(u, v)$ terms in two dimensions to give: $F(u, v)$ for $u & v = -N/2, \dots, 0, \dots, N/2-1$

This allows us to have $F(0, 0)$ to appear at the centre of the Fourier array.

For that a [translation in the frequency](#) domain has to be done for $u_0=N/2$ and $v_0=N/2$. From:

$$e^{j2\pi(\frac{\frac{N}{2}x+\frac{N}{2}y}{N})} = e^{j\pi(x+y)} = (-1)^{x+y}$$

$$f(x, y)(-1)^{x+y} \longleftrightarrow F(u - N/2, v - N/2)$$

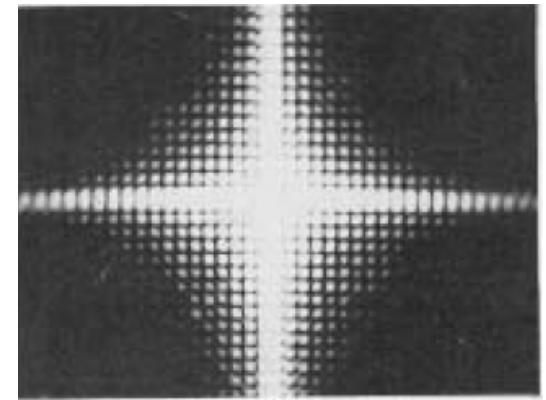
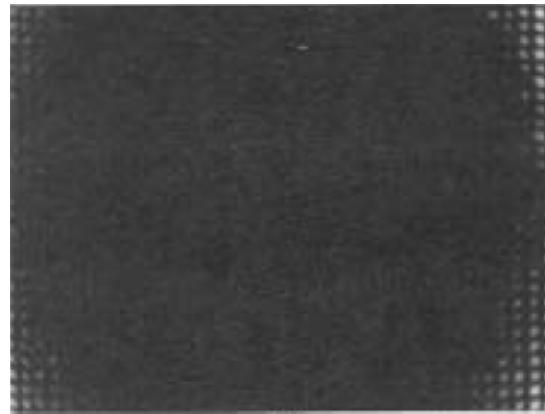
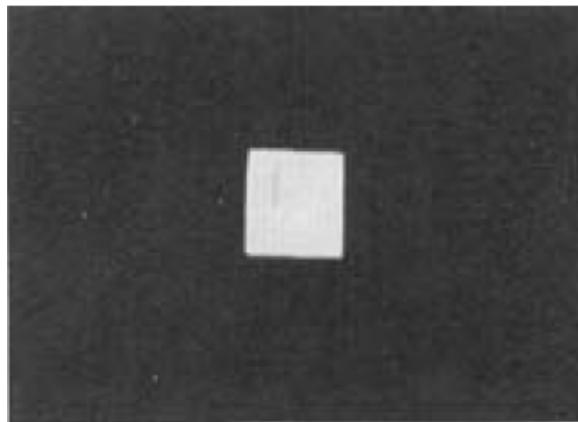
Typically we display $|F(u, v)|$ but the dynamic range is too high.

The solution is to scale the signal:

$$D(u, v) = c \log (1 + |F(u, v)|)$$

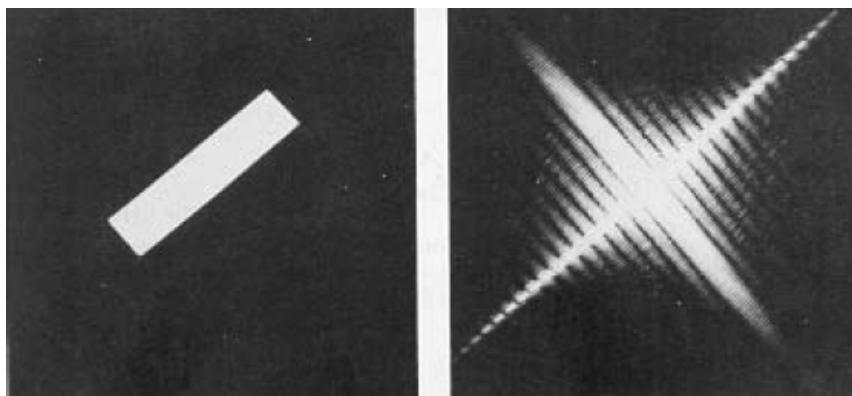
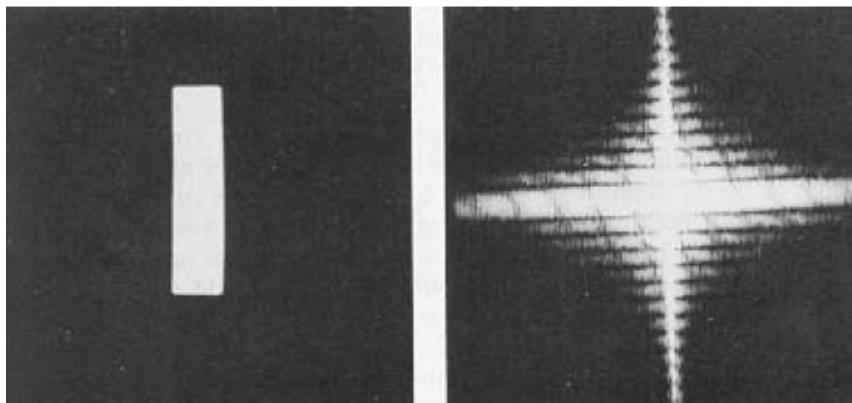


Fourier Transform





Fourier Transform



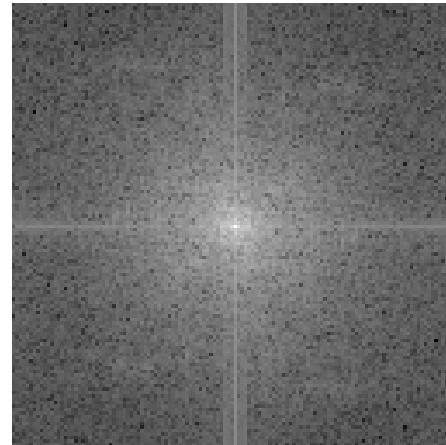


Fourier Transform

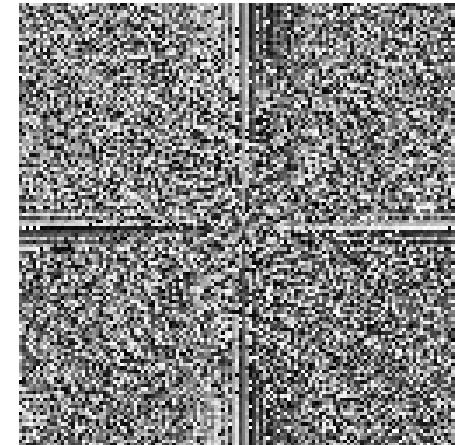
Direct Fourier Transform



Original

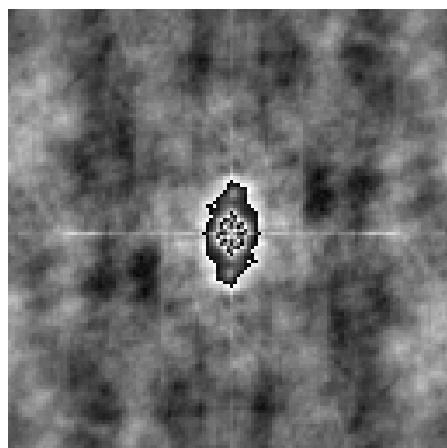


$\log(1+|F(u,v)|)$



$\phi(u,v)$

Invers Fourier Transform



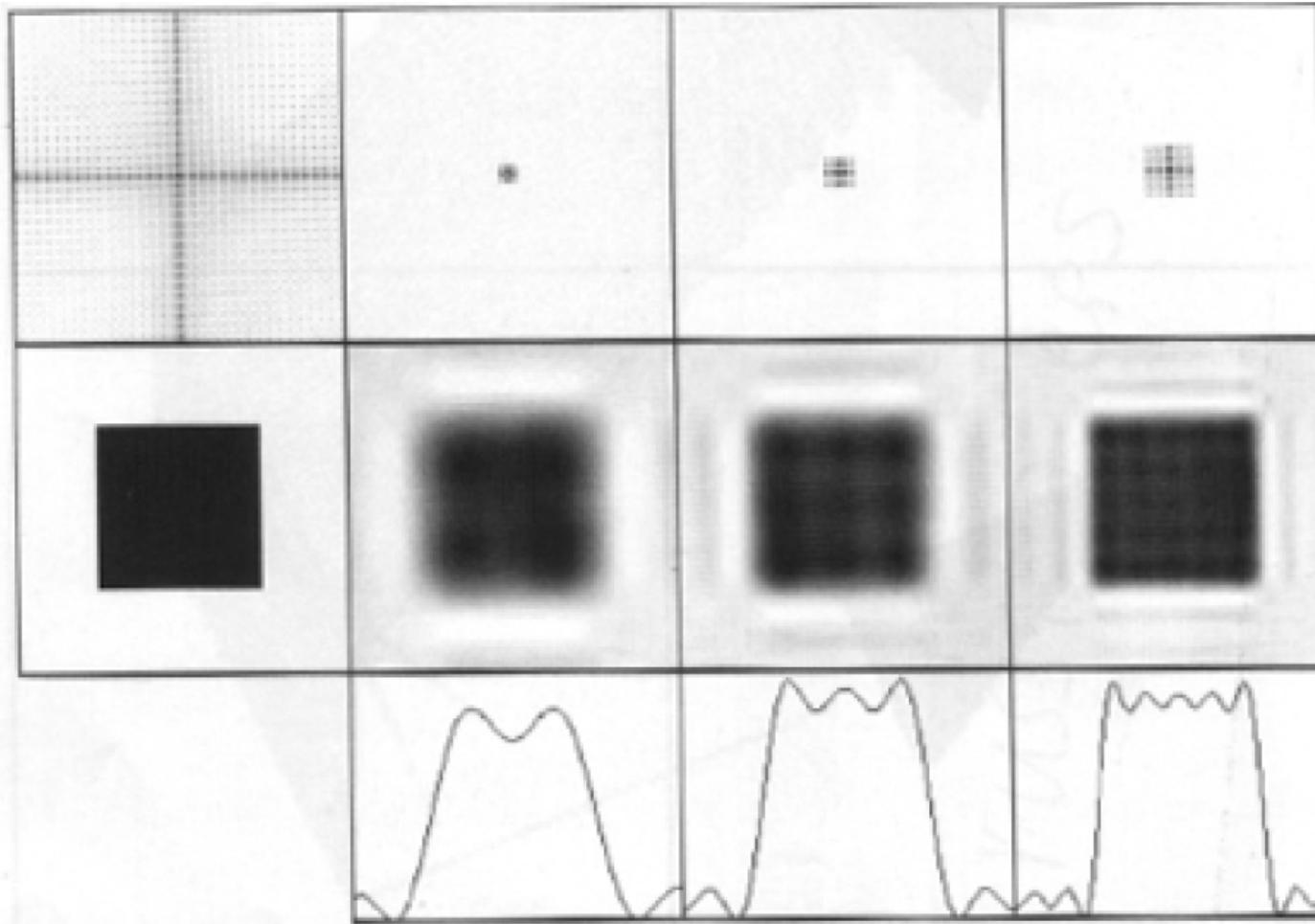
$\phi(u,v)=0$



$|A(u,v)|=0$



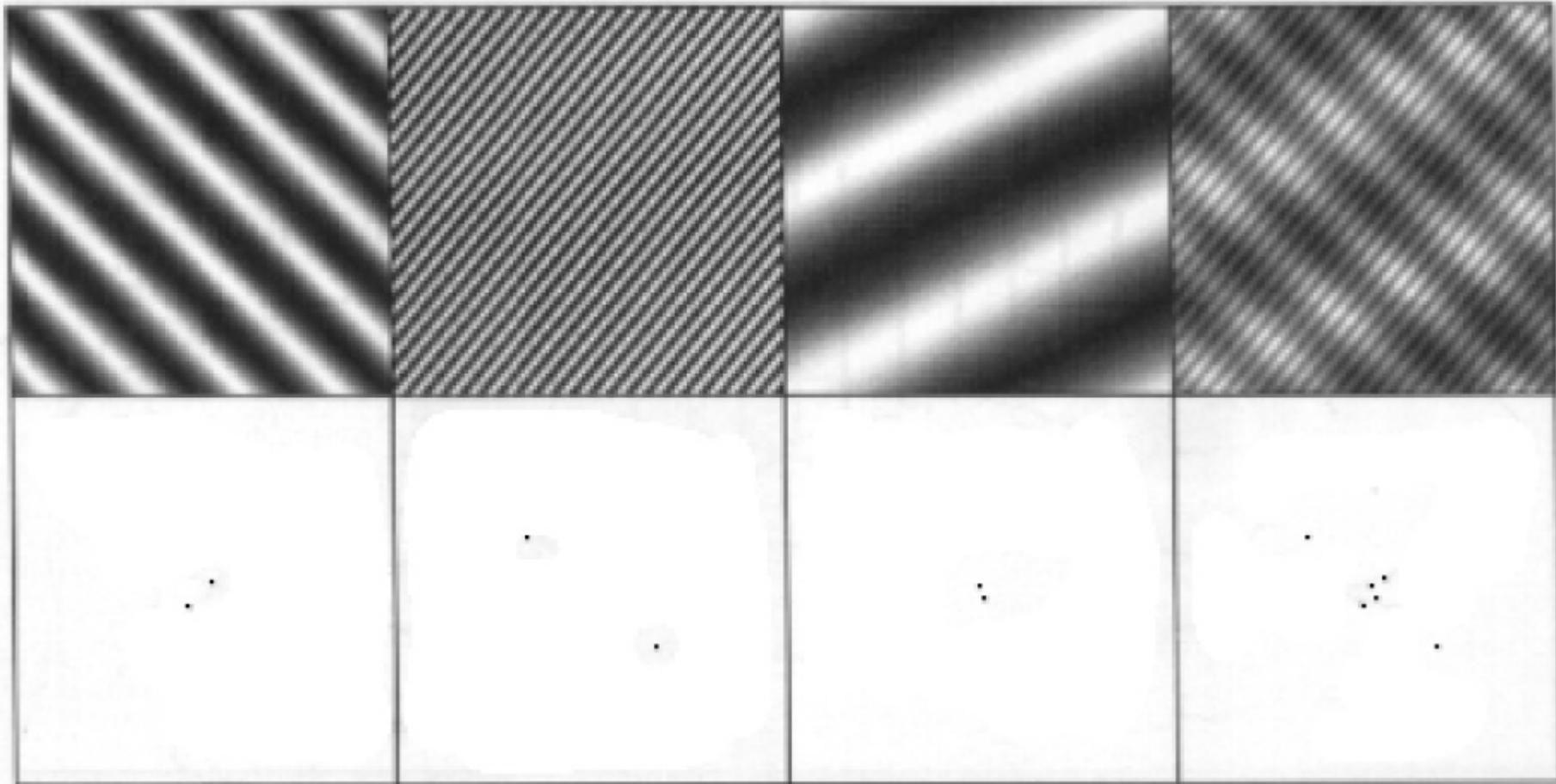
Fourier Transform



Images in the spatial domain are in the middle row, and their frequency space are shown on the top row. The bottom row shows the varying brightness of the horizontal line through the center of an image. The low frequency terms are on the center of the square, and terms with higher magnitude are on the outer edges.



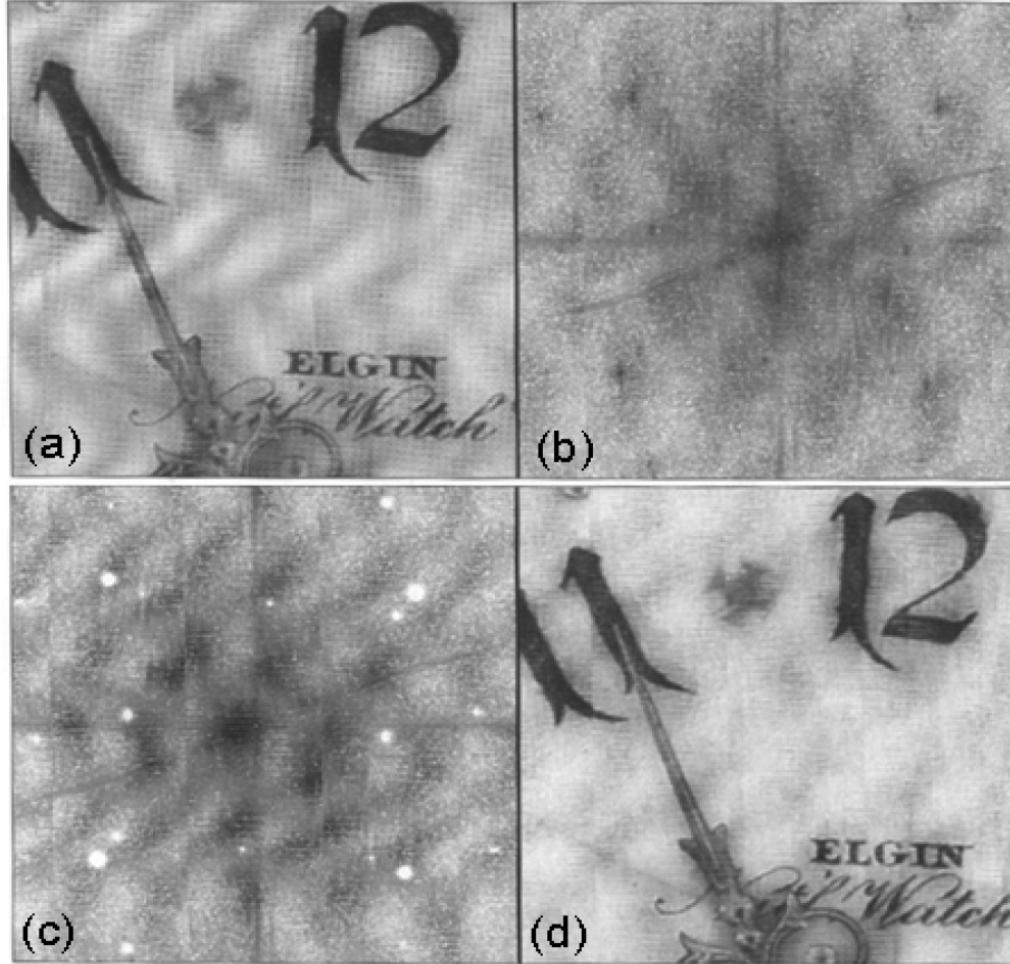
Fourier Transform



Images with perfectly sinusoidal variations in brightness: The first three images are represented by two dots. You can easily see that the position and orientation of those dots have something to do with what the original image looks like. The 4th image is the sum of the first three.



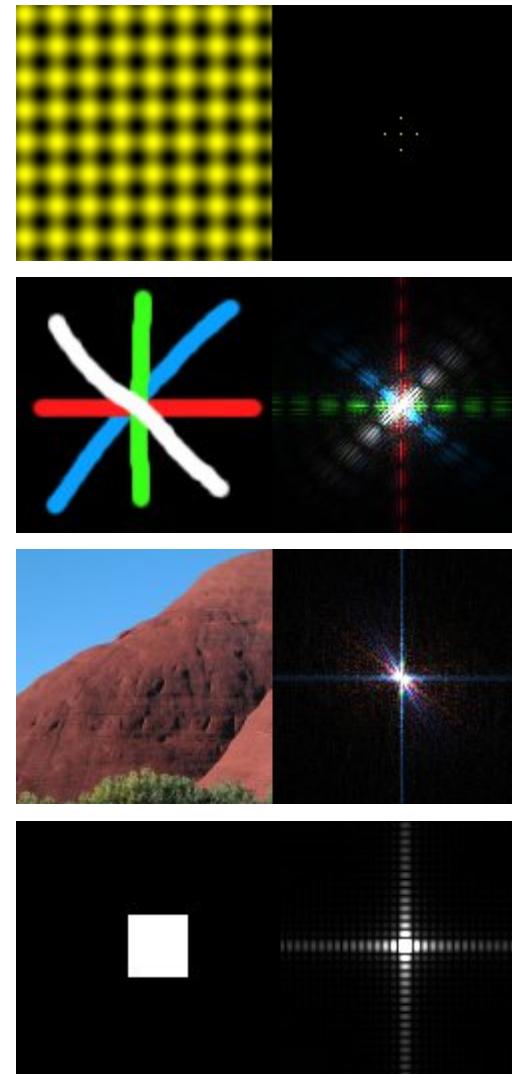
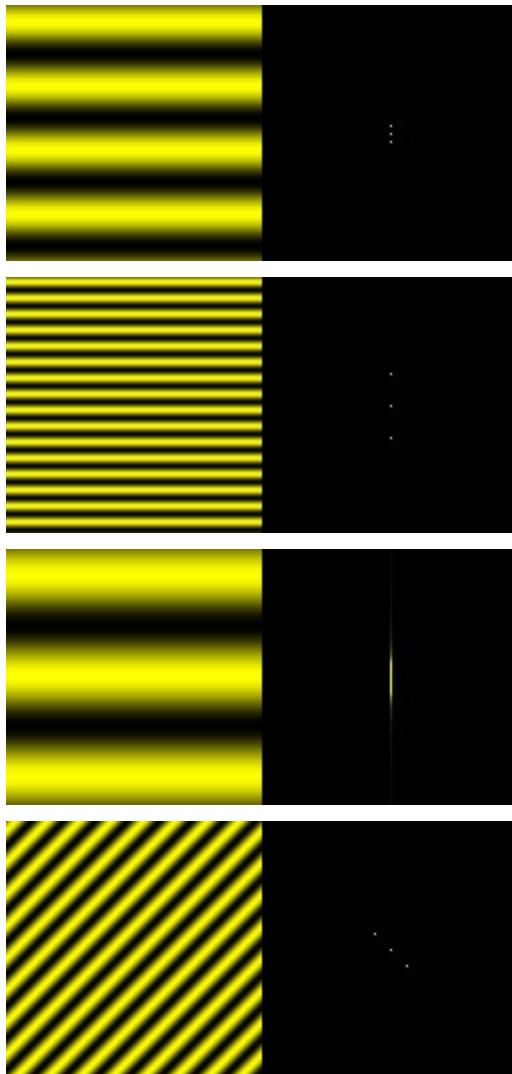
Fourier Transform



- (a) Dirty looking photo copied image. (b) The representation of image in the frequency space, i.e. the star diagram.
c) Those stars, however, do no good to the image, so we rub them out. (d) Reconstruct the image using (c) and those dirty spots on the original image are gone!



Fourier Transform





Fourier Transform

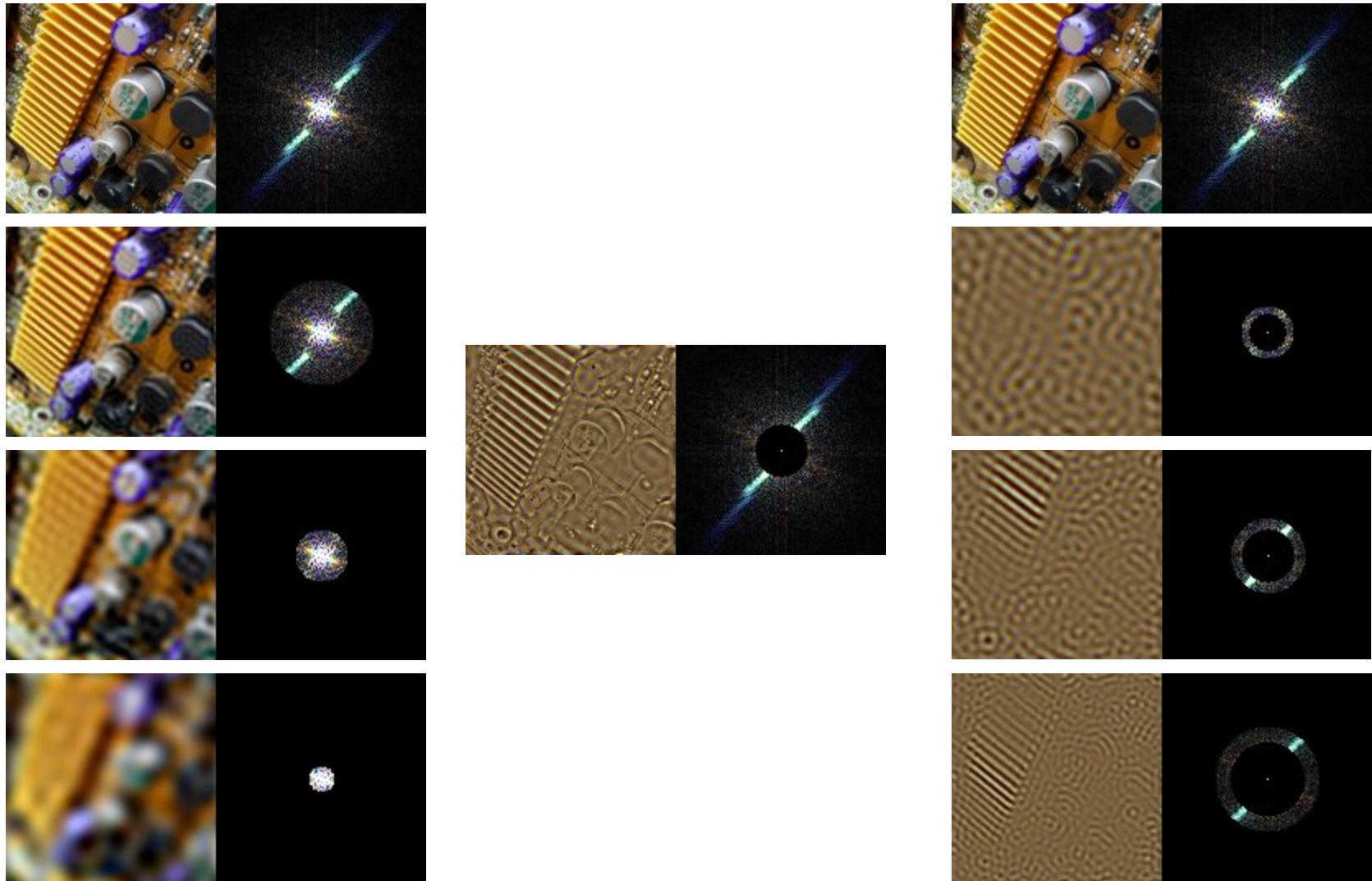


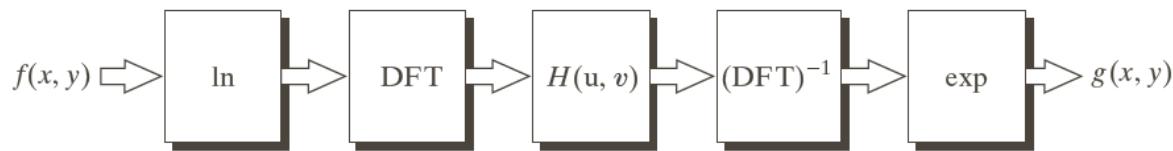
IMAGE PROCESSING

Technical University of Cluj Napoca
Computer Science Department



Fourier Transform

FIGURE 4.60
Summary of steps
in homomorphic
filtering.



Fourier Filtering Methods

Convert the spatial image into a frequency signal via the Fourier transform.

Modify frequencies as desired using a frequency filter.

Convert frequency image back to a spatial image using the inverse Fourier transform.



References

- <http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf>
- <https://www.ibiology.org/talks/fourier-transform/>
- <http://www.cs.uregina.ca/Links/class-info/425/Lab5 /index.html>



Topic 5: Noise in Images

Aim: Covers the origins of noise in image system. This includes fixed pattern noise from CCD systems, imaging striping and methods for removal. The Guassian noise model is detailed showing the approximations needed to obtain zero meaned additive signal independant Guassian noise. The baics propertis of additive noise and signal to noise ratio are introduced.

Contents:

- Introduction
- Data Drop-Out Noise
- Fixed Pattern Noise
- Detector or Shot Noise
- Properties of Additive Noise
- Signal to Noise Ratio
- Summary



Introduction

Consider all processes effecting the image **NOT** related to the object as being *Noise*

Range of origins:

1. Discrete nature of radiation
2. Detector sensitivity
3. Electrical noise
4. Film grain
5. Data transmission errors
6. Air turbulence
7. Image Quantisation

In this section we will consider some of the simple noise models, including data transmission errors, and intrinsic noise resulting from the discrete nature of radiation.

Data Drop-Out Noise

In many data transmission systems, random bits **corrupted or lost** on a data channel.

Typically appears as *snow* on images.



1 in 100 bits



1 in 20 bits

Corruption very common in satellite images and video systems where bits are set wrong,

Type of corruption is not correlated with the image data, and can be significantly reduced by;

- Threshold Average Filter
- Median Filter

With filtering, error rate of about 1.5%, can be removed without significant degradation of the image. These filters will be discussed in detail in Lectures 9-10.



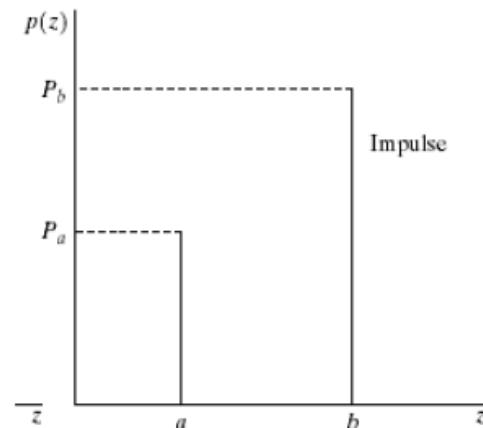
Salt and pepper noise

Impulse noise

- Sharp and sudden disturbances in image signals
- Randomly scattered white and black pixels in image
- Faulty image memory locations and impaired sensors
- An effective noise reduction method for this type of noise is a **median filter** or a **morphological filter**.

Impulse noise model

$$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$



If either $P_a=0$ or $P_b=0 \Rightarrow$ unipolar impulse noise

If $P_a \approx P_b \neq 0 \Rightarrow$ bipolar impulse noise or salt-and-pepper noise \Rightarrow normally, $a=0$ (black) and $b=255$ (white)





Salt and pepper noise

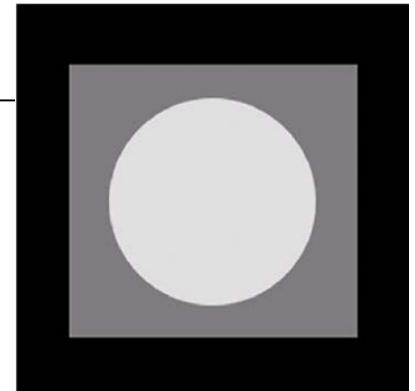
Original image



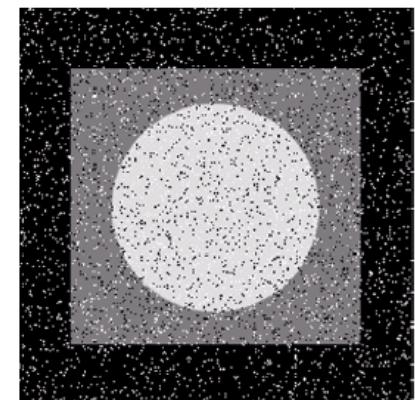
Add 'Salt and Pepper' Noise



Test pattern with
3 gray levels



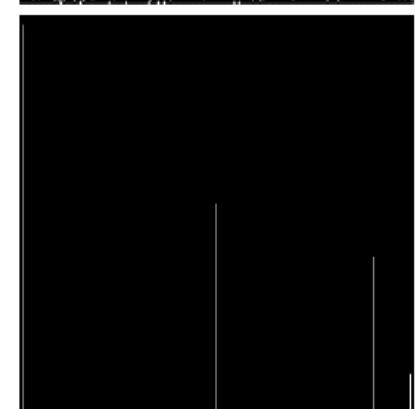
Test pattern with
salt and pepper
noise



After Noise Removal



Histogram of the
noisy test pattern



Salt & Pepper

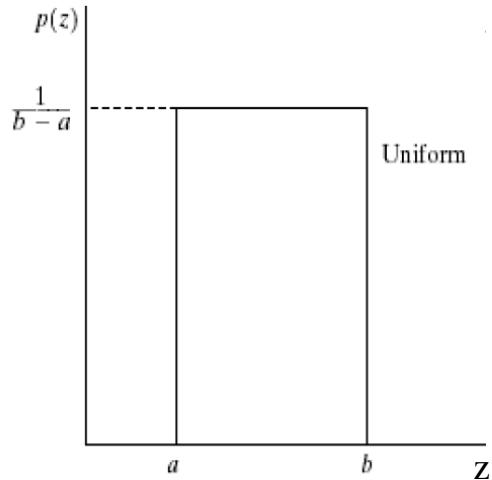


Uniform noise

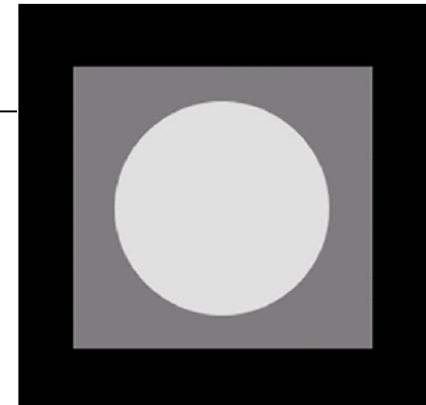
$$p(z) = \begin{cases} \frac{1}{b-a}, & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance are given by

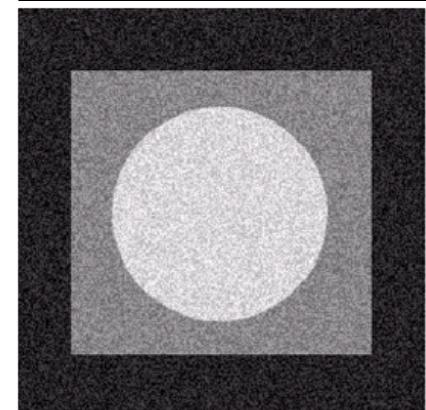
$$\mu = \frac{a+b}{2}, \quad \sigma^2 = \frac{(b-a)^2}{12}$$



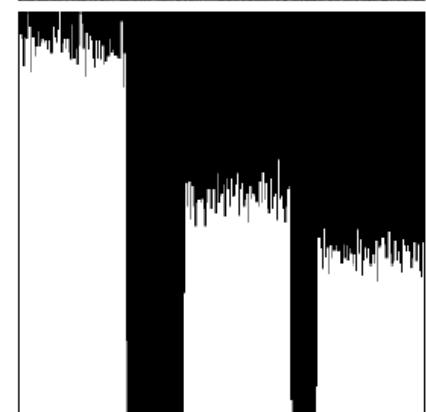
Test pattern with 3 gray levels



Test pattern with uniform noise



Histogram of the noisy test pattern



Fixed Pattern Noise

Two-Dimensional **CCD** systems, variable sensitivity of detectors. Typical problem on CCD sensors due to variability in manufacturing.

Noise can be corrected on a **point by point** basis by calibration of each sensor.



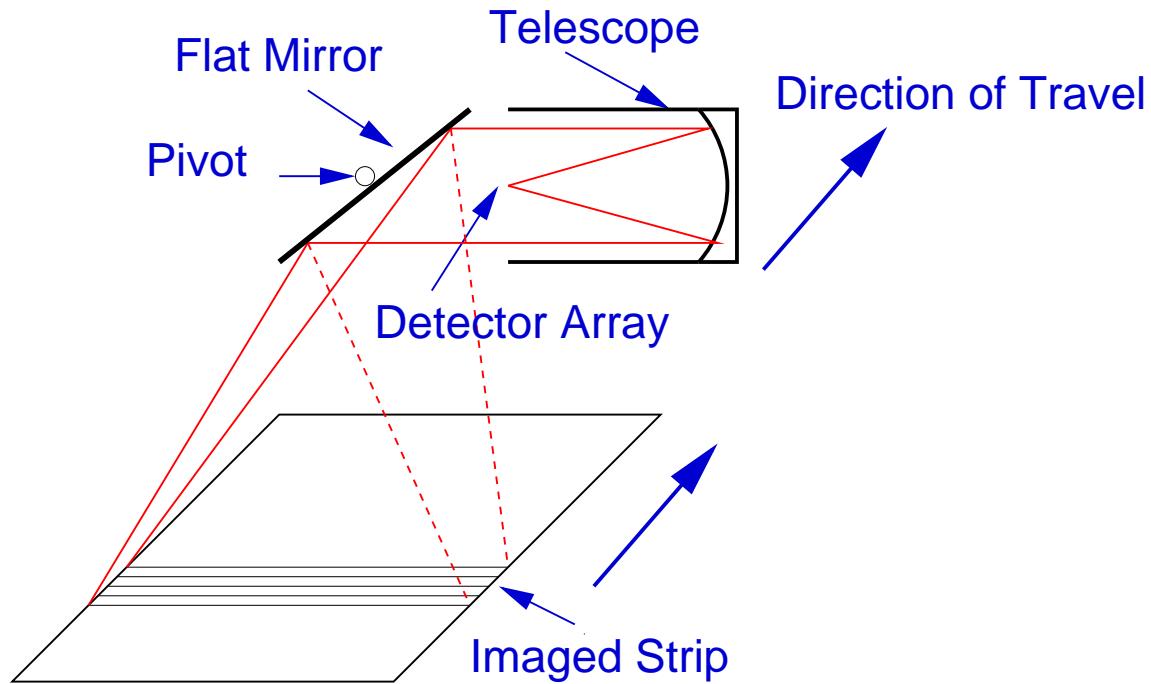
Typical fixed pattern noise from a CCD camera, (image stretch to about 5 grey levels).

Take measures at a range of light intensities, build-up sensitivity profile for each pixel point, use for corrections.

Typically only needed for critical applications, such as astronomy, or when sensor is very poor (Infra-red detector arrays).

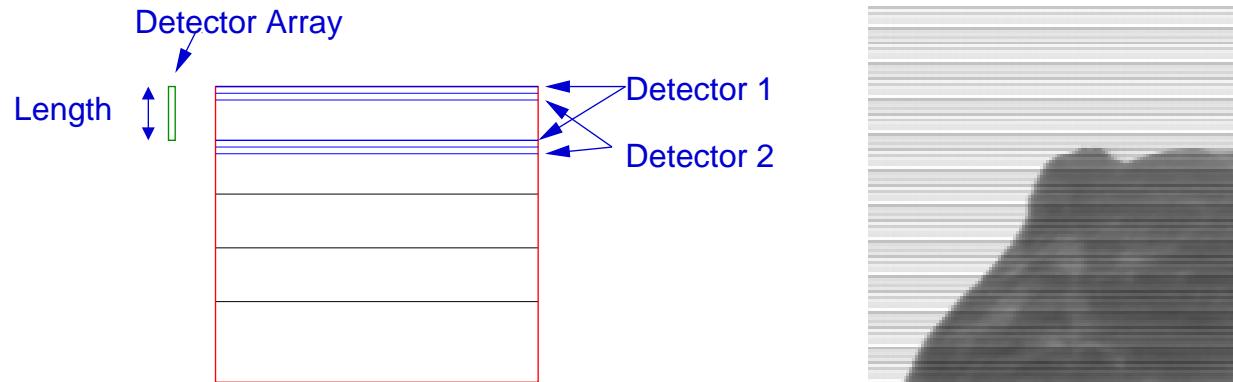
Satellite Image Striping

Many scanning systems use a one-dimensional



Satellite Image Striping I

Giving an image geometry of



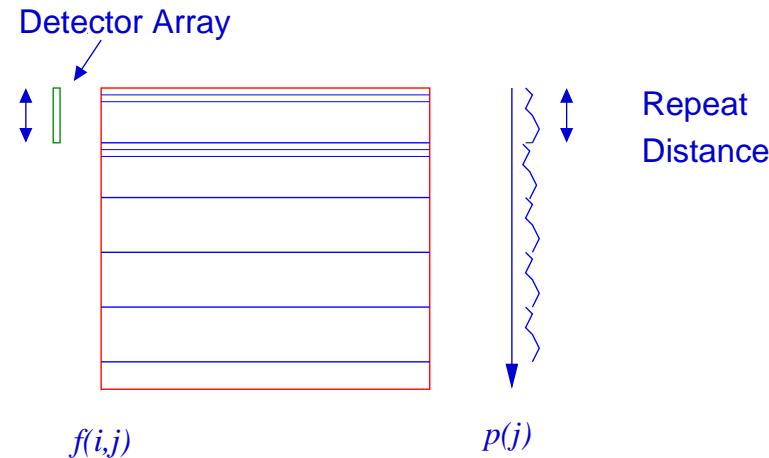
1-D scanning system show **striping** due to varying detector sensitivity.

Satellite Image Striping II

Form a projection,

$$p(j) = \sum_{i=0}^{N-1} f(i, j)$$

sensor of K pixels long, periodic structure of period K associated with detector sensitivity, so can be calculated and corrected for.

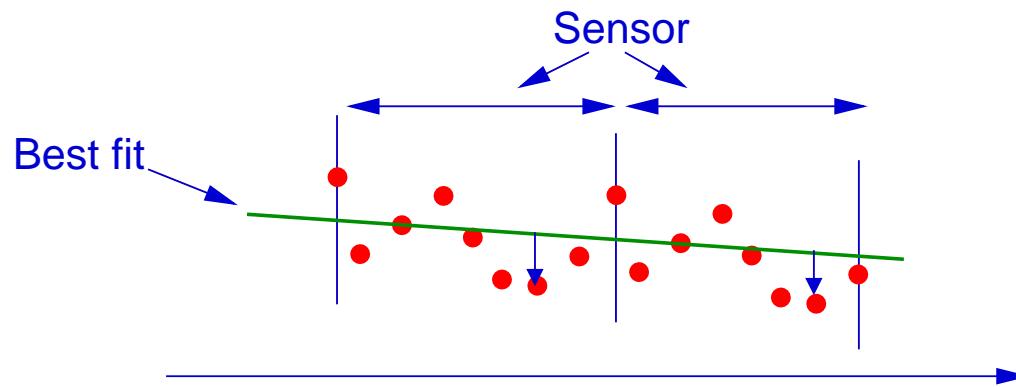


Satellite Image Striping II

If this is repeated on many images taken with the same sensor, able to obtain a good estimate of sensor variability.

- Additive error (charge leakage from CCD)
- Multiplicative error (gain variability in amplifier)
- General non-linear errors, need look-up-table for each detector.

Assume a that projection is smooth, deviation from best fit will give sensor partameters



Also possible: Fourier transform filtering, information causing striping at particular spatial frequency, so can be removed by suitable Fourier filters.



Statistical Method

If an image is scaled by a multiplicative factor a and additive term

$$g(i, j) = af(i, j) + b$$

we have that

$$\langle g \rangle = a \langle f \rangle + b$$

and

$$\sigma_g^2 = a^2 \langle f^2 \rangle - a^2 \langle f \rangle^2 = a^2 \sigma_f^2$$

If we have a detector with K elements, we effectively detect K images

$$g_k = a_k f + b_k$$

with each detected image being of size $N/K \times N$, being a reduce resolution version of $f(i, j)$.

The image statistics on mean and variance do not depend on the resolution, so and

$$\langle g_k \rangle = a_k \langle f \rangle + b_k \quad \text{and} \quad \sigma_{g_k}^2 = a_k^2 \sigma_f^2$$

so allowing us to solve for a_k and b_k .

Once you have the a_k and b_k you can easily reform the real image $f(i, j)$ from the subimages.

Basis of the DIA Programming task



Detector or Shot Noise

Noise process *intrinsic* to the process of measurement, not related to an imaging system.

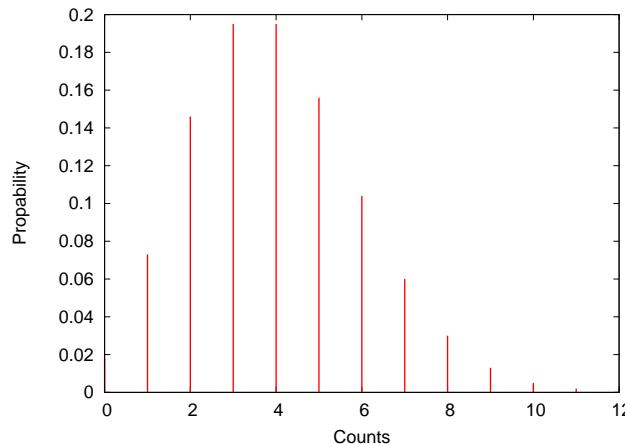
All imaging systems actually count *particles*, (electrons or photons), which are governed by statistical and physical laws.

For a source of average brightness $\langle \mu \rangle$ *expected* observed value is

$$\langle f \rangle = \Delta t \langle \mu \rangle$$

However a single observation is random variable from the PDF

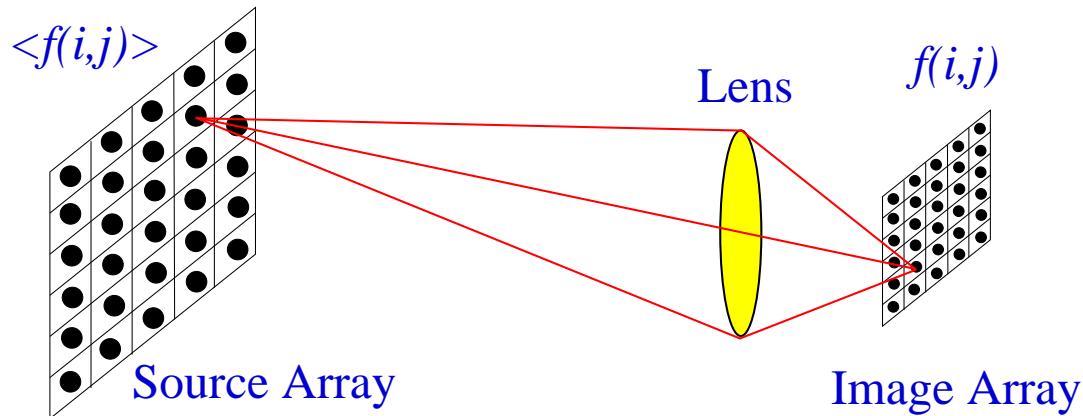
$$p(f) = \frac{\langle f \rangle^f \exp(-\langle f \rangle)}{f!}$$



PDF of a poisson distribution with $\langle f \rangle = 4$

Two-Dimensional Image

For the 2-D case we have, we assume that we have a two-dimensional array of sources.

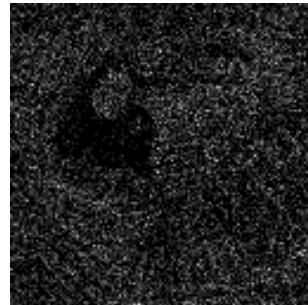


Source brightness of: $\langle f \rangle(i, j)$ then PDF for each pixel,

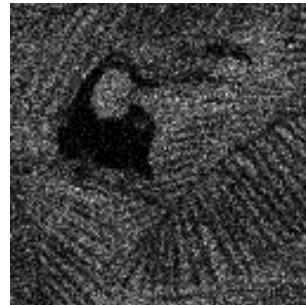
$$p(f(i, j)) = \frac{\langle f \rangle(i, j)^{f(i, j)} \exp(-\langle f \rangle(i, j))}{f(i, j)!}$$

Simulated Example

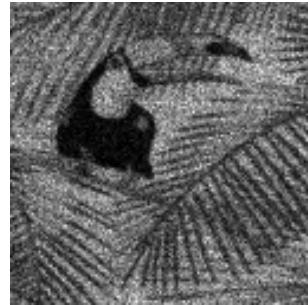
Digital simulation with average number of photons per pixel specified.



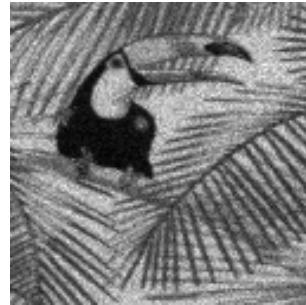
1 Photons/Pixel



4 Photons/Pixel



16 Photons/Pixel



64 Photons/Pixel

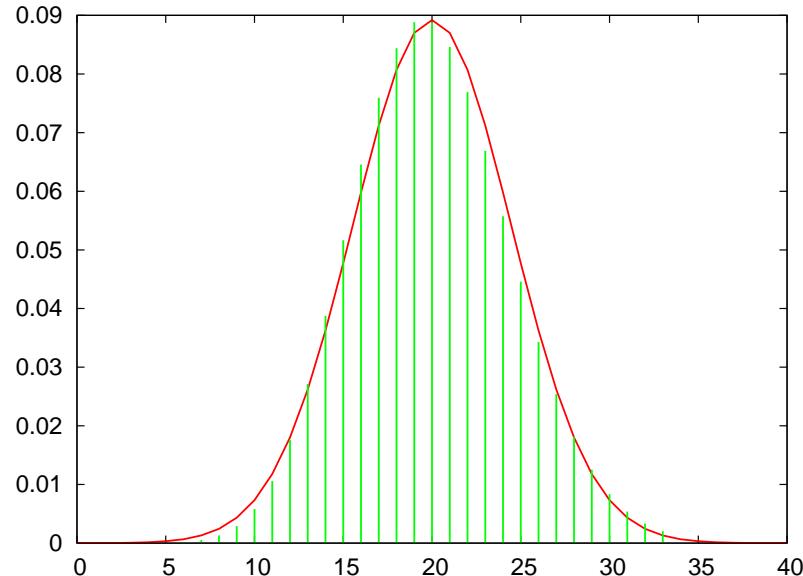
Low numbers of photons, noise dominates, but as number increase the image become usable.

Gaussian Approximation

The Poisson distribution is mathematically difficult, (discrete distribution).

For large **expected** values, this may be approximated by a Gaussian of **mean u** and **halfwidth $\sqrt{2u}$**

$$p(n) = \frac{u^n \exp(-u)}{n!} \rightarrow \frac{1}{(2\pi u)^{1/2}} \exp\left(\frac{-(n-u)^2}{2u}\right)$$



This has an error of $\approx 1\%$ for $u > 20$.

Gaussian Approximation I

So for mean of $\langle f \rangle$, approximate $p(f)$ by

$$p(f) = \frac{1}{(2\pi\langle f \rangle)^{1/2}} \exp\left(\frac{-(f - \langle f \rangle)^2}{2\langle f \rangle}\right)$$

So if we Regard the measured value f as

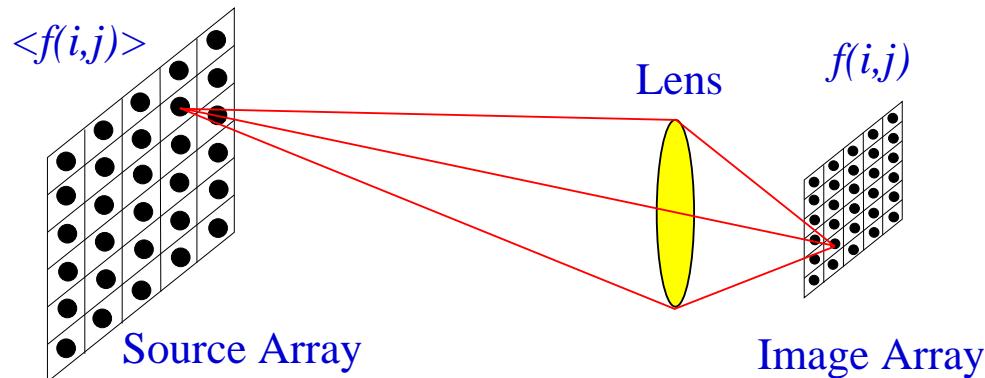
$$f = \langle f \rangle + n$$

where n is the noise so the PDF of the noise is,

$$p(n) = \frac{1}{(2\pi\langle f \rangle)^{1/2}} \exp\left(\frac{-n^2}{2\langle f \rangle}\right)$$

Two-Dimensional Image

For the 2-D case we have, we assume that we have a two-dimensional array of **sources**.



Source brightness of: $\langle f \rangle(i, j)$ then we measure,

$$f(i, j) = \langle f \rangle(i, j) + n(i, j)$$

where the PDF of the **noise** is given by.

$$p(n(i, j)) = \frac{1}{(2\pi\langle f \rangle(i, j))^{1/2}} \exp\left(\frac{-n(i, j)^2}{2\langle f \rangle(i, j)}\right)$$

This is an **additive noise model**, where the PDF of the noise **depends** on the signal. Known as

Signal Dependent Additive Noise.

This model assumes the imaging system is space invariant and linear.

Low Contrast Approximation

If we assume the image is *Low Contrast* so

$$\langle f \rangle(i, j) \approx \text{const} = \mu$$

we have that

$$p(n(i, j)) = \frac{1}{(2\pi\mu)^{1/2}} \exp\left(\frac{-n(i, j)^2}{2\mu}\right)$$

which is **independent** of the structure of the image.

Additive noise, with PDF being **Zero Mean Gaussian** with variance μ , the **mean** of the expected image value,

$$\mu = \langle \langle f \rangle(i, j) \rangle$$

This is the typical assumption for image noise models which are

Signal Independent Additive Gaussian

This assumes imaging system is:

- Linear and Space Invariant
- High brightness (many photons/electrons)
- Low contrast

for this noise model to be valid.



Validity of Additive Noise

In taking *additive* signal independent model, we assume *High brightness & Low contrast*.

Assumption valid in many imaging systems,

1. Video images (Many thousands on photons/electrons)
2. Infra-red (Usually very low contrast)
3. Electron microscope (Usually very low contrast)
4. CT and MRI Medical imaging (low contrast)

Not valid in,

1. γ -camera (small count number)
2. Astronomical images (small count number **and/or** very high contrast).
3. Image intensifier systems (small count number)
4. High magnification electron microscope (small count number)

If the additive signal independant noise model is not valid, processing is much more difficult.

Most processing used additive signal independant noise model, even if it is not *really* valid.



Properties of Additive Noise

Take the additive noise model as:

$$f(i, j) = s(i, j) + n(i, j)$$

where $s(i, j)$ is signal and $n(i, j)$ is noise.

The PDF of $n(i, j)$ is zero mean Gaussian so:

$$\langle n(i, j) \rangle = 0 \quad \text{and} \quad \langle |n(i, j)|^2 \rangle = \sigma_n^2$$

so that

$$\langle s(i, j) \rangle = \langle f(i, j) \rangle = \mu$$

Noise is independent of signal $s(i, j)$, so un-correlated, mathematically that:

$$\langle s(i, j)n(i, j) \rangle = 0$$

so the variance of $f(i, j)$ is:

$$\sigma_f^2 = \left\langle |f(i, j) - \langle f(i, j) \rangle|^2 \right\rangle$$

By substitution, and above properties, this can be expanded to give

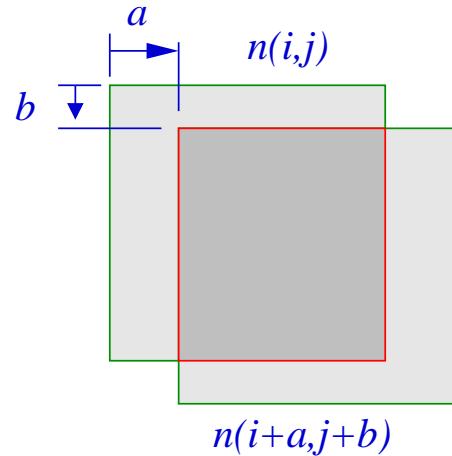
$$\sigma_f^2 = \sigma_s^2 + \sigma_n^2$$

where σ_s^2 is the Variance of the signal.

So the addition of noise alters the Variance but not the Mean.

Properties of Additive Noise I

We have assumed that each image point is independent, so the noise is not correlated **with-itself**,



mathematically this means:

$$n(i, j) \otimes n(i, j) = \delta_{i,j} \langle |n(i, j)|^2 \rangle = \delta_{i,j} \sigma_n^2$$

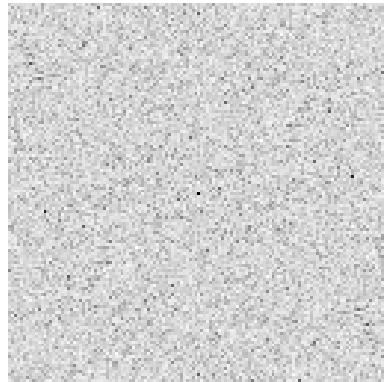
so the Auto-correlation is a **δ -Function**.

From the (auto)-correlation theorem, the Fourier Transform of the **Auto-correlation** is the **Power Spectrum**, so:

$$|N(k, l)|^2 = \text{Constant}$$

Properties of Additive Noise I

Known as *White Noise* with equal power at all spatial frequencies.



Power Spectrum of single realisation not actually constant, but equal power over each region of Fourier space.

Parseval's Theorem

We have that the power in real space and Fourier space is the same, so that

$$\langle |N(k, l)|^2 \rangle = \langle |n(i, j)|^2 \rangle = \sigma_n^2$$

Processing of Noisy Images

In Fourier space we have that

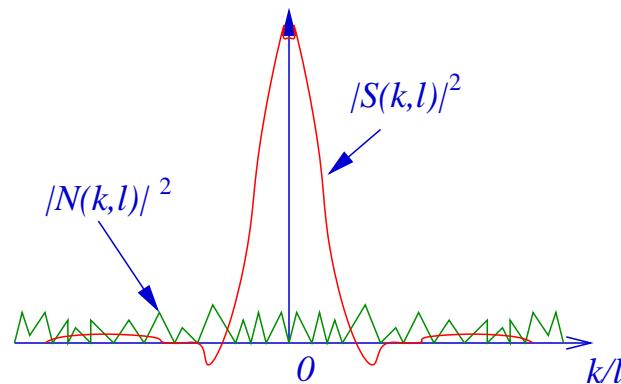
$$F(k, l) = S(k, l) + N(k, l)$$

we know that

- $S(k, l)$ = Sharply peaked about low spatial frequencies
- $N(k, l)$ = Constant at all spatial frequencies

so that when

- $|S(k, l)| \gg |N(k, l)|$ Little effect
- $|S(k, l)| \approx |N(k, l)|$ Signal corrupted



Greatest effect at high spatial frequencies where $S(k, l)$ is small. So high spatial frequencies corrupted by the noise.



Processing of Noisy Images I

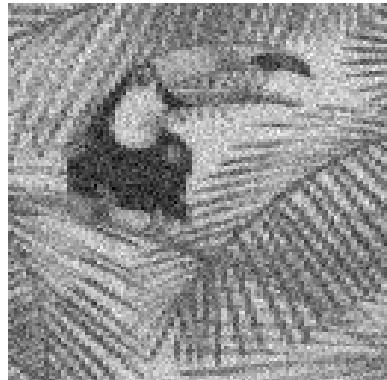
Reduce effect of noise by Low-Pass filtering

- Fourier low pass
- Real Space averaging
- Median filter

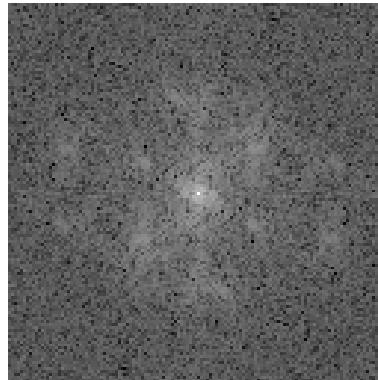
all of which remove or reduce high spatial frequencies, thus the effect of noise.

Details in next two lectures.

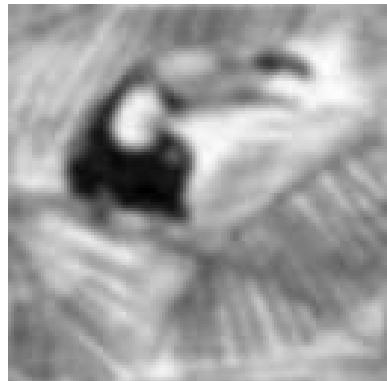
Digital Example



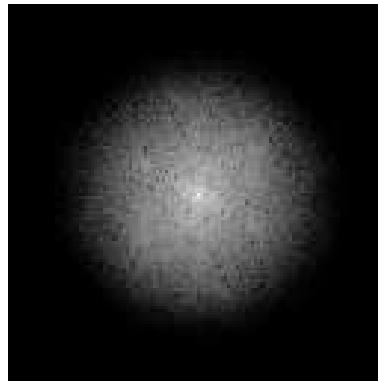
Noisy Image



Fourier Transform



Low-pass filtered



Fourier Transform

Signal to Noise Ratio

The use of **SNR** (Signal to Noise Ratio) is a confusing topic, since

- There is a range (about 10) definitions.
- Can a single number really classify **how good an image is**.
- In practice image quality is very strongly image dependant.

For signal independent additive noise,

$$f(i, j) = s(i, j) + n(i, j)$$

Signal & Noise variances are:

$$\begin{aligned}\sigma_s^2 &= \langle |s(i, j) - \langle s(i, j) \rangle|^2 \rangle \\ \sigma_n^2 &= \langle |n(i, j)|^2 \rangle\end{aligned}$$

Define SNR by

$$\text{SNR} = \frac{\sigma_s}{\sigma_n}$$



Signal to Noise Ratio I

Noting that the signal and the noise are uncorrelated, we have

$$\sigma_f^2 = \sigma_s^2 + \sigma_n^2$$

so we can write the SNR as:

$$\text{SNR} = \sqrt{\frac{\sigma_f^2}{\sigma_n^2} - 1}$$

To calculate **SNR**, need 2 of σ_s , σ_f , or σ_n .

Calculation of SNR for Single Image

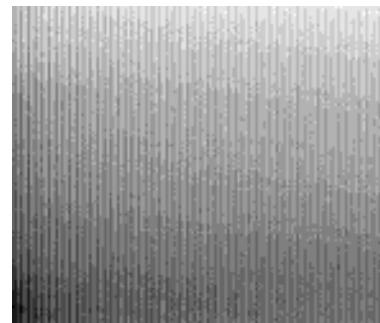
From single image can **only** find σ_f .

If **region** of image with **NO** signal can estimate σ_n from that region.

This method works for an image that contains a large region of “water” or “sky” where there is no signal.



Whole Image



Piece of Sky

Sky region shows typical CCD array fixed pattern noise.

Calculated values are $\sigma_f^2 = 5287$ and $\sigma_n^2 = 1.85$, so that

$$\text{SNR} \approx 53.4$$

Calcualte SNR for Multiple Images

Assume that we have **two** realisation of same scene, so:

$$\begin{aligned} f(i, j) &= s(i, j) + n(i, j) \\ g(i, j) &= s(i, j) + m(i, j) \end{aligned}$$

which is equivalent to two image of the same scene take at different times.

The noise in each realisation have the same PDF, so that

$$\begin{aligned} \langle n(i, j) \rangle &= \langle m(i, j) \rangle = 0 \\ \langle |n(i, j)|^2 \rangle &= \langle |m(i, j)|^2 \rangle = \sigma_n^2 \end{aligned}$$

The two noise realisations were measured at *different* times, so they are uncorrelated, so:

$$\langle n(i, j) m(i, j) \rangle = 0$$

In both cases the noise is uncorrelated with the signal, so that

$$\langle s(i, j) n(i, j) \rangle = \langle s(i, j) m(i, j) \rangle = 0$$



Calcualte SNR for Multiple Images I

If we then define the *Normalised* Correlation between $f(i, j)$ and $g(i, j)$ as:

$$r = \frac{\langle (fg - \langle f \rangle \langle g \rangle) \rangle}{[\langle |f - \langle f \rangle|^2 \rangle \langle |g - \langle g \rangle|^2 \rangle]^{1/2}}$$

Then by expanding, collecting terms, and cancelling all zero terms, *it-can-be-shown* that

$$r = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2}$$

so we can form the SNR from

$$\text{SNR} = \sqrt{\frac{r}{1-r}}$$

So allowing direct calculation of the SNR independant of the type of the signal.



Calcualte SNR for Multiple Images II

If there is **more than two realiastration** available a better estimate for SNR can be found by forming the normalised correlation between pairs of images and averaging.

In practice this measure of SNR **looks about right** for most images. For examples:

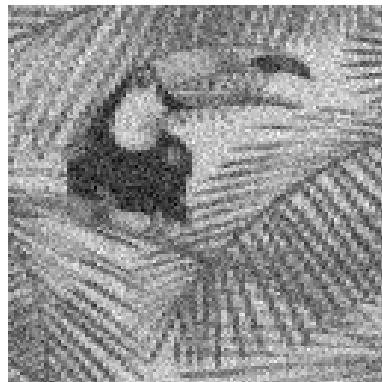
- SNR > 20 Little visible noise
- SNR ≈ 10 Some noise visible
- SNR ≈ 4 Noise clearly visible
- SNR ≈ 2 Image severly degraded
- SNR ≈ 1 Is there an image?

Digital Simulation

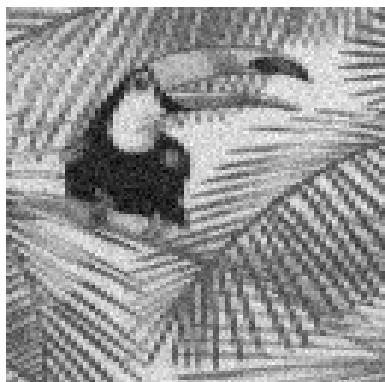
Digitally simulated “noisy” images:



SNR = 1



SNR = 2



SNR = 4



SNR = 8

Images formed by addition of Gaussian random noise.



Summary

In this section we have considered:

- Types of noise.
- Data drop-out noise
- Fixed pattern noise for two-dimensional CCDs
- Fixed pattern noise of one-dimensional sensors and how to correct.
- Detector or shot noise and Gaussian approximation.
- Properties of additive noise and underlying assumptions.
- Signal to Noise Ratio and methods of measuring.



References

- <https://www2.ph.ed.ac.uk/~wjh/teaching/dia/documents/noise.pdf>
- <https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture4.pdf>



Image Processing

(Year III, 2-nd semester)

Lecture 9-10: Digital filtering



Contents

- **Linear Digital Filtering**
 - Fourier Space Filters
 - Real Space Filters
 - Use of Linear Filters
- **Non-Linear Filters**
 - Real Space Non-Linear Filters
 - Homomorphic Filtering
- **Summary**



Linear Digital Filtering

- Main image processing operation, used for 90% of image processing operations.
- Objective is to **Convolve** image $f(i,j)$, with filter function $h(i,j)$
 - In Real Space: $g(i,j) = h(i,j) \odot f(i,j)$
 - In Fourier Space (using the convolution theorem) :
$$G(k,l) = H(k,l)F(k,l)$$
- This operation can be performed in Real **OR** Fourier space. Mathematical operation is identical, but computational cost varies.
- The operation of the filter is controlled by the convolution kernels or filter functions
 - $h(i,j)$ - In real space
 - $H(k,l)$ - In Fourier space



Fourier Space Convolutions

- The result of a Fourier space convolution can be converted in spatial domain by an inverse Fourier transform:

$$g(i,j) = F^{-1}\{H(k,l)F(k,l)\}$$

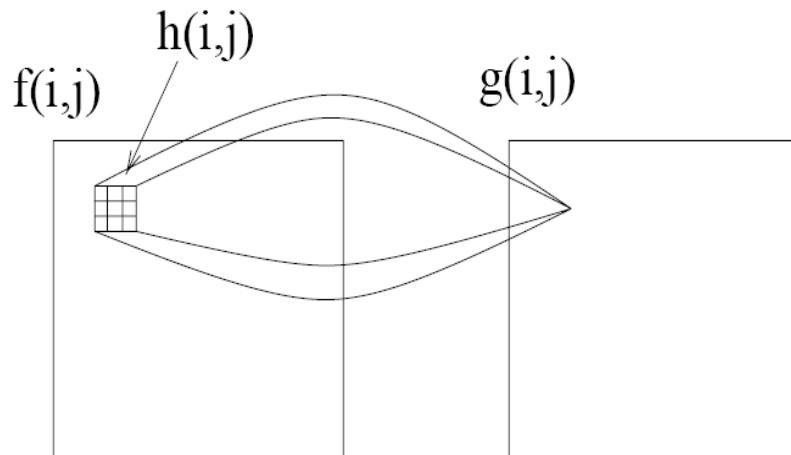
- Processing requires **TWO** FTs (one DFT and one IFT) and a complex multiply, (a third DFT required if $H(k,l)$ is formed from $h(i,j)$).
- **Note:** FTs and ‘x’ must be performed in floating point format.
- Computational time independent of filter type.



Real Space Convolutions

- For filter $h(i,j)$ of size $M \times M$:

$$g(i, j) = \sum_{m,n=-M/2}^{M/2} h(m, n) f(i - m, j - n)$$

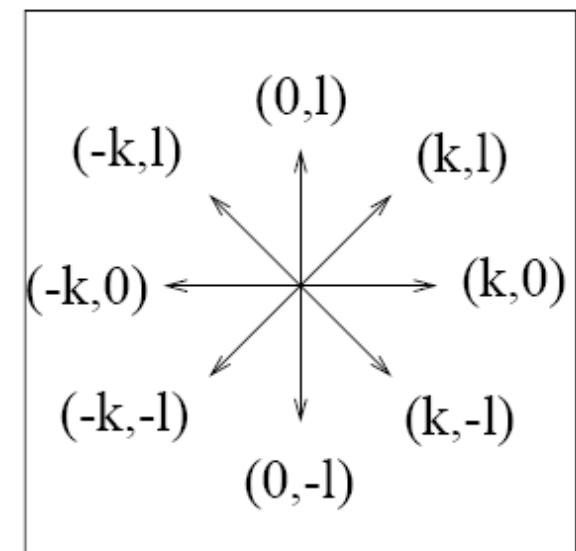


- “shift, multiply and accumulate” scheme
- Computation time $\sim M^2$
- All calculations can be integer or byte.
- Filter to 5x5 available in custom hardware at video rates.
- For serial machines, filters bigger than 9x9, are typically faster by Fourier space technique.



Fourier Space Filters

- Filtering operation applied to $F(k,l)$ and determined by $H(k,l)$ → filtered Fourier domain image $G(k,l)$.
- Most applications input & output images are **REAL**.
- The Fourier transform of a real image $F(k,l)$ is complex and obeys the familiar properties of
 - *Real part symmetric* and
 - *Imaginary part anti-symmetric*
- To obtain a real output the Fourier filter $H(k,l)$ must also obey these symmetry relations
- In practice $H(k,l)$ is Real and Symmetric





Low Pass Filter

- Low pass filters allow LOW spatial frequencies to pass while attenuating or blocking HIGH spatial frequencies. Used extensively in the Reduction of Noise.
- Low pass filters:
 - Ideal
 - Gaussian
 - Smooth: Butterworth, Trapezoidal



Ideal Low Pass Filter

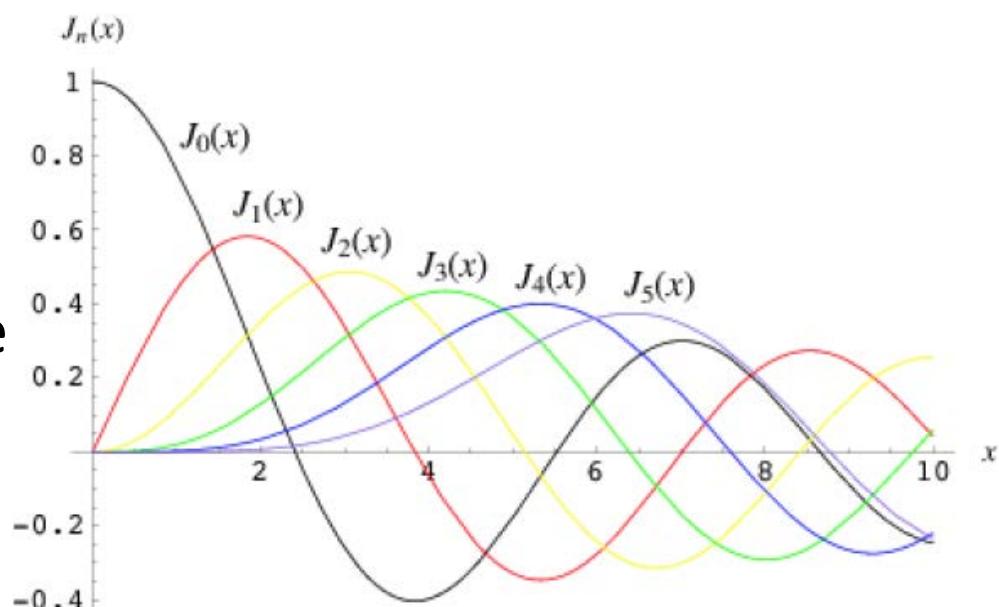
- Block all frequencies greater than some limit:

$$H(k, l) = \begin{cases} 1, & k^2 + l^2 \leq w_0^2 \\ 0, & \text{otherwise} \end{cases}$$

- The ideal low pass filter in spatial domain:

$$h(i, j) = \frac{J_1(r / w_0)}{r / w_0} \quad \text{with } r^2 = k^2 + l^2$$

where J_1 is a Bessel function



- Which results in “ringing” effects in the output image $g(x, y)$ due to the lobes associated with $h(x, y)$.

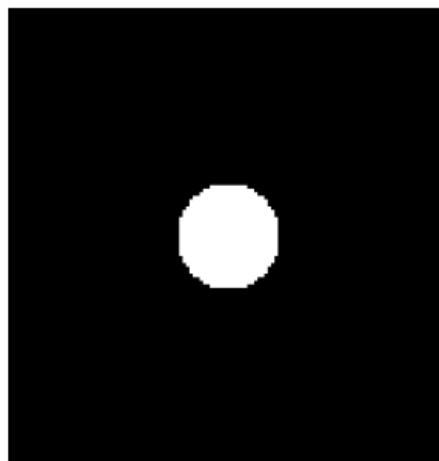


Digital Example

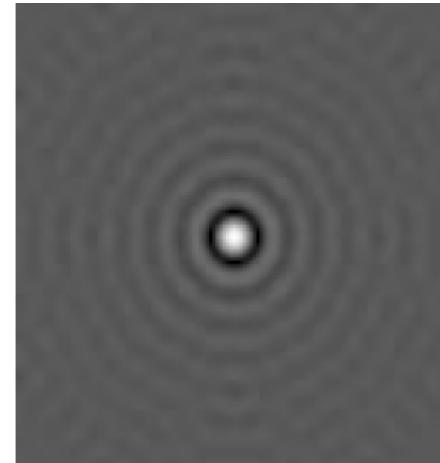
- 128x128 image, low-pass filter with $w_0 = 15$:



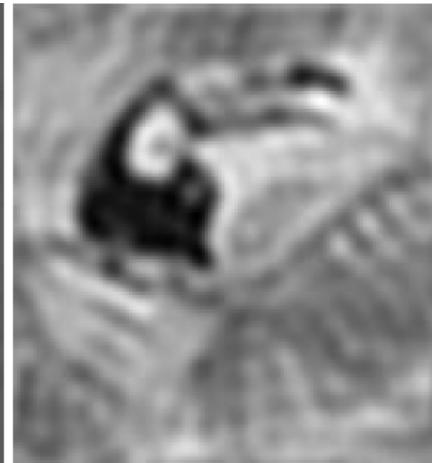
Input image



Low Pass filter



Real space filter



Filtered image

- Useful to reduce the effect of random noise, but too much “ringing” to be actually useful.



Gaussian Low Pass Filter

- Filter profile in Fourier space is a two dimensional Gaussian of the type:

$$H(k, l) = \exp\left(-\frac{w^2}{w_0^2}\right)$$

where $w^2 = k^2 + l^2$ and w_0 is the $1/e$ point of the Gaussian.

- In real space, $h(i, j)$ is also Gaussian:

$$h(i, j) = \frac{\pi}{w_0^2} \exp(-\pi^2 w_0^2 r^2)$$

where $r^2 = i^2 + j^2$

- The filter is infinite in both Fourier and real space, so **attenuates** rather than totally removing the high spatial frequencies.
- $H(u, v)$ does not remove high spatial frequencies
- **The image is smoothed but there is no edge ringing**

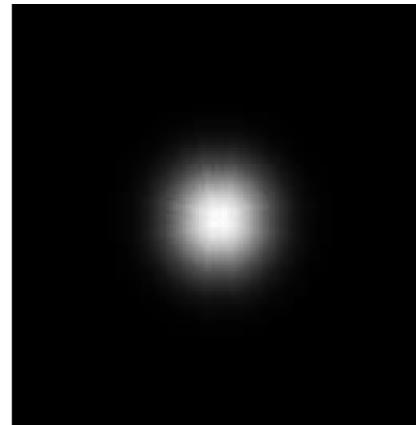


Digital example

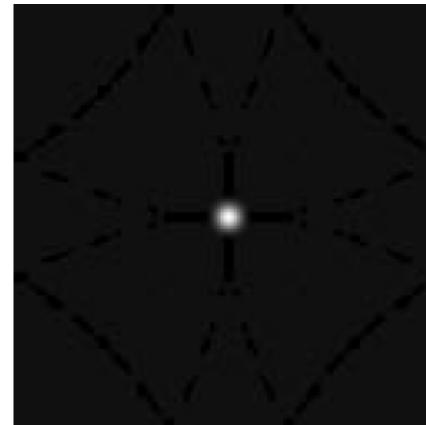
- 128x128 image filtered with low-pass filter; $w_0 = 15$



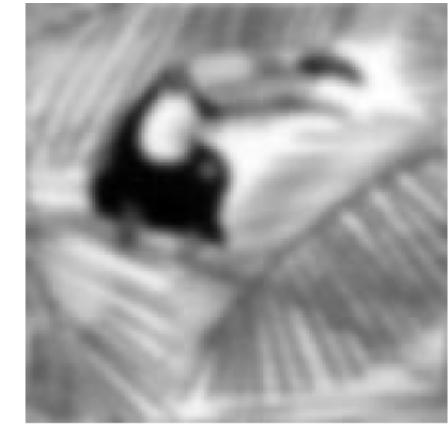
Input image



Low Pass filter



Real space filter

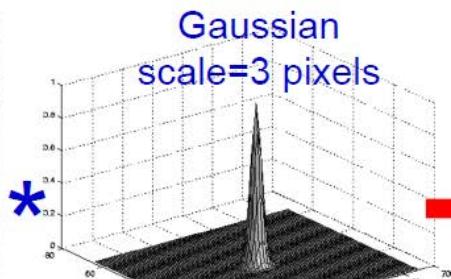
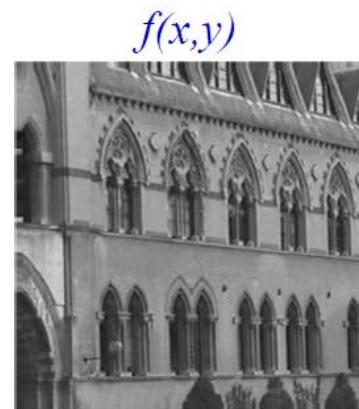


Filtered image

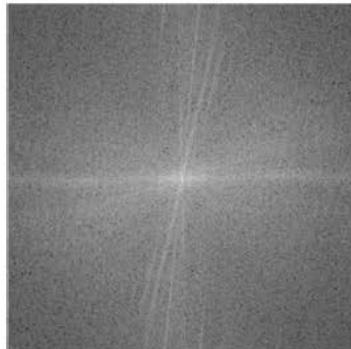
- Very useful digital filter for noise reduction giving a very “smooth” filtered image. Tends to severely smooth edges making further “edge detection” difficult.



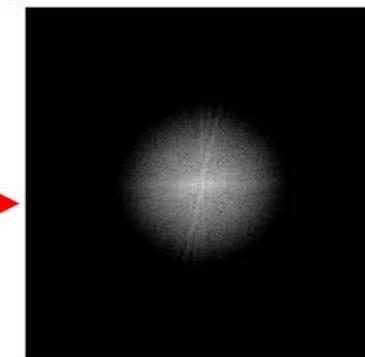
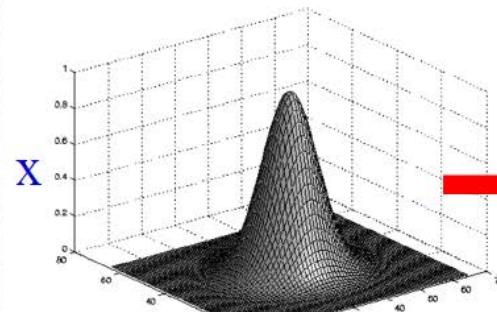
Gaussian Low Pass filter - example 2



Fourier transform



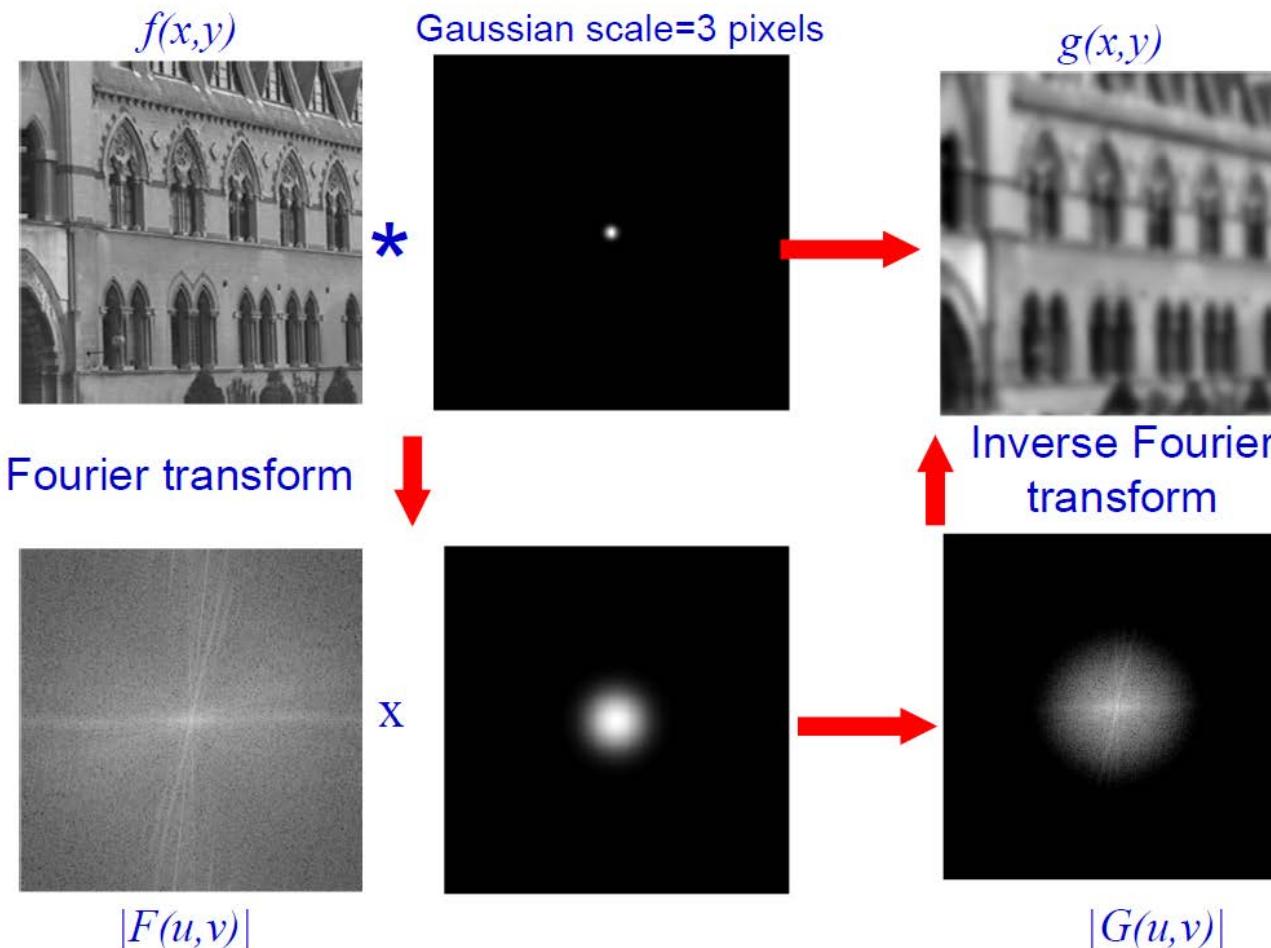
$$|F(u,v)|$$



$$|G(u,v)|$$



Gaussian Low Pass filter - example 2



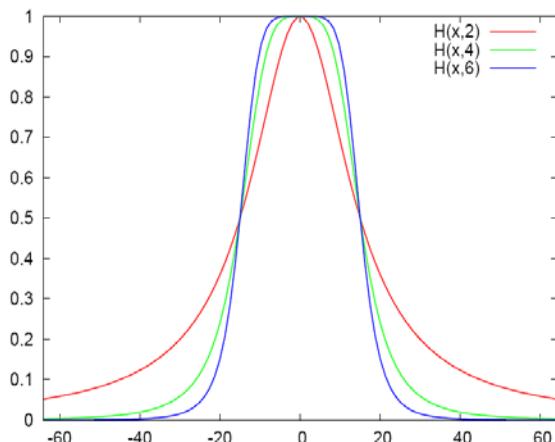


Other Smooth Low-Pass Filters

- Butterworth Filter:

$$H(k, l) = \frac{1}{1 + \left(\frac{w}{w_0}\right)^n}$$

- where w_0 is half point and n is the order.



- Plot with $w_0 = 15$ and $n = 2; 4; 6$.
- Very similar properties to Gaussian, filter inherited from analogue signal processing.

- Trapezoidal filter

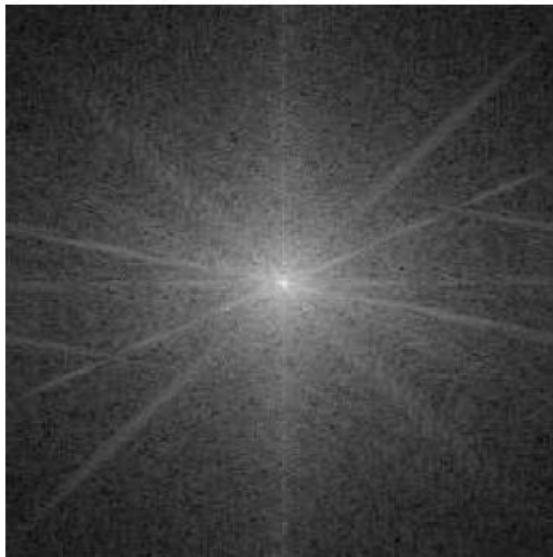
$$H(k, l) = \begin{cases} 1, & \text{if } w < w_0 \\ \frac{w - w_1}{w_0 - w_1}, & w_0 \leq w \leq w_1 \\ 0, & \text{if } w > w_1 \end{cases}$$

- This will exhibit more ringing than Gaussian or Butterworth, but less than ideal filter.



Low Pass Butterworth filtering example

- Low Pass filtering removes high frequencies, blurs image
- Gentler cutoff eliminates ringing artifact



DFT of Image after low
pass Butterworth filtering



Resulting image
after inverse DFT



High Pass Filters

- High pass filters allow HIGH spatial frequencies to pass while attenuating or blocking LOW spatial frequencies. Used for the enhancement of high frequencies (and thus edges).
- Types:
 - Ideal
 - Gaussian
 - Smooth: Butterworth



Ideal High Pass filter

- Block all frequencies less than some limit,

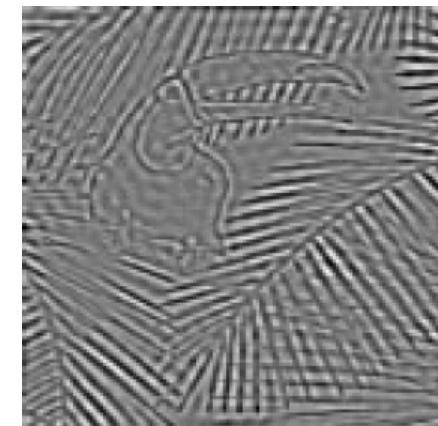
$$H(l, k) = \begin{cases} 0, & l^2 + k^2 \leq w_0^2 \\ 1, & \text{otherwise} \end{cases}$$

Example with $w_0=25$



Filter

- This filter suffers from such severe ringing artifacts that it is rarely used and much better operations can be obtained from the smooth high pass filters.



Output
Image



Gaussian High Pass Filter

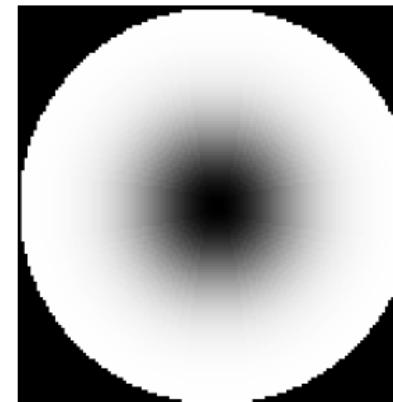
- smooth reduction of **low** spatial frequencies while the high spatial frequencies are pass unaltered.

$$H(k, l) = 1 - \exp\left(-\frac{w}{w_0}\right)^2$$

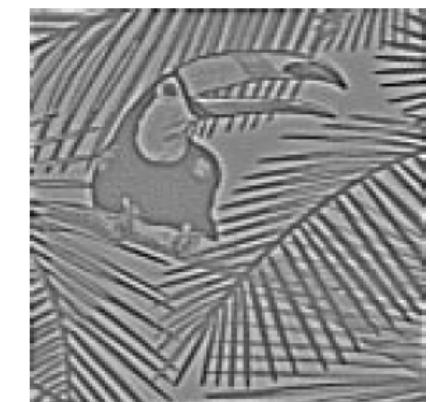
with $w^2 = k^2 + l^2$, which is a smooth filter in Fourier space.

- This gives a smooth $h(i, j)$ in real space, so enhances edges without introduction of “ringing”

- Example with $w_0 = 25$.



Filter



Output Image

In practice often combined with the Gaussian Low Pass filter to form a composite Difference of Gaussians (DOG) filter.



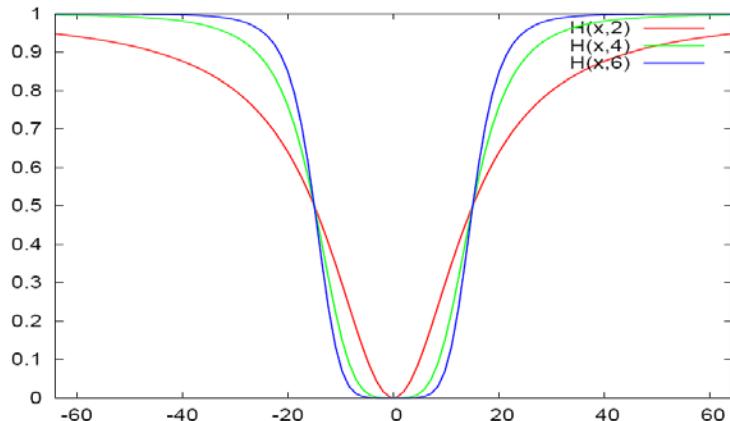
Other Smooth High Pass Filters

High Pass Butterworth:

$$H(k, l) = 1 - \frac{1}{1 + \left(\frac{w}{w_0}\right)^n} = \frac{1}{1 + \left(\frac{w_0}{w}\right)^n}$$

where w_0 is half point and n is the order.

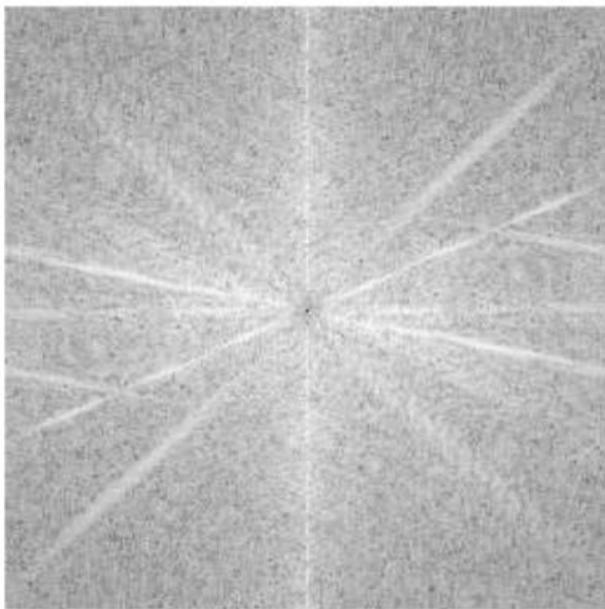
- Plot with $w_0 = 25$ and $n = 2; 4; 6$. Gives almost identical results to



- **Lowpass:** Ideal, Gaussian, Butterworth.
- **Highpass:** Ideal, Gaussian, Butterworth.
- **Bandpass:** Difference of Gaussians (DOG) filter.



High Pass Butterworth filter



DFT of Image after high
pass Butterworth filtering

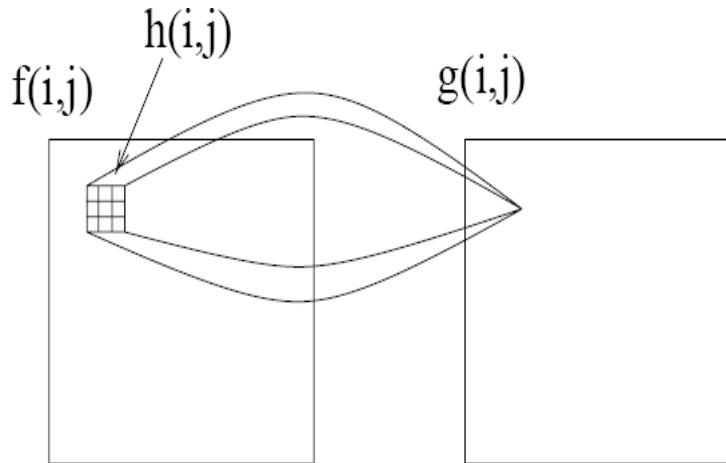


Resulting image
after inverse DFT



Real Space Filters

- Filter is specified in real space by the mask $h(i,j)$ of finite size, typically 3x3, 5x5 or sometimes 7x7.

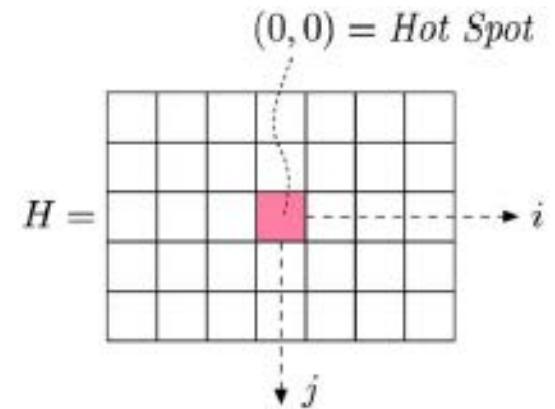
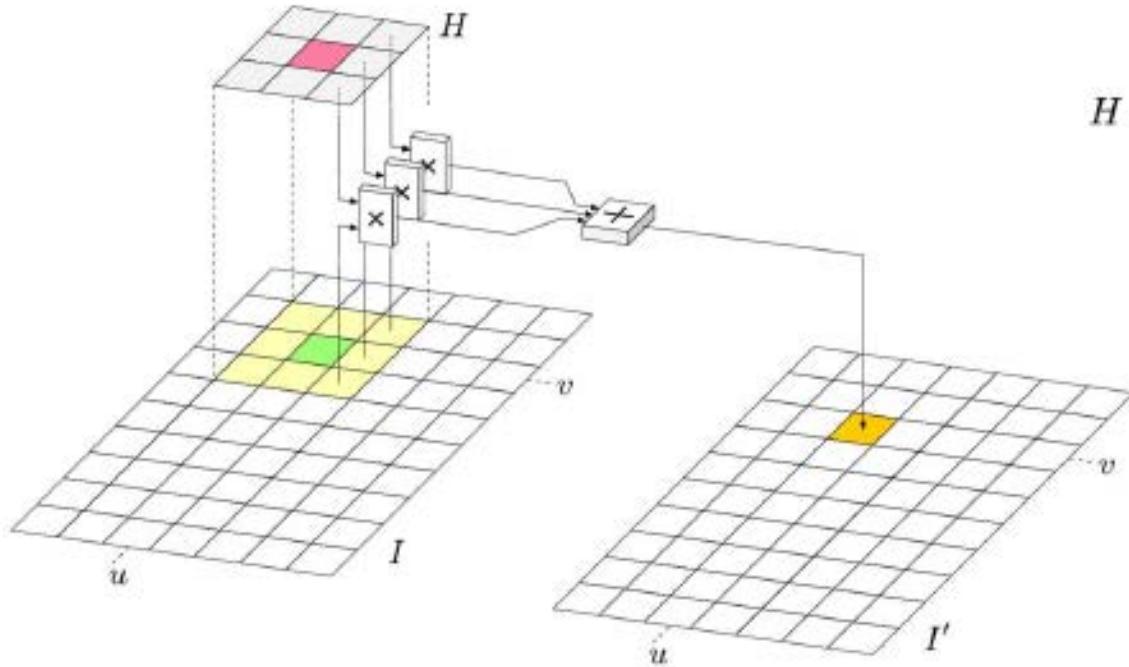


- Mask elements are Real, usually integer.
- Able to use integer, or fixed point arithmetic.
- For masks bigger than 7x7, faster to use Fourier technique.
- **Note :** Convolution can be easily implemented by a parallel computer system, or custom parallel hardware.

- The filter operation is then specified by the mask elements $h(i; j)$.



Real Space Filters



$$I'(u, v) \leftarrow \sum_{(i,j) \in R_H} I(u + i, v + j) \cdot H(i, j)$$

$$I'(u, v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u + i, v + j) \cdot H(i, j)$$



Real Space Averaging

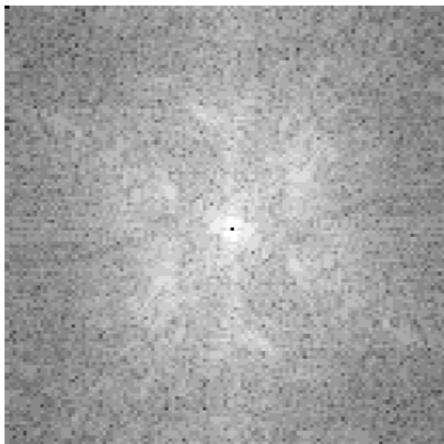
- Replace each pixel by the average of its neighbours, has the effect of “Low pass” filtering, so used to reduce the effect of noise.
 - Effect of reducing high spatial frequencies, but not removing them, (actually removes a range of spatial frequencies).
 - **5 Pixel Average**
0 1 0
1 1 1
0 1 0
 - gives a filter with an effective radius of 1 pixel
 - **9 Pixel Average**
1 1 1
1 1 1
1 1 1
 - gives a filter with an effective radius of $\sqrt{2}$ pixels
- It is approximately equivalent to multiplication in Fourier space with a function:
- $$H(k; l) = \text{sinc}(Nk/3) \text{ sinc}(Nl/3)$$
- where N x N is the size of the image.



Digital Example



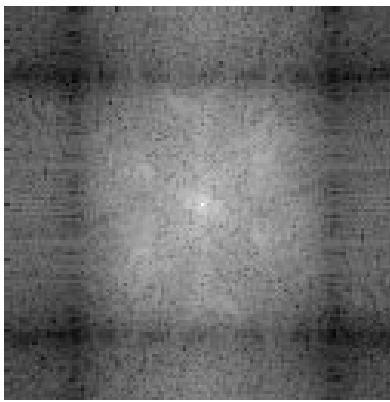
Input image



Fourier Transform



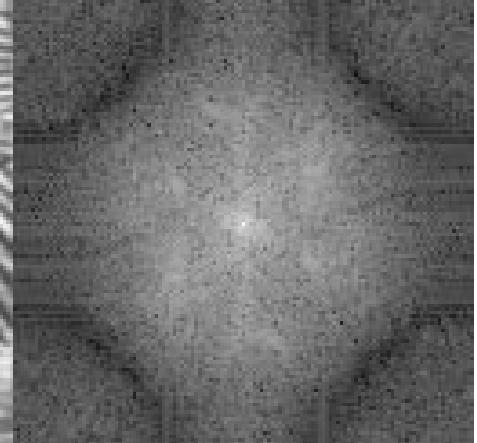
9 point averaging



Fourier Transform



5 point averaging



Fourier Transform

IMAGE PROCESSING



Real Space Gaussian Filters

Gaussian Kernel

1D Case $g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$

2D Case $G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

Separability of 2D Gaussian

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} = g_\sigma(x) \cdot g_\sigma(y)$$

Consequently, convolution with a Gaussian is separable: $I * G = I * G_x * G_y$, where G is the 2D discrete Gaussian Kernel and G_x and G_y are “horizontal” and “vertical” 1D discrete Gaussian kernels.

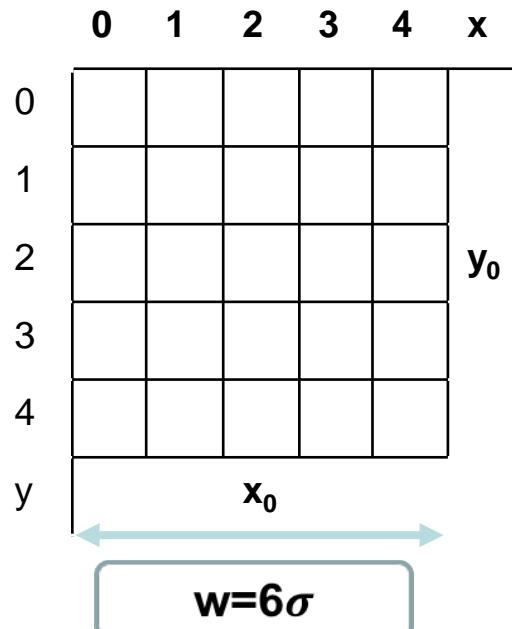


Real Space Gaussian Filters

$$G_{2D}(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

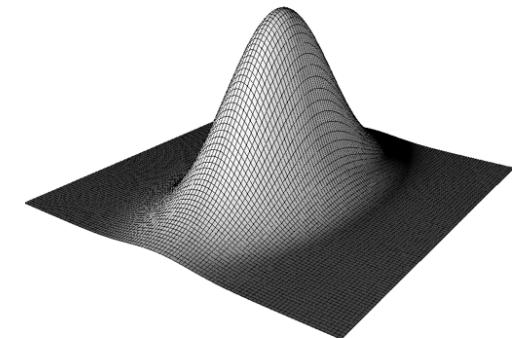
with $\mu=0$ and $\sigma = 1.0$

Design of a Gaussian kernel with given σ



$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1



1. Estimate the size of the kernel $w=6\sigma$
2. Evaluate the values in each cell of the kernel:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x_0-x)^2+(y_0-y)^2}{2\sigma^2}}$$

3. A rounding and a scaling is necessary
4. The middle row and column represent the 1 D kernels



Real Space Differentiation

- For a one-dimensional continuous function we have the definition of differentiation being:

$$\frac{df(x)}{dx} = \lim_{\delta \rightarrow 0} \frac{f(x + \delta) - f(x)}{\delta}$$

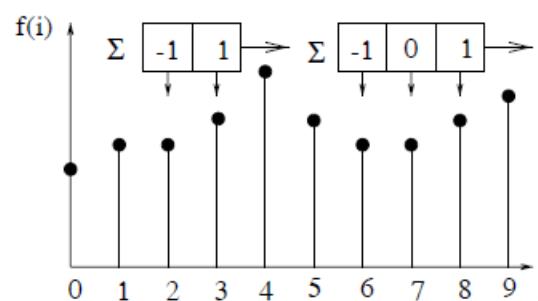
$$\frac{df(i)}{di} = f(i+1) - f(i)$$

- In the discrete case $\delta=1$:
- Which if we consider convolution as “**Shift-fold-multiply-add**” then differentiation can be written as:

$$\frac{df(i)}{di} = [-1 \quad 1] \odot f(i)$$

$$\frac{df(i)}{di} = [-1 \quad 0 \quad 1] \odot f(i)$$

- Simmilarly with $\delta=2$ we get:



Either of these convolutions can be used to approximate the first order differential of a sampled function.



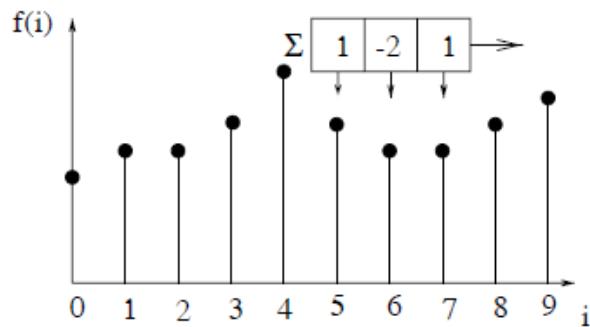
Second order differentials

- The second order differential is given by:

$$\frac{d^2 f(i)}{di^2} = f(i+1) - 2f(i) + f(i-1)$$

Which can be written as:

$$\frac{d^2 f(i)}{di^2} = [1 \ -2 \ 1] \odot f(i)$$



Note also that $[1 \ -2 \ 1] = [-1 \ 1] \odot [-1 \ 1]$

As would be expected since convolution is a linear operation.



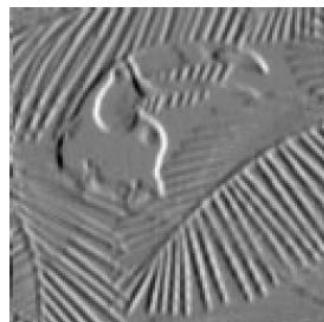
Two dimensional differentials

- In two dimensions we have:
- $\frac{\partial f(i,j)}{\partial i} = [-1 \ 0 \ 1] \odot f(i,j)$ and $\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \odot f(i,j)$
- However to reduce the effect of noise, it is conventional to average the differential over 3 rows/columns respectively to give:

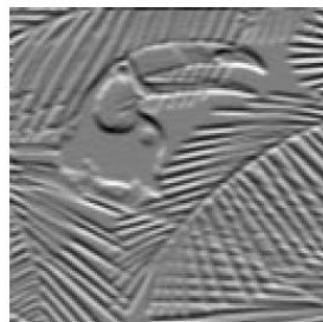
$$\frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \odot f(i,j)$$

$$\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \odot f(i,j)$$

Which will enhance the vertical and horizontal edges respectively.



x-differential

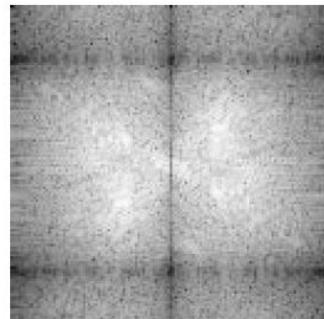


y-differential

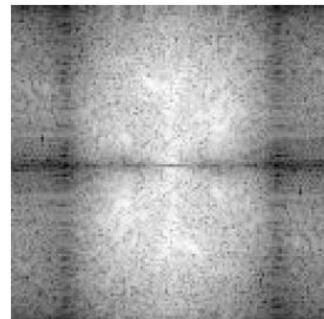


Fourier space differentials

- Properties:
- $F \left\{ \frac{\partial f(x,y)}{\partial x} \right\} = i2\pi u F(u, v)$ and $F \left\{ \frac{\partial f(x,y)}{\partial y} \right\} = i2\pi v F(u, v)$
- Differential is equivalent to Fourier space multiplication by $i2\pi u/v$.
- This has the effect of enhancing high frequency at the expense of low frequencies, so is essentially a “high-pass” filter.



x-differential (FT)



y-differential (FT)

Note: the Fourier transforms show zero through vertical/horizontal lines as expected, plus additional line zeros due to averaging effect.



Second order differentials

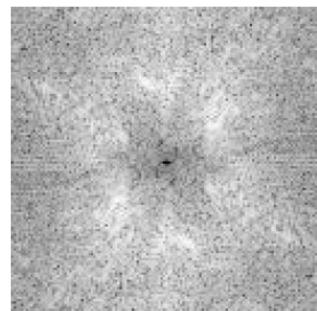
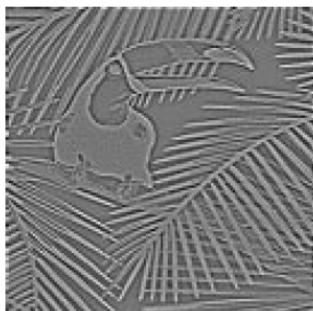
- For the second order differentials we have:

- $\frac{\partial^2 f(i,j)}{\partial i^2} = [1 \ -2 \ 1] \odot f(i,j)$ and $\frac{\partial^2 f(i,j)}{\partial j^2} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \odot f(i,j)$

- So that the Laplacian,

- $\nabla^2 f(i,j) = \frac{\partial^2 f(i,j)}{\partial i^2} + \frac{\partial^2 f(i,j)}{\partial j^2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \odot f(i,j)$

- Which forms the Laplacian of the 2-dimensional image.
- We also have: $F\{\nabla^2 f(x,y)\} = -(2\pi w)^2 F(u,v)$ where $w^2 = u^2 + v^2$ giving:



Enhances edges in all directions.



Laplacian Variations

- We can form an 8 point Laplacian by using

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

which also takes the Laplacian, but is less sensitive to noise.

- Edge Enhancement:**

- Edges of an image may be enhanced by the subtraction of the Laplacian from an image, which can be formed by,

$$f(i, j) - \nabla^2 f(i, j) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \odot f(i, j)$$



Input image



Edge Enhanced



Use of linear filters

- **Low Pass Filters:** are used to smooth images and reduce the effect of noise, in particular used to smooth image prior to edge detection.
- **High Pass Filter:** (also differentiations filters), have the effect of enhancing high frequencies and thus edges.
- Filters can be combined to form *Bandpass* that attenuates both low & high spatial frequencies allowing middle frequencies to pass.
- Due to linear nature, filters can be combined in Fourier space by \times or in real space by \odot operation.

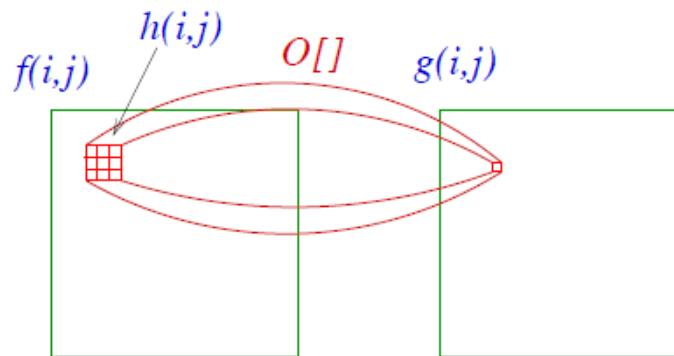


Non-Linear Real Space Filters

- The real space shift & multiply operation can be modified to:

$$g(i, j) = O_{m,n \in w}[h(m, n)f(i - m, j - n)]$$

- Range of $h(m, n)$ defined by w .
- The operation is now defined by the mask $h(i, j)$ and operator $O[]$



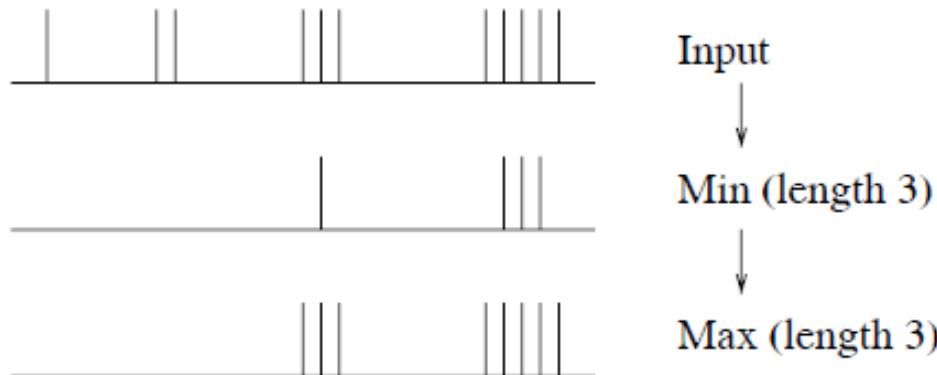
In most non-linear filters we have: $h(i, j) = 1, i, j \in w$

With the operation of the filter controlled by $O[]$ and the size of w only.



Shrink and Expand Filters

- Taking $O[] = \text{Min}[]$ the operator will act as a Shrink operation with bright objects reduced in size by approximately the “size” of the filter.
- Taking $O[] = \text{Max}[]$ the operator will act as an Expand filter, with bright objects increasing in size by approximately the “size” of the filter.
- These operators typically used as a pair on binary image to remove small, isolated regions.
- These filters are not commutative, i.e. $E[S[f(i,j)]] \neq S[E[f(i,j)]]$





Two Dimensional Case

- In two dimensions the Min and Max will selectively remove small bright objects.
- Very useful in “cleaning-up” isolated points in a binary thresholded image



Input



Binary Threshold

Can be used with Grey Scale image, but you tend to get funny results.



Binary Shrink



Binary Expand



Two Dimensional Case



(a)

Original Image with
Salt-and-pepper noise

(b)

Minimum filter removes
bright spots (maxima) and
widens dark image structures

(c)

Maximum filter (opposite effect):
Removes dark spots (minima) and
widens bright image structures



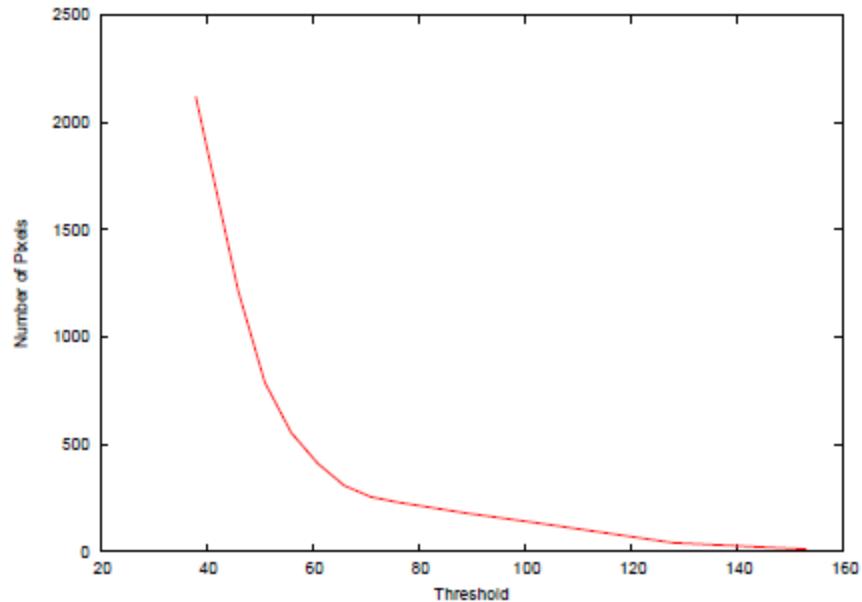
Threshold Average Filter

- For “data-dropout” noise we have isolated “noise points” that differ from the neighbour pixels.
- Compare each pixel with average of neighbours and smoothes only if pixel deviates significantly
- For each point form $A = \sum_{m,n=-M/2}^{M/2} h(m,n)f(i - m, j - n)$
- For 3x3 filter we have:
 - $h(i,j) = \begin{bmatrix} k & k & k \\ k & 0 & k \\ k & k & k \end{bmatrix}$ where $k = 1/(M^2-1) = 0.25$, then output is:
 - $g(i,j) = \begin{cases} A & \text{if } |A - f(i,j)| > T \\ f(i,j) & \text{otherwise} \end{cases}$
 - Selectively removes points that differ from neighbours.



Random bit Error Example

- 8 bit image and we corrupt 1:50 bits. Large corruption when *most significant bit* is corrupted.
- For 128×128 pixel image expect about 325 seriously corrupted pixels.
- Apply *Average Threshold Filter* and count number of changed pixels
- Typical threshold value
 $T=0.25 f_{MAX}$



1:50 Bit Error

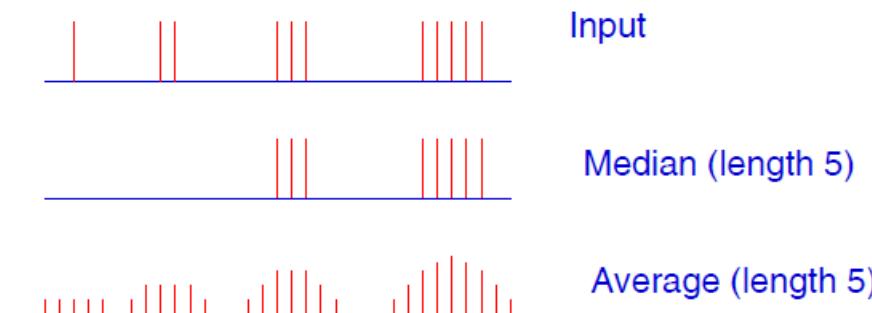


Threshold of 66



Median Filter

- The Median filter is formed by setting $O[] = \text{Median}[]$
- Where the median is defined as the *middle* value.
 - Eg. for the 5 values $f(i) = 6, 1, 0, 9, 11, 9$ then $\text{Median}[f(i)] = 10$
- **Note:** it effectively ignores the out-of-place large values, so removes noise points.
- In 1D the median filter removes all features of less than $M/2+1$ in size but preserves all other features.

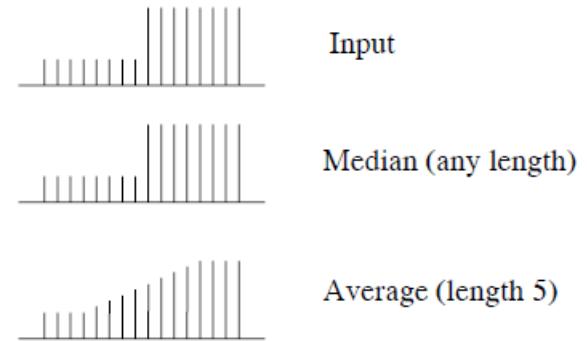


Similar to Shrink/Expand, but is also valid for Grey Level images.



Edge preserving property

- The most useful feature of the Median filter is its edge preserving property
- In 2D it removes all feature of size $< M/2 - 1$ while retaining all other features, **and** retaining edges.
- Very useful noise reduction filter used throughout image processing.
- Filter effectively smoothens the image into regions of constant intensity but retains edges.
- So acts as a selective Low-Pass filter.



3×3 Median



5×5 Median



Median Filter Example



(a)
Original Image with
Salt-and-pepper noise



(b)
Linear filter removes some of
the noise, but not completely.
Smears noise



(c)
Median filter salt-and-pepper noise
and keeps image structures largely
intact. But also creates small spots
of flat intensity, that affect sharpness

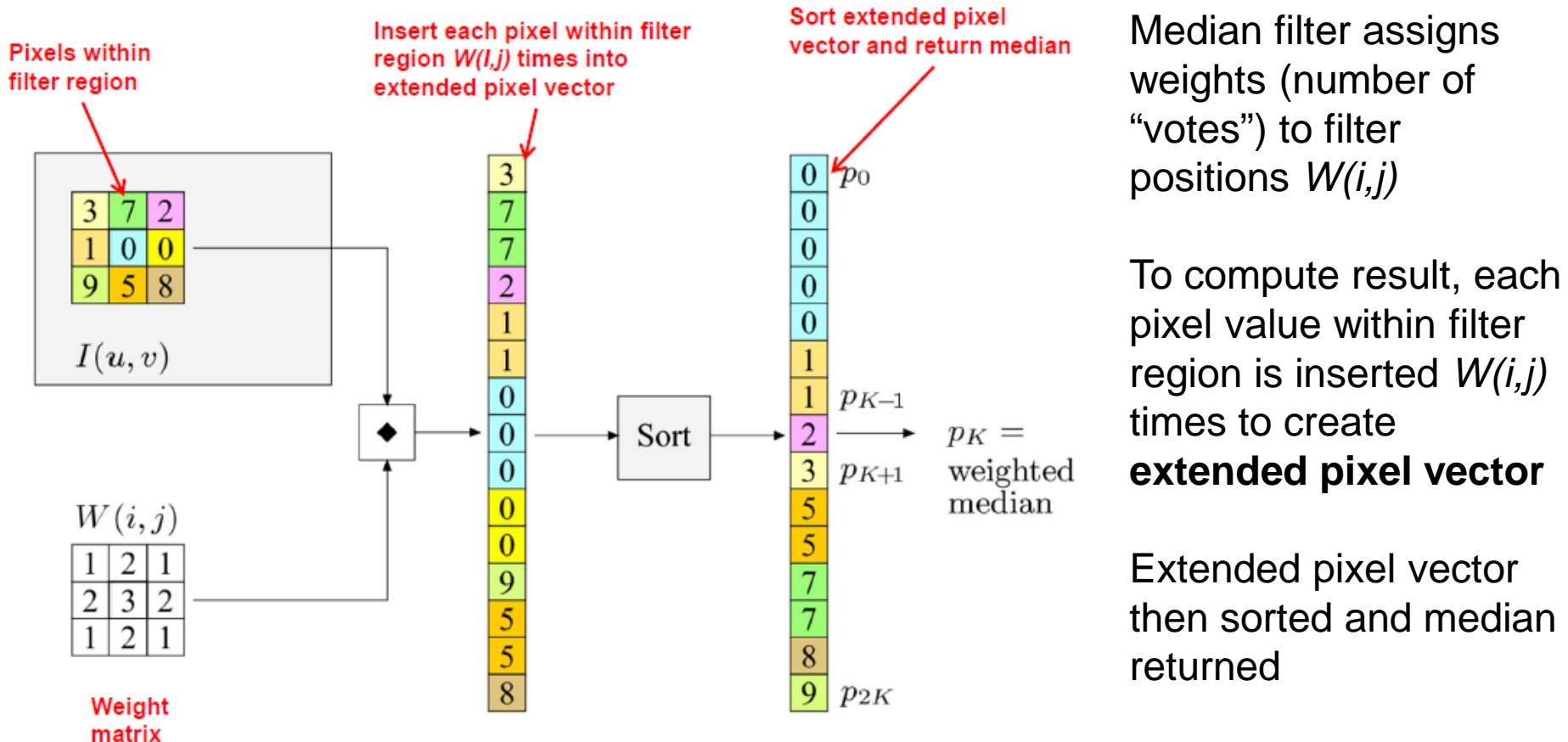


Implementation of Median Filter

- To calculate Median over each window the data must be (partly) sorted.
- Computationally expensive, and typically 5×5 Median filter about the same computational time as DFT.
- *Aside: Medians of large arrays are very slow to calculated by “thick” (SelectSort) way. Fast sorting techniques should be used.*
- One of the most useful real space filters available.



Weighted Median Filter

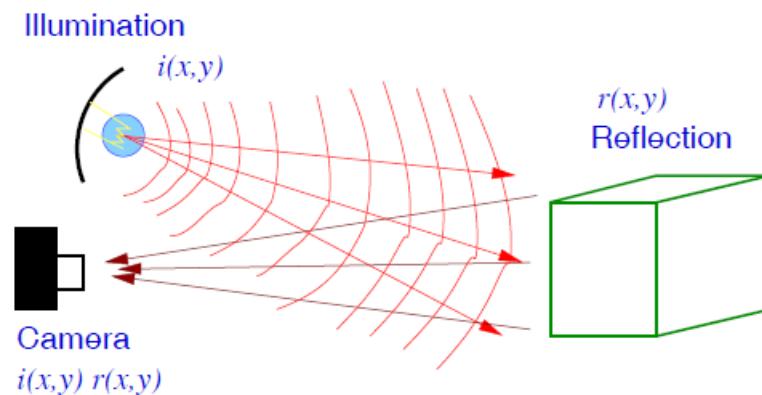


Weighting can be applied to implement non-rectangular filters and/or to increase the influence of a specific area.



Homomorphic Filtering

- For the case of a multiplicative process in Real space,
 - $f(x,y) = i(x,y)r(x,y)$
 - Where $i(x,y)$ = Illumination and $r(x,y)$ = reflectance.





Homomorphic Filtering cont'd

- Apply $\ln()$ to separate terms:

$$z(x,y) = \ln(i(x,y)) + \ln(r(x,y))$$

- Apply Fourier transform

$$Z(u,v) = F(\ln(i(x,y))) + F(\ln(r(x,y))) \text{ known as Cepstrum}$$

- Consider the frequency characteristic of each term:

- $i(x,y)$ is smooth then $\ln(i(x,y))$ is smooth

- $r(x,y)$ is rough then $\ln(r(x,y))$ is rough

- Filter $Z(u,v)$ to get $Y(u,v) = Z(u,v)H(u,v)$ where

- High Pass filtering improves $i(x,y)$

- Low Pass filtering improves $r(x,y)$

- Then the improved image $g(x,y) = \exp(F^{-1}(Y(u,v)))$

- Typically used to correct the illumination but can also be used for dealing with multiplicative noise



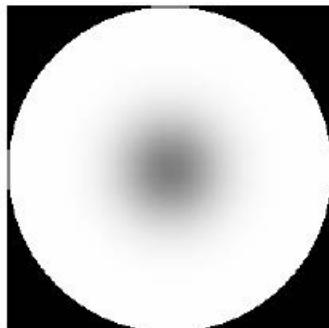
Homomorphic Filtering example



Input Image



Log of Input



Filter



Output

Low frequency variation in illumination has been (partially) removed.



Bibliography

- <https://www2.ph.ed.ac.uk/~wjh/teaching/dia/documents/filtering.pdf>
- <https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture4.pdf>
- <https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture10.pdf>
- <https://www.ibiology.org/talks/fourier-transform/>
- <http://www.robots.ox.ac.uk/~az/lectures/ia/lect1.pdf>
- <http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf>
- <https://www.pearson.com/us/higher-education/program/Gonzalez-Digital-Image-Processing-4th-Edition/PGM241219.html>



EDGE BASED SEGMENTATION

Image Processing
(Year III, 2-nd semester)

Lecture XI- XII: EDGE BASED SEGMENTATION



Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy



EDGE BASED SEGMENTATION

Edge detection

Edges are generated by:

- changes in the surface reflectance
- changes in the surface orientation
- an object being covered partially by another
- indirect covering by shadows

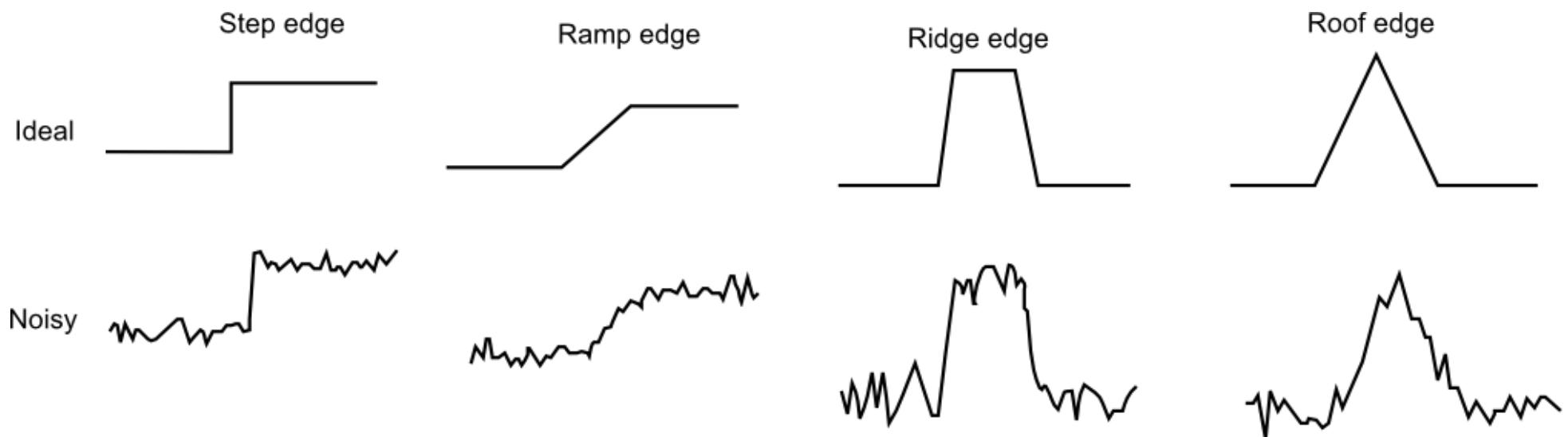
Edges are rapidly varying parts of the image so are represented by high spatial frequencies.



EDGE BASED SEGMENTATION

Typical edge profiles

Edges can be modeled according to their intensity profiles:

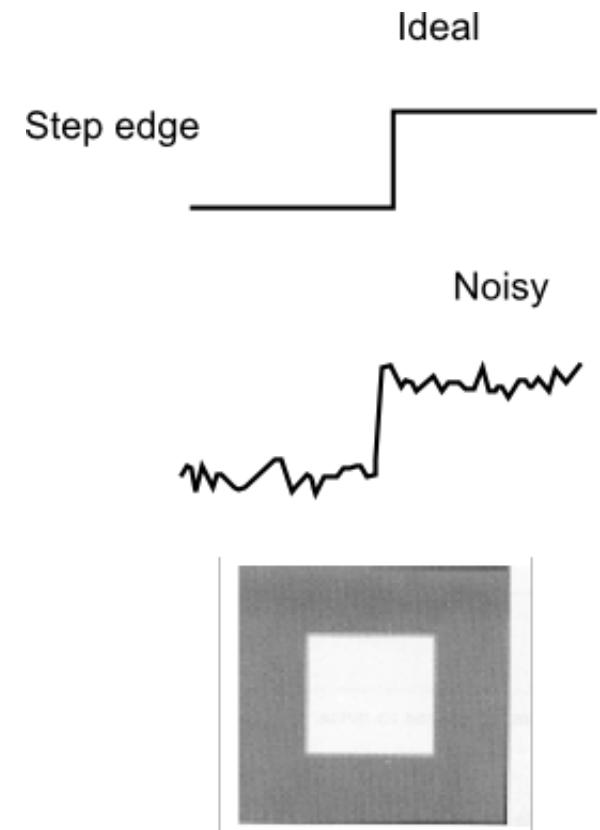




EDGE BASED SEGMENTATION

Edge types: Step edge

The image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side.

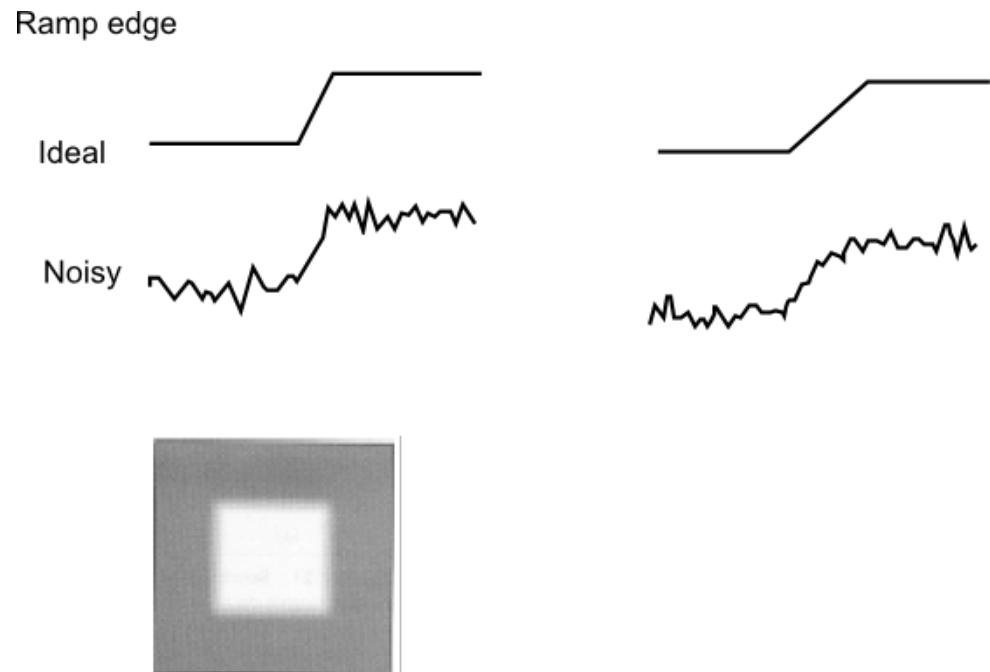




EDGE BASED SEGMENTATION

Edge types: Ramp edge

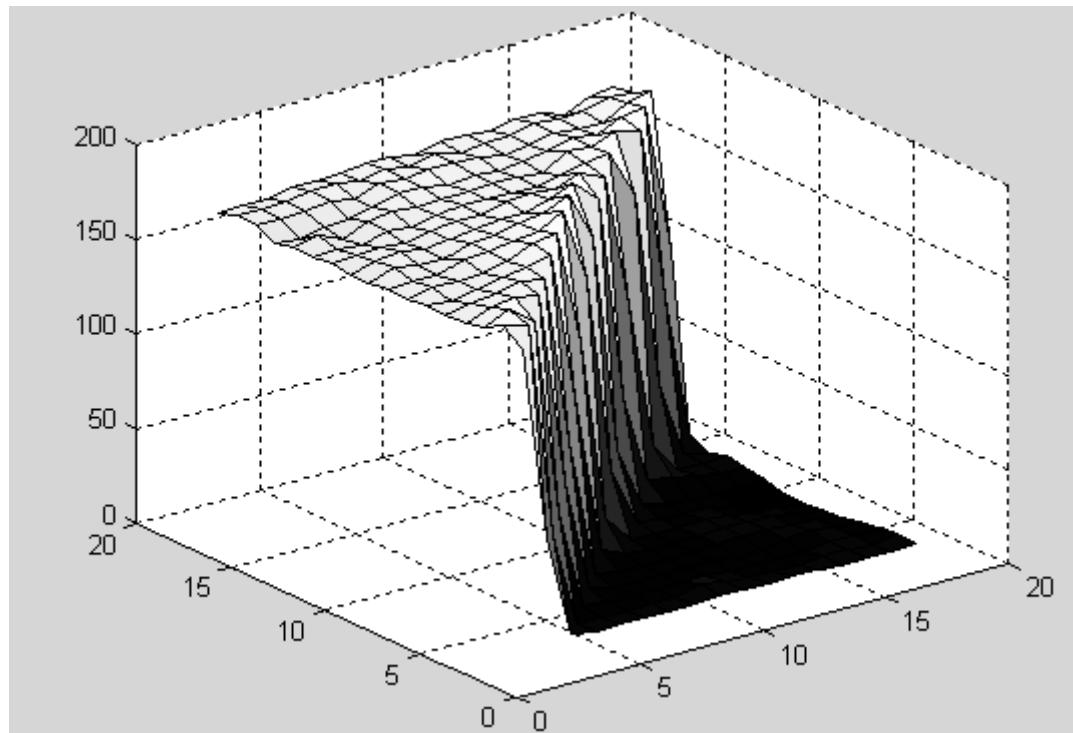
A step edge where the intensity change is not instantaneous but occurs over a finite distance.





EDGE BASED SEGMENTATION

Example:



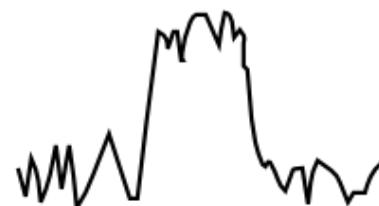
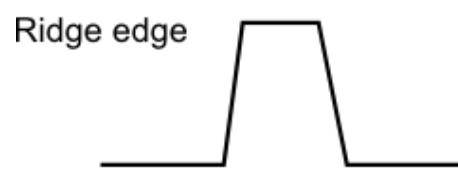
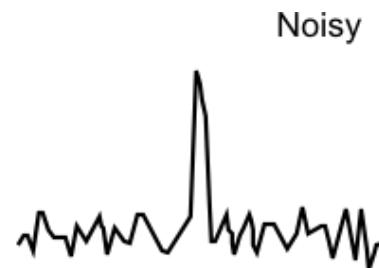
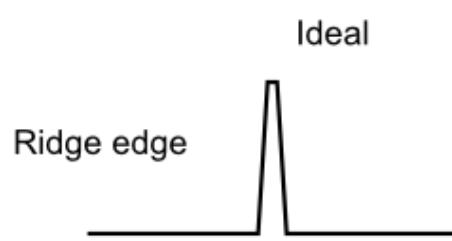
¹ Images taken from [3]



EDGE BASED SEGMENTATION

Edge types: Ridge edge

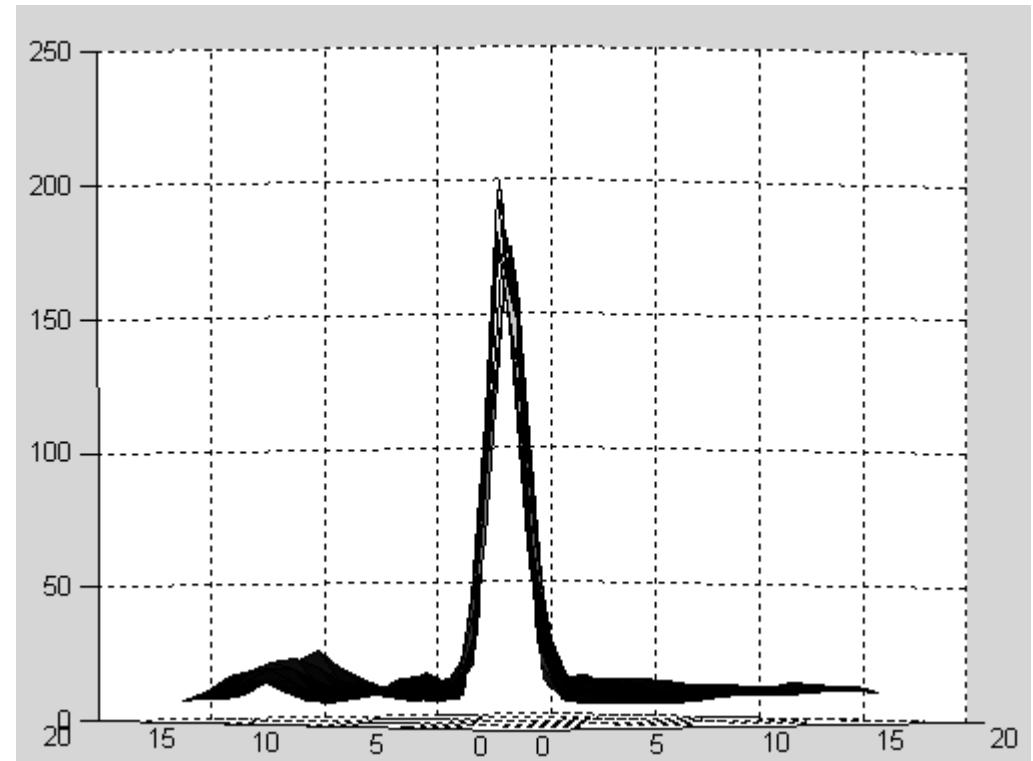
The image intensity abruptly changes value but then returns to the starting value within some short distance
– generated usually by lines





EDGE BASED SEGMENTATION

Example:



2

² Images taken from [3]

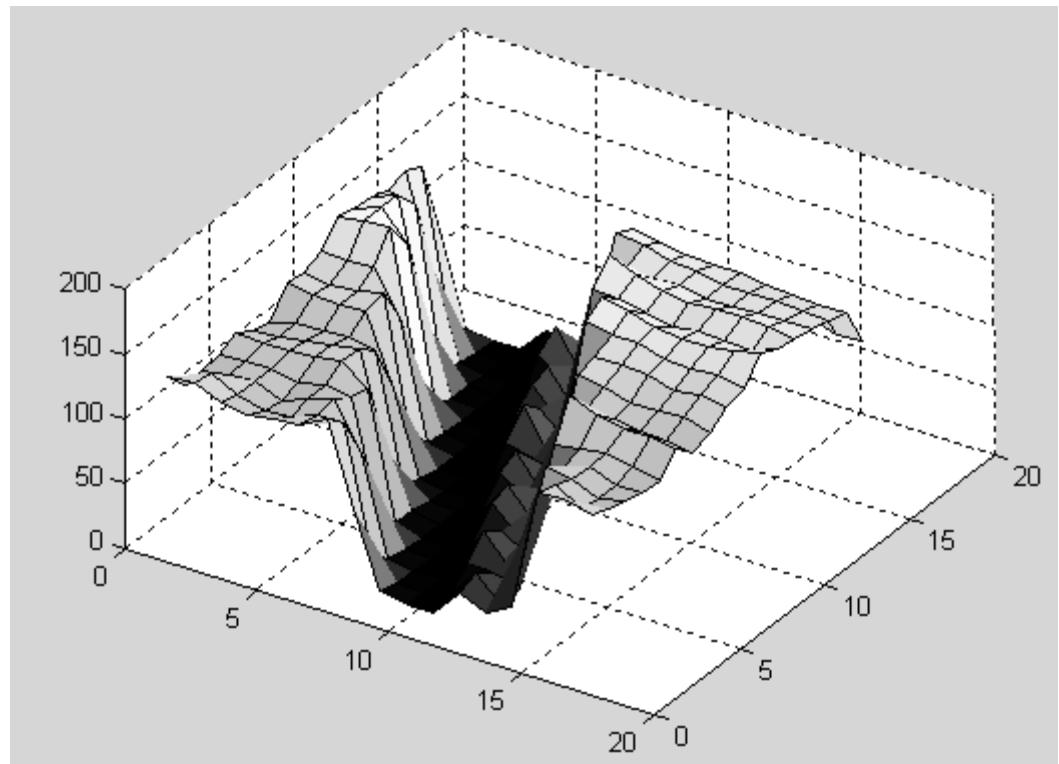


EDGE BASED SEGMENTATION

Example:



3



³ Images taken from [3]

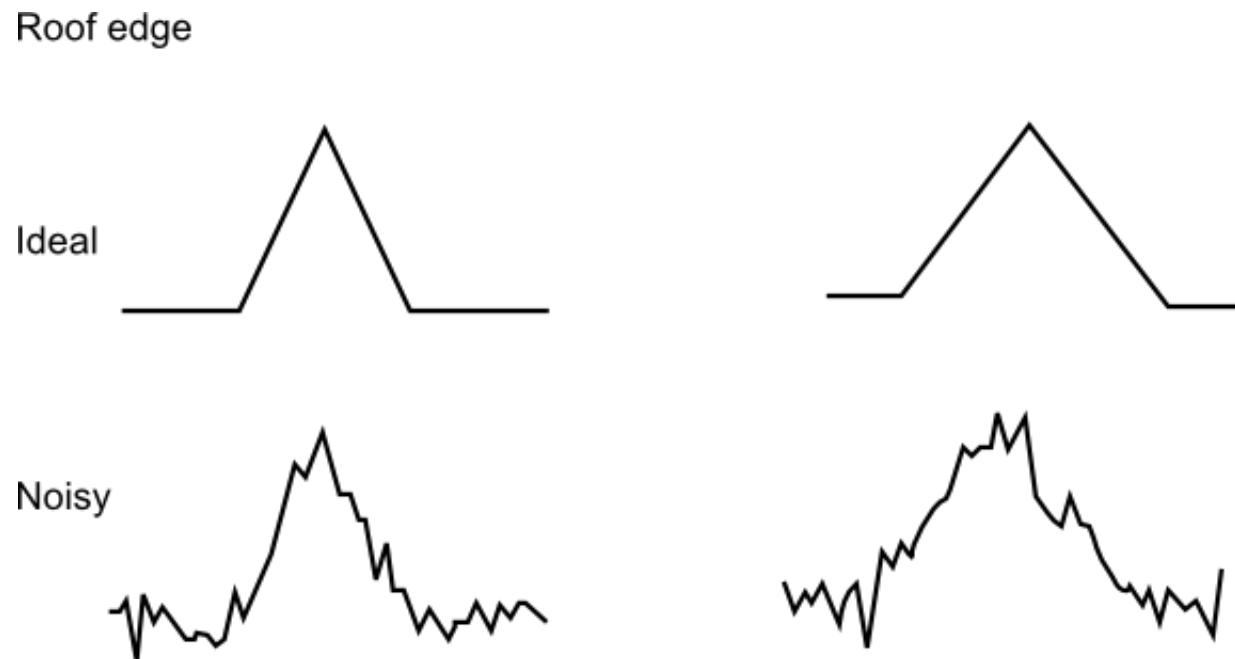


EDGE BASED SEGMENTATION

Edge types: Roof edge

A ridge edge where the intensity change is not instantaneous but occurs over a finite distance.

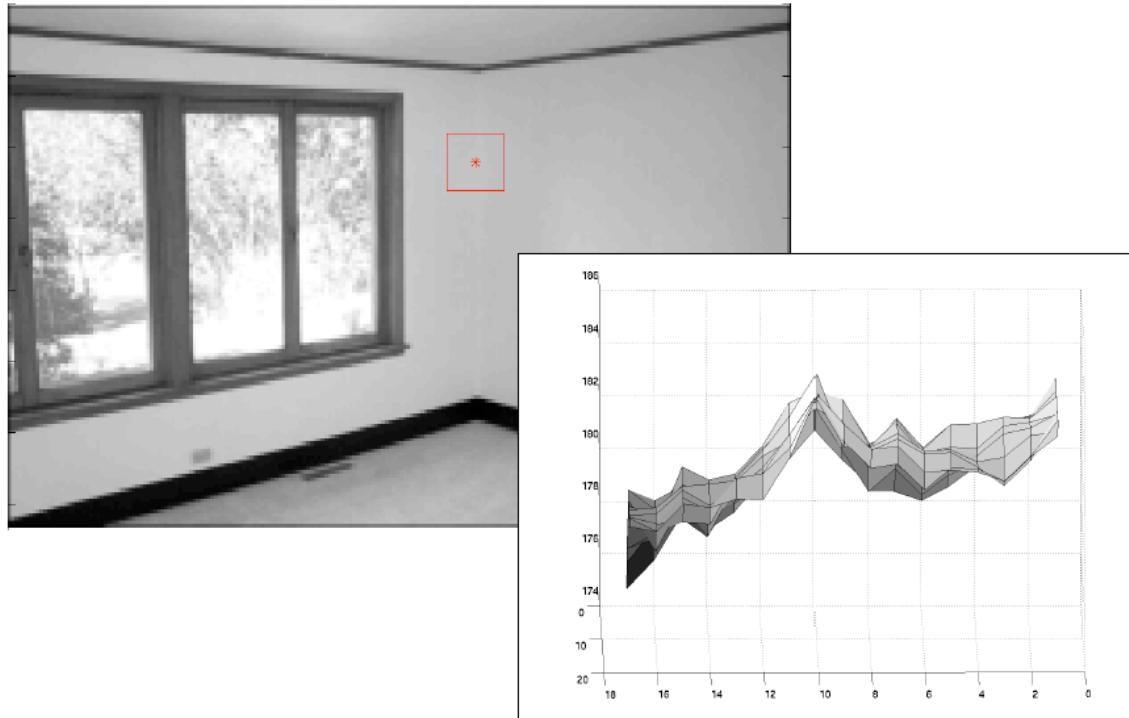
- generated usually by the intersection of surfaces





EDGE BASED SEGMENTATION

Example:



4

⁴ Images taken from[3]



Steps in edge segmentation

The edge based segmentation process:

- *preprocessing* - noise elimination
- *edge pixels detection and identification*
- *post-processing* - improve the detection results, extract edges, close the contours.



EDGE BASED SEGMENTATION

Classification
Criteria:

Principle used for edge
detection:

Way of performing the
derivation and filtering
operations

Methods based on applying differential operators

Methods based on filtering followed by differential operators

Methods based on approximation by one-dimensional surfaces
having a similar profile with the edge

Residues method

Methods that work on discrete surfaces, using discrete
approximation of differential operators by finite differences

Methods that work on surfaces approximated by a continuous
function and use the analytic form of the differential operators



EDGE BASED SEGMENTATION

Edge Detection

- The aim of all edge detection techniques is to enhance and mark edge pixels and then extract them.
- All need some type of High-pass filter, which can be viewed as either **First or Second order differentials**.

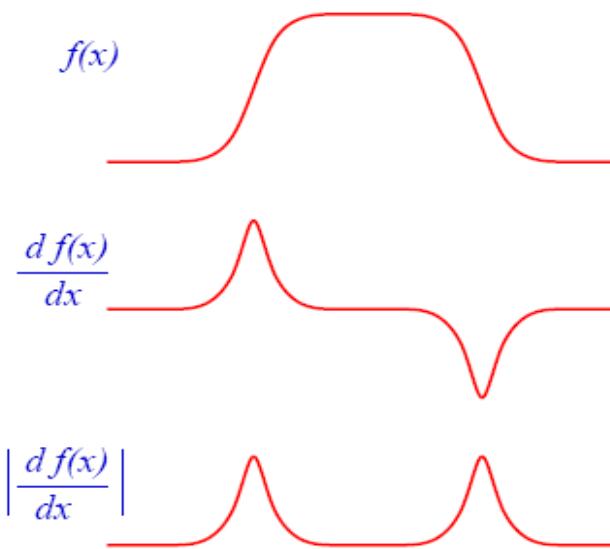


First Order Differentials

The One-Dimensional case:

- Given a function $f(x)$
- Compute the derivative $df(x)/dx$

We can then detect the edge by a simple threshold of: $|\frac{\partial f(x)}{\partial x}| > T \Rightarrow Edge$





EDGE BASED SEGMENTATION

Edge detection: First Order Differentials

- In two-dimensions, being given a function $f(x, y)$:

$$|\frac{\partial f(x, y)}{\partial x}| \Rightarrow \text{Vertical Edge} \text{ and } |\frac{\partial f(x, y)}{\partial y}| \Rightarrow \text{Horizontal Edge}$$

- But we want to detect edges in all directions. The variation of intensity in a given direction, θ is given by:

$$\frac{\partial f(x, y)}{\partial x} \cdot \cos \theta + \frac{\partial f(x, y)}{\partial y} \cdot \sin \theta$$

- In two-dimensions, the 1st order differential, $\nabla f(x, y)$ called **gradient** is a

vector given by:
$$\frac{\partial f(x, y)}{\partial x} \vec{i} + \frac{\partial f(x, y)}{\partial y} \vec{j}$$

- The modulus of the gradient is: $|\nabla f(x, y)| = \sqrt{(\frac{\partial f(x, y)}{\partial x})^2 + (\frac{\partial f(x, y)}{\partial y})^2}$

- Threshold to obtain edges:

$$|\nabla f(x, y)| > T \Rightarrow \text{Edge}, \quad |\nabla f(x, y)| < T \Rightarrow \text{no Edge}$$



EDGE BASED SEGMENTATION

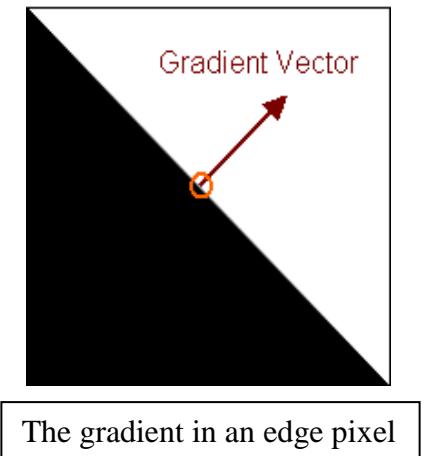
Edge detection: First Order Differentials

Digital implementation - form the first order differentials by convolution of the image with two kernels:

$$\frac{\partial f(i, j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i, j) ,$$

$$\frac{\partial f(i, j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \circ f(i, j)$$

$$|\nabla f(i, j)| = \sqrt{\left(\frac{\partial f(i, j)}{\partial i}\right)^2 + \left(\frac{\partial f(i, j)}{\partial j}\right)^2} \quad (1)$$



- faster approximation of the modulus of the gradient is given by:

$$|\nabla f(i, j)| = \left| \frac{\partial f(i, j)}{\partial i} \right| + \left| \frac{\partial f(i, j)}{\partial j} \right| \quad (2)$$



EDGE BASED SEGMENTATION

First Order Differentials Examples

Original image

**Image
Processing**

Horizontal edges

**Image
Processing**

Vertical edges

**Image
Processing**

Gradient
Magnitude
(1)

**Image
Processing**

Gradient
Magnitude
(2)

**Image
Processing**



EDGE BASED SEGMENTATION

Edge Detection: Sobel filter

- is a slight variation on the first order differential filter :

$$\bullet \quad \frac{\partial f(i, j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i, j), \text{ and } \frac{\partial f(i, j)}{\partial j} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \circ f(i, j)$$

- This is the most common simple first order differential edge detector.





EDGE BASED SEGMENTATION

Sobel filter: Examples

Original
image

**Image
Processing**

Vertical
edges

Image
Processing

Horizontal
edges

Image
Processing

Gradient
magnitude

**Image
Processing**



EDGE BASED SEGMENTATION

Sobel filter: Examples



Original image



Horizontal edges



Vertical edges



Gradient magnitude



$T = 20$



$T = 120$



$T = 210$



$T = 250$



First Order Diferentials: Problems

- **Arbitrary threshold value:** for avoiding this, the threshold is set such that x% of image is classified as edge
- **Thick edges:** if threshold is too low edges frequently are thick. Range of **edge thinning techniques** that try to thin edges to a single pixel by removing edge pixels while keeping the edges connected.
- **Broken Edges:** Range or **edge-joining techniques** to try bridge gaps also Hough Transform, to fit lines.
- **Noise Points:** Modified threshold filter to **remove isolated points** or non-connected double points.



EDGE BASED SEGMENTATION

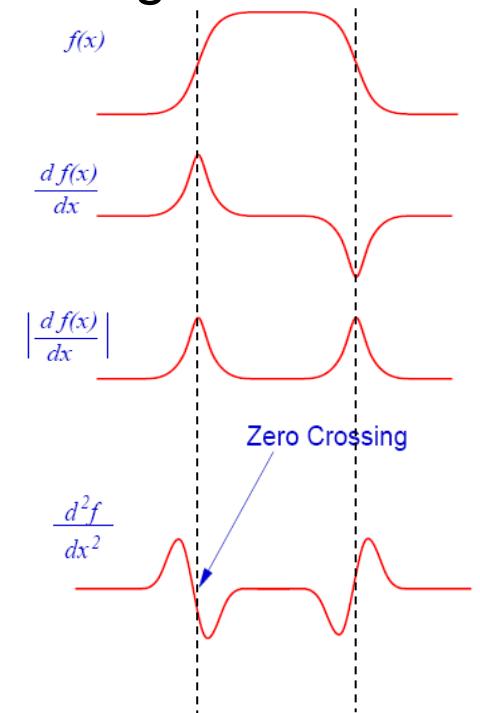
Edge Detection: Second Order Differentials

- In one dimension the edge is located by the zero crossing:
- In two dimensions one has to compute the Laplacian:

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

It can be implemented by a single convolution of :

$$\nabla^2 f(x, y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ f(i, j)$$

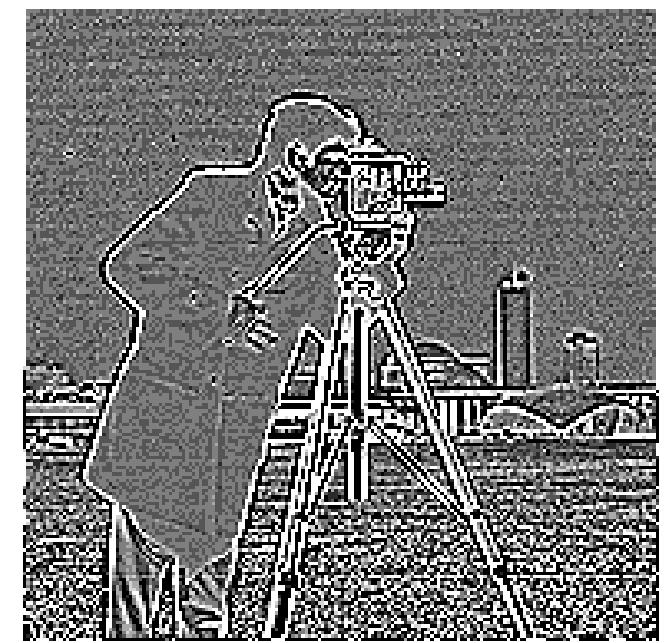




EDGE BASED SEGMENTATION

Laplacian example

•

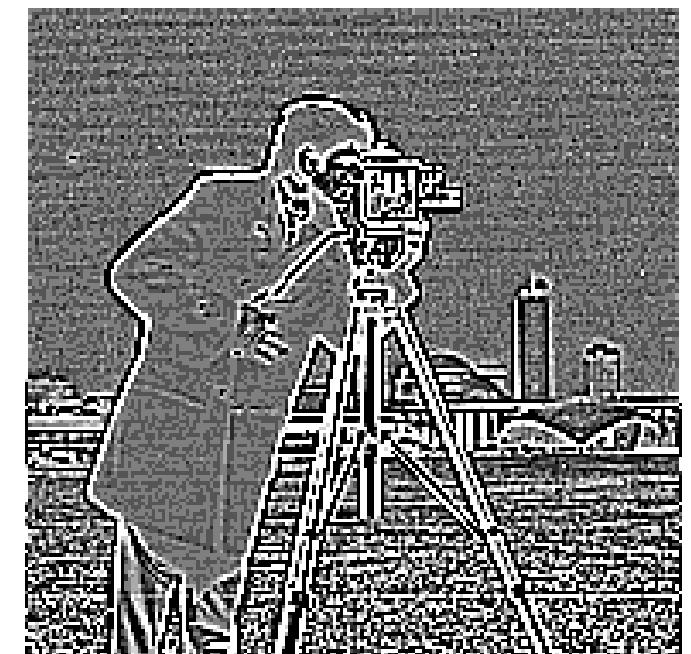


$$\nabla^2 f(x, y) = \frac{1}{8} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ f(i, j)$$



EDGE BASED SEGMENTATION

Laplacian example

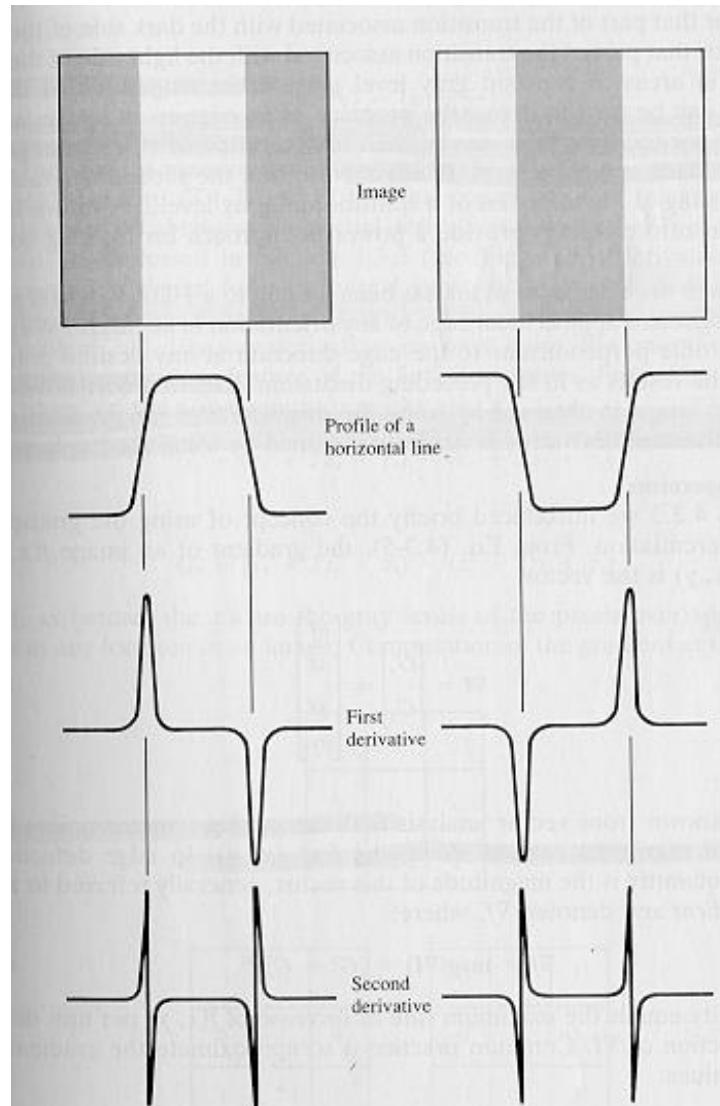


$$\nabla^2 f(x, y) = \frac{1}{16} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \circ f(i, j)$$



EDGE BASED SEGMENTATION

First and second order derivatives





Second Order Differentials: Problems

- **Thin Edges:** the edges occur between pixels. Always get thin edges, but difficult to display on a digital image.
- **Closed Loops:** edges always form closed loops, reduces break-up of edges, but can cause problems at corners.
- **Noise Problems:** Laplacian is a high pass filter, so enhances high frequencies, and thus noise.



Pre Processing

- Difficult to enhance the edge image, so to reduce the effect of noise we typically want to smooth the image *before* we apply the Laplacian.
- For example use Nine point average, then Laplacian.
- Noting that the convolution is a linear operation, the two [3x3] convolutions can be implemented as a single [5x5] convolution.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & -2 & -1 & -2 & 1 \\ 1 & -1 & 0 & -1 & 1 \\ 1 & -2 & -1 & -2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$



EDGE BASED SEGMENTATION

Edge Enhancement



$$f(x, y) - \nabla^2 f(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \circ f(i, j)$$



Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy



Methods based on filtering followed by differential operators

1. The 1D situation:

- The edge is represented by a step signal, and the edge point is represented by its coordinates
- A Gaussian noise is overlapped over the step signal
- The edge point is given by a maximum variation in intensity, and it can be identified by the maximum of the 1st order derivative or the zero crossings of the 2nd order derivative
- Noise generates several local maxima of close amplitude, and several zero crossings



Analysis of the 2D situation

- The edge point is characterized by coordinate and direction
- Use as filtering operator a 2D Gaussian $G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = g(x)g(y)$
- Edge points can be emphasized by:
 1. Maximum of gradient
 2. Zero crossings of the Laplacian
 3. Zero crossings of the 2nd order directional derivative in the gradient direction



Canny edge detector

Criteria:

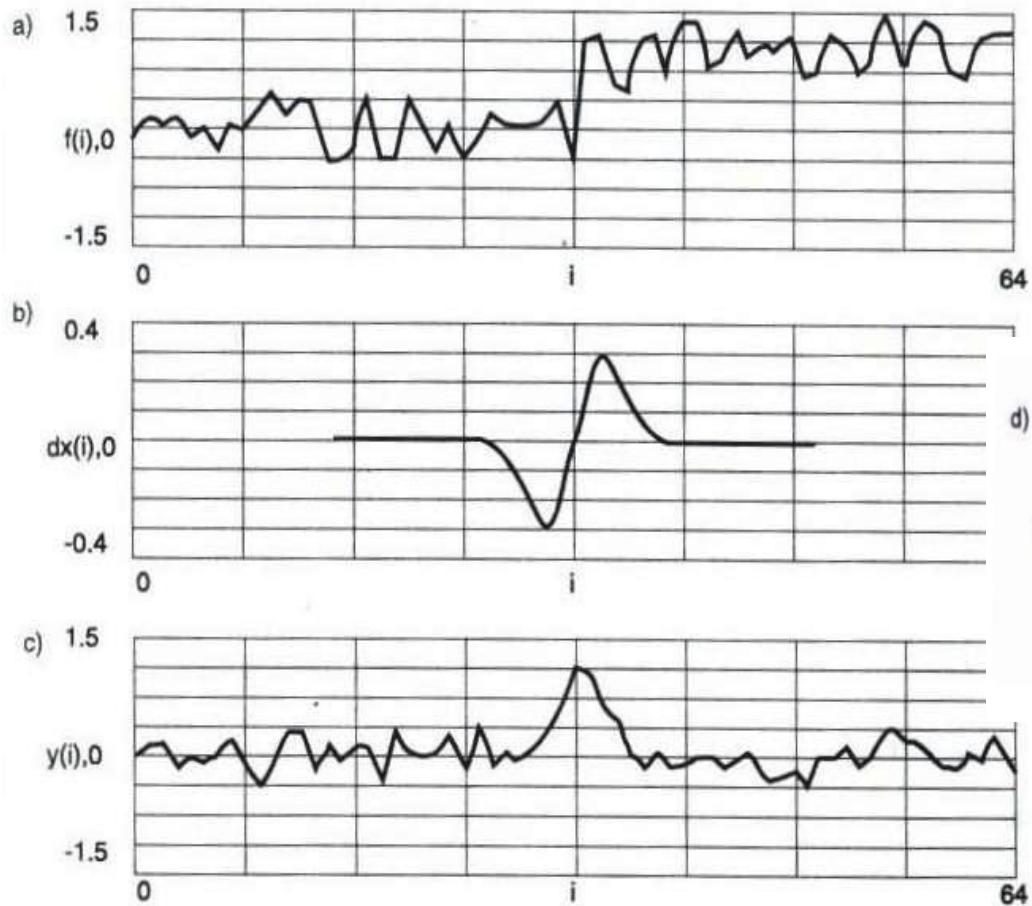
1. **Optimal detection:** the probability of marking a real edge point should be high, and the probability of marking a false edge point should be low (maximize the signal-noise ratio)
2. **Correct localization:** the points marked as belonging to an edge should be as close as possible to the real edge points
3. **Single response to one edge**

Canny gave the mathematical formulation of the three criteria, showing that **the first order derivative** of the Gaussian models its filter. A similar result can be found by using the **2nd order derivative** of the Gaussian.

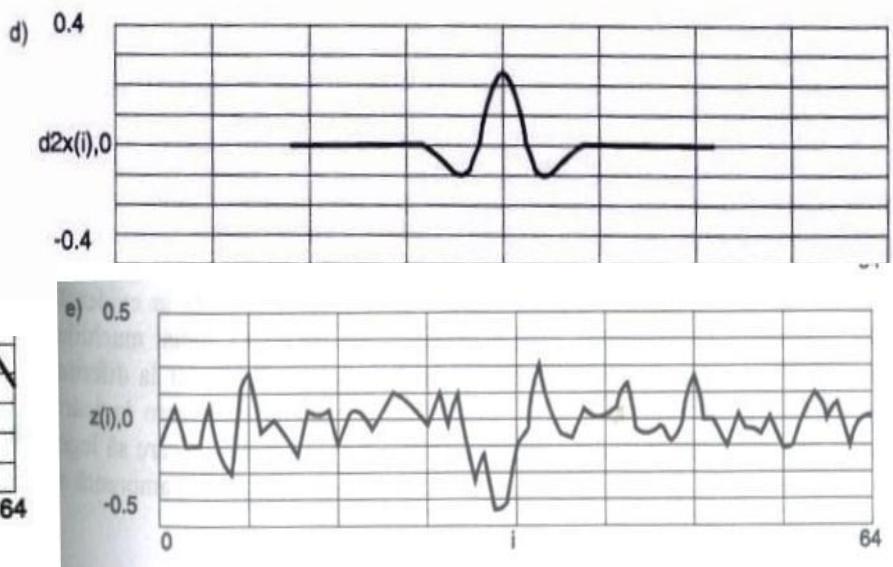


EDGE BASED SEGMENTATION

Canny – examples



- a) step signal summed with white noise of mean=0 and deviation 0.5
b) first order derivative of the Gaussian
c) filter response
d) second order derivative of the Gaussian
e) the filter's response





EDGE BASED SEGMENTATION

Maximum of the gradient method

1. For each filtered image point the modulus and direction of the gradient are evaluated
2. The edge points are detected as local extremes of the gradient modulus in the gradient's direction

The filtered image:

$$G * I = I_f(x, y) = \iint I(x - \zeta, y - \eta) \cdot G(\zeta, \eta) \cdot d\zeta \cdot d\eta = \sum_{\zeta} \sum_{\eta} I(x - \zeta, y - \eta) \cdot g(\zeta) \cdot g(\eta)$$

$$\frac{\partial I_f(x, y)}{\partial x} = \iint I(\zeta, \eta) \cdot g_x(x - \zeta) \cdot g(y - \eta) d\zeta \cdot d\eta = g_x(x) * I(x, y) * g(y)$$

$$\frac{\partial I_f(x, y)}{\partial y} = \iint I(\zeta, \eta) \cdot g(x - \zeta) \cdot g_y(y - \eta) d\zeta \cdot d\eta = g_y(y) * I(x, y) * g(x)$$

the gradient: $\nabla I_f(x, y) = \frac{\partial I_f(x, y)}{\partial x} \vec{i} + \frac{\partial I_f(x, y)}{\partial y} \vec{j}$



EDGE BASED SEGMENTATION

Maximum of the gradient method

the gradient magnitude:

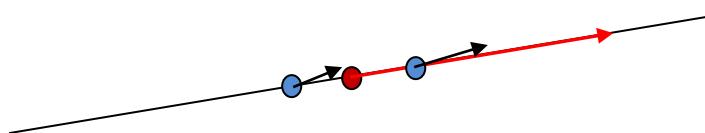
$$|\nabla I_f(x, y)| = \sqrt{\left(\frac{\partial I_f(x, y)}{\partial x}\right)^2 + \left(\frac{\partial I_f(x, y)}{\partial y}\right)^2} \quad \text{and}$$

the gradient orientation:

$$\alpha = \arctan \frac{\frac{\partial I_f(x, y)}{\partial y}}{\frac{\partial I_f(x, y)}{\partial x}},$$

- A point is a local maximum of the gradient modulus if:

The modulus of its gradient is greater than the modulus of the gradient of two opposite points located in the direction of the gradient, at equal distance





EDGE BASED SEGMENTATION

Zero crossing of the Laplacian

1. The image is filtered with a LoG kernel representing the composition of a Gaussian low pass filter and a Laplacian 2nd order derivative filter

2. The edge points are detected as the zero crossings of the filtered image



Zero crossing of the Laplacian

The *Gaussian distribution* function in two variables: $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

- The shape of the distribution and hence the amount of smoothing can be controlled by varying σ .
- In order to smooth an image $f(x,y)$, we convolve it with $G(x,y)$ to produce a smoothed image $s(x,y) = f(x,y) * G(x,y)$.
- Having smoothed the image with a Gaussian operator we can now take the Laplacian of the smoothed image:
- Therefore the total operation of edge detection after smoothing on the original image is: $\nabla^2(f(x,y) * g(x,y))$.
- This method of edge detection was first proposed by Marr and Hildreth at MIT who introduced the principle of the *zero-crossing* method.



Zero crossing of the Laplacian

- The LoG (Laplacian of Gaussian) kernel:

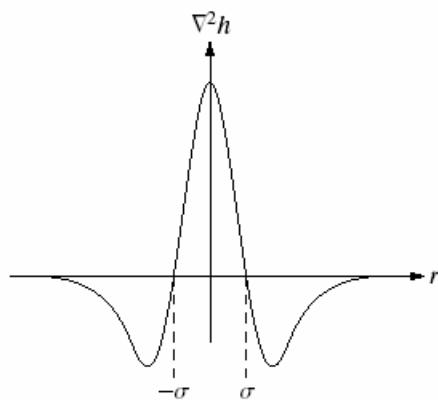
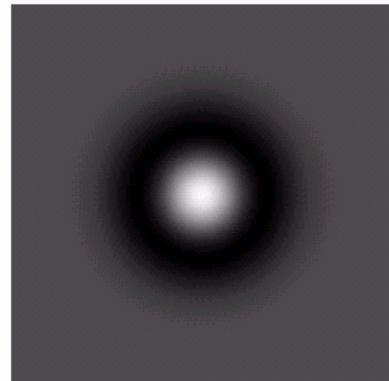
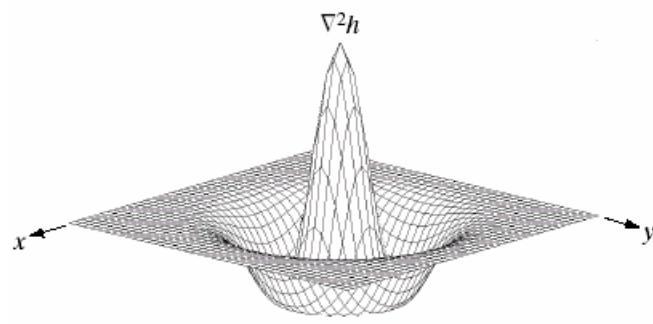
$$\nabla^2 \cdot G = \frac{x^2 + y^2 - 2 \cdot \sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Where σ is the standard deviation of the Gaussian and it determines the scale of the filter
- The width (in pixels) of the central region of the filter is, $w = \sigma \cdot 2 \cdot \sqrt{2}$
- The useful width of the filter is $s = 3w$
- A low value for s will emphasize many edges, a high value emphasizes only high variations in intensity



EDGE BASED SEGMENTATION

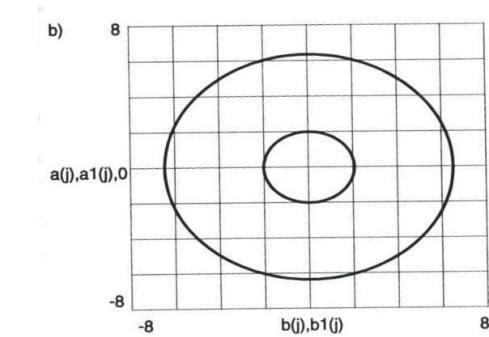
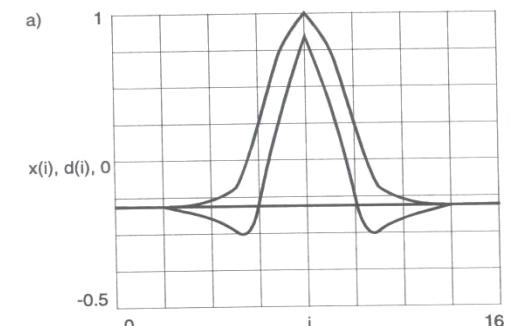
Zero crossing of the Laplacian



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

a b
c d

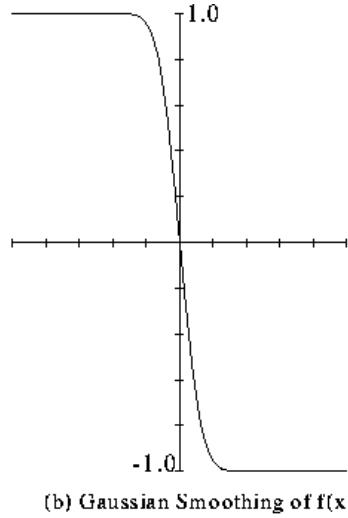
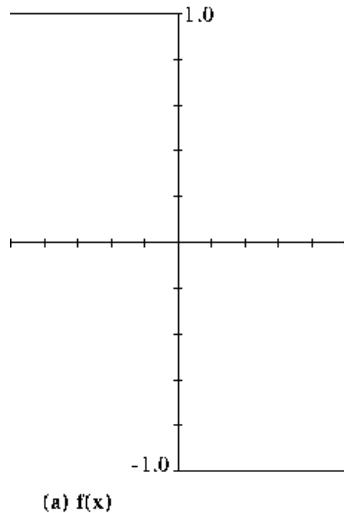
FIGURE 10.14
Laplacian of a Gaussian (LoG).
(a) 3-D plot.
(b) Image (black is negative, gray is the zero plane, and white is positive).
(c) Cross section showing zero crossings.
(d) 5×5 mask approximation to the shape of (a).



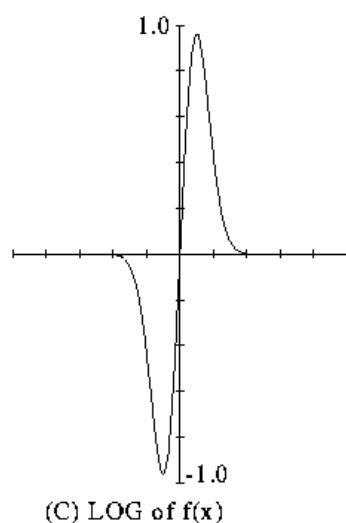


EDGE BASED SEGMENTATION

Zero crossing of the Laplacian



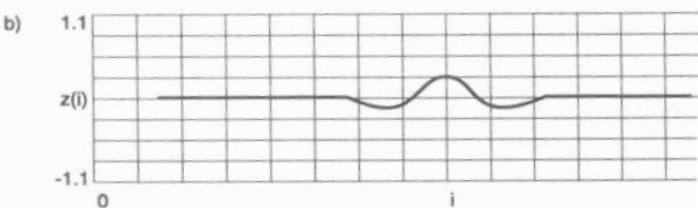
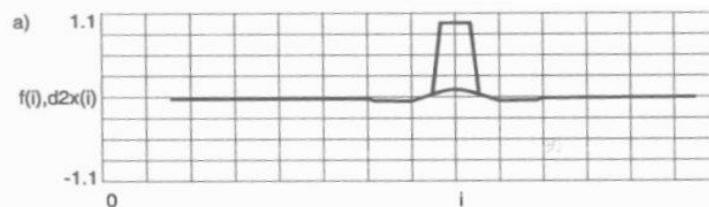
The basic principle of this method is to find the position in an image where the second derivatives become zero. These positions correspond to edge positions as shown in the figure:



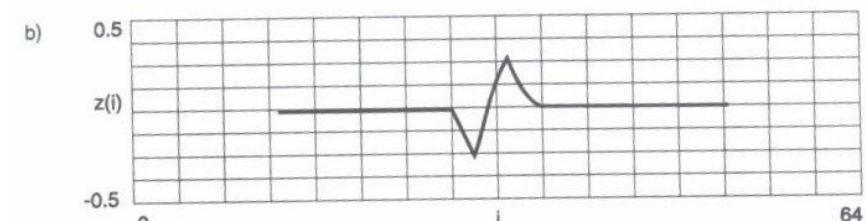
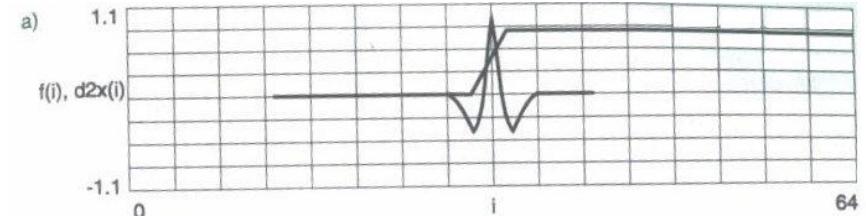


EDGE BASED SEGMENTATION

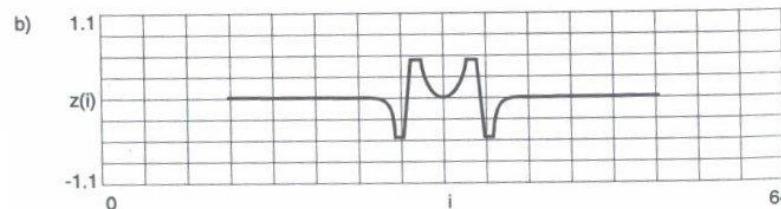
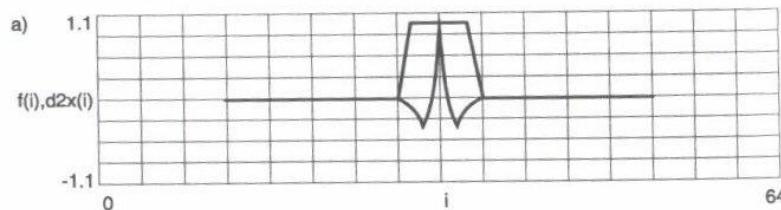
LOG for different edge types



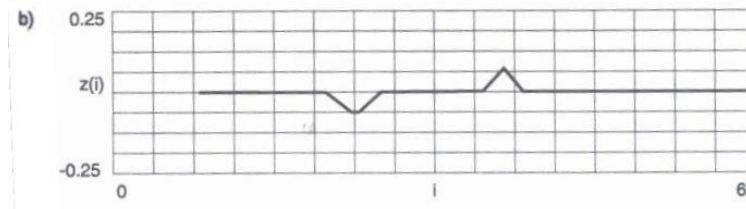
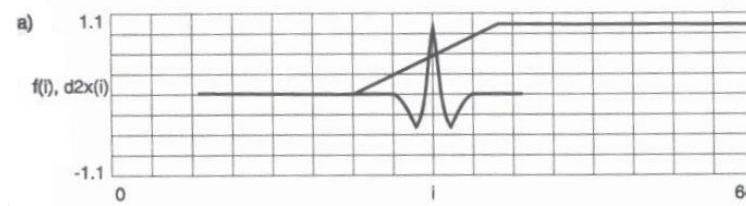
(a) muchie impuls de lățime mai mică decât w ($\sigma=3$);
(b) răspunsul filtrului LoG, se observă deplasarea
trecerilor prin zero cu $(w-d)/2$.



(a) semnal rampă cu lățimea mai mică decât lățimea operatorului s ($\sigma=1$);
(b) răspunsul filtrului LoG, se observă o trecere prin zero corespunzătoare
mijlocului rampei.



(a) muchie impuls, cu lățimea mai mare decât w și mai mică decât s ($\sigma=1$);
(b) răspunsul filtrului LoG, se observă influența reciprocă dintre muchii
care se menține atâtă timp cât lățimea impulsului este mai mică decât s .



(a) muchie rampă cu lățimea mai mare decât dimensiunea operatorului ($\sigma=1$);
(b) răspunsul filtrului LoG nu pune în evidență treceri prin zero, este
necesar ca lățimea rampei să fie mai mică decât lățimea operatorului.



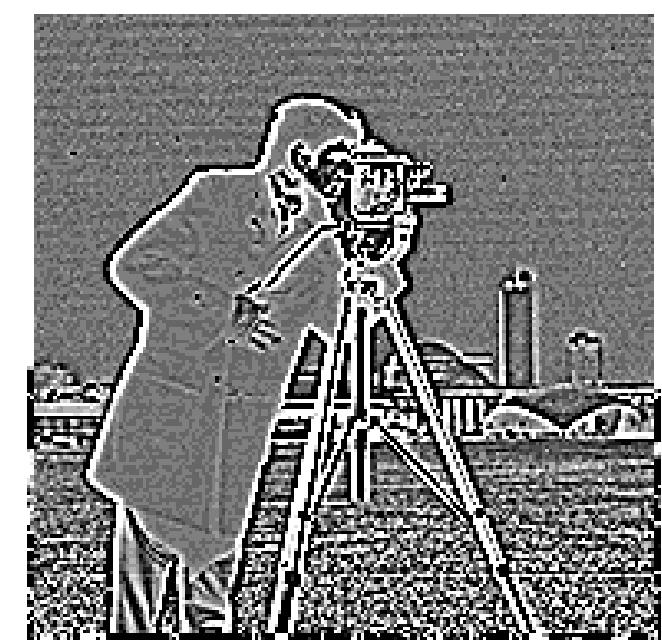
EDGE BASED SEGMENTATION



Original Image



LoG



LoG - Shrunked



EDGE BASED SEGMENTATION

Zero crossing of the second order directional derivative, in the direction of the gradient

1. Filter the image with a Gaussian kernel, G
2. The edge points are the zero-crossings of the 2nd order directional derivative in the direction of the gradient

The 1st order directional derivative in the direction of the gradient:

$$G_n = \frac{\partial G}{\partial n} = n \cdot \nabla G \text{ where}$$

- $n = \frac{\nabla(G * I)}{|\nabla(G * I)|}$, and
- $\nabla(G * I) = \frac{\partial I_f(x, y)}{\partial x} \vec{i} + \frac{\partial I_f(x, y)}{\partial y} \vec{j}$

An edge point is a local maximum in the direction n of the operator G_n applied to the image.



EDGE BASED SEGMENTATION

**Zero crossing of the second order directional derivative,
in the direction of the gradient**

For a local maximum: $\frac{\partial}{\partial n} G_n * I = 0$ or $\frac{\partial^2}{\partial n^2} G * I = 0$

$$\frac{\partial^2}{\partial n^2} G * I = \frac{\frac{\partial^2 I_f}{\partial x^2} \cdot (\frac{\partial I_f}{\partial x})^2 + 2 \cdot \frac{\partial I_f}{\partial x} \cdot \frac{\partial I_f}{\partial y} \cdot \frac{\partial^2 I_f}{\partial x \partial y} + \frac{\partial^2 I_f}{\partial y^2} \cdot (\frac{\partial I_f}{\partial y})^2}{(\frac{\partial I_f}{\partial x})^2 + (\frac{\partial I_f}{\partial y})^2}$$

The zero crossings of the 2nd order derivative represent a change of sign
in the resulted image



EDGE BASED SEGMENTATION

Mathematics for 2nd order derivative

$$\frac{\partial^2 I_f(x, y)}{\partial x^2} = \iint I(\zeta, \eta) \bullet g''_{xx}(x - \zeta) \bullet g(y - \eta) d\zeta d\eta =$$
$$\sum_{\zeta} \sum_{\eta} I(\zeta, \eta) \bullet g''_{xx}(x - \zeta) \bullet g(y - \eta) = \sum_{\zeta} g''_{xx}(x - \zeta) \sum_{\eta} I(\zeta, \eta) \bullet g(y - \eta) =$$
$$g''_{xx}(x) * I(x, y) * g(y)$$

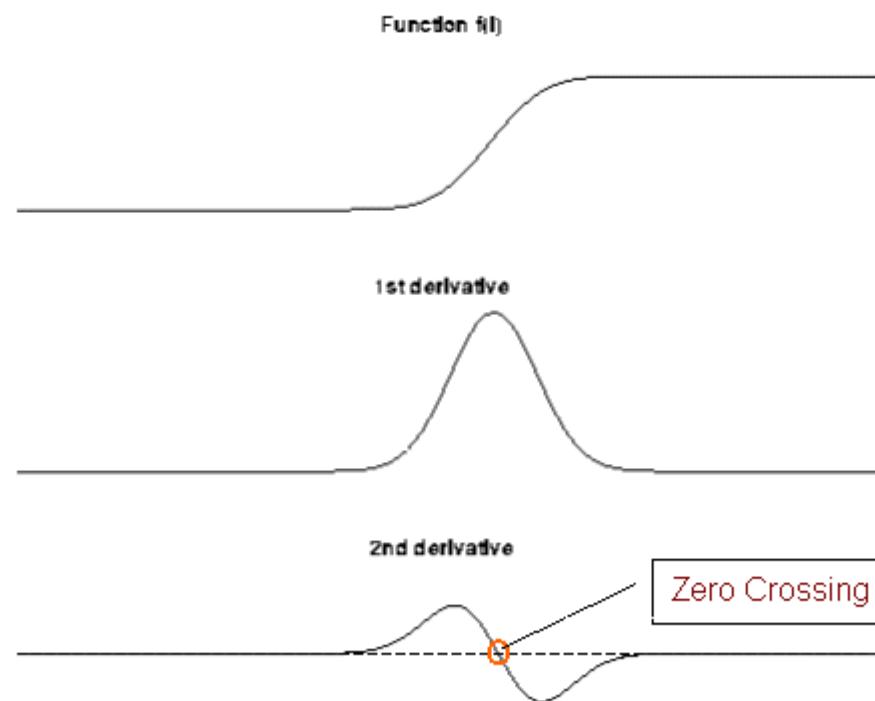
$$\frac{\partial^2 I_f(x, y)}{\partial y^2} = \iint I(\zeta, \eta) \bullet g(x - \zeta) \bullet g''_{yy}(y - \eta) d\zeta d\eta =$$
$$\sum_{\zeta} \sum_{\eta} I(\zeta, \eta) \bullet g(x - \zeta) \bullet g''_{yy}(y - \eta) = \sum_{\eta} g''_{yy}(y - \eta) \sum_{\zeta} I(\zeta, \eta) \bullet g(x - \zeta) =$$
$$g''_{yy}(y) * I(x, y) * g(x)$$

$$\frac{\partial^2 I_f(x, y)}{\partial x \partial y} = \iint I(\zeta, \eta) \bullet g'_x(x - \zeta) \bullet g'_y(y - \eta) d\zeta d\eta =$$
$$\sum_{\zeta} \sum_{\eta} I(\zeta, \eta) \bullet g'_x(x - \zeta) \bullet g'_y(y - \eta) = \sum_{\zeta} g'_x(x - \zeta) \sum_{\eta} I(\zeta, \eta) \bullet g'_y(y - \eta) =$$
$$g'_x(x) * I(x, y) * g'_y(y)$$



EDGE BASED SEGMENTATION

Zero crossing of the second order directional derivative





Contents

- Edge based segmentation, edge detection
- Methods based on filtering followed by differential operators
- **Edge extraction**
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy



EDGE BASED SEGMENTATION

Edge extraction

- Edge detection defines some thick contours of the real edges, and because of noise, it introduces false edges
- For high level processing, one needs to mark the edge by a curve of 1 pixel width, very close to the real position of the edge
- Methods:
 - Gradient variation:
 1. extraction of local extremes of the gradient
 2. discrimination of the local extremes
 - Zero crossing of the Laplacian or the 2nd order derivative
 1. extraction of the zero crossings
 2. discrimination of the zero crossings



The gradient method – extraction of the local gradient extremes

- Let M a point in the image, having ∇M , d a given distance
- M_1 and M_2 points on a line passing through M and having the direction of the gradient of M, located in opposite positions with respect to M
- $M_1 = M + d \cdot \frac{\nabla M}{|\nabla M|}$ and $M_2 = M - d \cdot \frac{\nabla M}{|\nabla M|}$
- M is selected if $|\nabla M| \geq |\nabla M_1|$ and $|\nabla M| > |\nabla M_2|$



The gradient method – discrimination gradient extremes

- **Hysteresis thresholding:** select all extremes that have the gradient greater than a low threshold t_l and form a connected path to an extreme that has the value greater than a high threshold t_h
- **Algorithm:**
 1. Find the binary images f_h and f_l obtained by thresholding the extremes of the gradient with t_h and t_l and compute the image $f_p = f_l - f_h$
 2. Expand all the non-zero points in f_h to form connected chains with all the non-zero points of f_p (i.e. find the adjacency graph of the non-zero points in f_p and select the connected components with non-zero points in f_h)
 3. Select all the connected components that have a length greater than l_{min}



EDGE BASED SEGMENTATION

Zero crossings methods

1. Extraction of the zero crossings:

Let A a point of coordinates (x, y) , having as neighbors $B(x+1, y)$, $C(x, y+1)$, $D(x, y-1)$, $E(x-1, y)$

- If $A \cdot B < 0$ and $A \cdot C < 0$, then there is a zero-crossing of coordinates $[x + \text{interpolation}(A,B), y + \text{interpolation}(A,C)]$
- If $A \cdot B < 0$ and $A \cdot C >= 0$ then there is a zero-crossing of coordinates $[x + \text{interpolation}(A,B), y]$
- If $A \cdot B >= 0$ and $A \cdot C < 0$ then there is a zero-crossing of coordinates $[x, y + \text{interpolation}(A,C)]$

Interpolation(A,B) returns a value in $[0,1]$, based on the linear interpolation of the zero point between A and B

2. Discrimination of the zero crossings

eliminates the zero-crossings generated by noise:

- In the case of the zero crossings of the LoG use the slope condition or the gradient's amplitude for discrimination.
- In the case of the 2nd order directional derivative a first condition is the similitude of the gradient's direction and the transition direction.



Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- **Edge closing**
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy



EDGE BASED SEGMENTATION

Edge closing

For eliminating false edges use high thresholds and apply an algorithm for tracking and closing the contours.

Algorithm: identifies the end points and for each

1. it determines the direction of the contour (the tangent to the contour in the end point)
2. select the point having the maximum gradient compared to the gradient values of 3 neighboring pixels, in the direction of the contour with a deviation of $+/- 45^0$
3. check the stop conditions
4. if the stop conditions are not satisfied, go to step 1 otherwise go to another end point

The stop conditions are:

- no more points for which to continue the search
- contour is closed
- obtain a maximum length



EDGE BASED SEGMENTATION

Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy



EDGE BASED SEGMENTATION

**Algorithm for edge detection, tracking and extraction,
based on the zero crossings of the 2nd order derivative,
with sub-pixel accuracy**

Steps:

1. Filter the image
2. Find the partial derivatives of 1st and 2nd order
3. Compute the 2nd directional derivative in the gradient's direction
4. Detect the zero-crossings at sub-pixel level
5. Track, extract and close the contours

The last two steps are done simultaneously. They are generated by the detection of a start point obtained by the systematic scanning of the 2nd order directional derivative.



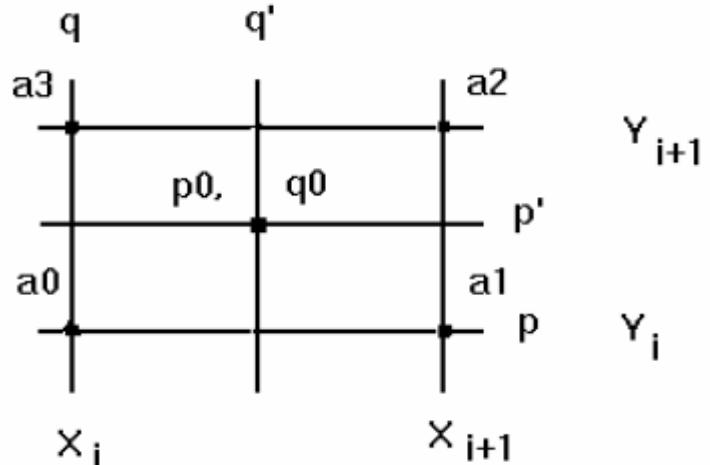
Algorithm (2)

- The starting points obey the conditions:
 - A change of the sign in the 2nd order directional derivative between the current point and the next point
 - The point should not have been marked previously
 - The gradient should be greater than a threshold
 - The direction of the gradient should coincide with the sense in which the 2nd order directional derivative changes
- From the start point one calls the following procedures:
 - Detect zero crossings
 - Track the contour
 - Extract the edges



EDGE BASED SEGMENTATION

Algorithm – detect zero crossings



$$a_0 = \frac{\partial^2 I_f(x_i, y_i)}{\partial n^2} \quad a_1 = \frac{\partial^2 I_f(x_{i+1}, y_i)}{\partial n^2}$$

$$a_2 = \frac{\partial^2 I_f(x_i + 1, y_i + 1)}{\partial n^2} \quad a_3 = \frac{\partial^2 I_f(x_i, y_{i+1})}{\partial n^2}$$

The bilinear approximation of $f(x, y)$ in the 2×2 cell of pixels, in the directions:

- 1 dir. Y_i : $f(x, y) = f(x_i, y_i) + [f(x_{i+1}, y_i) - f(x_i, y_i)]x = z_1(x)$
- 2 dir. y_{i+1} : $f(x, y) = f(x_i, y_{i+1}) + [f(x_{i+1}, y_{i+1}) - f(x_i, y_{i+1})]y = z_2(y)$
- 3 dir. X : $f(x, y) = [z_2(y) - z_1(x)]y + z_1(x)$

$$f(x, y) = f(x_i + p, y_i + q) \approx g(p, q) = (1-p)(1-q)a_0 + p(1-q)a_1 + q(1-p)a_3 + pqa_2, \text{ where } p, q \in [0, 1]$$

The zero crossings depend on the sign of $s_0 = a_0 \cdot a_1$, $s_1 = a_1 \cdot a_2$, $s_2 = a_2 \cdot a_3$, $s_3 = a_3 \cdot a_0$

1. If all the products are >0 , the function has no zero crossings in the interpolation interval
2. If two products are <0 then there is a single contour line that intersects the corresponding edges
3. If all products are negative then the cell is intersected by two distinct contours.

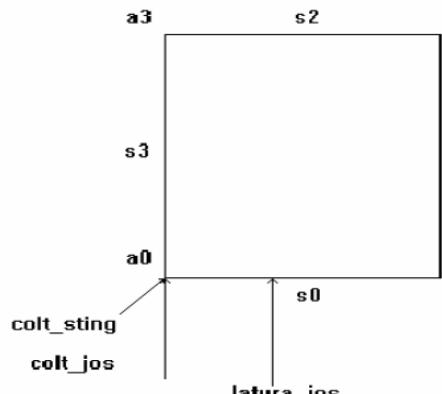
Let $D = a_1 \cdot a_3 - a_0 \cdot a_2$

- For $D > 0$, a contour intersects the cell through the edges 0 and 3, and another through 1, 2
- For $D < 0$, a contour intersects the cell through the edges 0 and 1, and another through 2, 3
- If $D = 0$, the two contours intersect and pass through opposite edges of the cell.

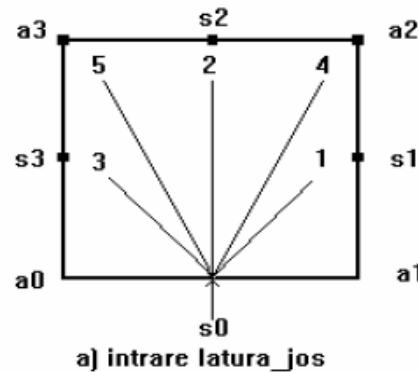


EDGE BASED SEGMENTATION

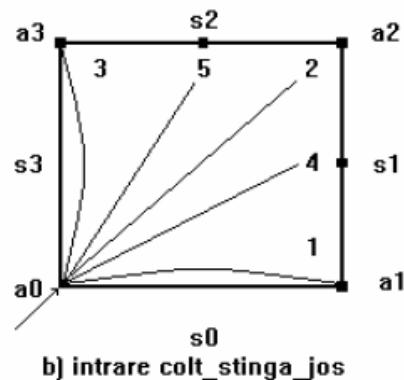
Algorithm – contour tracking and extraction



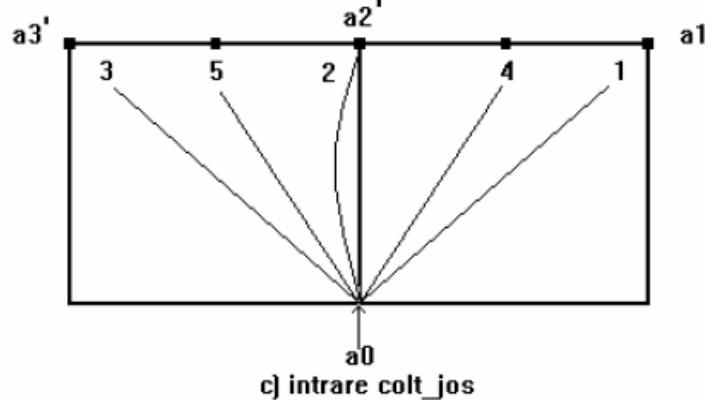
a₂ Cons. the entry point on S₀ or in node a₀.
Three entrance directions :
1. through edge
2. through a corner orthogonal on the edge
3. oblic in the corner



Exit points: s₀<0
1. if s₃≥0^s₁<0 v s₃<0^s₁<0^D<0
2. if s₃>0^s₁>0 v s₃<0^s₁<0^D=0
3. if s₃<0 ^ s₂≥0 v s₃<0^s₁<0^D>0
4. if s₃>0^s₁=0
5. if s₃=0 ^s₁>0



Exit points: a₀=s₀=s₃=0
1. if s₁=0^a₁=0^a₂≠0^a₃≠0
2. if s₁=0^a₁≠0^a₃≠0
3. if s₁>0 ^a₃=0
4. if s₁<0 ^ s₂>0
5. if s₁>0 ^ s₂<0

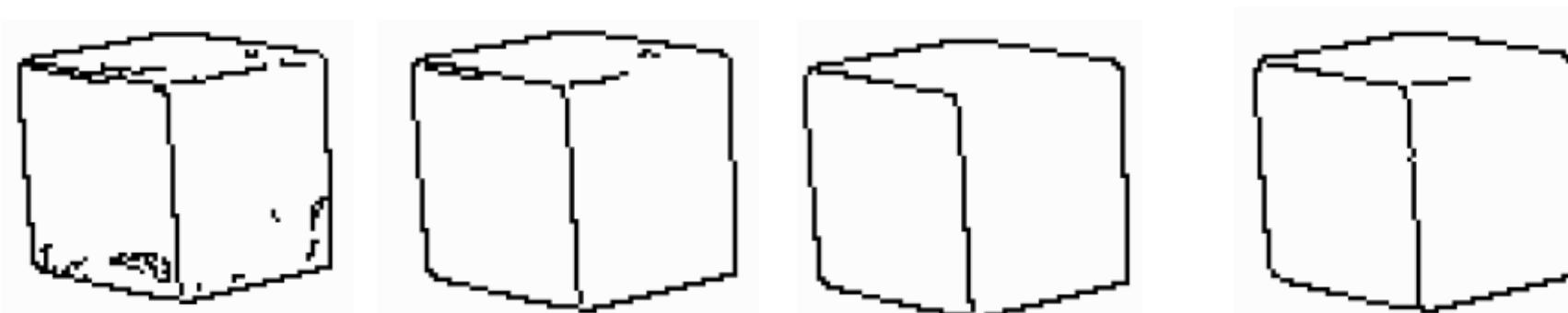




EDGE BASED SEGMENTATION

Algorithm – last step

- By the linear interpolation the exact coordinate of the edge point is found.
- The coordinates are maintained in a list of contour points. In the result matrix, the contour points are marked as checked points
- If the contour point is already marked as checked point, or if the gradient in that point is less than a low threshold, the contour extraction ends. (Only contours having a length greater than a minimum length, l_{min} are considered).
- The gradient threshold can be reduced greatly (as this algorithm is based on a rigorous tracking of the contour)





EDGE BASED SEGMENTATION

References

- [1] Sergiu Nedevschi: “Prelucrarea Imaginilor si Recunoasterea Formelor”,
Ed. Microinformatica, 1997
- [2] Emanuele Trucco, Alessandro Verri: Introductory Techniques for 3-D Computer Vision
- [3] Robert Collins, CSE486, Penn State, “Lecture 5: Gradients and Edge Detection”

Introduction to binocular stereo vision

Kristian Kirk
Computer Vision and Media Technology Laboratory (CVMT)
Aalborg University (AAU)

What is binocular stereo vision?

- A way of getting depth (3-D) information about a scene from two 2-D views (images) of the scene

What is binocular stereo vision?

- A way of getting depth (3-D) information about a scene from two 2-D views (images) of the scene
- Used by humans and animals

What is binocular stereo vision?

- A way of getting depth (3-D) information about a scene from two 2-D views (images) of the scene
- Used by humans and animals
- Computational stereo vision
 - Programming machines to do stereo vision
 - Studied extensively in the past 25 years
 - Difficult; still being researched

Purpose of this talk:

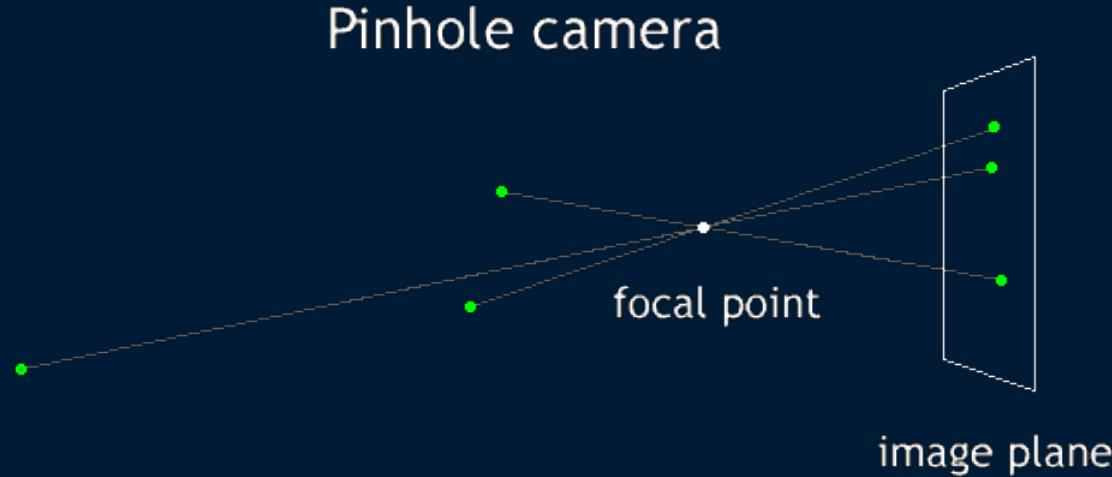
- An introduction to:
 - Basic principle of stereo vision
 - Computational stereo analysis
 - How does it work?
 - What is required?
 - Where are the difficulties?

Purpose of this talk:

- An introduction to:
 - Basic principle of stereo vision
 - Computational stereo analysis
 - How does it work?
 - What is required?
 - Where are the difficulties?

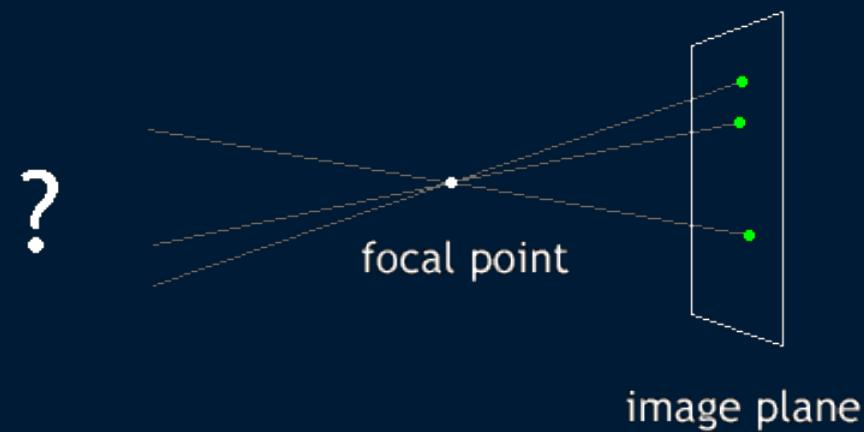
Fundamentals of stereo vision

- A camera model:
 - Models how 3-D scene points are transformed into 2-D image points
 - The pinhole camera: a simple linear model for perspective projection



Fundamentals of stereo vision

- The goal of stereo analysis:
 - The inverse process: From 2-D image coordinates to 3-D scene coordinates
 - Requires images from at least two views



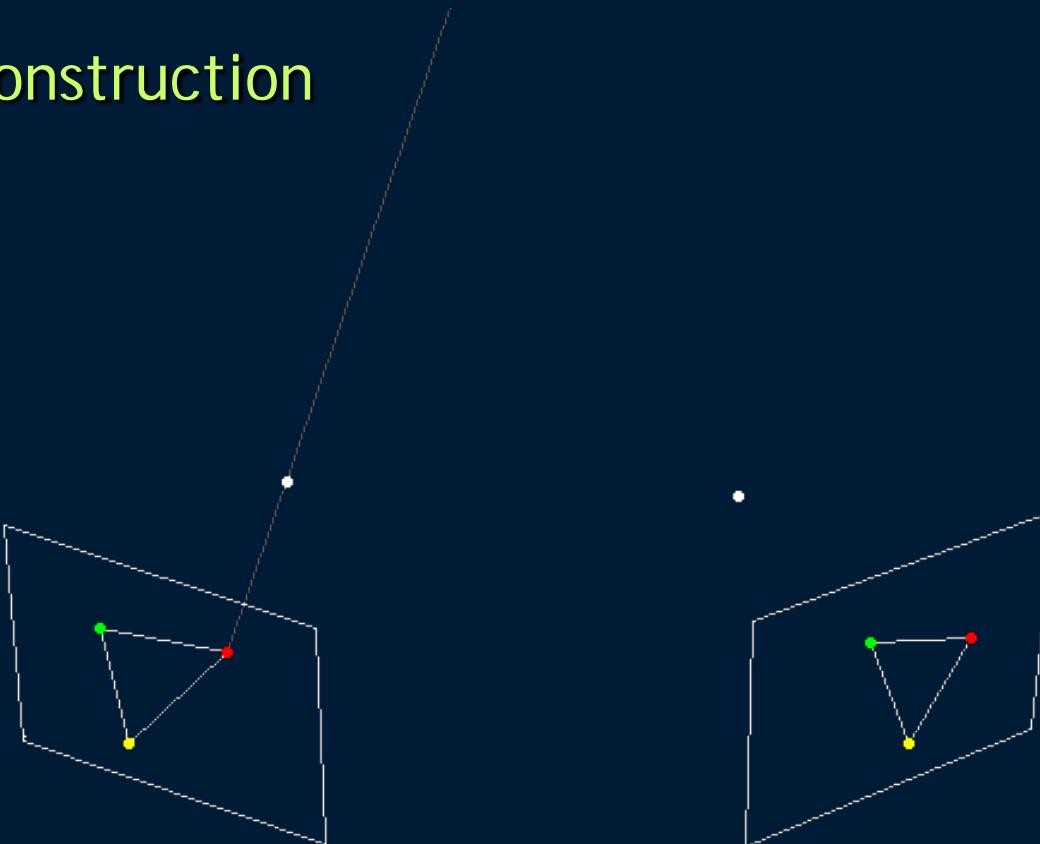
Fundamentals of stereo vision

- 3-D reconstruction



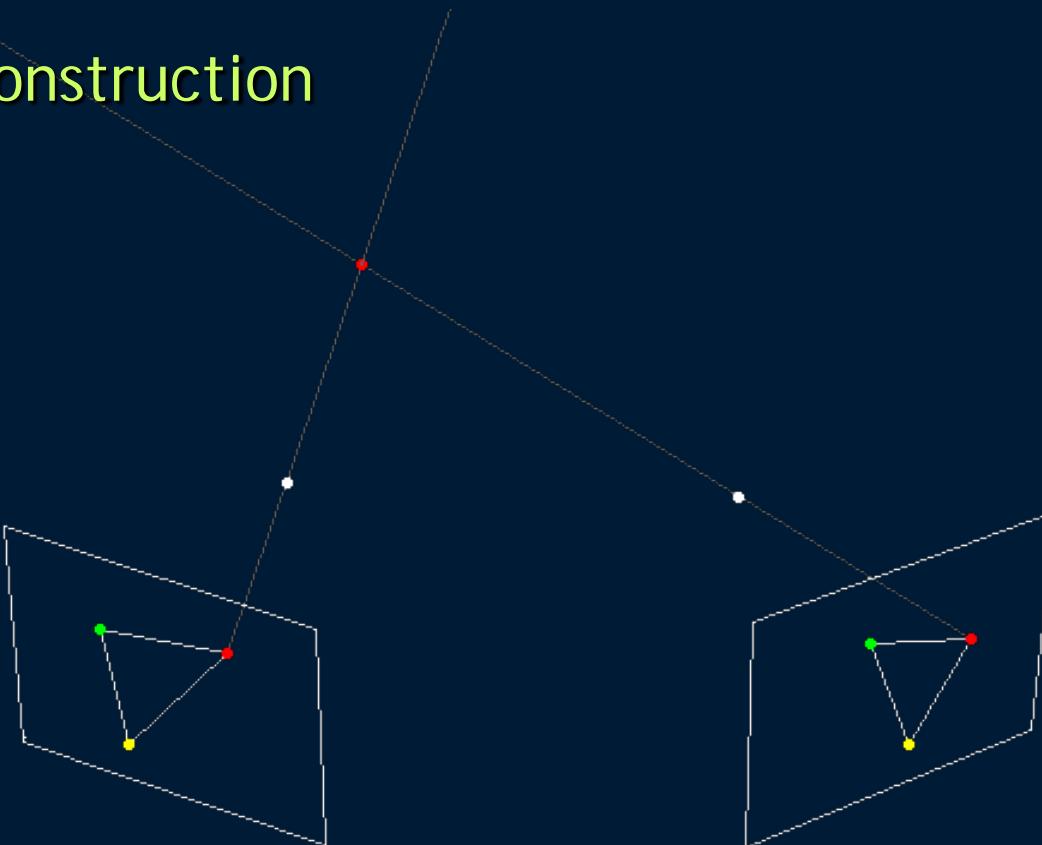
Fundamentals of stereo vision

- 3-D reconstruction



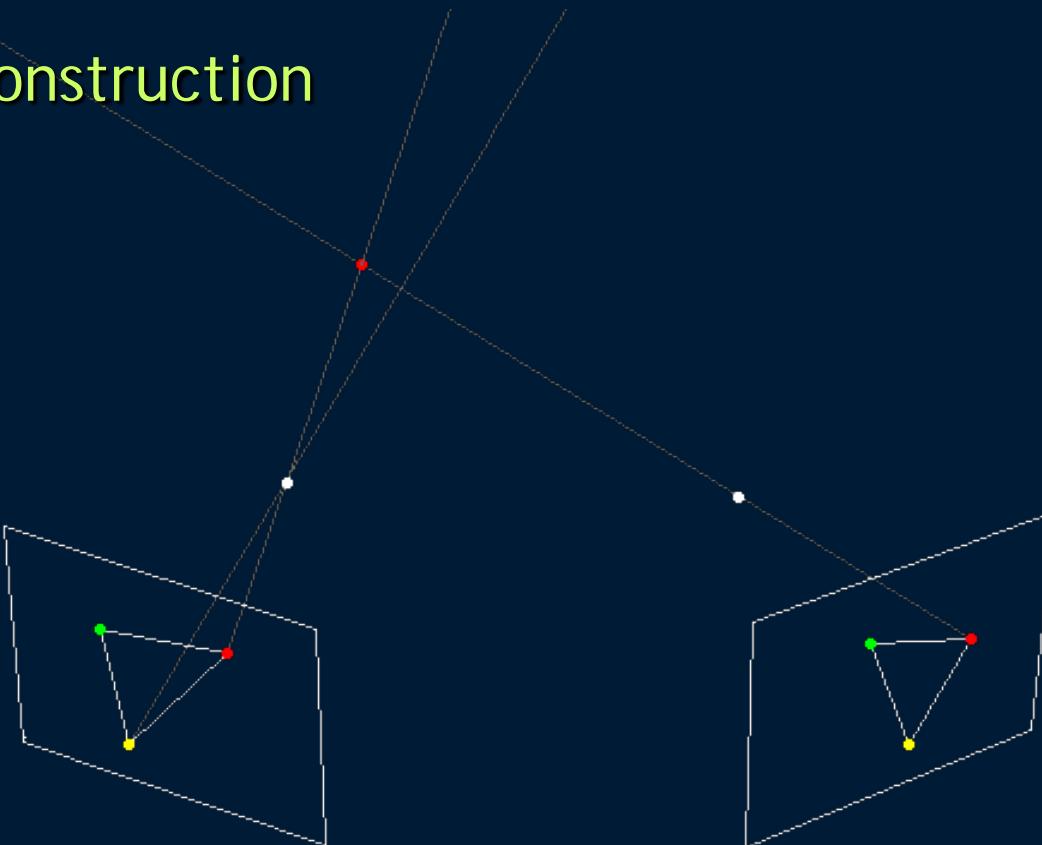
Fundamentals of stereo vision

- 3-D reconstruction



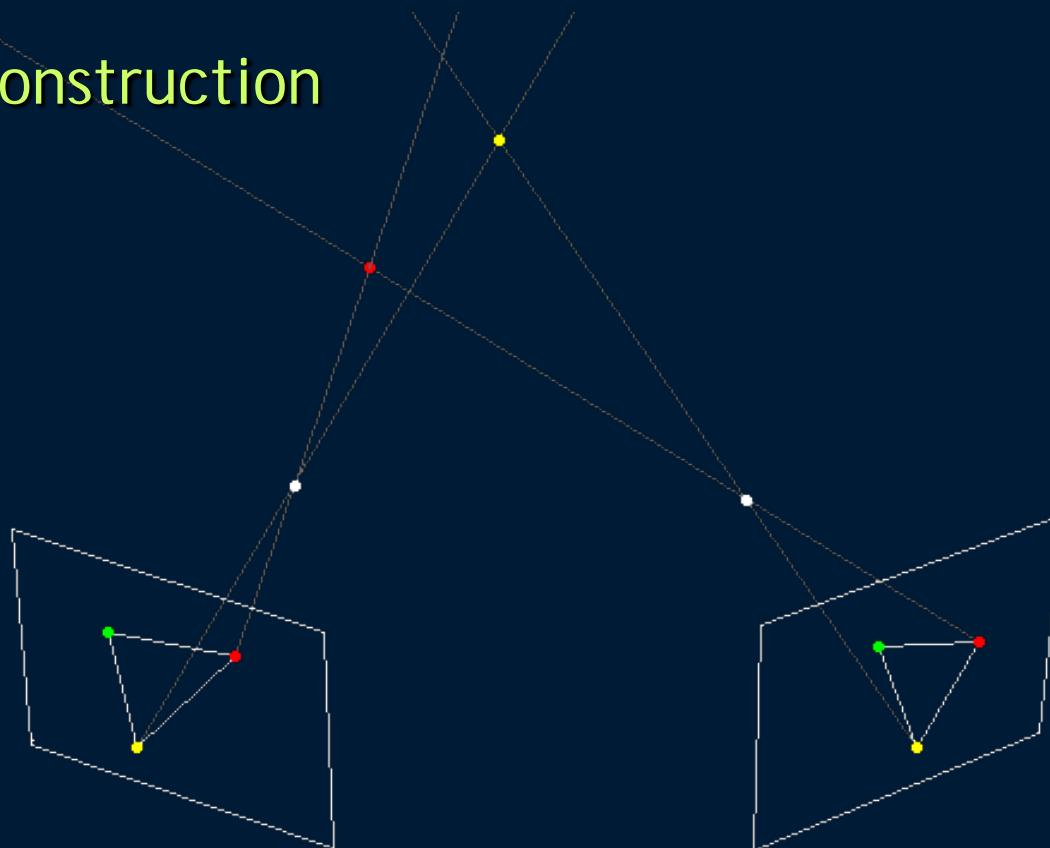
Fundamentals of stereo vision

- 3-D reconstruction



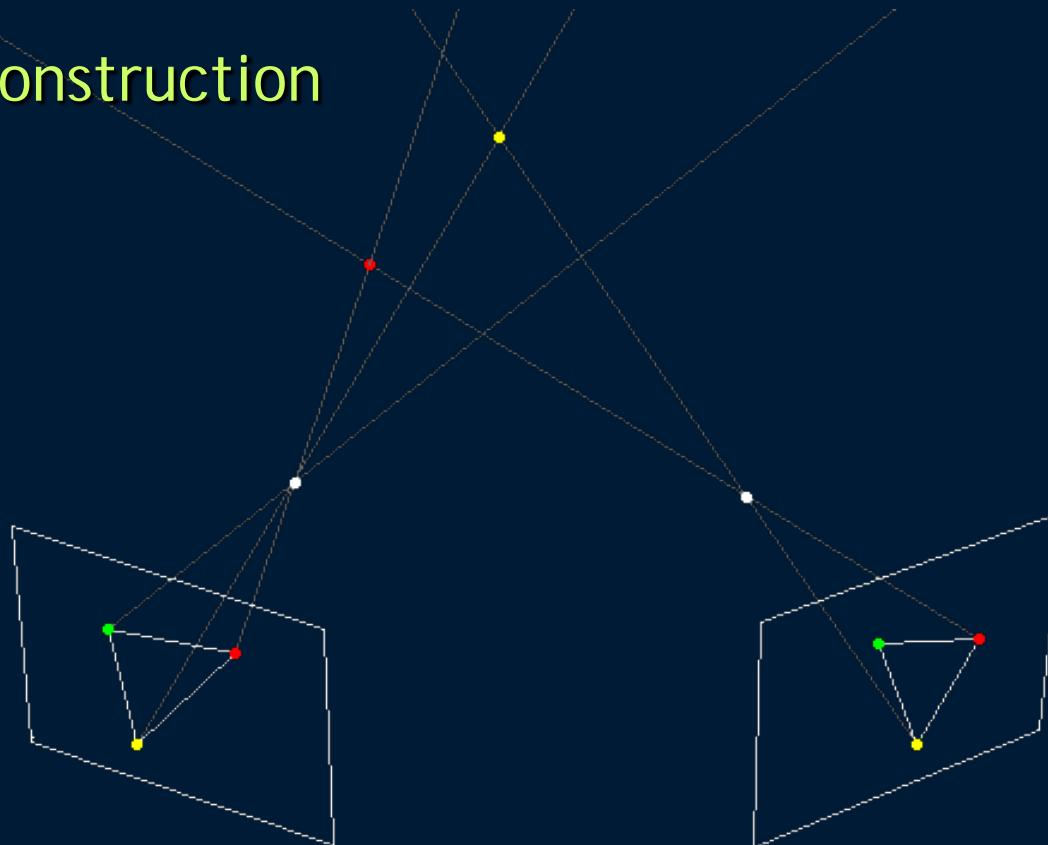
Fundamentals of stereo vision

- 3-D reconstruction



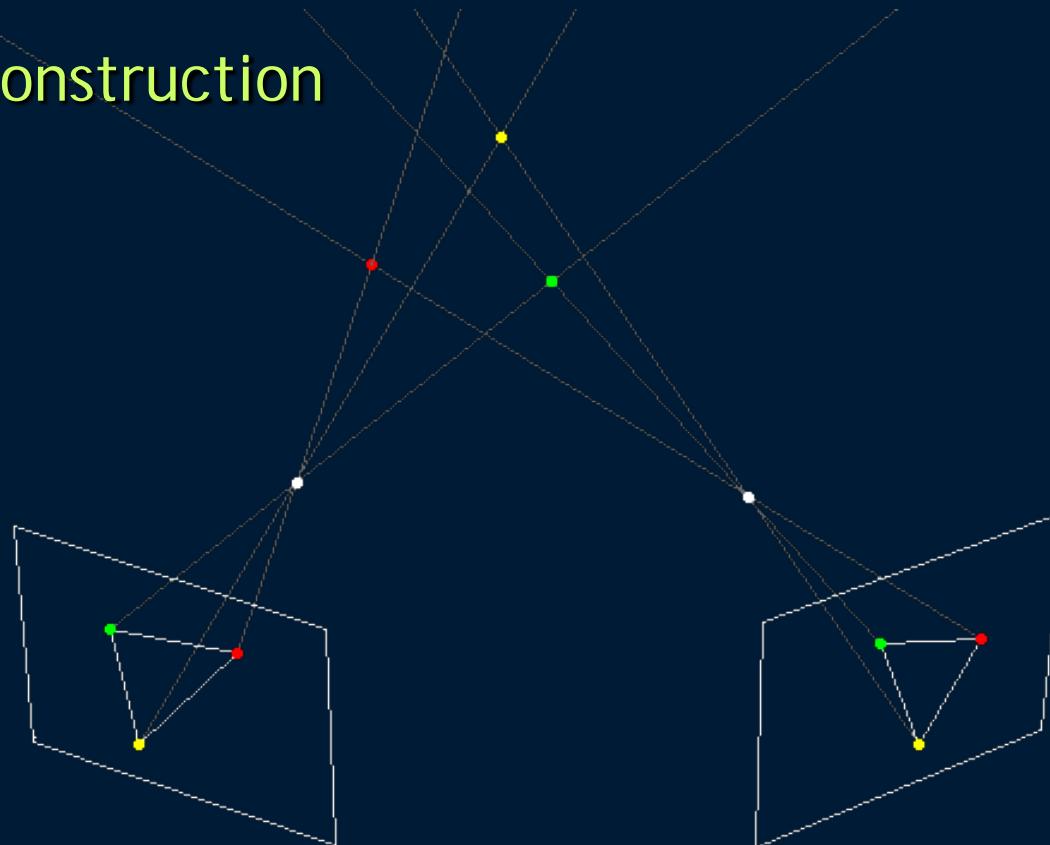
Fundamentals of stereo vision

- 3-D reconstruction



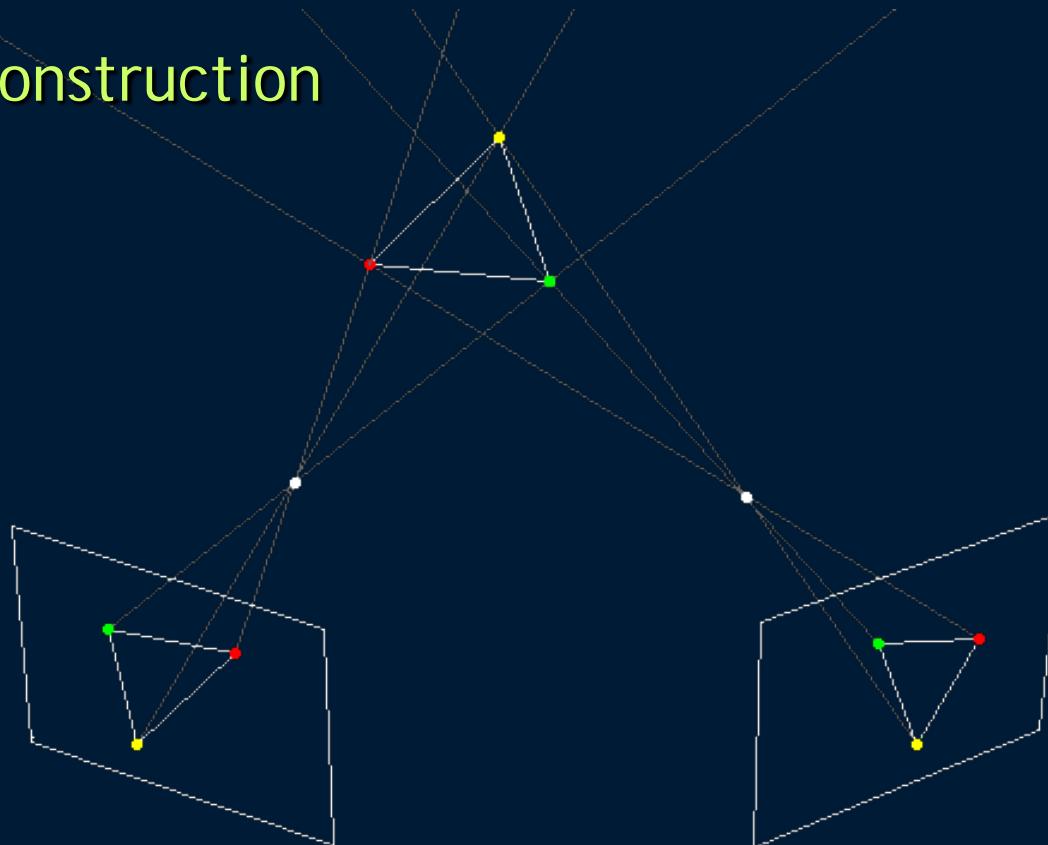
Fundamentals of stereo vision

- 3-D reconstruction



Fundamentals of stereo vision

- 3-D reconstruction

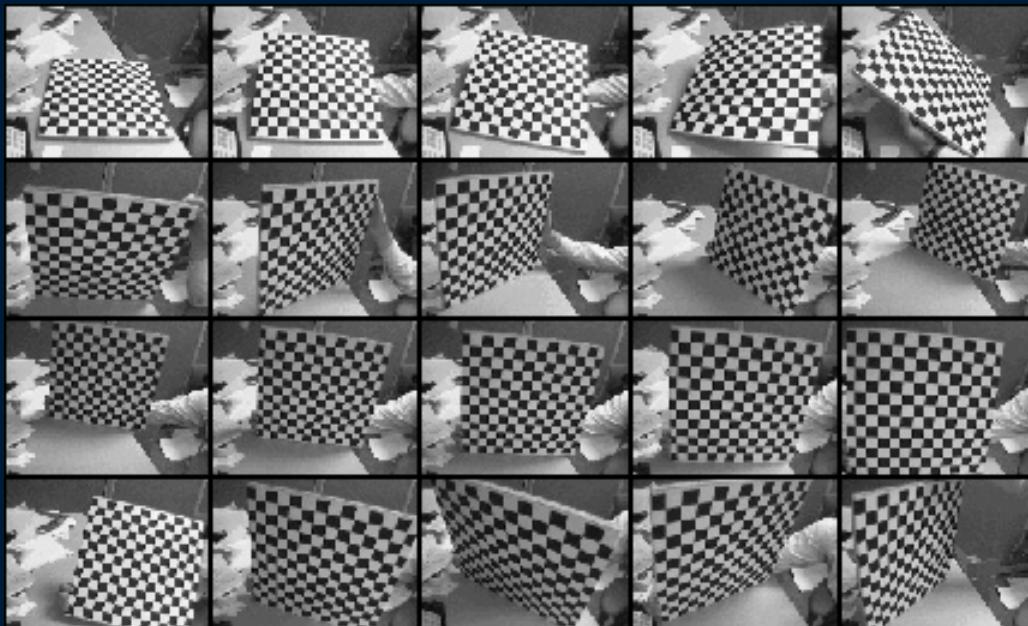


Prerequisites

- Camera model parameters must be known:
 - External parameters:
 - Positions, orientations
 - Internal parameters:
 - Focal length, image center, distortion, etc..

Prerequisites

- Camera calibration



Two subproblems

- Matching
 - Finding corresponding elements in the two images
- Reconstruction
 - Establishing 3-D coordinates from the 2-D image correspondences found during matching

Two subproblems

- Matching (hardest)
 - Finding corresponding elements in the two images
- Reconstruction
 - Establishing 3-D coordinates from the 2-D image correspondences found during matching

The matching problem

- Which image entities should be matched?
 - Two main approaches
 - Pixel/area-based (lower-level)
 - Feature-based (higher-level)



Matching challenges

- Scene elements do not always look the same in the two images
 - Camera-related problems
 - Image noise, differing gain, contrast, etc..
 - Viewpoint-related problems:
 - Perspective distortions
 - Occlusions
 - Specular reflections

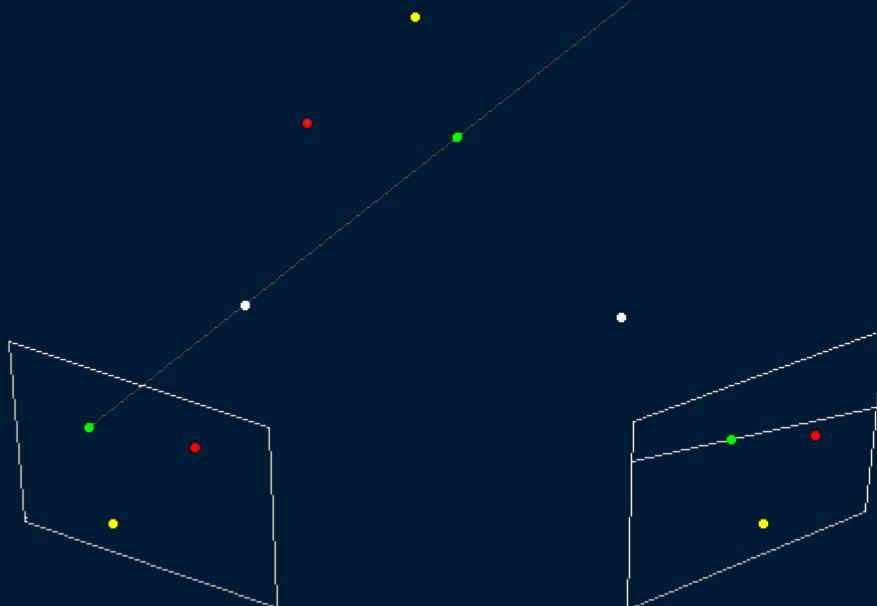


Choice of camera setup

- Baseline
 - distance between cameras (focal points)
- Trade-off
 - Small baseline: Matching easier
 - Large baseline: Depth precision better

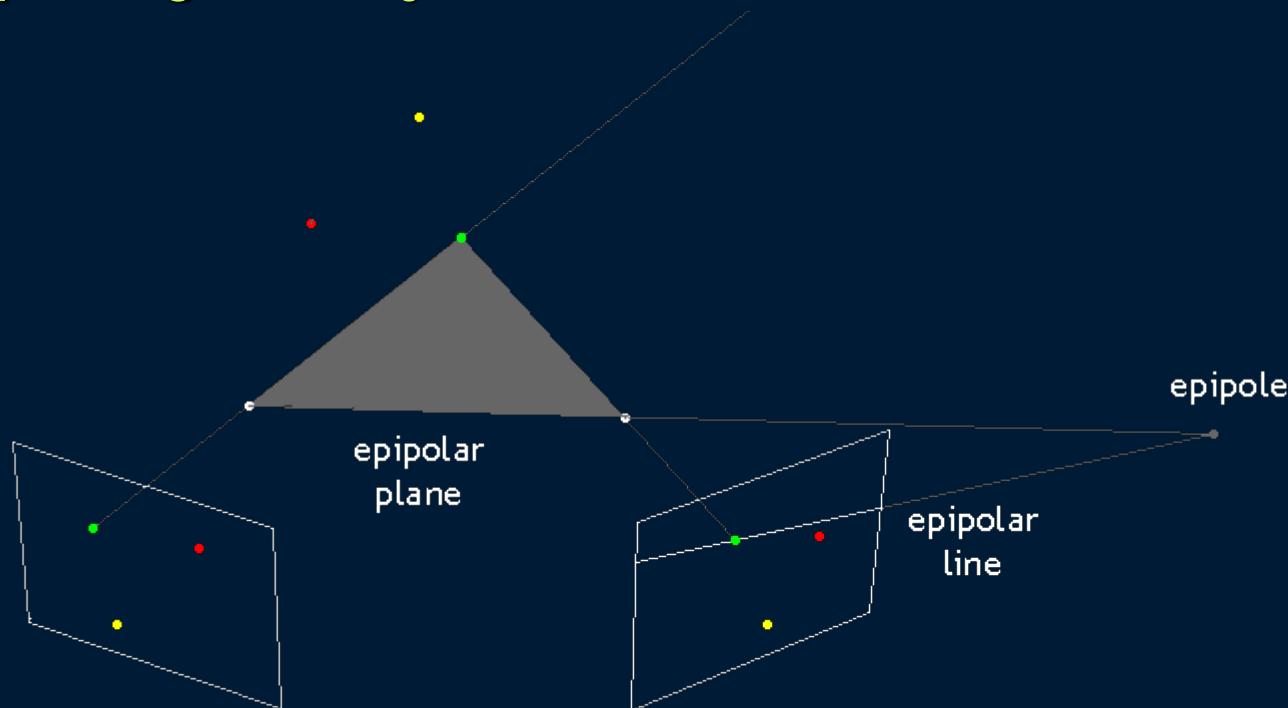
Matching clues

- Correspondance search is a 1-D problem
 - Matching point must lie on a line



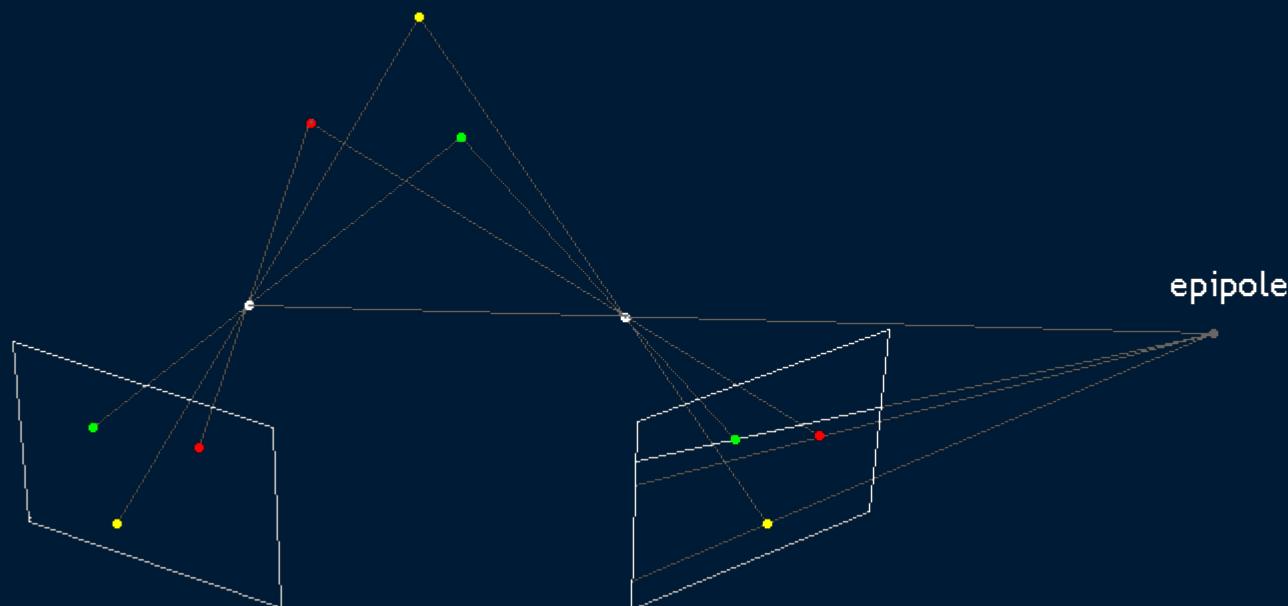
Matching clues

- Epipolar geometry



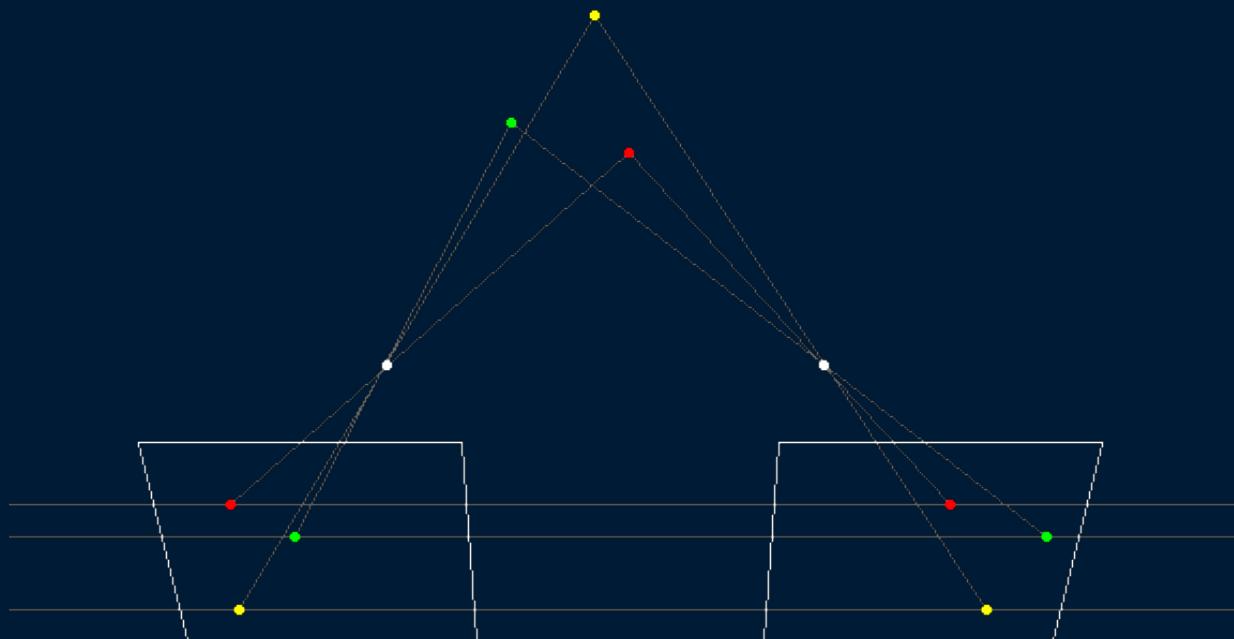
Matching clues

- Epipolar geometry



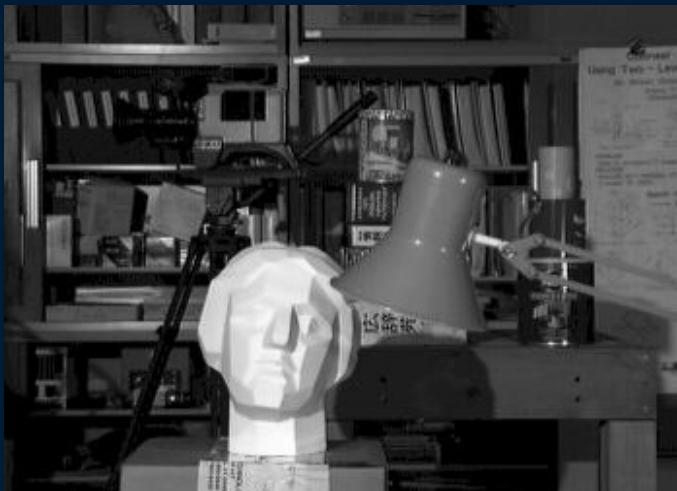
Rectification

- Simplifies the correspondance search
 - Makes all epipolar lines parallel and coincident
 - Corresponds to parallel camera configuration



Goal: disparity map

- Disparity:
 - The horizontal displacement between corresponding points
 - Closely related to scene depth

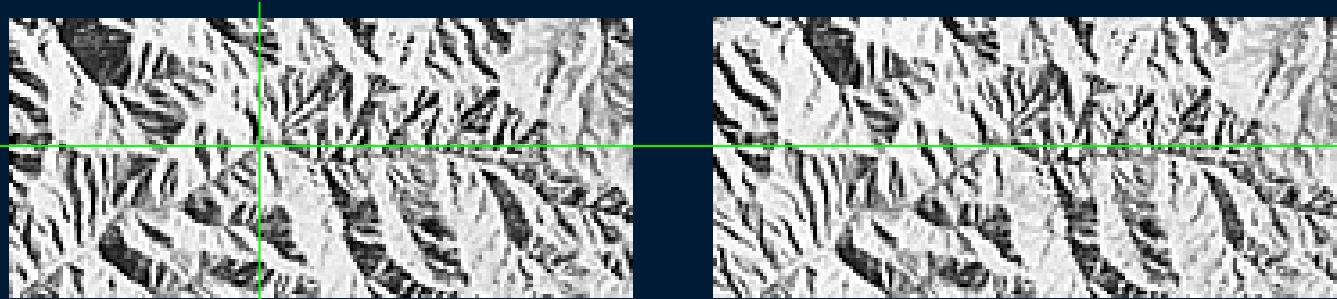


More matching heuristics

- Always valid:
 - (Epipolar line)
 - Uniqueness
 - Minimum/maximum disparity
- Sometimes valid:
 - Ordering
 - Local continuity (smoothness)

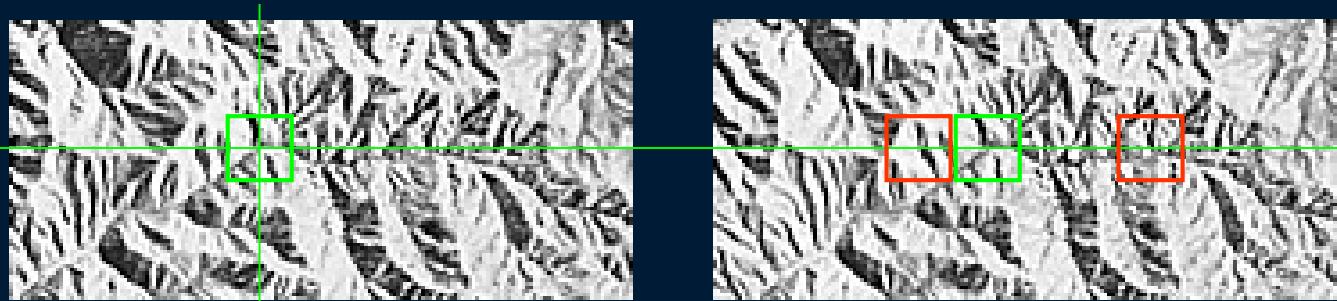
Area-based matching

- Finding pixel-to-pixel correspondences
 - For each pixel in the left image, search for the most similar pixel in the right image



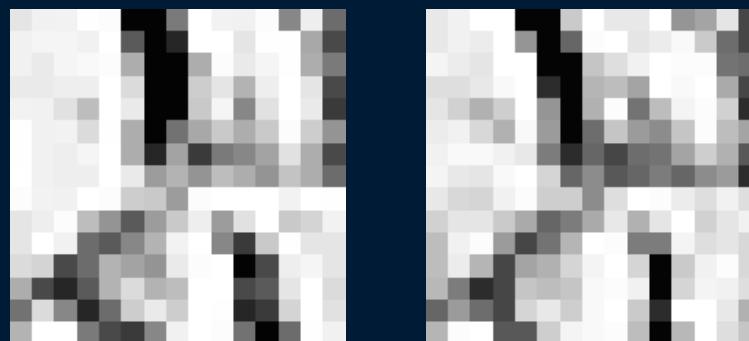
Area-based matching

- Finding pixel-to-pixel correspondences
 - For each pixel in the left image, search for the most similar pixel in the right image
 - Using neighbourhood windows



Area-based matching

- Similarity measures for two windows
 - SAD (sum of absolute differences)
 - SSD (sum of squared differences)
 - CC (cross-correlation)
 - ...

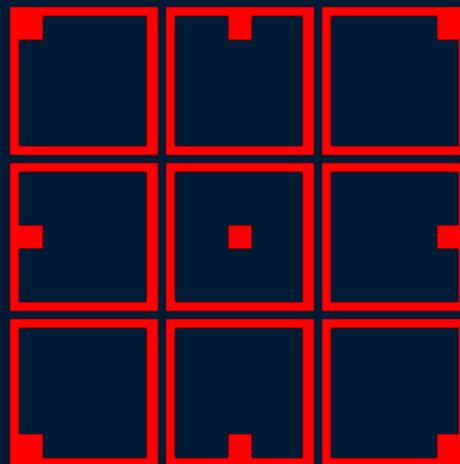


Area-based matching

- Choice of window size
 - Factors to consider:
 - Ambiguity
 - Noise sensitivity
 - Sensitivity towards viewpoint-related distortions
 - Expected object sizes
 - Frequency of depth jumps

Area-based matching

- Variable window position
 - Better matching at depth jumps (disparity edges)



Three or more viewpoints

- More matching information
 - Additional epipolar constraints
 - More confident matches

Summary

- Stereo vision:
 - A method for 3-D analysis of a scene using images from two or more viewpoints
- Two subproblems:
 - Matching
 - Reconstruction
- Most difficult part: Matching
- Area-based matching using windows
 - Low-level matching of image intensity patterns



Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester, English-class)

Lecture 13: Stereovision



STEREOVISION

Goal

The fundamental equations of the *pinhole camera model* are [Trucco98]:

$$\begin{cases} x = f \cdot \frac{X_c}{Z_c} \\ y = f \cdot \frac{Y_c}{Z_c} \end{cases}$$

P(X_C, Y_C, Z_C) 3D point in the camera coordinate system
p(x,y,-f) its projection on the image plane

Knowing the image coordinates (x,y) we cannot infer the depth (Z), only the projection equations

Measure depth (Z) \Rightarrow at least two cameras (stereo-system)

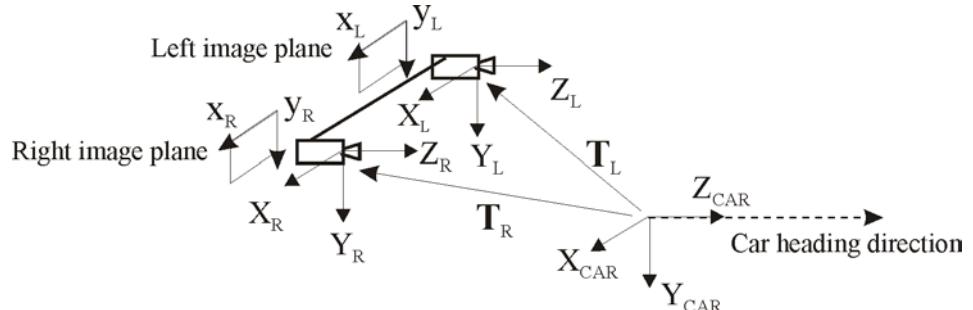
Stereo camera configurations

- Canonic (parallel axes) – theoretical model (impossible to obtain in practice)
 \Rightarrow image rectification
- Coplanar axes (but unparallel)
- General configuration



STEREOVISION

The model of a stereovision system



Parameters	Type of parameters	Set of parameters
Internal parameters of the stereo system	Intrinsic parameters set / camera	Principal points: $PP_L(x_{0L}, y_{0L})$, $PP_R(x_{0R}, y_{0R})$ Focal distance: $f_L(f_{XL}, f_{YL})$, $f_R(f_{XR}, f_{YR})$ Radial distortion: (k_1^L, k_2^L) , (k_1^R, k_2^R) Tangential distortion: (p_1^L, p_2^L) , (p_1^R, p_2^R)
	Relative extrinsic parameters set / stereo system	$T_{REL} = R_{CL}^T(T_{CR} - T_{CL})$ $R_{REL} = R_{CL}^T R_{CR}$
External parameters of the stereo system	Absolute extrinsic parameters set / camera	-For cameras: Translation vectors: T_{CL}, T_{CR} Rotation matrices: R_{CL}, R_{CR} Rotation vectors: r_{CL}, r_{CR} -For the stereo rig: Translation vector: $T_{C-rig} = T_{CL}$ Rotation matrix: $R_{C-rig} = R_{CL}$ Rotation vector: $r_{C-rig} = r_{CL}$



STEREOVISION

The stereovision process

1. Stereo-rig setup
2. Camera and stereo-rig calibration
3. Synchronous acquisition of image pairs
4. Image rectification (if the canonic approach is used)
5. Feature extraction and selection (edges or pixels)
6. Correspondence searching between left and right image features
7. 3D reconstruction in camera /world coordinate system

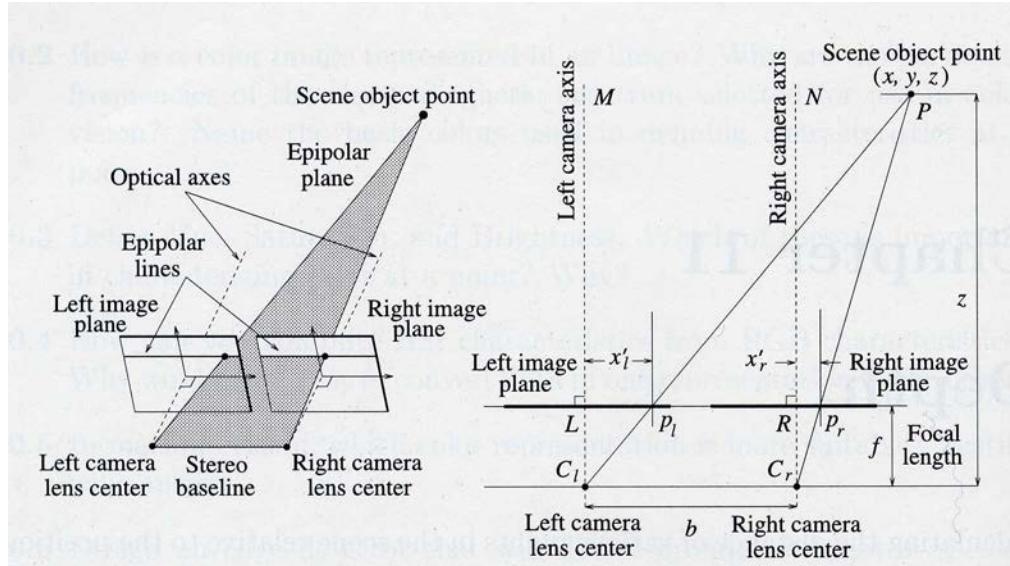
References:

E. Trucco, A. Verri, "Introductory techniques for 3D Computer Vision", Prentice Hall, 1998



STEREOVISION

The canonical model



Depth estimation (canonical config.)

$$x_l = f_x \cdot \frac{X_1}{Z_1}$$

$$x_r = f_x \cdot \frac{X_2}{Z_2}$$

$$d = x_l - x_r = f_x \cdot \left(\frac{X_1}{Z_1} - \frac{X_2}{Z_2} \right) = f_x \cdot \frac{X_1 - X_2}{Z} = f_x \frac{b}{Z}$$

$$Z = \frac{f_x \cdot b}{d}$$

$$X_1 = x_l \cdot Z / f_x; Y_1 = y_l \cdot Z / f_y$$

Assumptions

- Image planes are coplanar \Rightarrow optical axes are parallel
- Horizontal image axes are collinear
- Epipolar lines – horizontal
- $V_{OL} = V_{OR} \Rightarrow y_L = y_R$

Depth estimation (coplanar config.)

Coplanar but non-parallel optical axes: θ angle

$$Z = \frac{f_x \cdot b}{d + f_x \cdot \tan(\theta)}$$



STEREOVISION

Range Resolution

Often it's important to know the minimal change in range that stereo can differentiate, that is, the *range resolution* of the method. Range resolution is a function of the range itself. At closer ranges, the resolution is much better than at farther ranges.

Range resolution is governed by the following equation, [Konolige1999]:

$$|\Delta Z| = (Z^2/fb)\Delta d \quad (Z=fb/d; \Delta Z/\Delta d = -fb/d^2; \Delta Z/\Delta d = -Z^2/fb; |\Delta Z| = (Z^2/fb)\Delta d;)$$

The range resolution ΔZ is the smallest change in range Z that is discernible by the stereo geometry, given a change in disparity of Δd . The range resolution goes up (gets worse) by the square of the range.

The **baseline b** and **focal f length** both have an inverse influence on the resolution, so that larger baselines and focal lengths (telephoto) make the range resolution better.

Finally, the pixel size has a direct influence, so that smaller pixel sizes give better resolution.

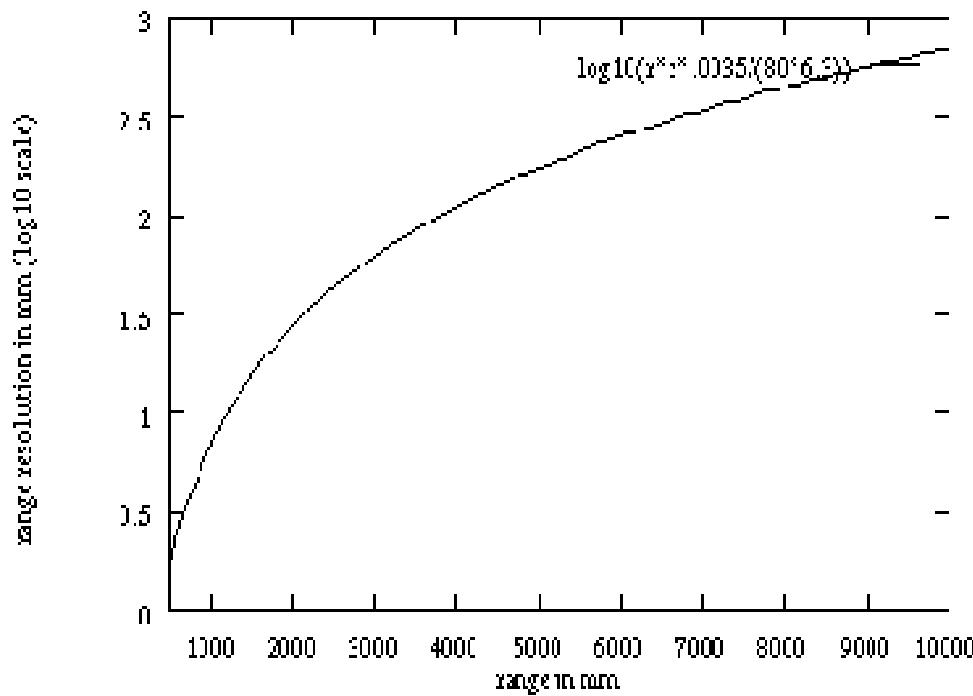
Typically, stereo algorithms can report disparities with sub-pixel precision, which also increases range resolution.



STEREOVISION

Range Resolution

The figure plots range resolution as a function of range for the STH-V1 stereo head, given a baseline of 8 cm and 6.3 mm lenses. The Stereo Engine interpolates disparities to 1/4 pixel, so Δd is $1/4 * 14 \text{ um} = 3.5 \text{ um}$.

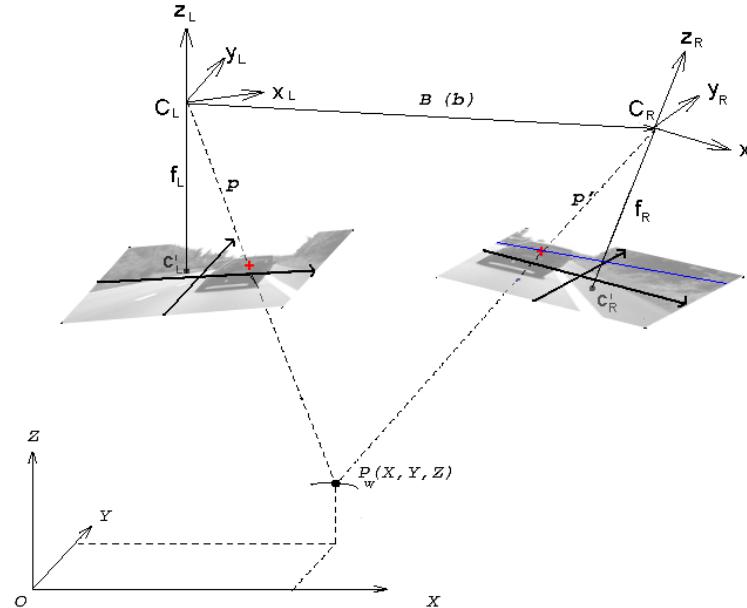


The range resolution is plotted on a log10 scale to show more detail at closer ranges. At 1 meter, the range resolution is about 10 mm. At 4 meters, it has grown to 100 mm; by 10 meters, it is almost a meter.



STEREOVISION

3D stereo-reconstruction for GENERAL CAMERA CONFIGURATION



Intrinsic parameters

Focal lengths

- Left camera: $f_L (f_{XL}, f_{YL})$
- Right camera: $f_R (f_{XR}, f_{YR})$

Principal points

- Left camera: (x_{CL}, y_{CL}) .
- Right camera: (x_{CR}, y_{CR}) .

Extrinsic parameters

$$\mathbf{T}_{CL} = \begin{vmatrix} X_{CL} \\ Y_{CL} \\ Z_{CL} \end{vmatrix} \quad \mathbf{R}_{CL} = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{vmatrix}$$

$$\mathbf{T}_{CR} = \begin{vmatrix} X_{CR} \\ Y_{CR} \\ Z_{CR} \end{vmatrix} \quad \mathbf{R}_{CR} = \begin{vmatrix} r'_{11} & r'_{12} & r'_{13} \\ r'_{21} & r'_{22} & r'_{23} \\ r'_{31} & r'_{32} & r'_{33} \end{vmatrix}$$



STEREOVISION

3D stereo-reconstruction problem := mapping 2D image pairs $(p_L(x_L, y_L), p_R(x_R, y_R))$ into a unique 3D point P_w :

$$\mathbf{P}_w = \begin{vmatrix} X_w \\ Y_w \\ Z_w \end{vmatrix}$$

The solution:

$$\mathbf{P}_w = \mu \mathbf{R}_L * \begin{vmatrix} x_L - x_{CL} \\ y_L - y_{CL} \\ -f_L \end{vmatrix} + \mathbf{T}_{CL} \iff \begin{vmatrix} X_w \\ Y_w \\ Z_w \end{vmatrix} = \mu \mathbf{R}_L * \begin{vmatrix} x_L - x_{CL} \\ y_L - y_{CL} \\ -f_L \end{vmatrix} + \begin{vmatrix} X_{CL} \\ Y_{CL} \\ Z_{CL} \end{vmatrix} \quad (1)$$

Where μ is a scaling factor depending on Z and can be expressed from the 3-rd equation of the following system:

$$\begin{vmatrix} x_L - x_{CL} \\ y_L - y_{CL} \\ -f_L \end{vmatrix} = \mu^{-1} \mathbf{R}_L^T * \begin{vmatrix} X_w - X_{CL} \\ Y_w - Y_{CL} \\ Z_w - Z_{CL} \end{vmatrix} \iff \begin{vmatrix} x_L - x_{CL} \\ y_L - y_{CL} \\ -f_L \end{vmatrix} = \mu^{-1} \begin{vmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{vmatrix} * \begin{vmatrix} X_w - X_{CL} \\ Y_w - Y_{CL} \\ Z_w - Z_{CL} \end{vmatrix} \quad (2)$$

$$\mu^{-1} = \frac{-f_L}{r_{13}(X_w - X_{CL}) + r_{23}(Y_w - Y_{CL}) + r_{33}(Z_w - Z_{CL})}$$



STEREOVISION

The 3D reconstruction solution:

Replacing μ^1 in the first two equations of (2), the following relations are obtained

The projection equations of P in the left camera coordinate system:

$$\begin{cases} x_L - x_{CL} = -f_L \frac{r_{11}(X_w - X_{CL}) + r_{21}(Y_w - Y_{CL}) + r_{31}(Z_w - Z_{CL})}{r_{13}(X_w - X_{CL}) + r_{23}(Y_w - Y_{CL}) + r_{33}(Z_w - Z_{CL})} \\ y_L - y_{CL} = -f_L \frac{r_{12}(X_w - X_{CL}) + r_{22}(Y_w - Y_{CL}) + r_{32}(Z_w - Z_{CL})}{r_{13}(X_w - X_{CL}) + r_{23}(Y_w - Y_{CL}) + r_{33}(Z_w - Z_{CL})} \end{cases} \quad (3)$$

The projection equations of P in the right camera coordinate system:

$$\begin{cases} x_R - x_{CR} = -f_R \frac{r'_{11}(X_w - X_{CR}) + r'_{21}(Y_w - Y_{CR}) + r'_{31}(Z_w - Z_{CR})}{r'_{13}(X_w - X_{CR}) + r'_{23}(Y_w - Y_{CR}) + r'_{33}(Z_w - Z_{CR})} \\ y_R - y_{CR} = -f_R \frac{r'_{12}(X_w - X_{CR}) + r'_{22}(Y_w - Y_{CR}) + r'_{32}(Z_w - Z_{CR})}{r'_{13}(X_w - X_{CR}) + r'_{23}(Y_w - Y_{CR}) + r'_{33}(Z_w - Z_{CR})} \end{cases} \quad (4)$$

Notation:

$$\begin{cases} x'_L = x_L - x_{CL} \\ y'_L = y_L - y_{CL} \\ x'_R = x_R - x_{CR} \\ y'_R = y_R - y_{CR} \end{cases}$$



STEREOVISION

The 3D reconstruction solution:

$$(3) + (4) \Rightarrow \mathbf{A} * \begin{vmatrix} X_w \\ Y_w \\ Z_w \end{vmatrix} = \mathbf{B} \Leftrightarrow \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{vmatrix} * \begin{vmatrix} X_w \\ Y_w \\ Z_w \end{vmatrix} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix} \quad (5)$$

where:

$$\mathbf{A} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{vmatrix} = \begin{vmatrix} r_{13}x'_L + r_{11}f_L & r_{23}x'_L + r_{21}f_L & r_{33}x'_L + r_{31}f_L \\ r_{13}y'_L + r_{12}f_L & r_{23}y'_L + r_{22}f_L & r_{33}y'_L + r_{32}f_L \\ r'_{13}x'_R + r'_{11}f_R & r'_{23}x'_R + r'_{21}f_R & r'_{33}x'_R + r'_{31}f_R \\ r'_{13}y'_R + r'_{12}f_R & r'_{23}y'_R + r'_{22}f_R & r'_{33}y'_R + r'_{32}f_R \end{vmatrix}$$

$$\mathbf{B} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix} = \begin{vmatrix} a_{11}X_{CL} + a_{12}Y_{CL} + a_{13}Z_{CL} \\ a_{21}X_{CL} + a_{22}Y_{CL} + a_{23}Z_{CL} \\ a_{31}X_{CR} + a_{32}Y_{CR} + a_{33}Z_{CR} \\ a_{41}X_{CR} + a_{42}Y_{CR} + a_{43}Z_{CR} \end{vmatrix}$$

The algebraic solution for system (5) can be obtained using the least squares method:

$$\mathbf{X} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{B}$$

Geometric meaning of the solution → the middle of the shortest segment connecting the two projection lines



STEREOVISION

The stereo-correlation problem

For a given left image point p_L a right image point p_R must be found so that the pair (p_L, p_R) represents the projections of the same 3D point P on the two image planes.

There are two major approaches:

- edge based reconstruction (sparse reconstruction) : only the highly relevant features like as edges are used
- dense reconstruction: all correlated pixels are reconstructed

The search space reduction is essential for real time software or hardware implementations. Is based on epipolar geometry constraints (epipolar lines). Has benefic effects also on reconstruction accuracy through diminishing of the false positive correspondences.



STEREOVISION

Basics of epipolar geometry

- Epipole (baseline intersection with the image plane) \Rightarrow one epipole / each image (is the intersection of all epipolar lines of the image)
- Epipolar plane \Rightarrow one plane for each 3D point
- Epipolar line \Rightarrow one line for each 3D point

Epipolar geometry constraint

- For every image point (x_L, y_L) there is a unique epipolar line (e_R) in the right image which will contain the corresponding right projection point (x_R, y_R) and vice-versa

Computing the epipolar lines [Trucco] :

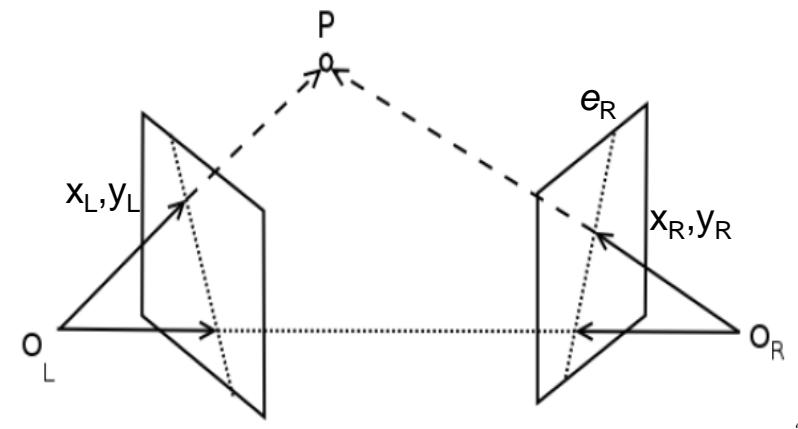
$$a * x + b * y + c = 0$$

$$\begin{vmatrix} a_R \\ b_R \\ c_R \end{vmatrix} = \mathbf{F} * \mathbf{P}_L \quad ; \quad \begin{vmatrix} a_L \\ b_L \\ c_L \end{vmatrix} = \mathbf{F}^T * \mathbf{P}_R$$

where:

\mathbf{F} – fundamental matrix

$$\mathbf{P}_L = \begin{vmatrix} x_L \\ y_L \\ 1 \end{vmatrix} \quad ; \quad \mathbf{P}_R = \begin{vmatrix} x_R \\ y_R \\ 1 \end{vmatrix}$$





STEREOVISION

Computing the fundamental (F) and essential (E) matrices

$$\mathbf{F} = (\mathbf{A}_R^{-1})^T * \mathbf{E} * \mathbf{A}_L^{-1}$$

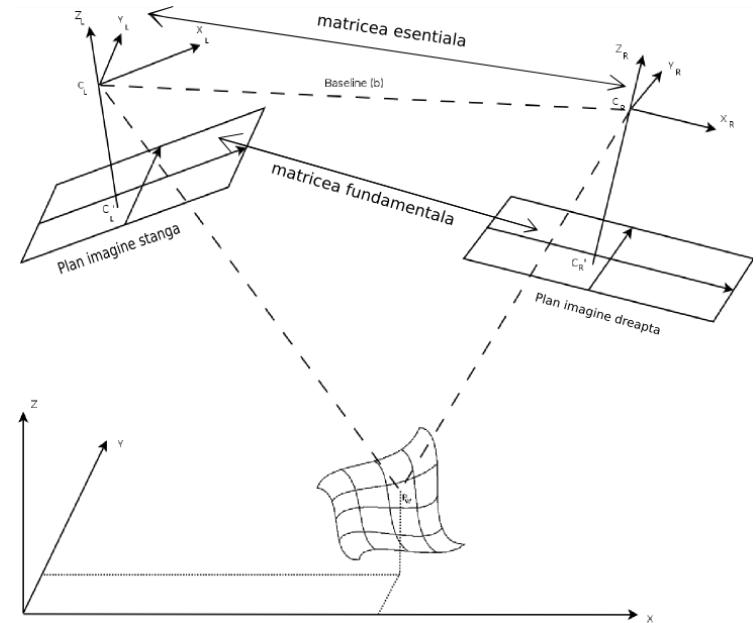
where:

$$\mathbf{E} = \mathbf{R}_{LR} * \mathbf{S}$$

$$\mathbf{R}_{LR} = \mathbf{R}_{CW-R}^T * \mathbf{R}_{CW-L}$$

$$\mathbf{T}_{LR} = \mathbf{T}_{CW-R} - \mathbf{T}_{CW-L} = [T_x \quad T_y \quad T_z]^T$$

$$\mathbf{S} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$



E – essential matrix

R_{LR} – relative left-to-right rotation matrix

T_{LR} – relative left-to-right translation matrix

A_L, A_R – the internal matrices of the two cameras



STEREOVISION

The stereo-correlation

a. *Features selection*

- - low level features: pixels.
- high level features: edge segments

b. *Features matching*

- use of epipolar geometry constraints (epipolar lines) for search space reduction
- for a left image point, a right correspondent point must be chosen out of a set of candidates.
- the correlation function is the measure used for discrimination.

c. *Increasing the resolution of the correlation*

- compute the disparity with sub-pixel accuracy \Rightarrow far range & high accuracy stereo-vision



STEREOVISION

Features selection

Low level features -
each image pixel is
reconstructed (dense
stereo)



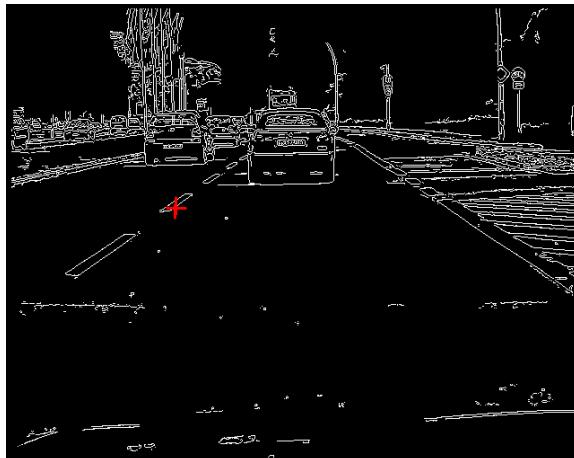
High level features –
only edge pixels are
reconstructed (edge
based stereo)



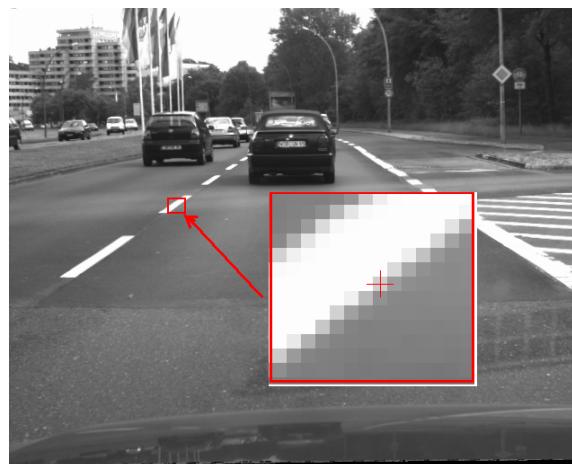


STEREOVISION

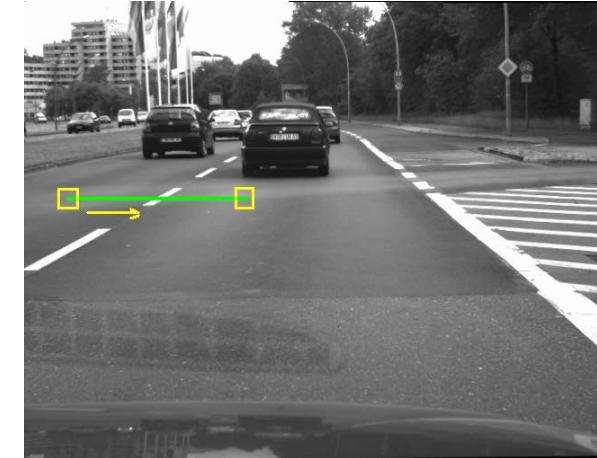
Features matching



Edge feature in left image



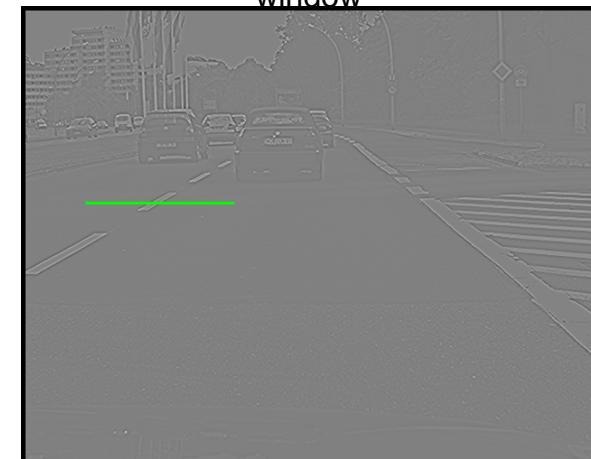
Grayscale left image correlation window



The correlation window slides along the epipolar line in the right image window



LoG left image correlation window



Technical University of Cluj Napoca
Computer Science Department

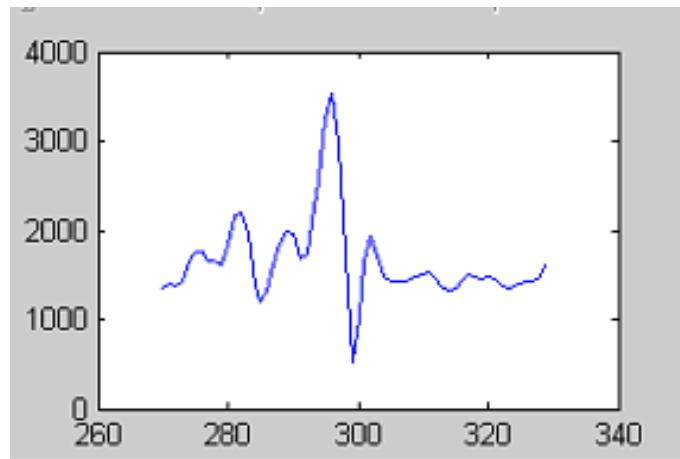


STEREOVISION

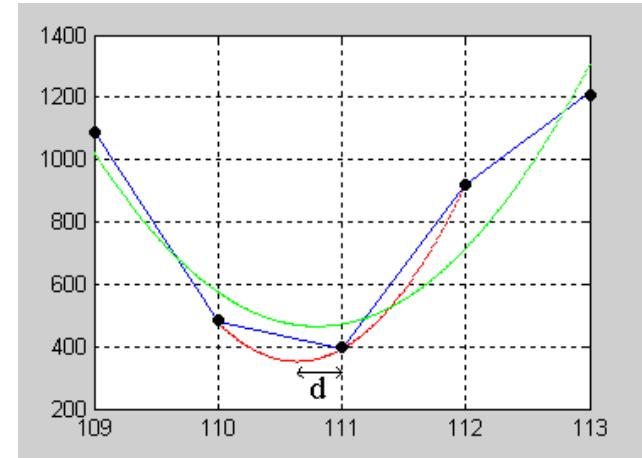
The correlation function

Any distance measure function SAD, SSD, normalized correlation

$$SAD(x_R, y_R) = \sum_{i=-\frac{w}{2}}^{\frac{w}{2}} \sum_{j=-\frac{w}{2}}^{\frac{w}{2}} |I_L(x_L + i, y_L + j) - I_R(x_R + i, y_R + j)|$$



Global minima of the correlation function



Detection of the sub-pixel position of global minima of the correlation function using a parabolic interpolator