

Neuronale Netze

Neural Networks for Image Processing

Joshua Winebarger & Christian Mohr

10.01.2012

Outline of presentation

- Intro. to Image Processing
- Traditional issues & techniques in image processing
 - Fundamental techniques
 - Point Processing
 - Filtering / Masking
 - Transforms
 - Applications
 - Edge Detection
 - Resampling
- Example Systems
 - Face detection
 - Focus of Attention

Sources

- Portions of this lecture were taken from the following sources:
 - The EECE/CS 253 Image Processing lecture series by Richard Alan Peters II at Vanderbilt University School of Engineering (marked with a *) under a creative-commons license
 - The « Visuelle Perzeption für Mensch-Maschine-Interaktion » lecture series by Mika Fischer and Rainer Stiefelhagen of the Karlsruhe Institute of Technology (marked with a †)

INTRO TO IMAGE PROCESSING & UNDERSTANDING

Applications of Image Processing

■ Medical diagnoses

- Object detection
 - Edge Detection

■ Image restoration

- Denoising

■ Robotics

- Scene analysis
- Object identification
 - Transform-Invariant features

■ Security

- Person detection
- Face recognition

■ Photography

- Automatic brightness and contrast correction
- High-Dynamic Range Photography

■ Astronomy

■ Military

- Target identification

■ Forensics

Image Understanding

- Pattern recognition – ID of objects
- Conventional pattern recognition:
 - Observation vector mapped onto a feature space
 - Decision Process: Partitioning of feature space

Image Formation

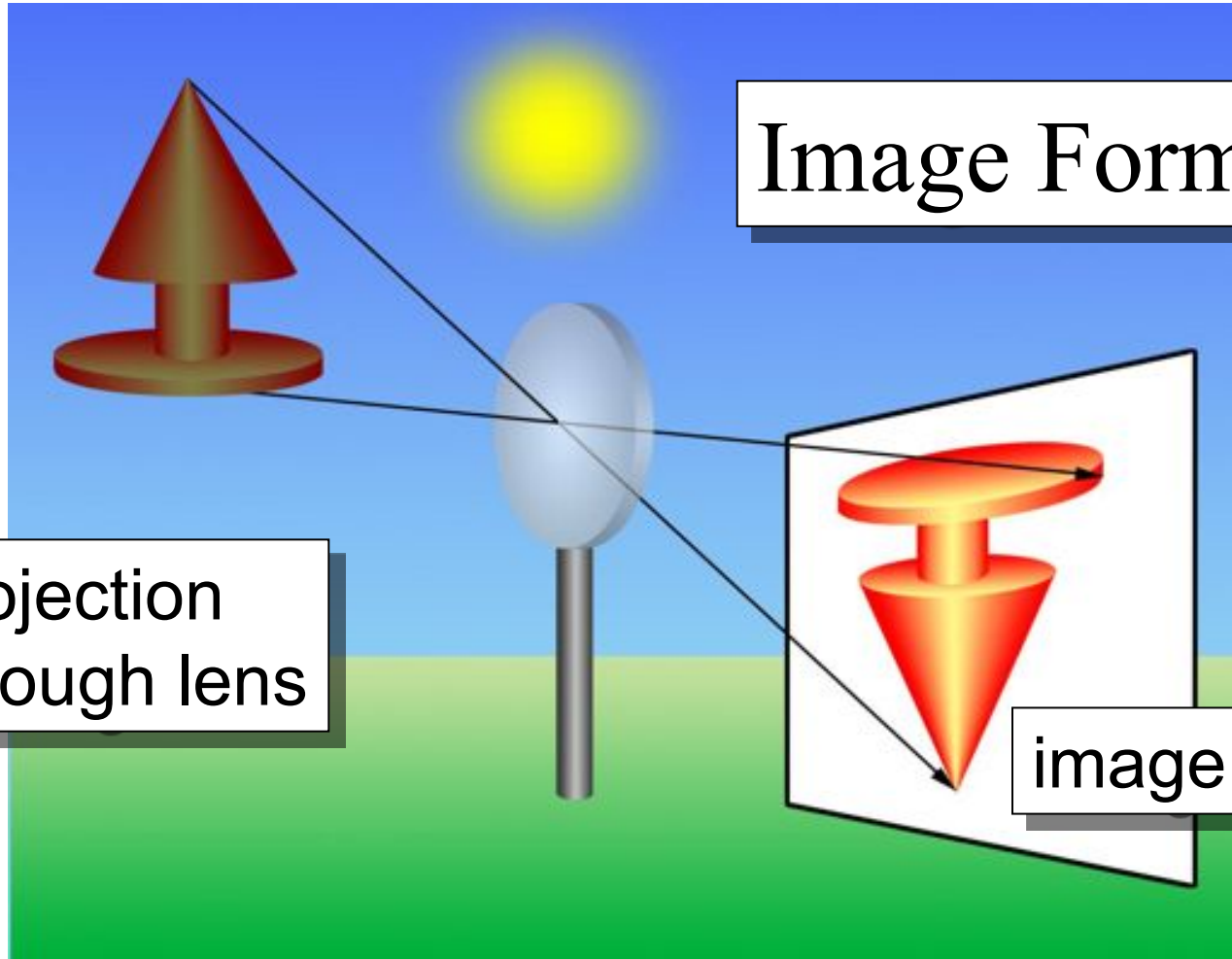
- Image formed is combination of:
 - Geometry of object and rotation
 - Surface properties of object (color, texture)
 - Lighting intensity, color, position
 - Perspective of viewer

How is an image formed?*

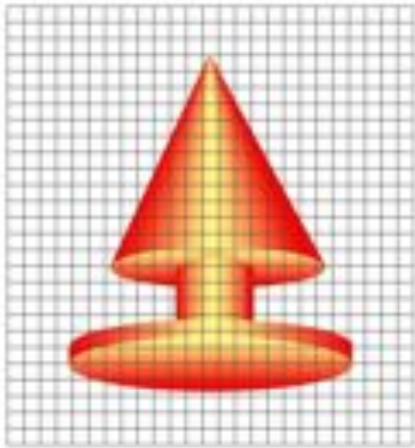
Image Formation

projection
through lens

image of object



How is an image formed?*



real image



sampled



quantized



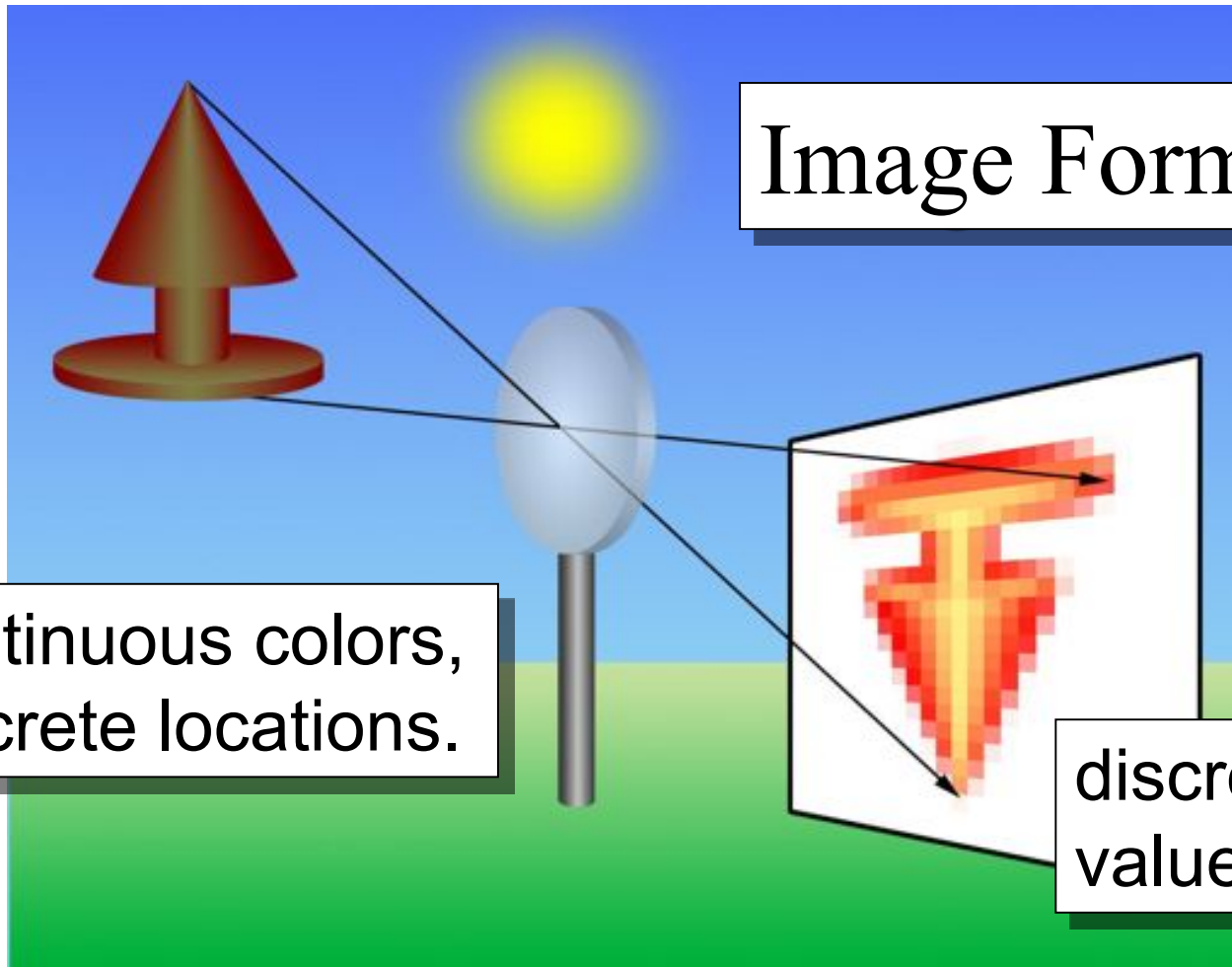
sampled &
quantized

How is an image formed?*

Image Formation

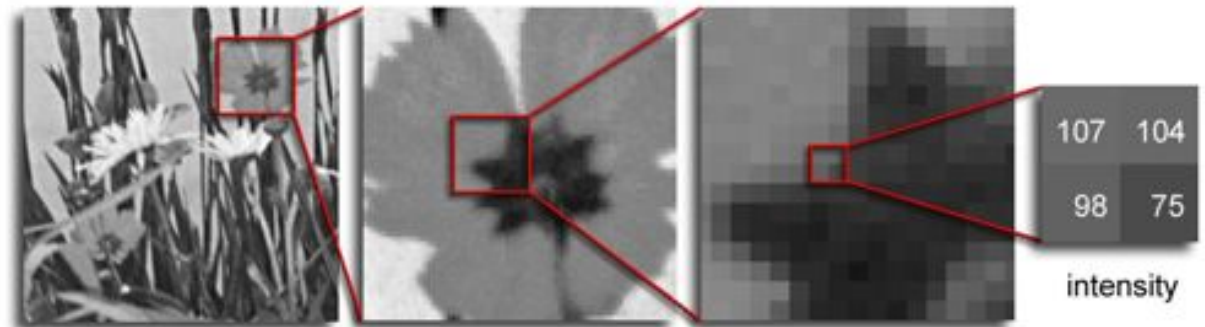
continuous colors,
discrete locations.

discrete real-
valued image



Composition of a Digital Image*

each square is called a pixel (for *picture element*)



Fundamental techniques in traditional image processing

■ Point Processing

- Perform operations on each pixel independent of neighboring pixels

■ Filtering / Masking - Convolution in 2D

- Perform operations on pixels, taking into account values of other pixels
- Applications:
 - Edge Detection
 - Resampling

■ Frequency-Domain Analysis

- Fourier Transform in 2D
- Conversion of image into Frequency Domain representation
- Applications: Feature Extraction

Point Processing

- Perform an operation on a pixel's value as a function of this value alone
 - Independent of value of the pixel's neighbors
- Examples:
 - Brightness and contrast adjustment
 - Gamma correction
 - Histogram equalization
 - Color correction

Point Processing*



- gamma



- brightness



original



+ brightness



+ gamma



histogram mod



- contrast



original



+ contrast



histogram EQ

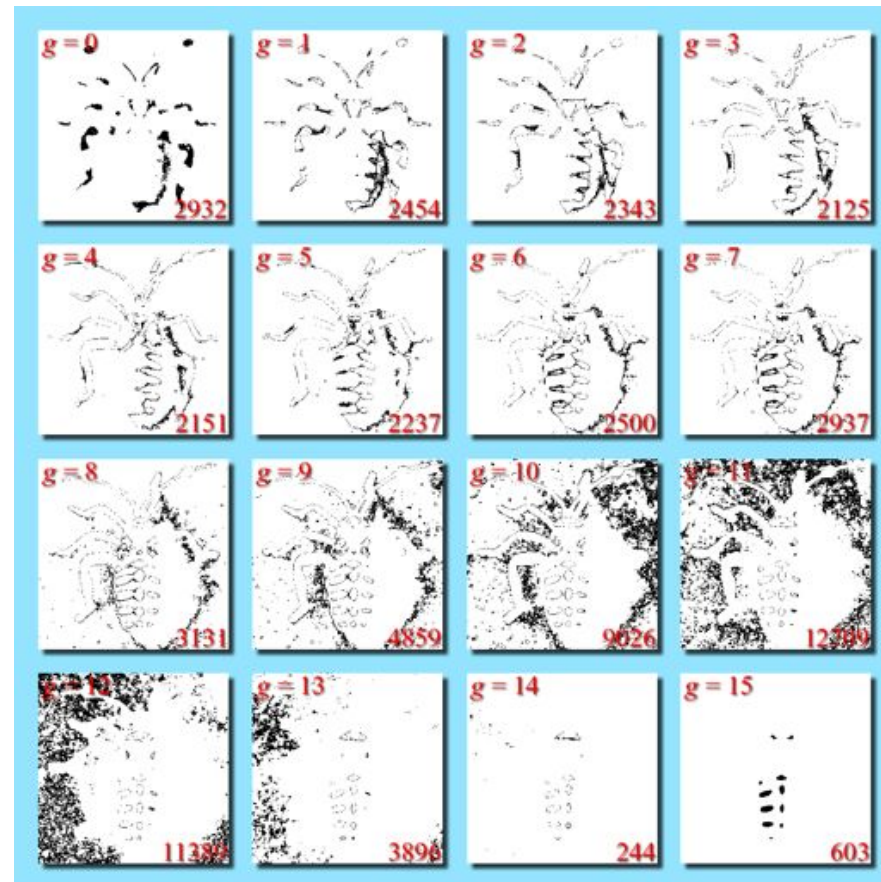
Histogram of a greyscale image*

The Histogram of a Grayscale Image



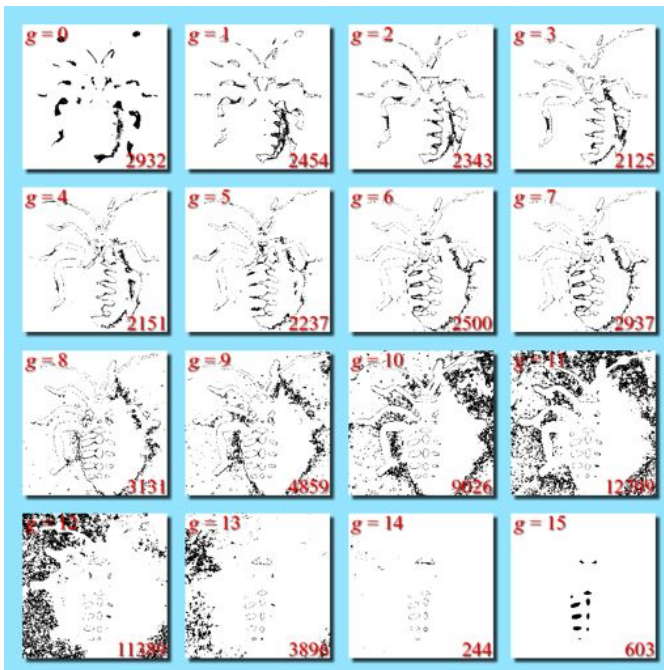
16-level (4-bit) image

lower RHC: number of pixels with intensity g

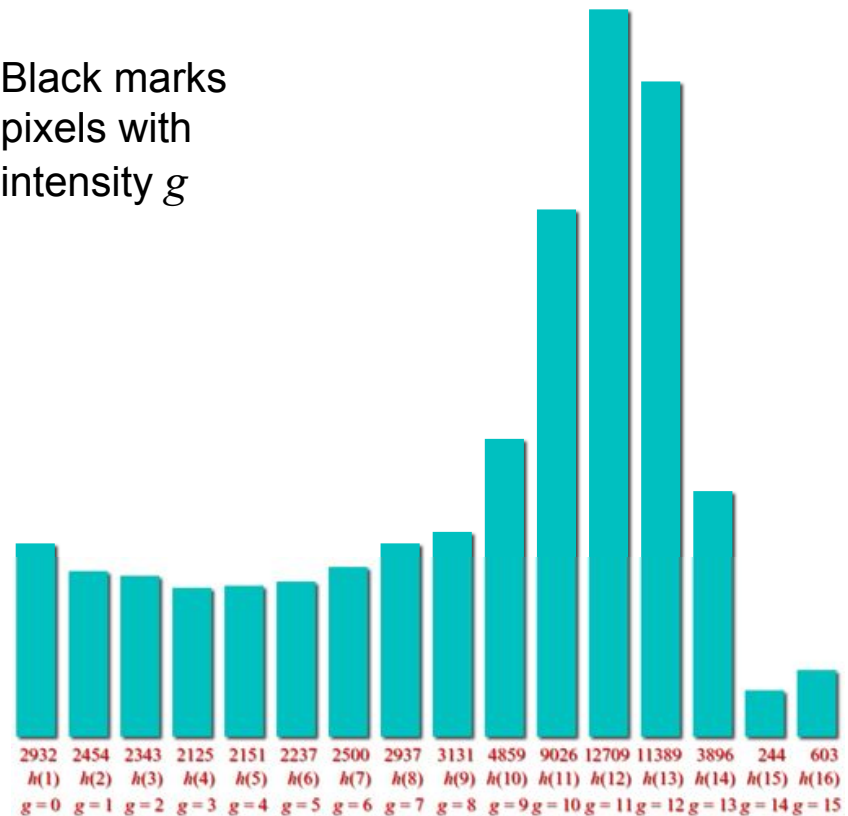


black marks pixels with intensity g

Histogram of a Greyscale Image*

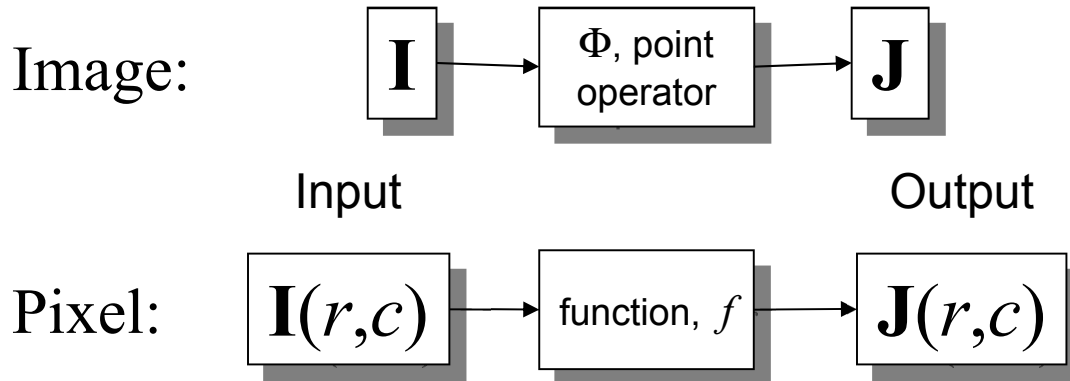


Black marks
pixels with
intensity g



Plot of histogram:
number of pixels with intensity g

Point Operations via Functional Mappings*



$$\mathbf{J} = \Phi[\mathbf{I}]$$

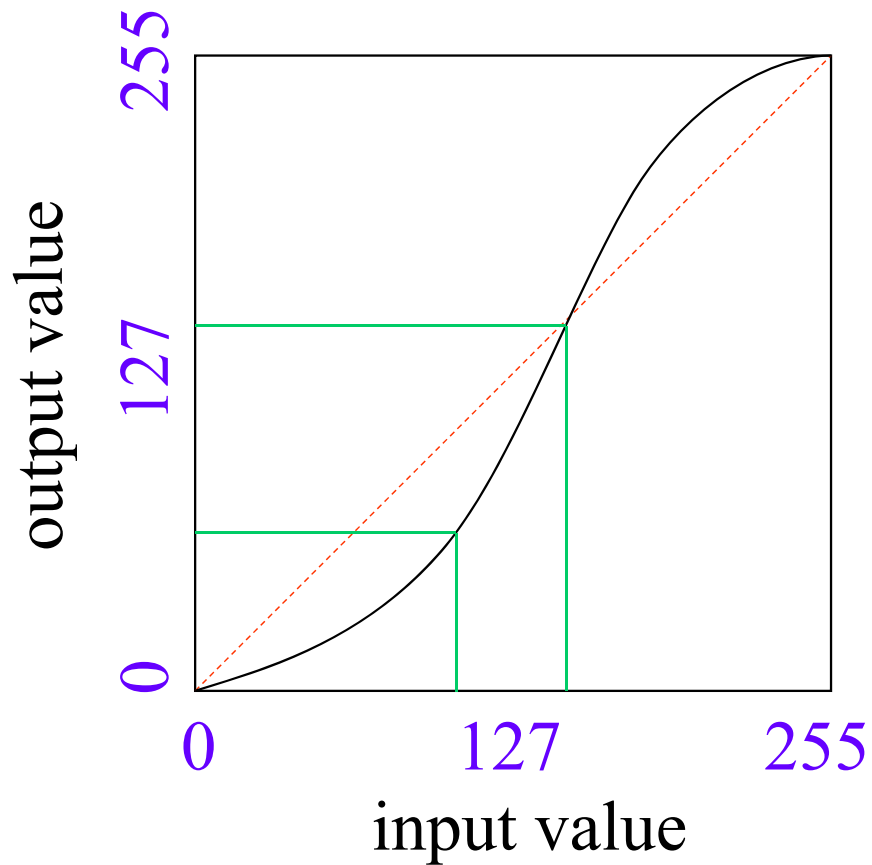
If $\mathbf{I}(r,c) = g$
 and $f(g) = k$
 then $\mathbf{J}(r,c) = k$.

The transformation of image \mathbf{I} into image \mathbf{J} is accomplished by replacing each input intensity, g , with a specific output intensity, k , at every location (r,c) where $\mathbf{I}(r,c) = g$.

A functional mapping is often implemented by a look-up table (LUT)

The rule that associates k with g is usually specified with a function, f , so that $f(g) = k$.

Look-Up Table Operations*

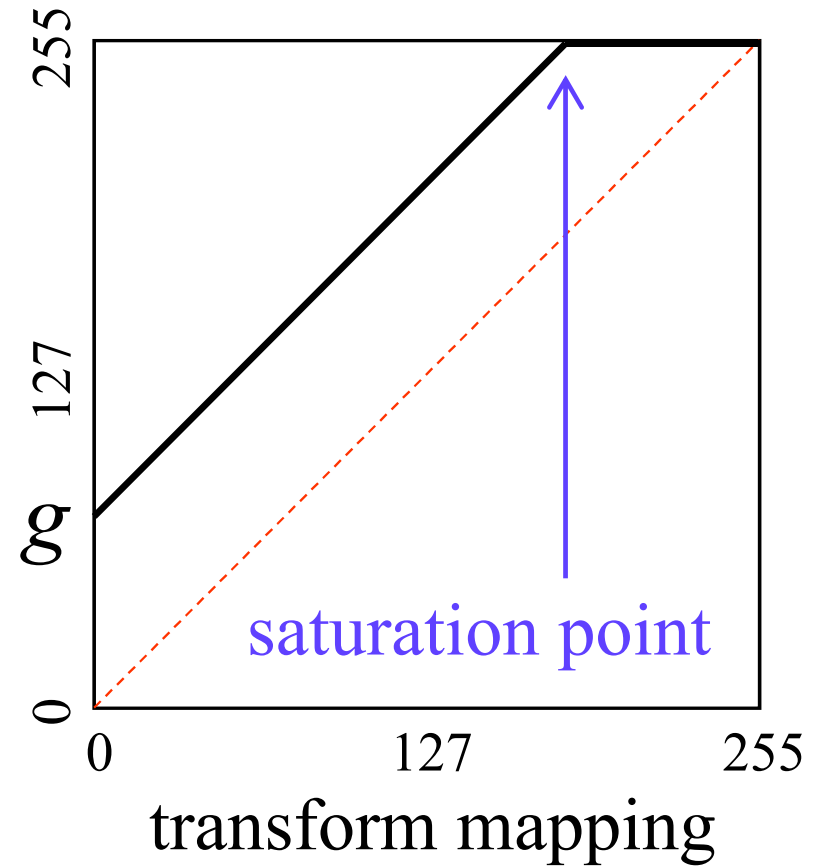


<i>E.g.:</i>	index	value

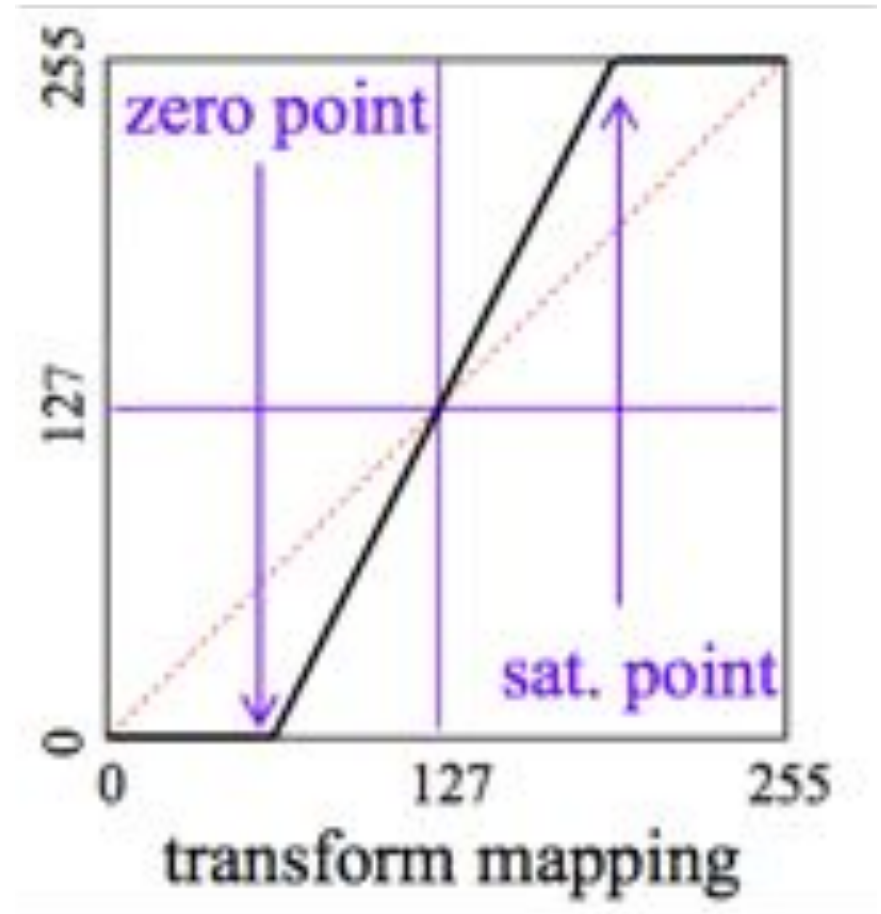
	101	64
	102	68
	103	69
	104	70
	105	70
	106	71

	input	output

Point Processing: Increase Brightness*

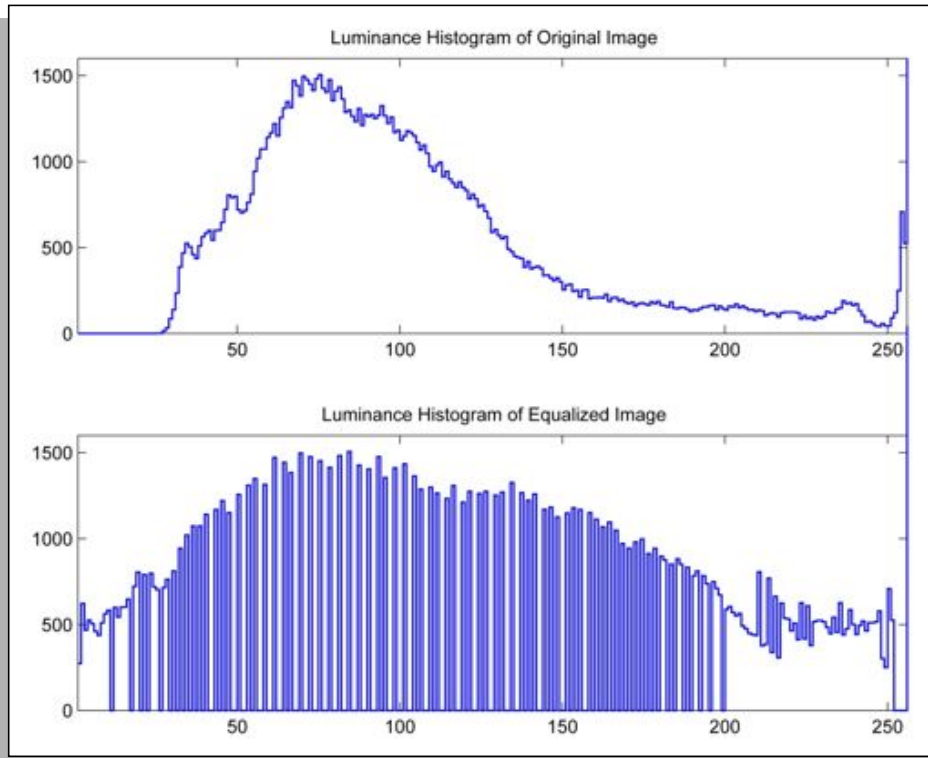


Point Processing: Increase Contrast*



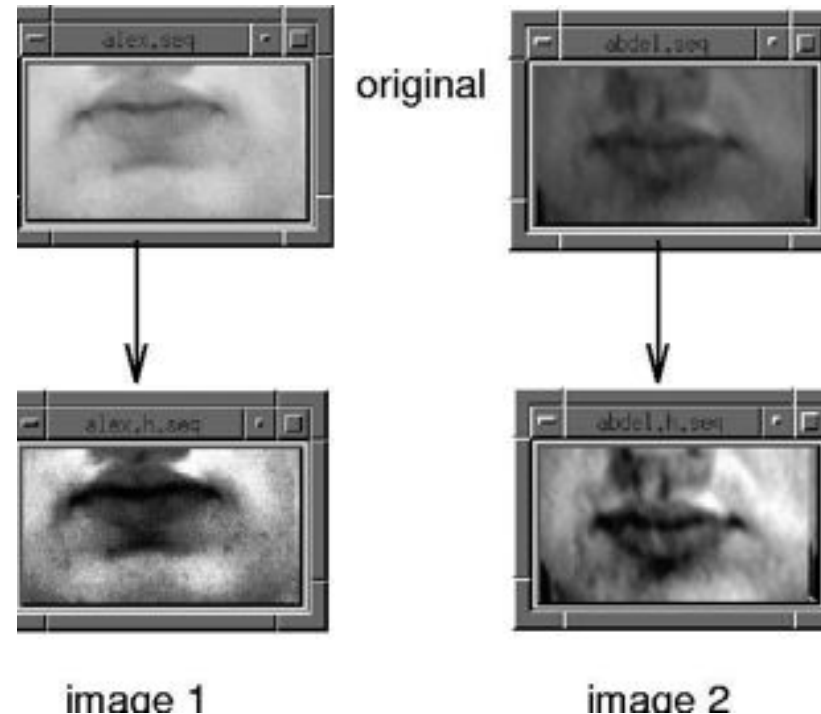
Point Processing: Histogram Equalization*

- Remap image so its histogram is as close to constant as possible
- CDF of image is used as conversion LUT



Point Processing: Histogram Equalization*

- Defines a mapping of gray levels p into gray levels q such that the distribution of q is close to being uniform
- Stretches contrast (expands the range of gray levels)
- Transforms different input images so that they have similar intensity distributions (thus reducing the effect of different illumination)
- Fast algorithm exists,
- Transformation can be defined in terms of the cumulative histogram



Point Processing: Histogram Equalization*

- The probability of an occurrence of a pixel of level i in the image is $p(x_i)$: define c as the cumulative distribution function:
- create a transformation of the form:
 - $y = T(x)$
- will produce a level y for each level x in the original image, such that the cumulative probability function of y will be linearized across the value range.
- L : number of gray levels, n_i : number of occurrences of gray level i

$$p(x_i) = \frac{n_i}{n}, i \in 0, \dots, L - 1$$

$$c(i) = \sum_{j=0}^i p(x_j)$$

$$y_i = T(x_i) = c(i)$$

$(y_i \in [0,1])$

$$y'_i = y_i \cdot (\max - \min) + \min$$

$(y'_i \in [\min, \max])$

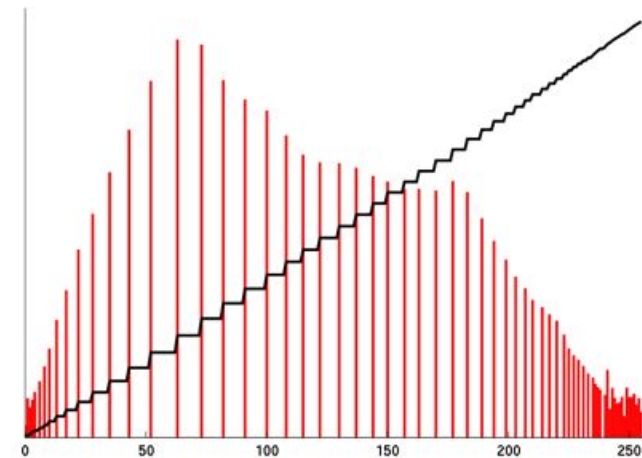
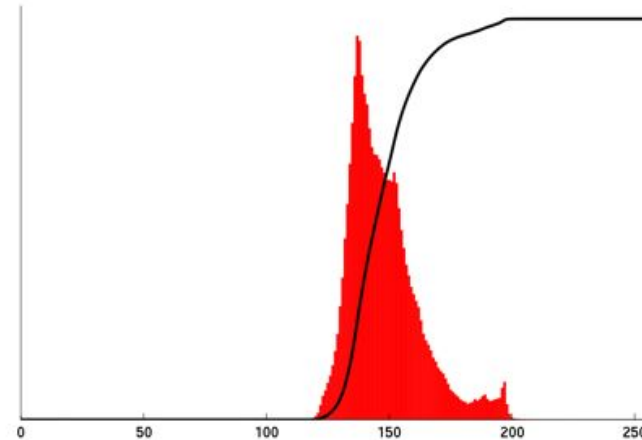
Point Processing: Histogram Equalization*



Unequalized image



Equalized image



Corresponding Histograms

Linear Filtering

- Modify a pixel's value to be some combination of the values around it
- Typically done using convolution with a kernel

Convolution in 2D

$$J(r, c) = [I * h](r, c) = \sum_{i=-s}^s \sum_{j=-d}^d I(r-i, c-j) h(i, j)$$

- I is the input image, J is the resulting image
- h is the kernel
- Three methods in practice:
 - Moving window
 - Shift, multiply, and add
 - Frequency domain

Convolution in 2D: Moving Window

- Consider a 3x3 image window:

$$\begin{bmatrix} g_1 & g_2 & g_3 \\ g_4 & g_5 & g_6 \\ g_7 & g_8 & g_9 \end{bmatrix}$$

- Convolution Kernel:

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}$$

- Replace center pixel with weighted average:

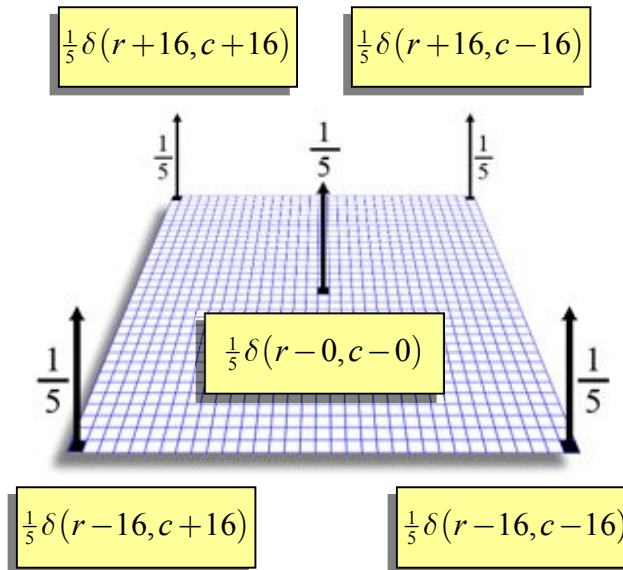
$$g_5 = \sum_{i=1}^9 g_i w_i$$

- Slide window over original image

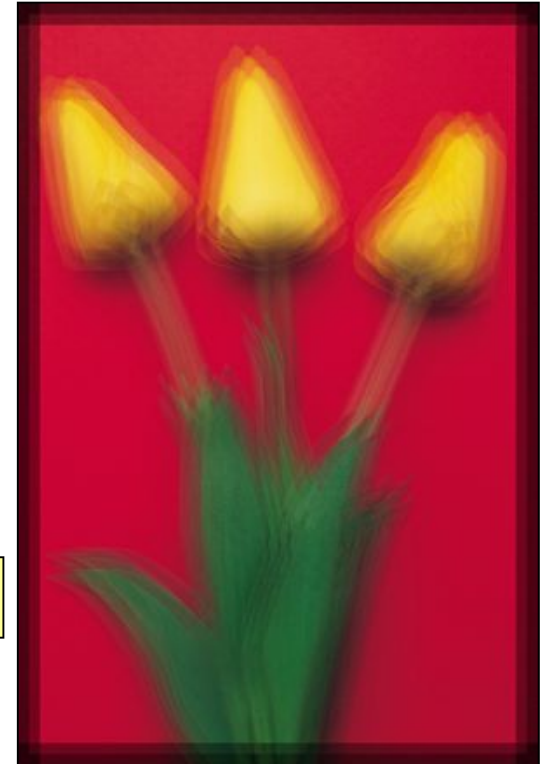
Convolution in 2D: Shift-Multiply-Add*

- The image is copied 1 time for each element in the convolution mask.
- Each copy is shifted relative to the original by the displacement of its associated mask element.
- Each copy is multiplied by the value of its associated mask element.
- The set of shifted and multiplied images is summed pixel wise.

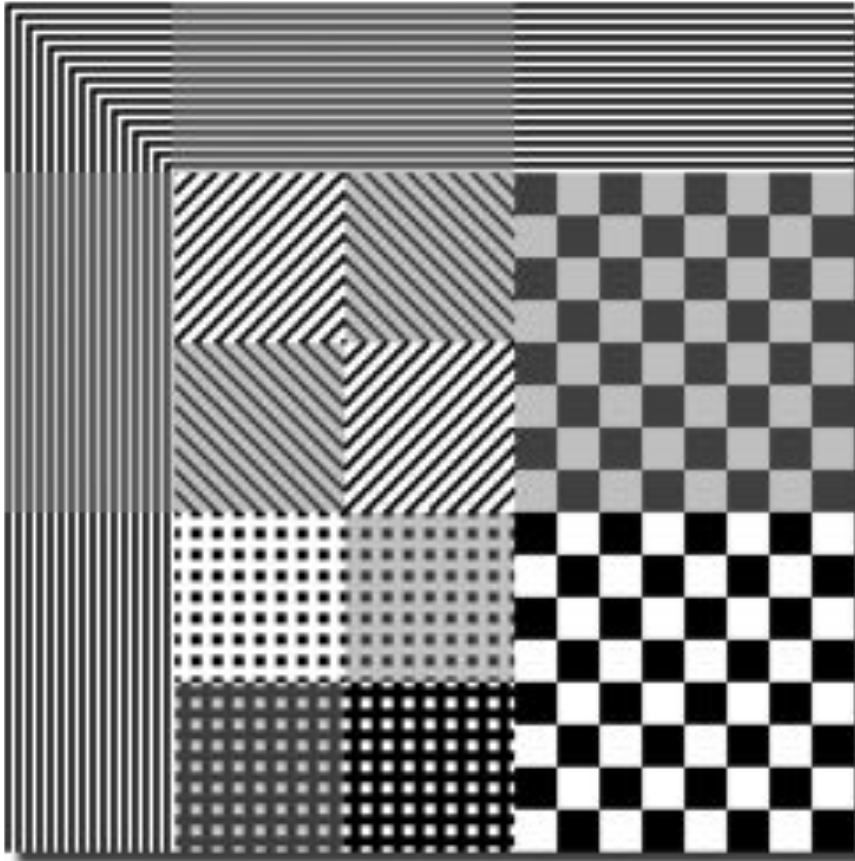
Convolution in 2D: Convolution by Five Impulses



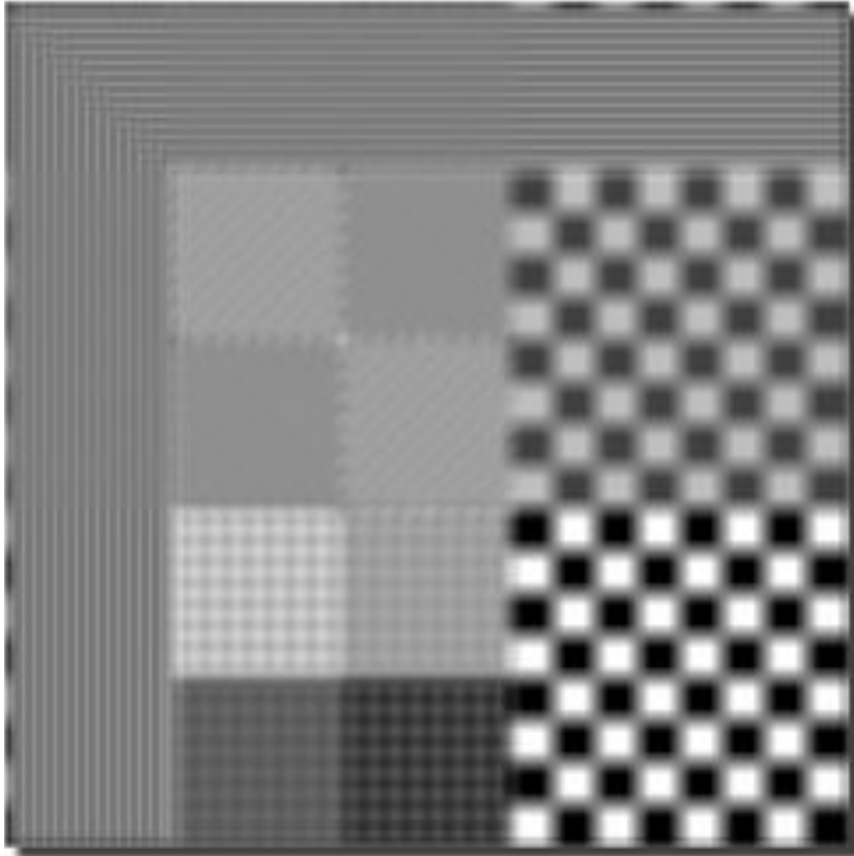
Five copies, four moved, one not moved, averaged.



Convolution Example: 9x9 Blur*



Convolution Example: 9x9 Blur*



Fourier Transform in Image Processing

- Useful in certain types of noise reduction, deblurring, and image restoration
- Useful in feature detection and enhancement

Fourier Transform in 1D

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt \quad \text{time} \rightarrow \text{freq}$$

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} dt \quad \text{freq} \rightarrow \text{time}$$

This is a sum of sine and cosine functions extending to infinity

$$e^{2\pi i f t} = \cos(2\pi f t) + i \sin(2\pi f t)$$

$$H(f) = \int_{-\infty}^{\infty} h(t) \cos(2\pi f t) dt + i \int_{-\infty}^{\infty} h(t) \sin(2\pi f t) dt$$

DFT in 1D

- DFT for signals in discrete time and frequency
- n measured complex values for time or frequency series
- Transform produces n such values
- Dot product of data with sine & cosine
- Strength of component of time series which repeats f times in interval $t=[0,n-1]$

$$H[f] = \sum_{t=0}^{n-1} h[t] e^{i \frac{2\pi t f}{n}}$$

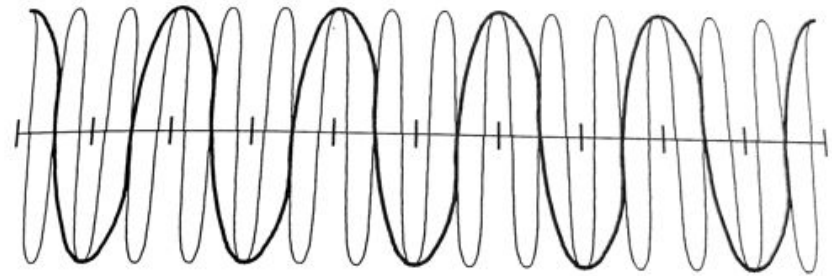
$$h[t] = \frac{1}{n} \sum_{f=0}^{n-1} H[f] e^{-i \frac{2\pi t f}{n}}$$

DFT in 1D

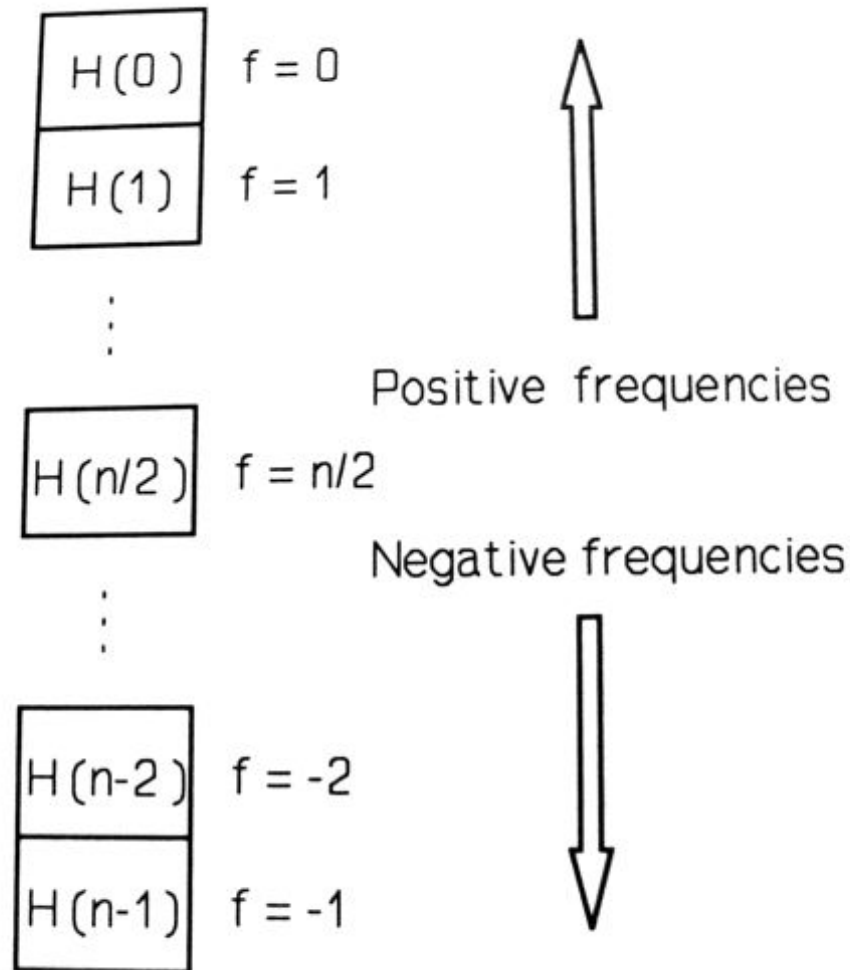
- fwd. & rev. xfm. can be accomplished with same algorithm
 - fwd. xfm. for rev. xfm.:
 - cpx. conj. \rightarrow fwd. xfm. \rightarrow cpx. conj. \rightarrow divide by n
- FFT
 - DFT: $O(n^2)$ multiplications & additions
 - FFT (if n is power of 2):
 $O(n \log (n))$

Aliasing & Nyquist Frequency

- Aliasing & Nyquist limit:
 - High-frequency component takes on character of non-existent low-freq component
- Nyquist frequency: Half the sampling frequency → highest frequency which can be reconstructed
- Corresponds to $H[n/2]$



Positive & Negative Frequencies



Fourier Transform in 2D

- Dealing with values in a plane, we need a two-dimensional Fourier basis function:

$$e^{-2\pi(ux+vy)}$$

- Fourier and inverse fourier transform use double integral:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i2\pi(ux+vy)} dx dy$$

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{i2\pi(ux+vy)} du dv$$

- Assumption that function exists in infinite plane

DFT for images

- Assumption that image exists on a torus!
- Image I has the following DFT:

$$\Xi(v, u) = \frac{1}{RC} \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} I(r, c) e^{-i2\pi \left(\frac{vr}{R} + \frac{uc}{C} \right)}$$

- and inverse DFT:

$$I(r, c) = \sum_{u=0}^{R-1} \sum_{v=0}^{C-1} \Xi(v, u) e^{i2\pi \left(\frac{vr}{R} + \frac{uc}{C} \right)}$$

2D Sinusoids*

- The basis function is a complex 2D sinusoid

$$e^{\pm i \frac{2\pi}{N}(vr+uc)} = e^{\pm i \frac{2\pi}{N}(r \sin \theta + c \cos \theta)}$$

- where

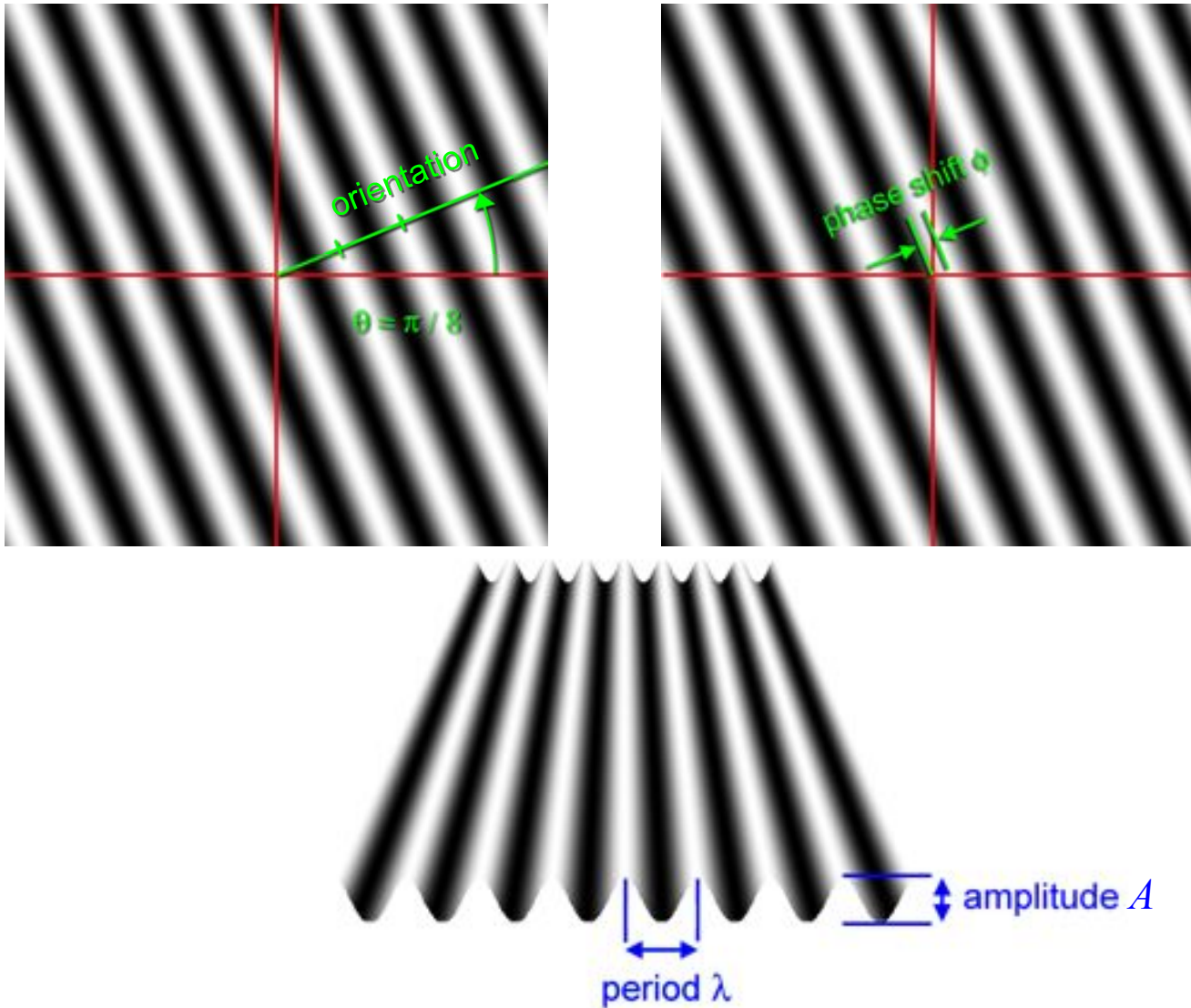
$$v = \omega \sin \theta, \quad u = \omega \cos \theta, \quad \omega = \sqrt{v^2 + u^2}, \quad \theta = \tan^{-1}\left(\frac{v}{u}\right), \quad \lambda = \frac{N}{\omega}$$

- by Euler's relation:

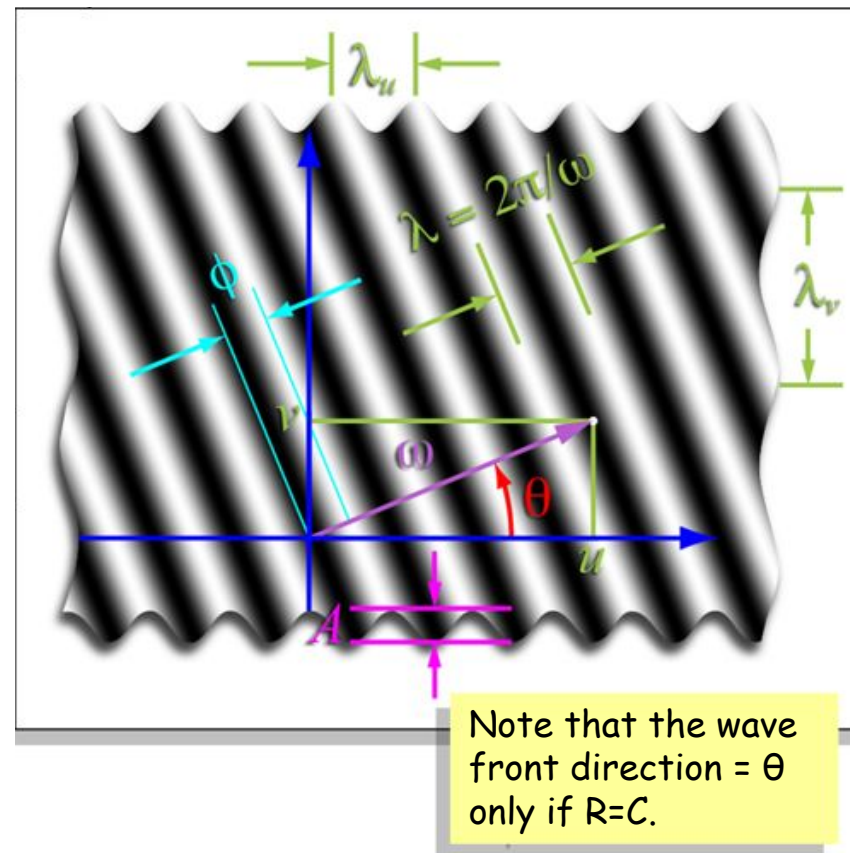
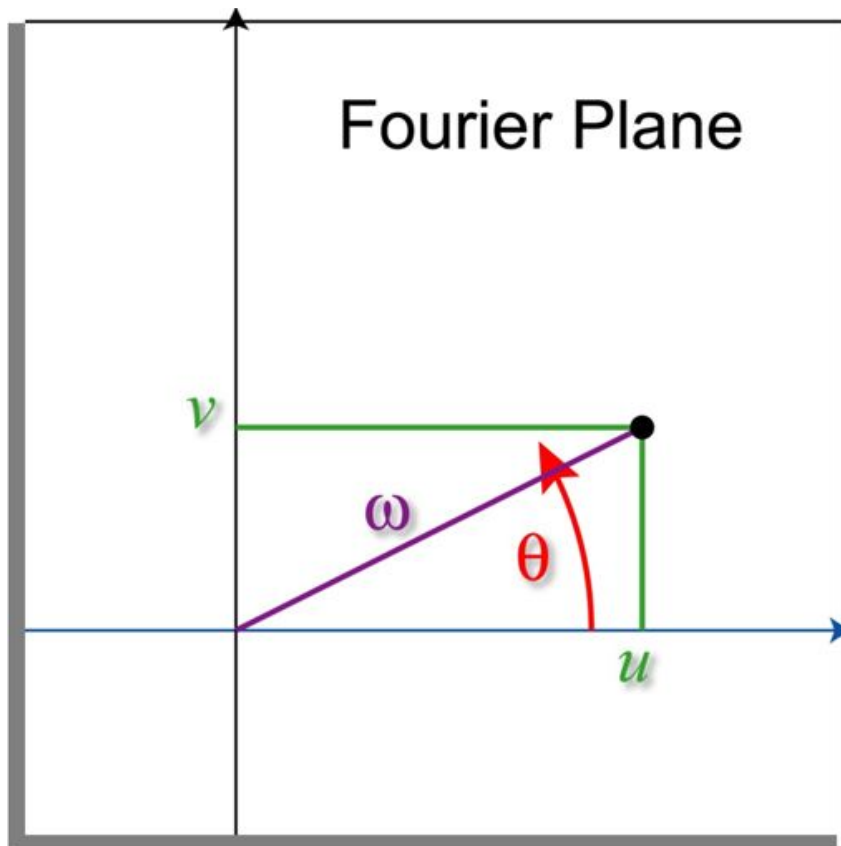
$$e^{\pm i 2\pi \frac{1}{\lambda}(r \sin \theta + c \cos \theta)} = \cos\left[\frac{2\pi}{\lambda}(r \sin \theta + c \cos \theta)\right] \pm i \sin\left[\frac{2\pi}{\lambda}(r \sin \theta + c \cos \theta)\right]$$

- The real and imaginary parts are sinusoidal « gratings »

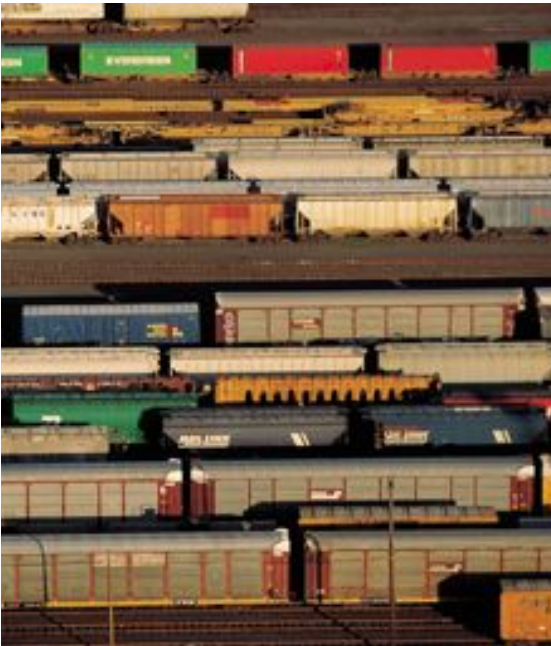
2D Sinusoids*



The Fourier Plane and the value of a Fourier Coefficient*



Example FT of an Image*



I



$$\log\{|\mathcal{F}\{\mathbf{I}\}|^2+1\}$$



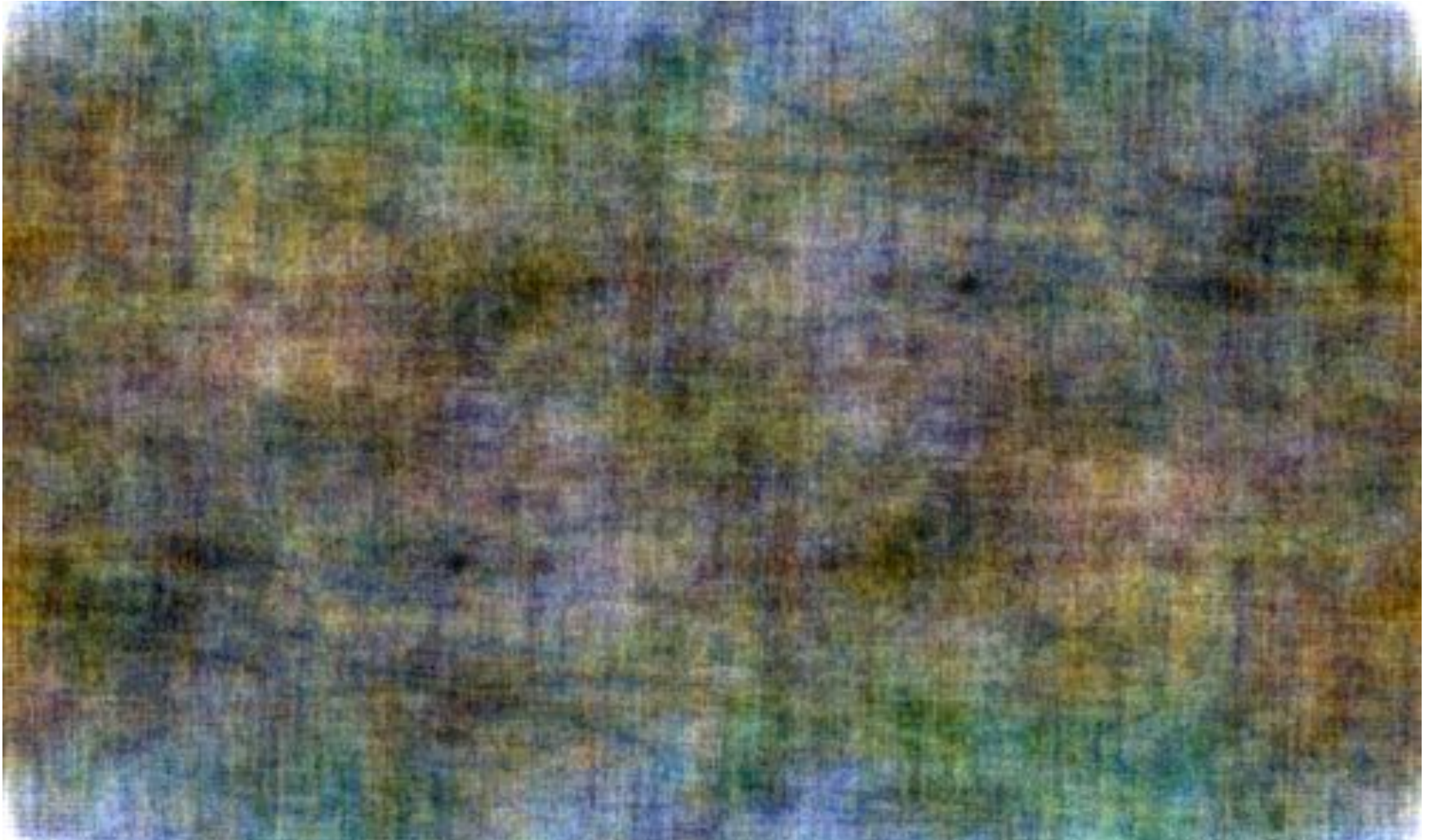
$$\angle[\mathcal{F}\{\mathbf{I}\}]$$

Importance of Magnitude & Phase*

I



Magnitude-Only Reconstruction



Phase-Only Reconstruction



Fourier-Domain Feature Extraction

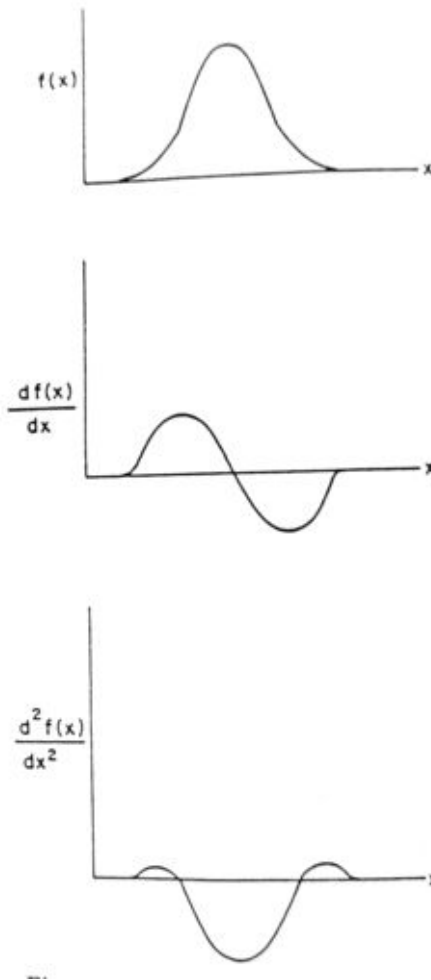
- Scale and rotation invariant features are very useful in object recognition
- Idea:
 - Sample FT plane with wedge-shaped elements:
 - Rotational information captured
 - Sample FT plane with ring-shaped elements
 - Scale information captured

TRADITIONAL IMAGE PROCESSING TECHNIQUES

Edge Detection

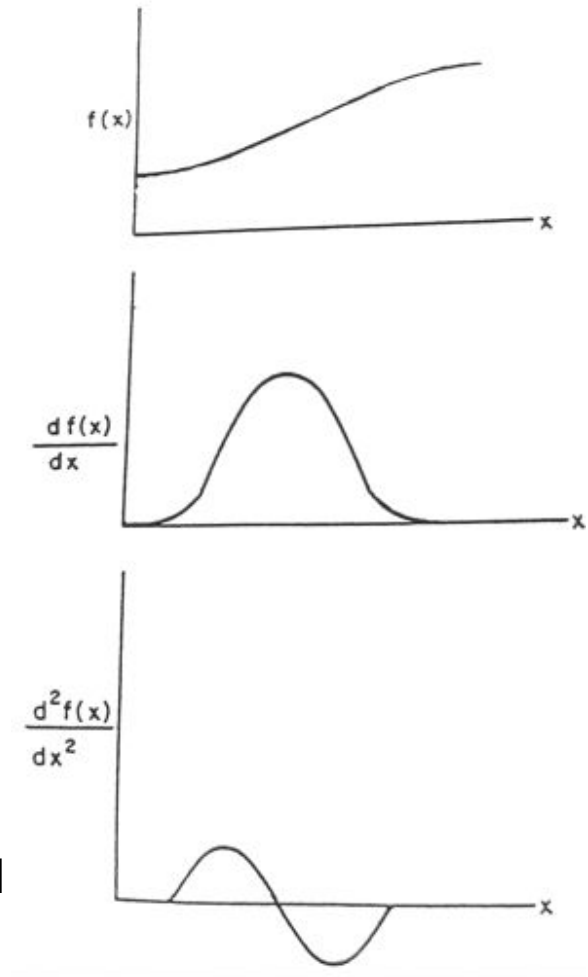
- Isolate edges or make them more visually prominent
- Increase grey-level difference between boundaries of two more homogenous regions
- Useful in object detection

Line Edge vs Step Edge



- Line Edge:
 - Second derivative max. at edge
- Step edge
 - First derivative max. at edge point
- Detect edges in image:
 - Fix threshold on first and second derivatives
$$g'(i, j) \geq T$$

$$g'(i, j) < T$$
- Directions:
 - Vertical vs. Horizontal
 - Vertical & Horizontal
- Difficulty with noise



Edge detection operators

- Laplacian

$$g'(i, j) = \text{abs} \left[4g(i, j) - g(i+1, j) - g(i-1, j) - g(i, j+1) - g(i, j-1) \right]$$

- Gradient

$$g'(i, j) = \sqrt{\left[g(i, j) - g(i, j-1) \right]^2 + \left[g(i, j) - g(i-1, j) \right]^2}$$

- Robert

$$g'(i, j) = \text{abs} \left[g(i, j) - g(i+1, j+1) \right]$$

Edge Detection with Convolution

- Edge detection can be implemented with 2D convolution

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

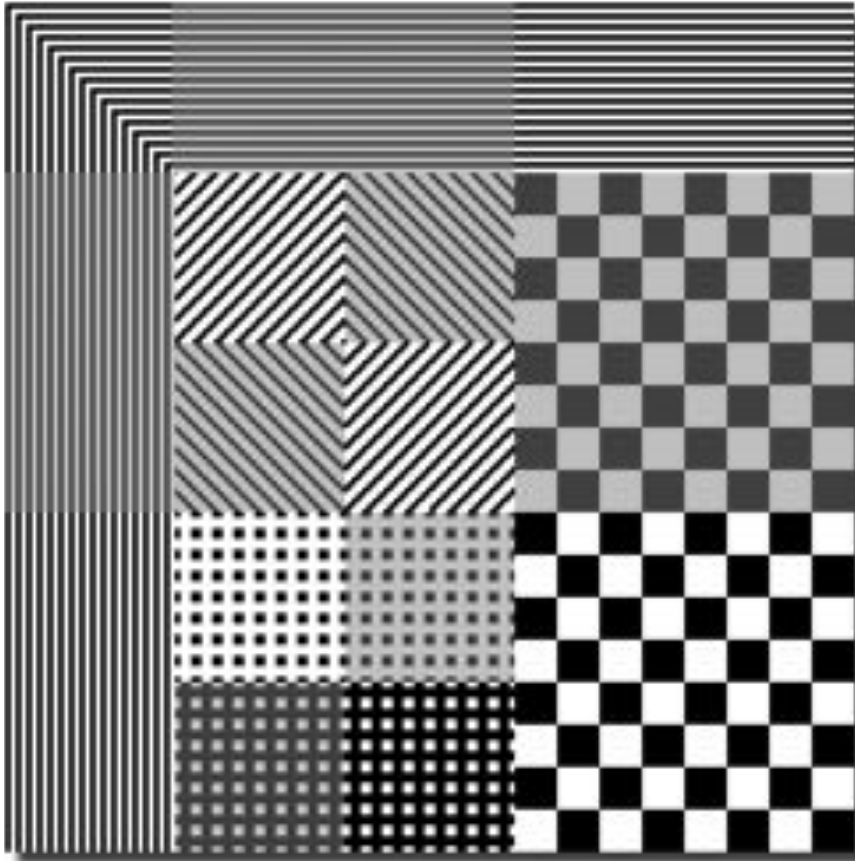
Masks for Sobel edge operator

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

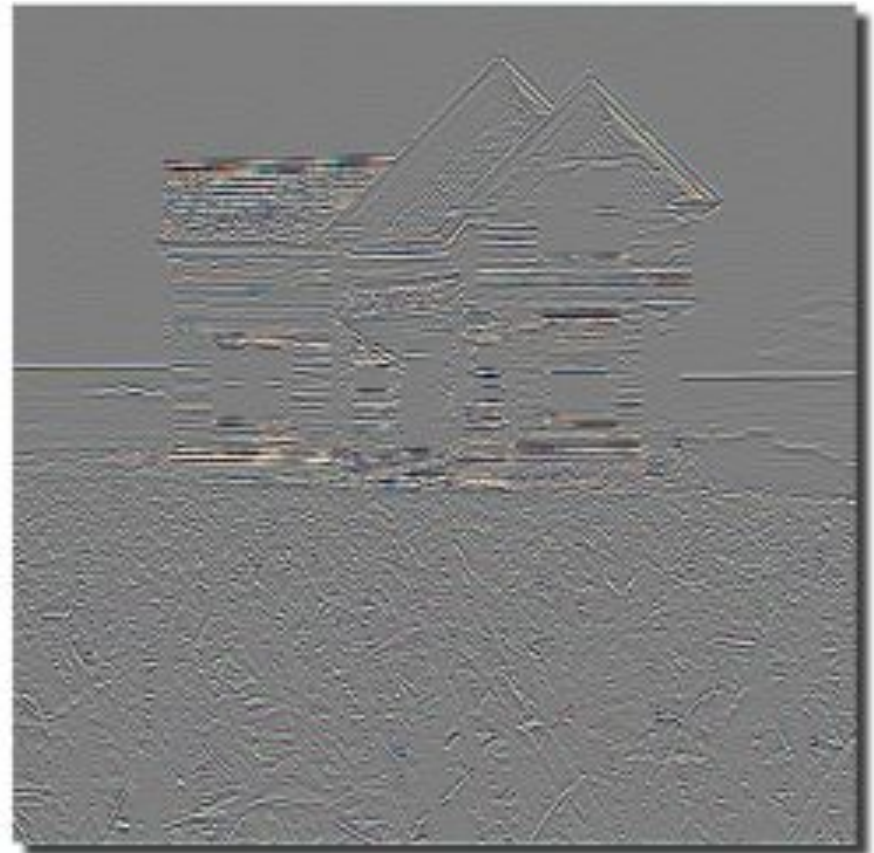
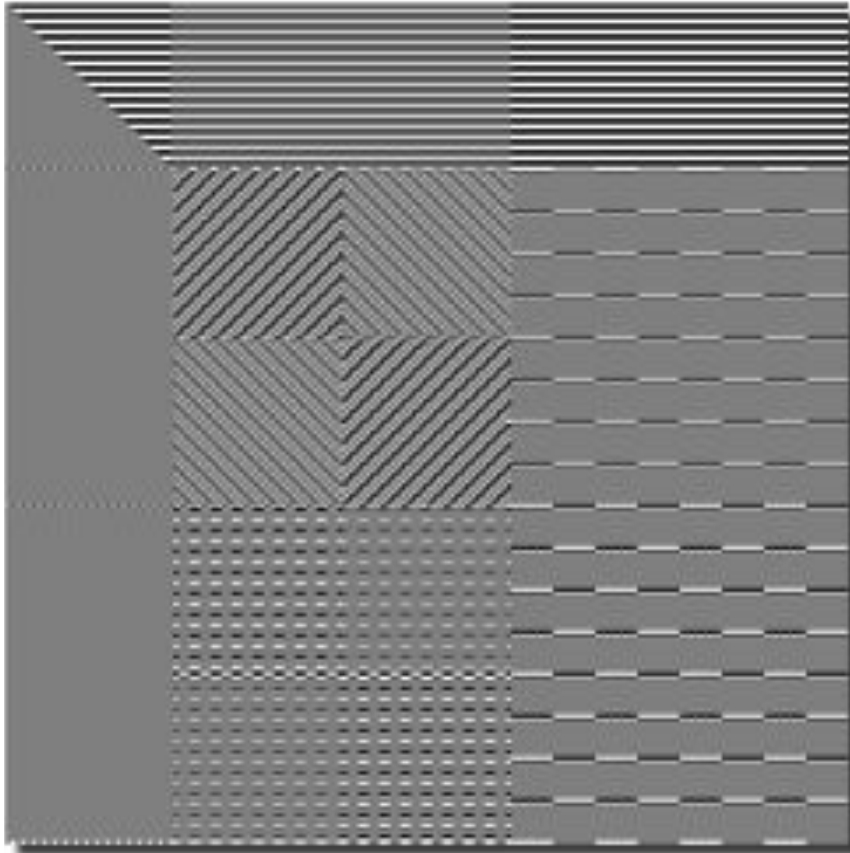
$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Masks for Kirsch operator

Edge Detection: Laplacian Example*



Edge Detection: Laplacian Example*



Resampling

- Reduce or Enlarge size of image
- Process of estimating intermediate values of a continuous function from discrete samples

- Several Techniques:
 - Replication
 - Decimation
 - Nearest-neighbor resampling
 - Bilinear
 - Bicubic
 - Cubic spline

Replication & Decimation

Replication by factor of N

- Start with blank N times the linear dimension of the original
- Fill in every $N \times N$ block with the value of every N th pixel

Decimation by a factor of N

- Take every N th pixel in every N th row

Nearest-Neighbor Resampling*

Nearest Neighbor Resampling

The “Nearest Neighbor” algorithm is a generalization of pixel replication and decimation.

It also includes fractional resizing, *i.e.* resizing an image so that it has p/q of the pixels per row and p/q of the rows in the original. (p and q are both integers.)



Nearest-Neighbor Resampling*

Nearest
Neighbor
Resampling

3/7 resize

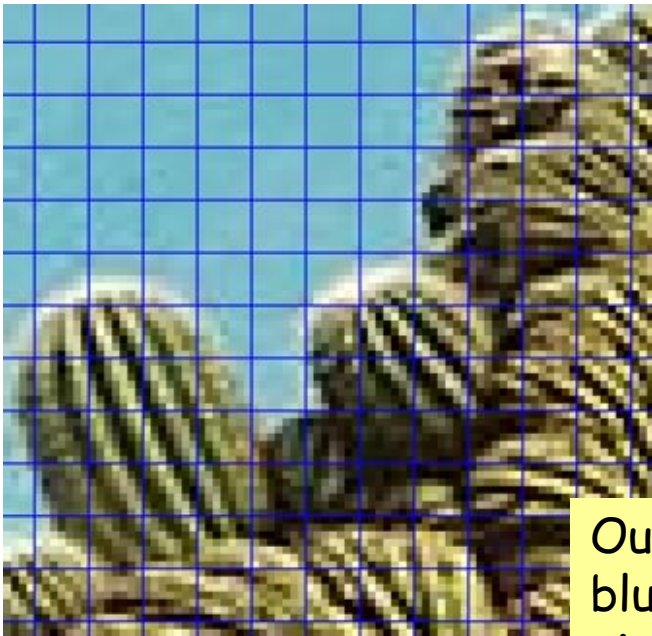


Zoom in for a
better look

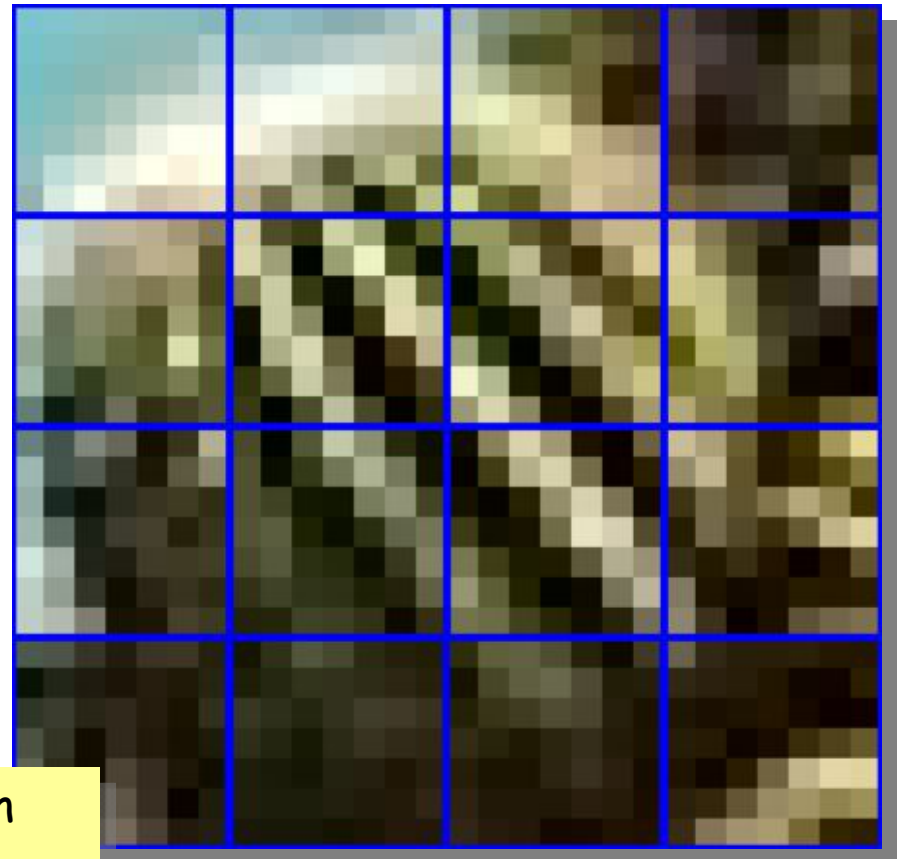
Nearest-Neighbor Resampling*

Nearest
Neighbor
Resampling

3/7 resize



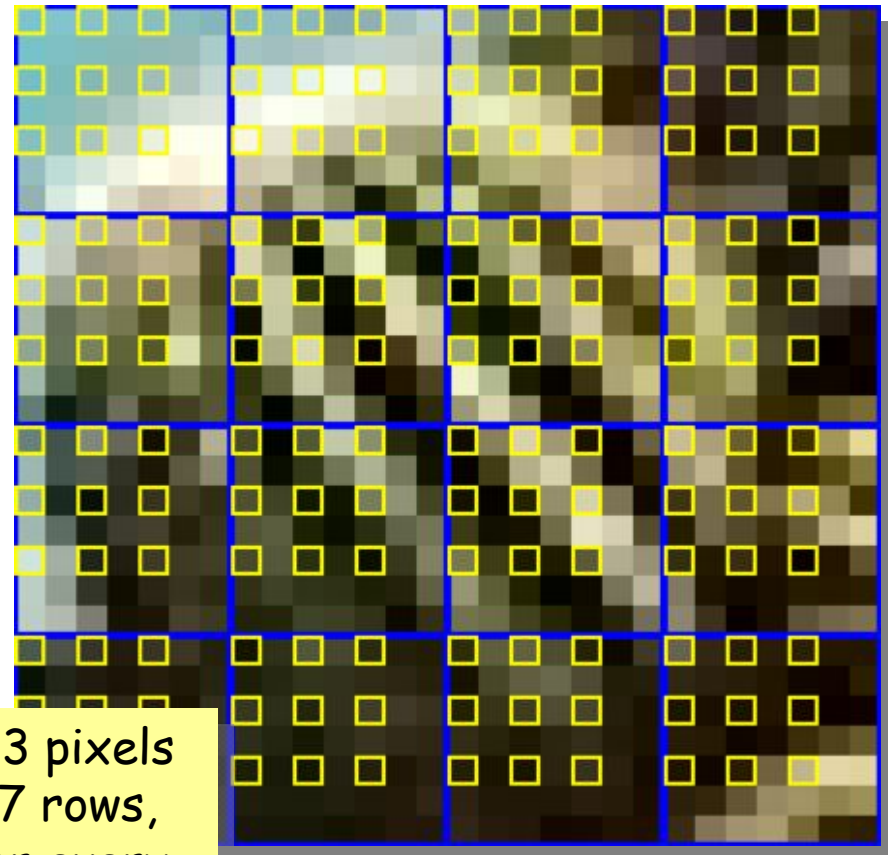
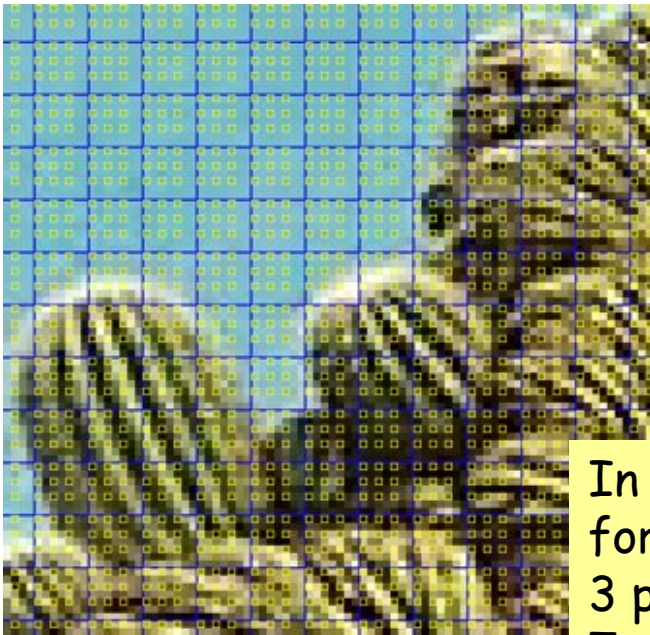
Outlined in
blue: 7x7
pixel squares



Nearest-Neighbor Resampling*

Nearest
Neighbor
Resampling

3/7 resize

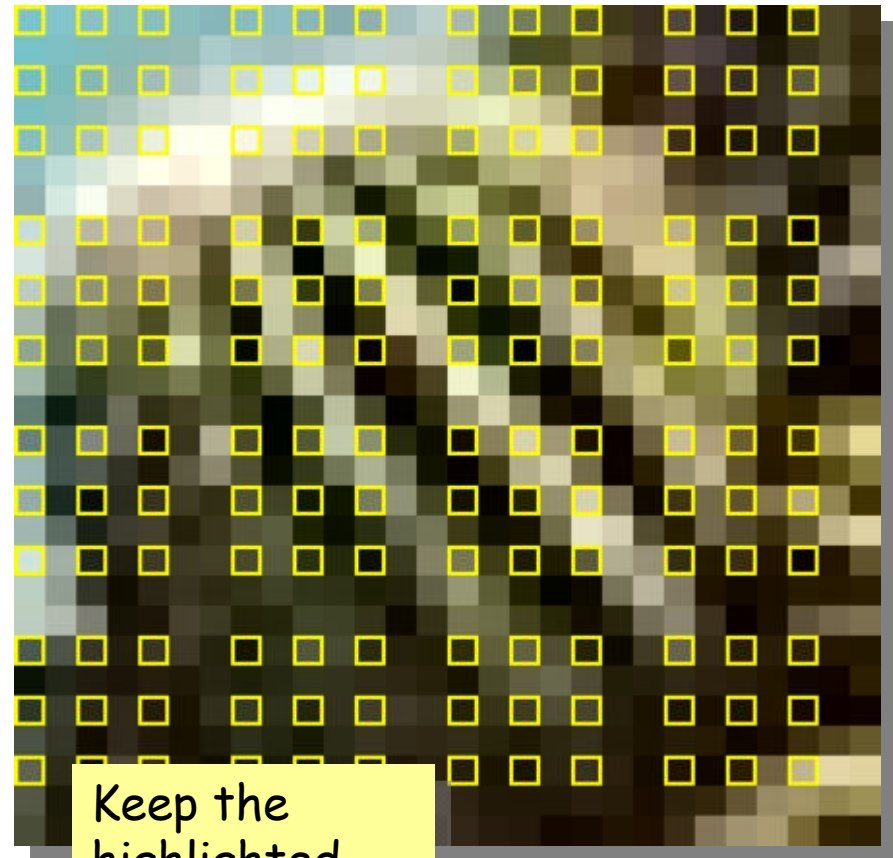
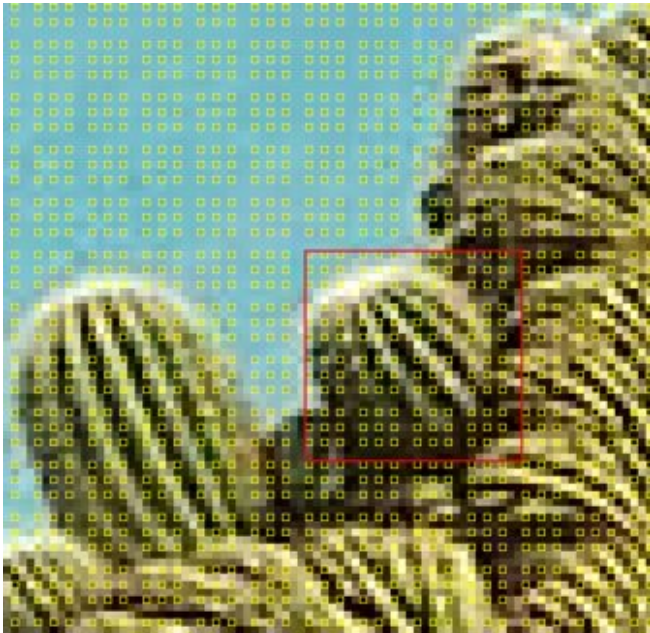


In yellow: 3 pixels
for every 7 rows,
3 pixels for every
7 cols.

Nearest-Neighbor Resampling*

Nearest
Neighbor
Resampling

3/7 resize

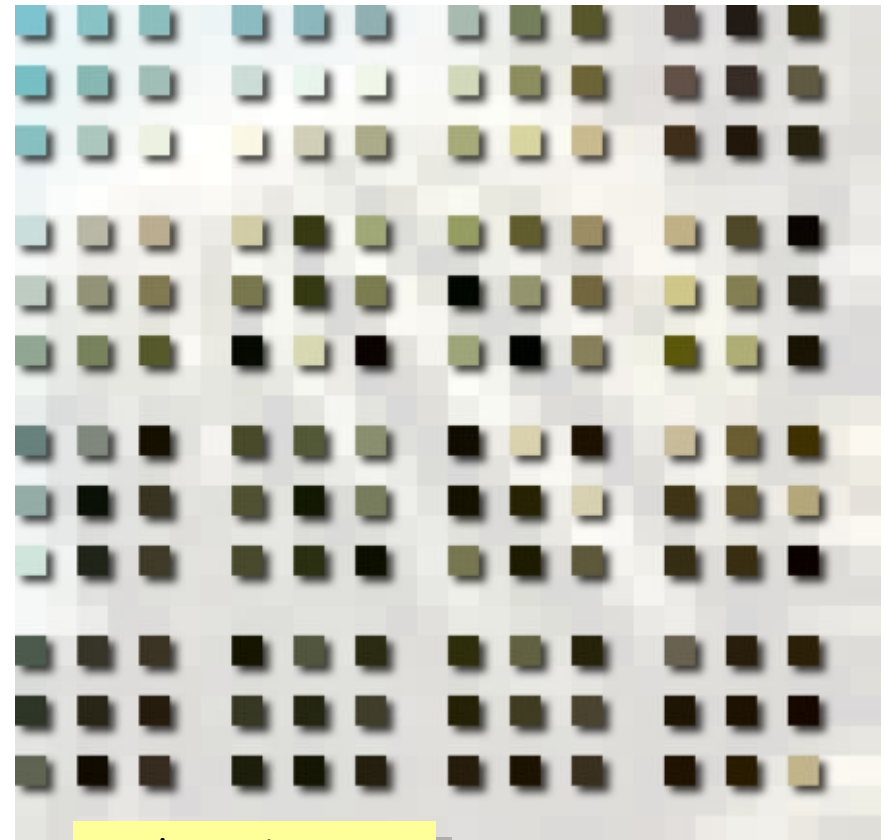
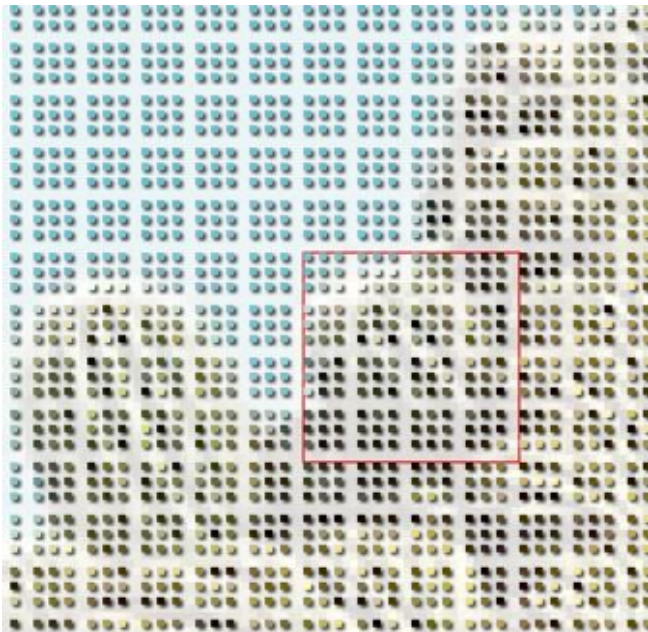


Keep the
highlighted
pixels...

Nearest-Neighbor Resampling*

Nearest
Neighbor
Resampling

3/7 resize

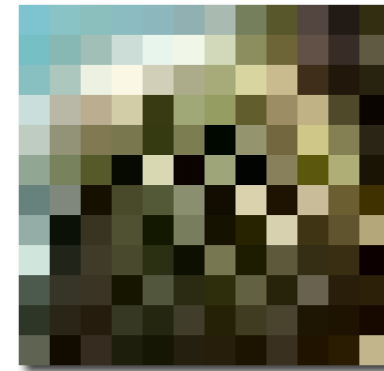


... don't keep
the others.

Nearest-Neighbor Resampling*

Nearest
Neighbor
Resampling

3/7 resize



3/7 times the
linear dimensions
of the original

Resampling Example

Size Reduction 3/7



original

Resampling Example

Size Reduction 3/7 (zoomed)



nearest neighbor

Resampling Example

Size Reduction $3/7$ (zoomed)



bilinear

Resampling Example

Size Reduction $3/7$ (zoomed)



bicubic

NEURAL NETWORKS IN IMAGE PROCESSING

Why ANNs

- Most prominent problems in image processing are already solved by classical methods
- So: Why use ANNs in Image Processing?
- Useful in problems for which no mathematical solution exists
 - Cheaper and easier to let a ANN find a solution than to analyze a problem and design an algorithm
- Computational Speed
 - ANNs are often faster themselves
 - They are easy to implement in parallel

Why ANNs

- Harsh reality of real-life problems
 - Unlikely that there is a solution for the *exact* problem
 - Adaption might destroy optimality
 - Assumptions might be not realistic (for example noise follows a particular distribution)
 - Often not tolerable
 - ANNs are very tolerant of unusual noise distributions
 - So in practice ANNs often outperform their “mathematically optimal” counterparts

Choosing the ANN

- Decisions to make when using ANNs
 - Supervised – Unsupervised
 - Real-Domain – Complex-Domain

Supervised – Unsupervised

- Types most commonly used for IP:
 - Supervised nets:
 - Multiple-Layer Feedforward Network (MLFN)
 - Probabilistic Neural Network (PNN)
 - Unsupervised nets:
 - Kohonen Network
- Most IP problems (for example visual component inspection) provide training samples with labels, so supervised training is more common

MLFN or PNN

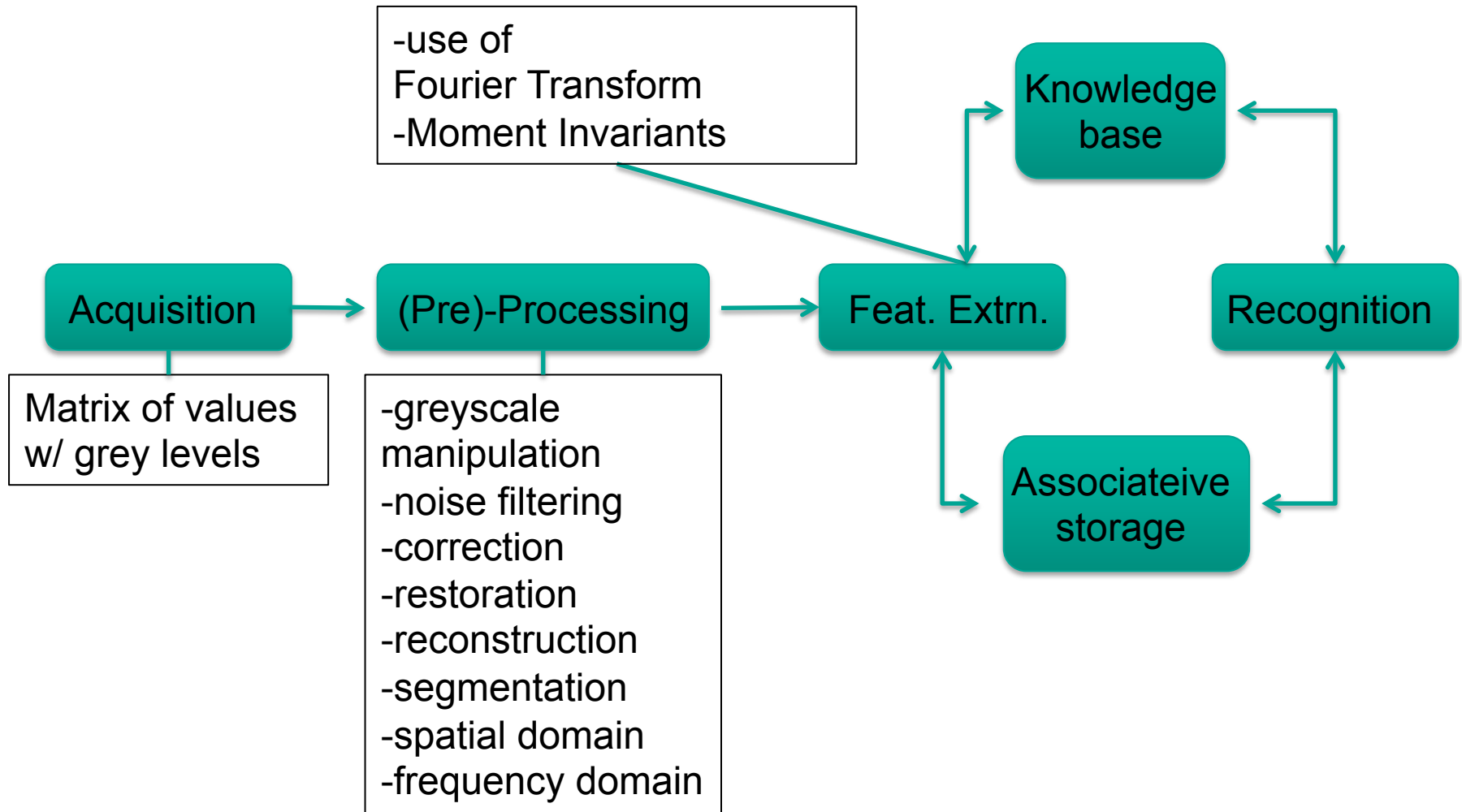
- Rather easy decision, since properties differ a lot

MLFN	PNN
Very long training periods required	Very fast training
Very fast in execution, intrinsically parallel, often best choice for Real-time applications	Execution can be very slow, relatively memory consuming
Analysis is difficult or sometimes impossible	A (asymptotically) mathematically optimal classifier, well understood and mathematically described method of operation
Good if training data is expensive and therefor sparse and better in generalizing	Needs a quite thorough training set but good in handling outliers
Does not provide confidence levels	Byproduct of the computation are the Bayesian posterior probabilities but limited to classification problems (no function approximation, etc.)

Real-Domain – Complex-Domain

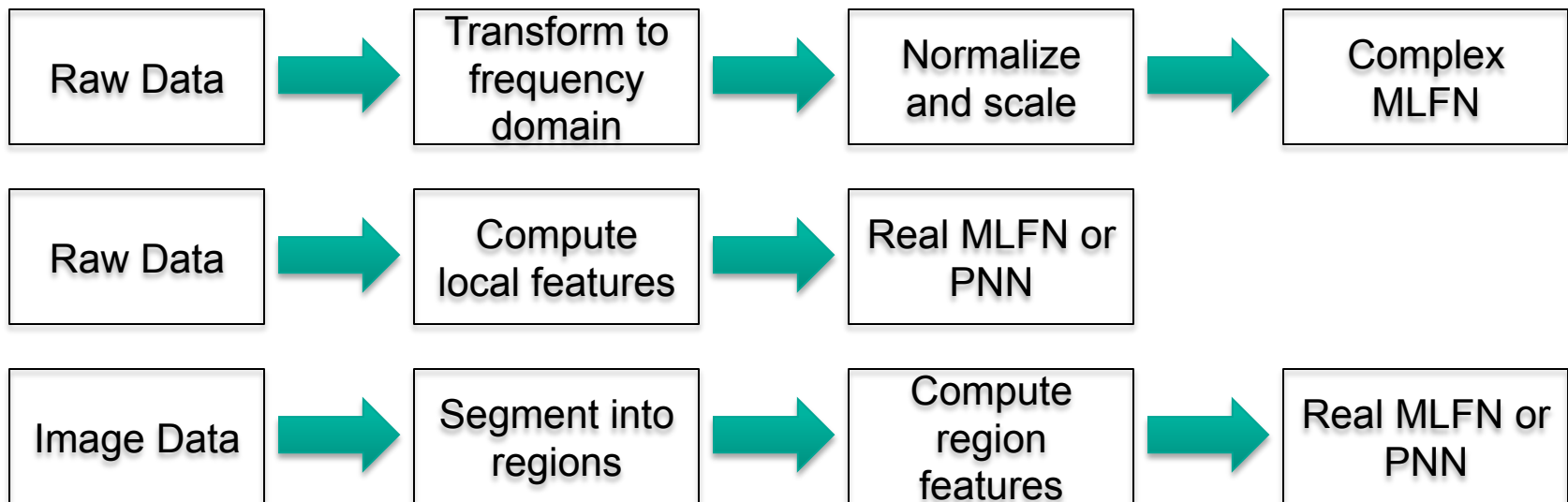
- Traditionally nearly all neural networks operate in the real domain (for complex problems half of the inputs are real numbers and half are imaginary)
- But when application data is inherently complex (frequency domain or phase plane data) performance can be significantly increased by using neurons working in the complex domain
- Training speed and reliability usually increase dramatically
- Generalization is almost always better
- ! Only better when phase information is needed !

Example System Concept



Integrating ANNs with traditional Algorithms

- Different ways of integrating neural networks in IP applications. Most common:



Complex-Domain Units

- When application data is complex it is advised to use neurons that accept complex-valued inputs and produce complex-valued outputs
- Network architecture is essentially the same as in real domain
- Use complex addition and multiplication in the neurons

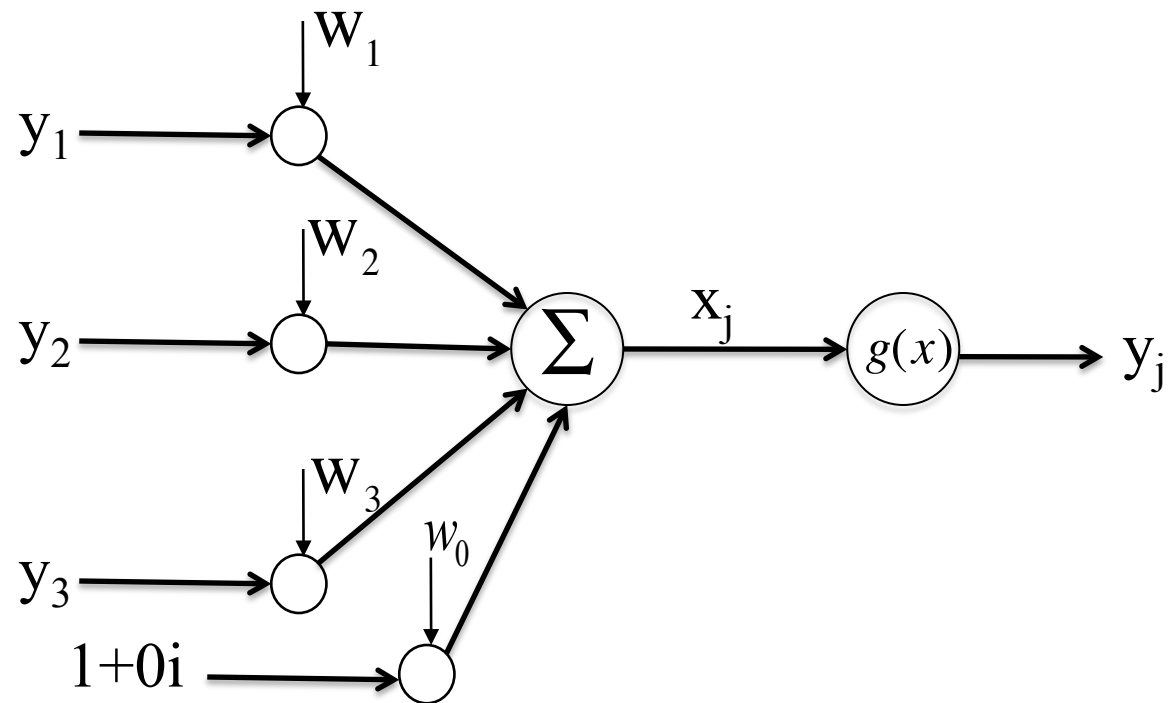
$$(a,b) + (c,d) = (a + c, b + d)$$

$$(a,b) \bullet (c,d) = (ac - bd, ad + bc)$$

- Fewer hidden layers are needed since there are twice as many weights for each
- Special considerations for the outputs when the network is a classifier

Complex-Domain Units

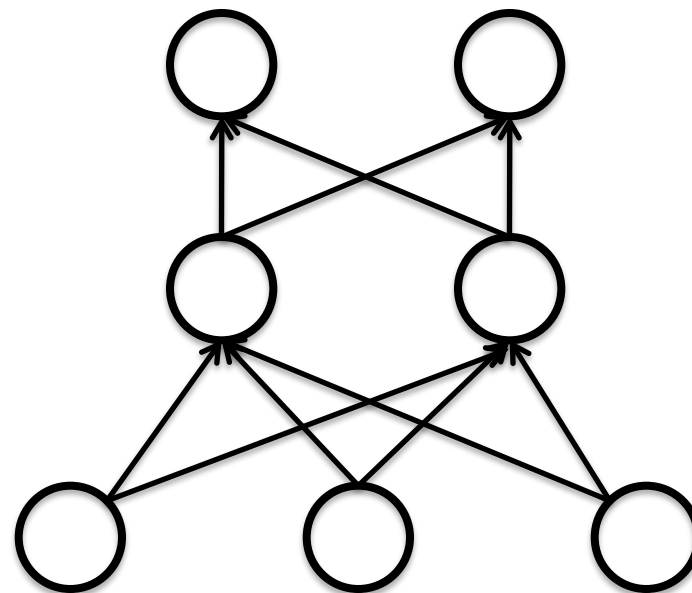
- All inputs and outputs are complex
- But in classification tasks real outputs are sufficient
- All sums and multiplications are complex



$$y_j = g(x_j) = g\left(\sum_i w_{ij} y_i + w_n\right)$$

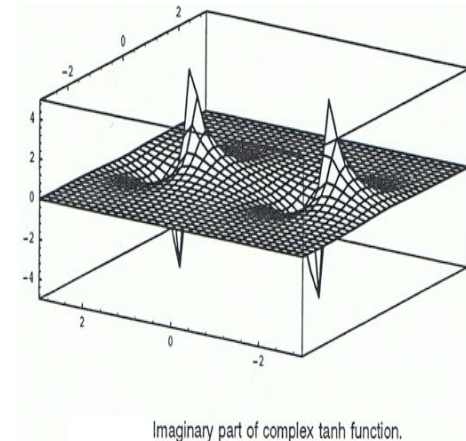
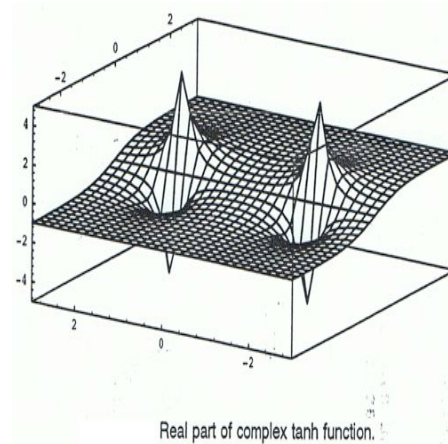
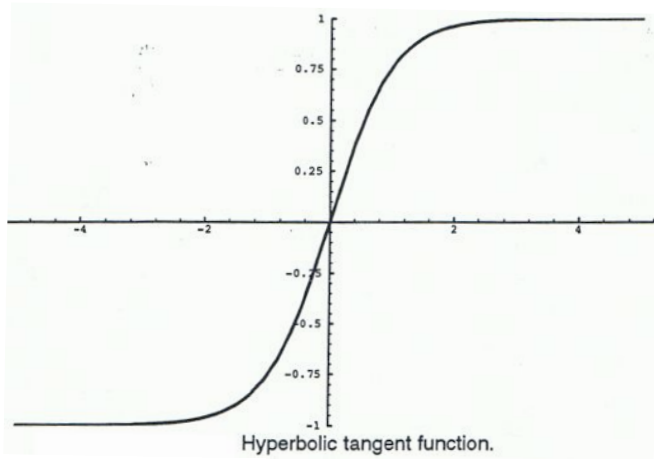
Complex-Domain Networks Network Topology

- Strictly feed-forward networks
- Some nets allow skipping of one or more layers
- Mostly only one hidden layer



Complex-Domain Networks Activation Function

- Choice of activation function in real domain already discussed (differentiable for back propagation etc.)
- More difficult in complex domain, for example hyperbolic tangent:



- Non-differentiable around 0 → Complex hyperbolic tangent not useful!

Complex-Domain Networks Activation Function

- Properties for an activation function:
 - Real and imaginary parts must be differentiable
 - Bounded magnitude
 - Approximately linear when magnitude is small
- Obvious possibility: apply the function separately, e.g.

$$f(x + yi) = \tanh(x) + \tanh(y)i$$

Complex-Domain Networks Activation Function

- Better solution:

$$f(x + yi) = px + pyi$$

$$p = \frac{s\left(\sqrt{x^2 + y^2}\right)}{\sqrt{x^2 + y^2}}$$

- Additional useful property: preserve direction of net input
- With $s(x)$ being a sigmoid function with $s(0)=0$ (like the hyperbolic tangent function)

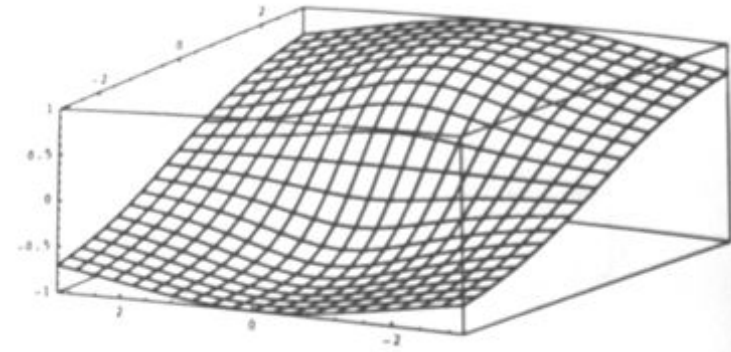


Fig. 2-8: Real part of complex squashing function.

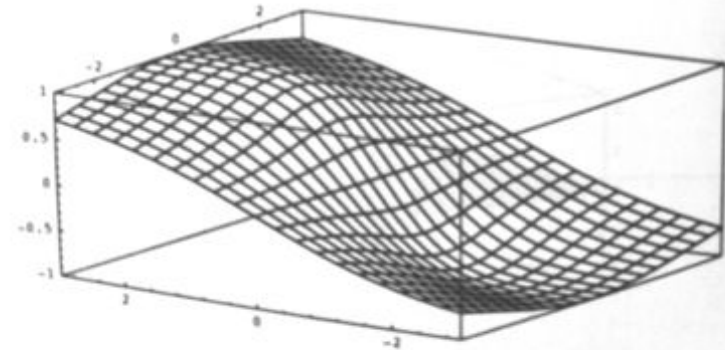


Fig. 2-9: Imaginary part of squashing function.

APPLICATIONS

Face Detection -- Motivation

- A face provides different functions:
 - Person identification
 - Perception of emotional expressions
 - Mouth as source of speech
 - Lipreading
 - Perception of intention
 - Perception of age
 - Perception of gender
 - ...

Face Detection Motivation†

- Computer-based face perception is important for:
 - Human-Machine Interfaces
 - Multimedia
 - Surveillance
 - Security
 - Telephone conferences
 - Communication
 - Animations
 - ...

What makes Face Detection so difficult? †



What makes Face Detection so difficult? †

- Numerous possible illuminations:
 - The same face looks different every time
 - There is a lot of work about normalizing shadow effects and reducing illumination changes
 - Different people's faces lit from the same direction are more similar than the same person's face lit from different directions



What makes Face Detection so difficult? †

- Rotation:
 - Undoing rotation too expensive
 - How to detect rotated faces directly? Different classifiers for different rotations?



What makes Face Detection so difficult? †

- Intrinsic variations of facial appearance

Source	Possible Tasks
Identity	Classification, known-unknown, verification, identification
Facial expressions	Inference of emotion
Speech	Lip-reading
Sex	Deciding whether male or female
Age	Estimating age

- Extrinsic variations of facial appearance

Source	Effects
Viewing geometry	Pose
Illumination	Shading, color, self-shadowing, specular highlights
Image Process	Resolution, focus, imaging noise, perspective effects
Other objects	Occlusion, shadowing, indirect illumination

What makes Face Detection so difficult? †

- Face Detection is comparable to detecting moving object in noisy scenes
 - Face handled as a sub-surface on the head
 - Appearance changes with its parameters!
- Boundaries of faces not clear: depends on hair styles (although there are approaches trying to do that)
- It is impossible to model intrinsic parameters analytically

What makes Face Detection so difficult? †

- Classification face / non face is very complex (we can not model the whole world!)
- Intrinsic and extrinsic parameters do not cover different make-up styles, glasses, jewelery, ...
- Biggest problem is head pose & lighting
 - Face appearance changes dramatically with its pose !
- Perception of faces is highly dynamic in space, time and its context
 - A person is more likely to be found behind a desk than on top of a book shelf.

Computerized Face Representation†

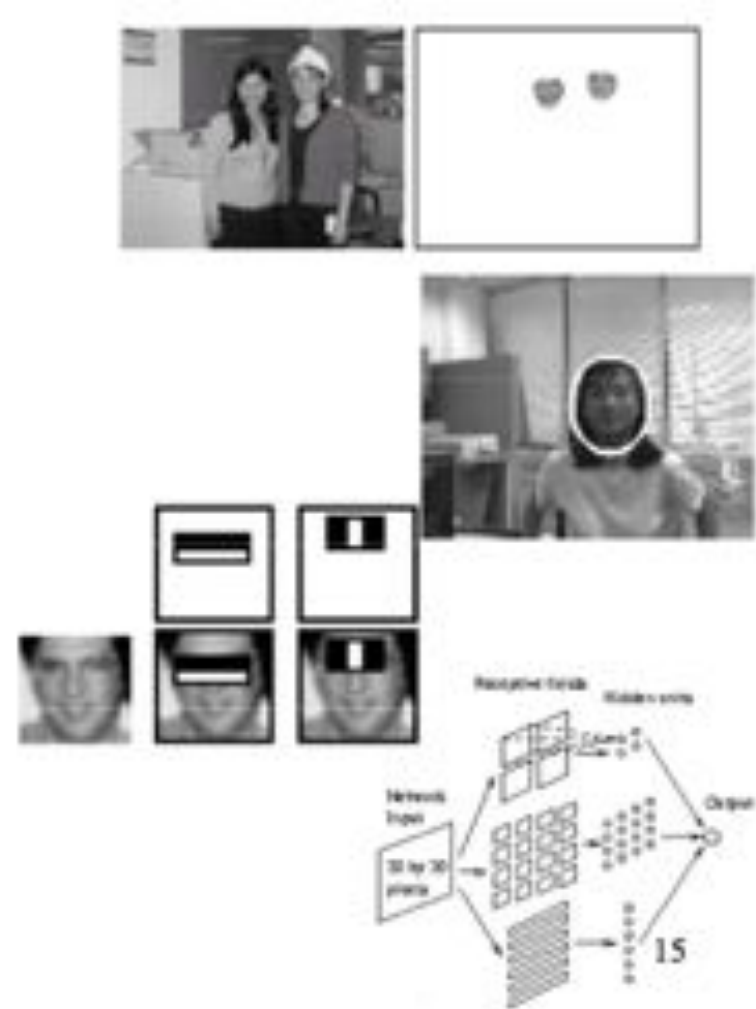
- Local feature-based face representation:
 - Find facial features (such as mouth corners, eyes, nostrils, ...) and check for well-defined spatial configuration (a priori knowledge)
 - Use of relatively high resolution images (> 60x60 pixels)
 - Humans can detect faces even within 15x15 pixels and lower !
 - Problems with Occlusions: what to do if several parts are not visible?
 - Problem of face detection is divided in multiple detection tasks, each nearly as difficult itself as face detection alone

Computerized Face Representation†

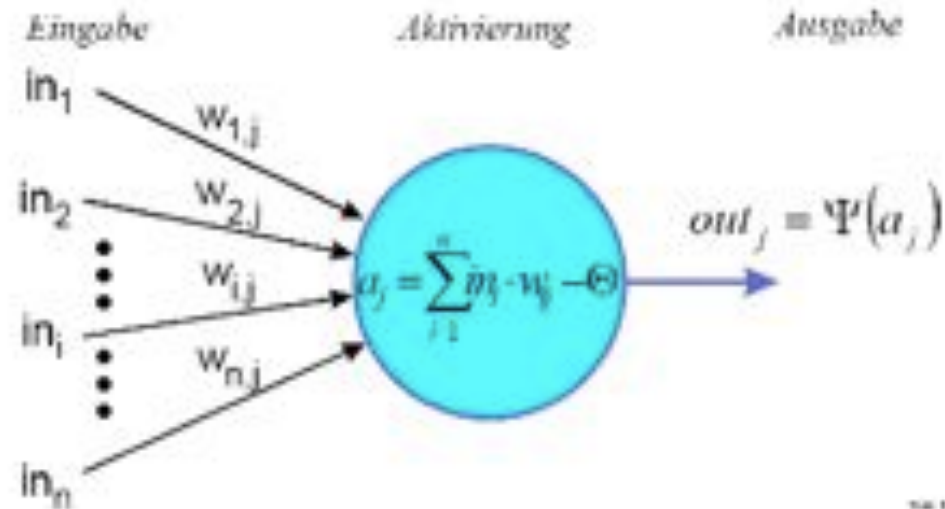
- Holistic face representation:
 - No a priori knowledge
 - No subdivision into single parts
 - Relatively low resolution possible
 - Occlusions handled as statistical outliers
 - Variances handled by different preprocessing steps, region of interest limitation and statistical learning
 - Detection can be achieved by
 - Classification between face images / non-face images
 - Application of generic face model on region of interests Detect faces by scoring how well models fit

Different Face Detection Approaches†

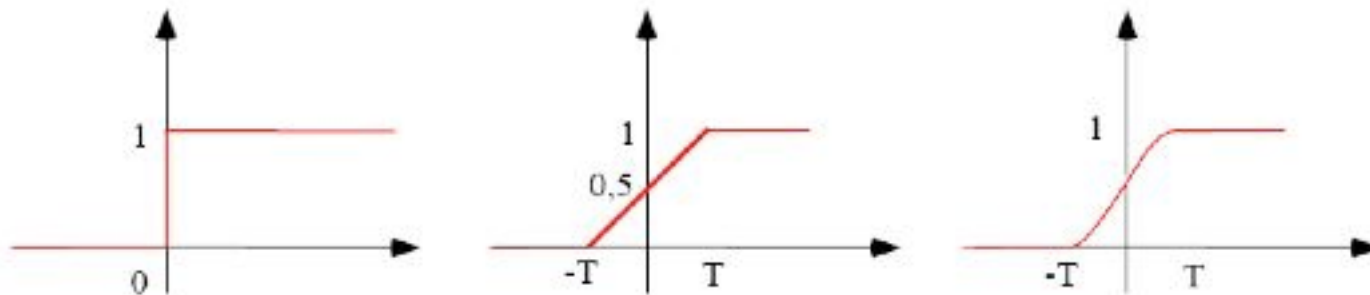
- Skin-color detection
- An elliptical head model
- Using Haar-Filter cascades (Viola & Jones)
- Artificial neural networks



ANN Face Detection Neuron Model†



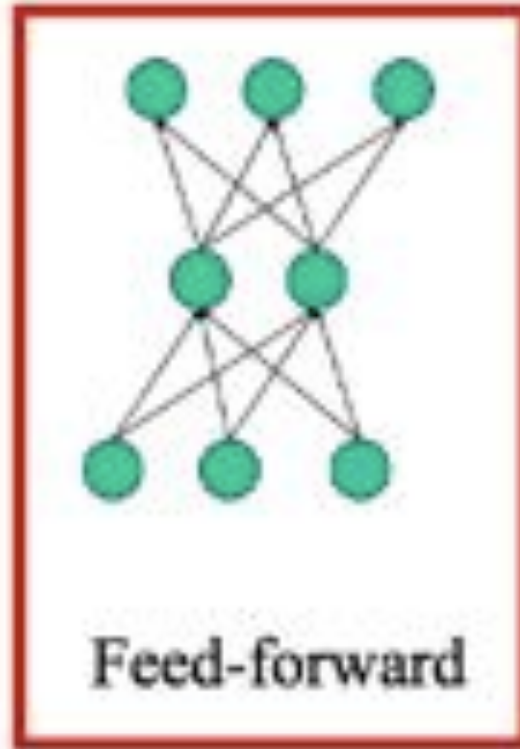
Different outputs possible, depending on activation function:



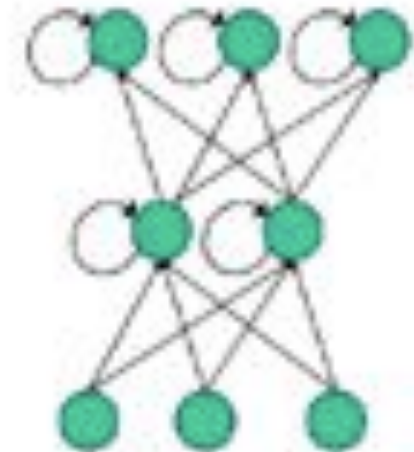
ANN Face Detection Topologies



Fully connected



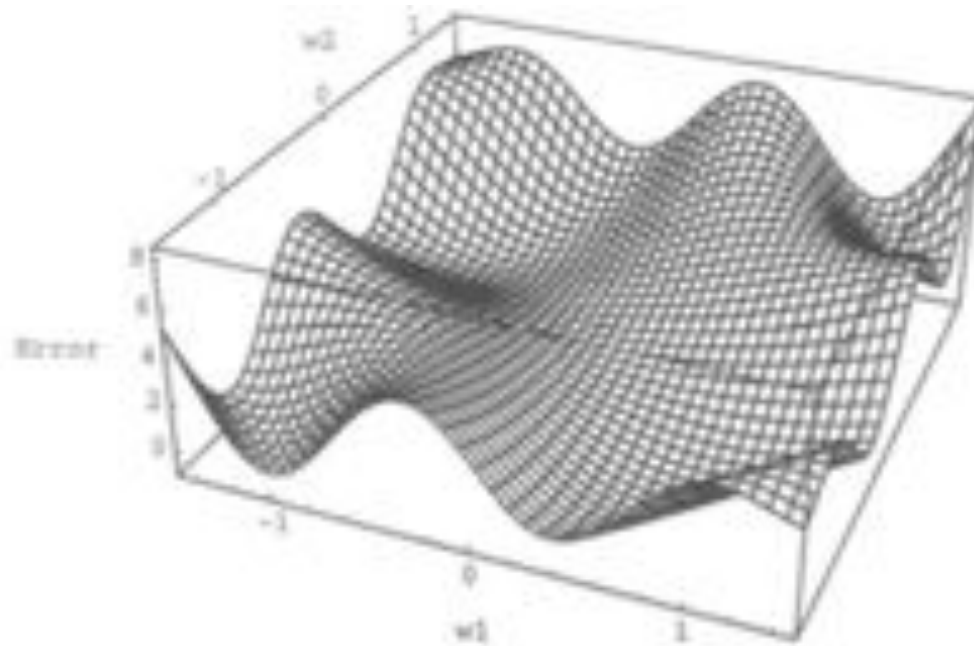
Feed-forward



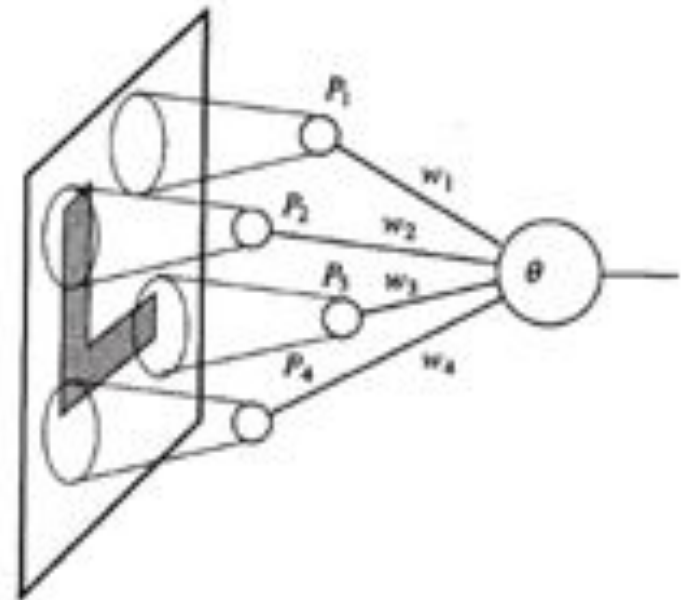
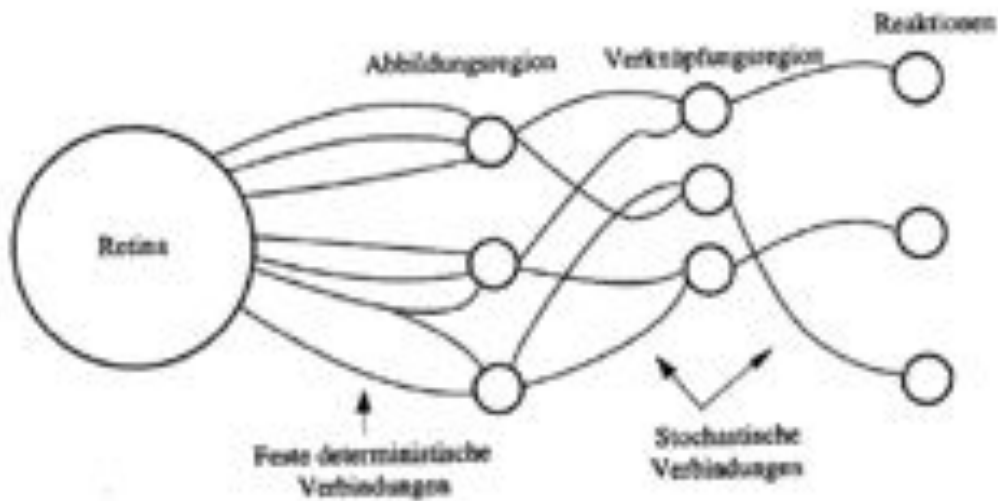
Recurrent

ANN Face Detection Training†

- Learning the connectionist weights is achieved by descending the gradient of the resulting output error E (Backpropagation Algorithm)



ANN Face Detection†



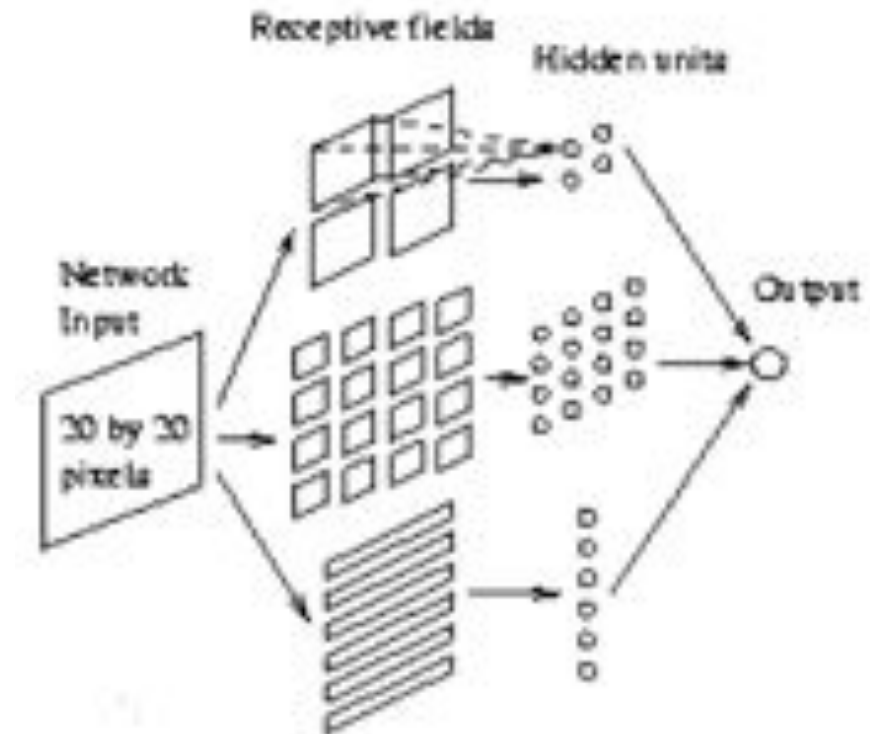
ANN Face Detection†

- *General approach for detecting (upright, frontal) faces with NN:*
- *Network receives as input a 20x20 pixel region of an image*
- *output ranges from -1 (no face present) to +1 (face present)*
- *the neural network „face-filter“ is applied at every location in the image*
- *to detect faces with different sizes, the input image is repeatedly scaled down*

Neural Network Based Face Detection, by Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 20, number 1, pages 23-38, January 1998.

ANN Face Detection Network Topology[†]

- ANN Topology:
 - 20x20 pixel input retina
 - 4 types of receptive hidden fields
 - One real-valued output



ANN Face Detection Network Topology†

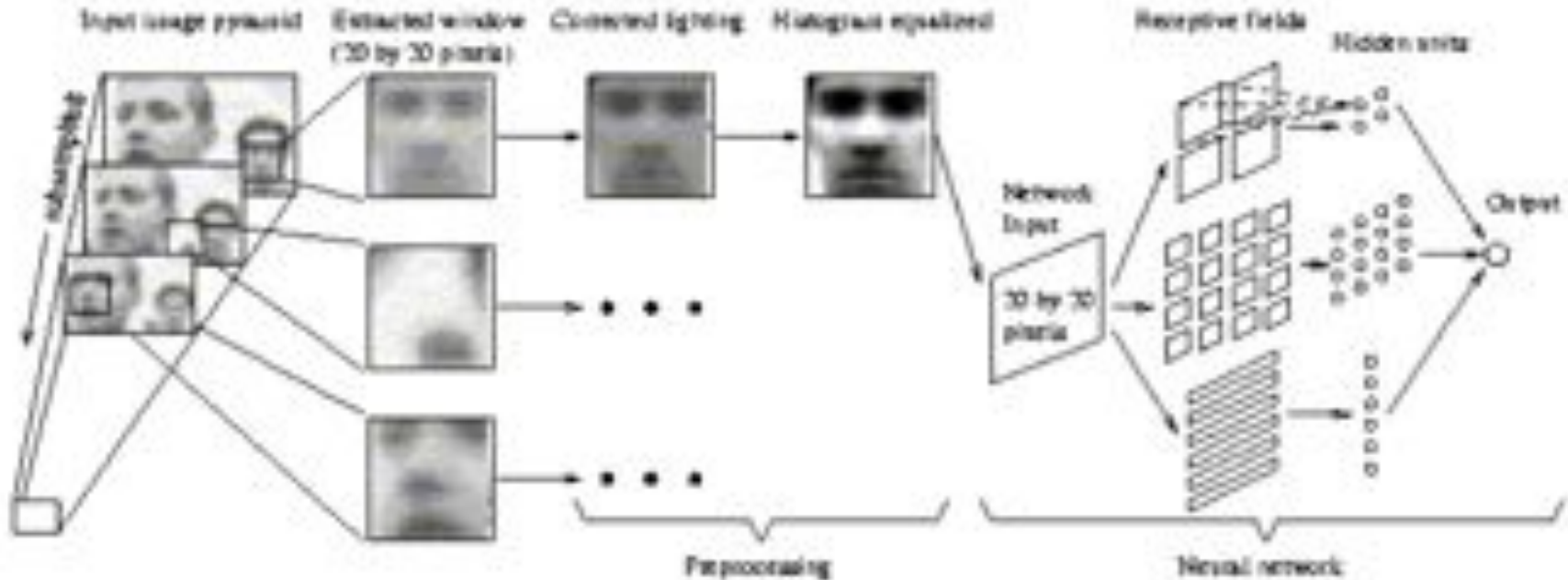


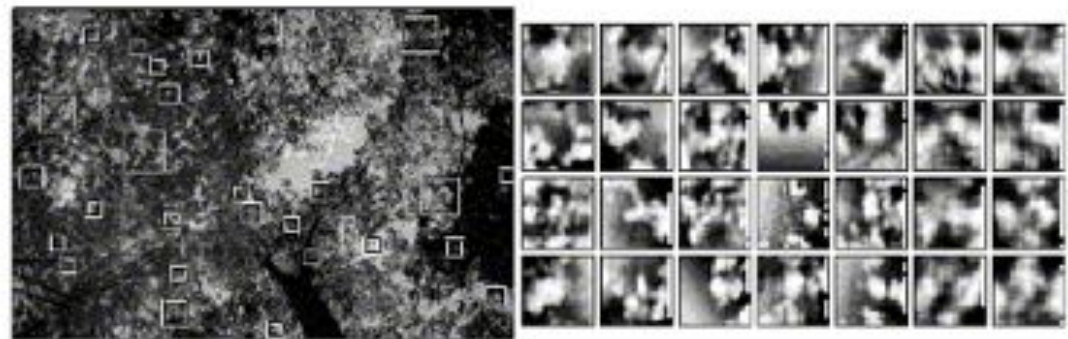
Figure 1: The basic algorithm used for face detection.

ANN Face Detection Training Set[†]

- 1050 normalized face images
- 15 face images generated by rotating and scaling original face images
- 1000 randomly chosen non-face images



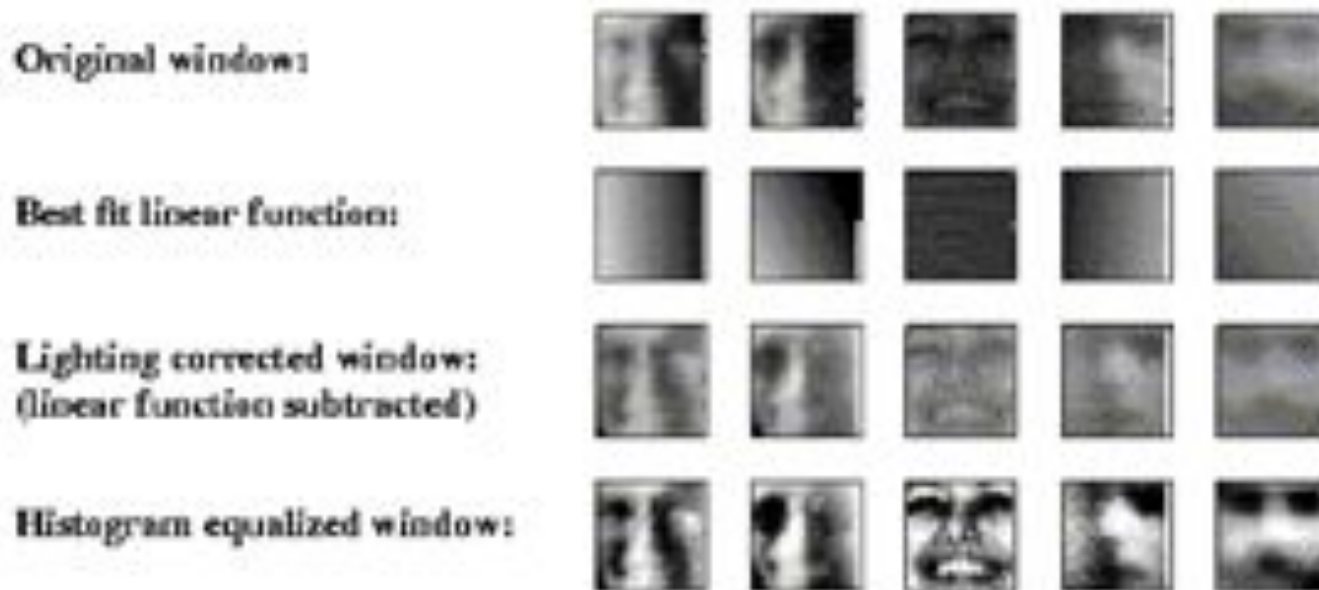
Face Samples



Non-Face Samples

ANN Face Detection Preprocessing†

- Correct for different lighting conditions (overall brightness, shadows)
- Rescale images to fixed size
- Often: extract relevant features (edges, FFT, wavelets, PCA, DCT, ...)



ANN Face Detection Training†

- Training Procedure:
 1. randomly choose 1000 non-face images
 2. train network to produce 1 for faces, -1 for non-faces
 3. run network on images containing no faces. Collect subimages in which network incorrectly identifies a face (output > 0)
 4. select up to 250 of these „false positives“ at random and add them to the training set as negative examples

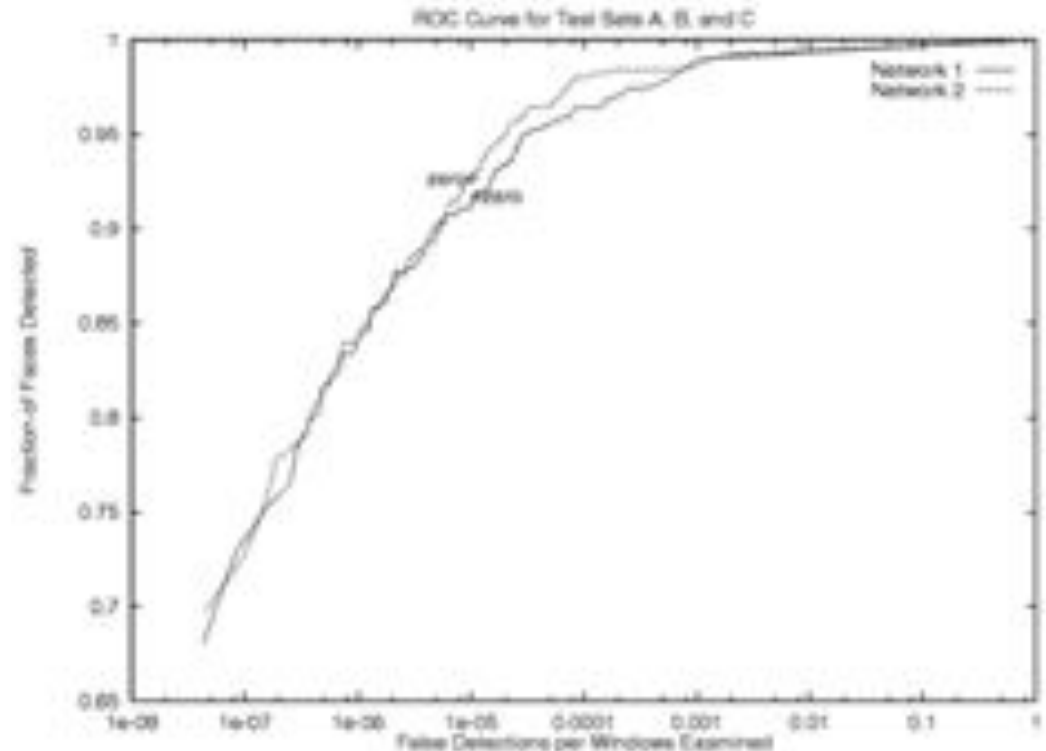
ANN Face Detection Results[†]

Results on 130 images containing 507 faces (83 Million subregions examined):

- 92.5 % detection rate at false detection rate of 1/96402 (862 false detections)
- 77.9 % detection rate at false detection rate of 1/41 Mio. (2 false detections)

Speed:

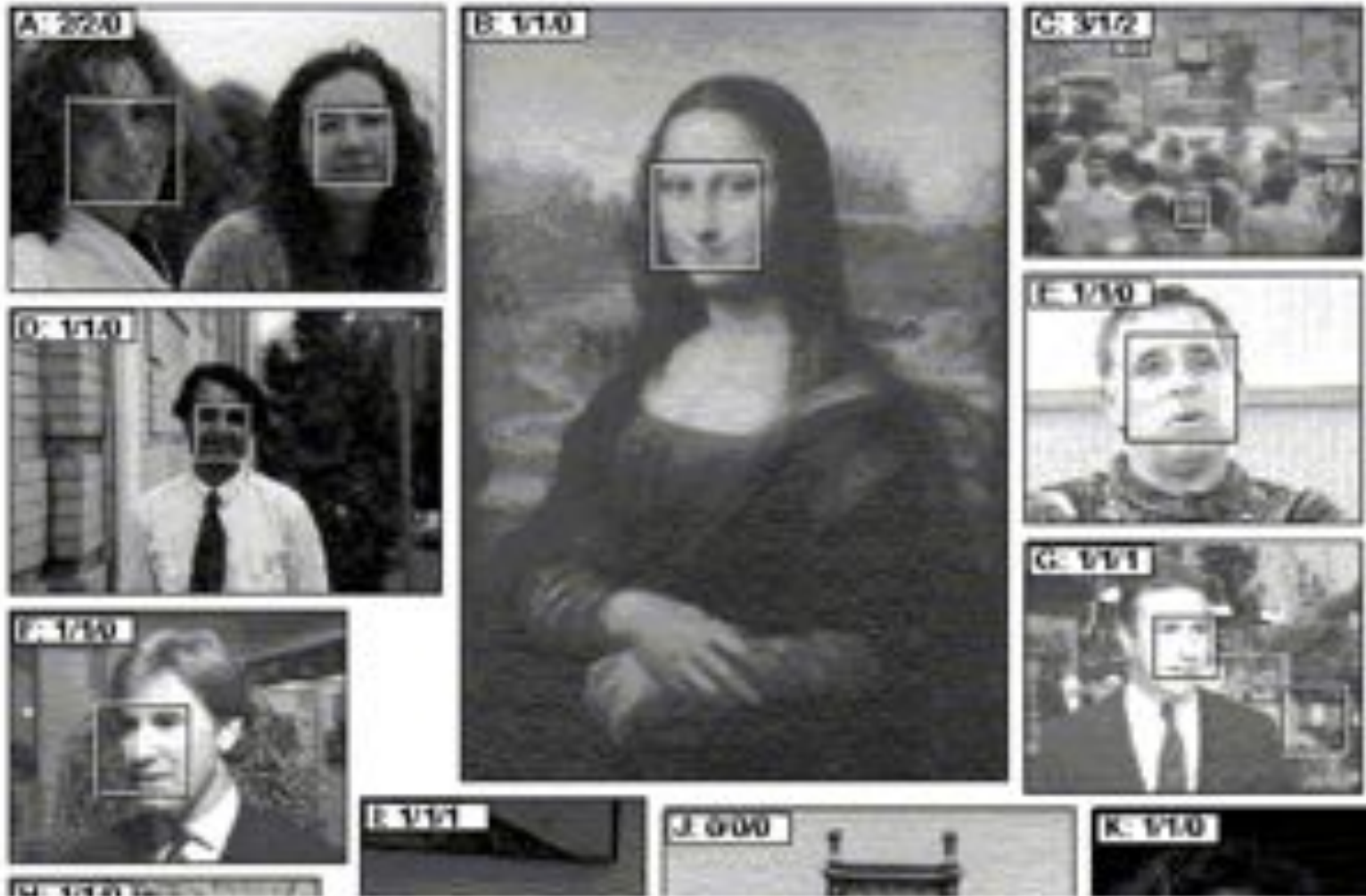
- best system, using two networks: 383 seconds per image
- tuned for speed (77% detection rate): 7.2 seconds per image



ANN Face Detection Results[†]



ANN Face Detection Results[†]



Focus of Attention and Head Pose Estimation†

- Tracking Gaze and Focus of Attention
 - In meetings:
 - to determine the addressee of a speech act
 - to track the participants' attention
 - to analyse, who was in the center of focus
 - for meeting indexing/retrieval
 - Interactive rooms
 - to guide the environment's focus to the right application
 - to suppress unwanted responses
 - Virtual collaborative workspaces (CSCW)
 - Human-Robot Cooperation
 - Cars (Driver monitoring)

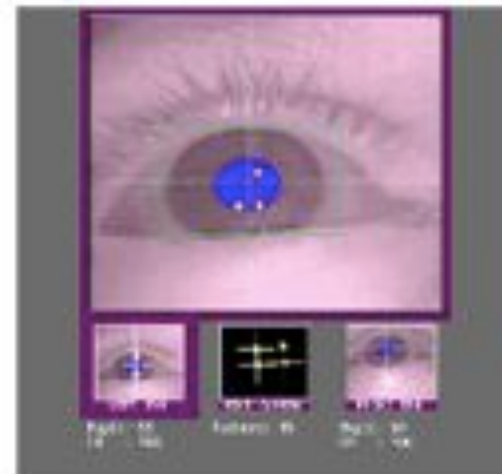
Focus of Attention and Head Pose Estimation†

- Attentional signals:
 - Eye gaze
 - Head orientation
 - Body posture
 - Gestures
- Head Orientation is a good cue to predict focus of attention !
- Eye-gaze is difficult to track ...

Focus of Attention and Head Pose Estimation†

Vision-based Eye Gaze Tracking – Problems:

- Requires the user to wear head gear or position of the user relative to the cameras rather fixed
- Certainly not acceptable for everyday use in multimodal rooms



Head Pose Estimation†

- Model-based approaches:
 - Locate and track a number of facial features
 - Compute head pose from 2D to 3D correspondences (Gee & Cipolla '94, Stiefelhagen et al '96, Jebara & Pentland '97, Toyama '98)

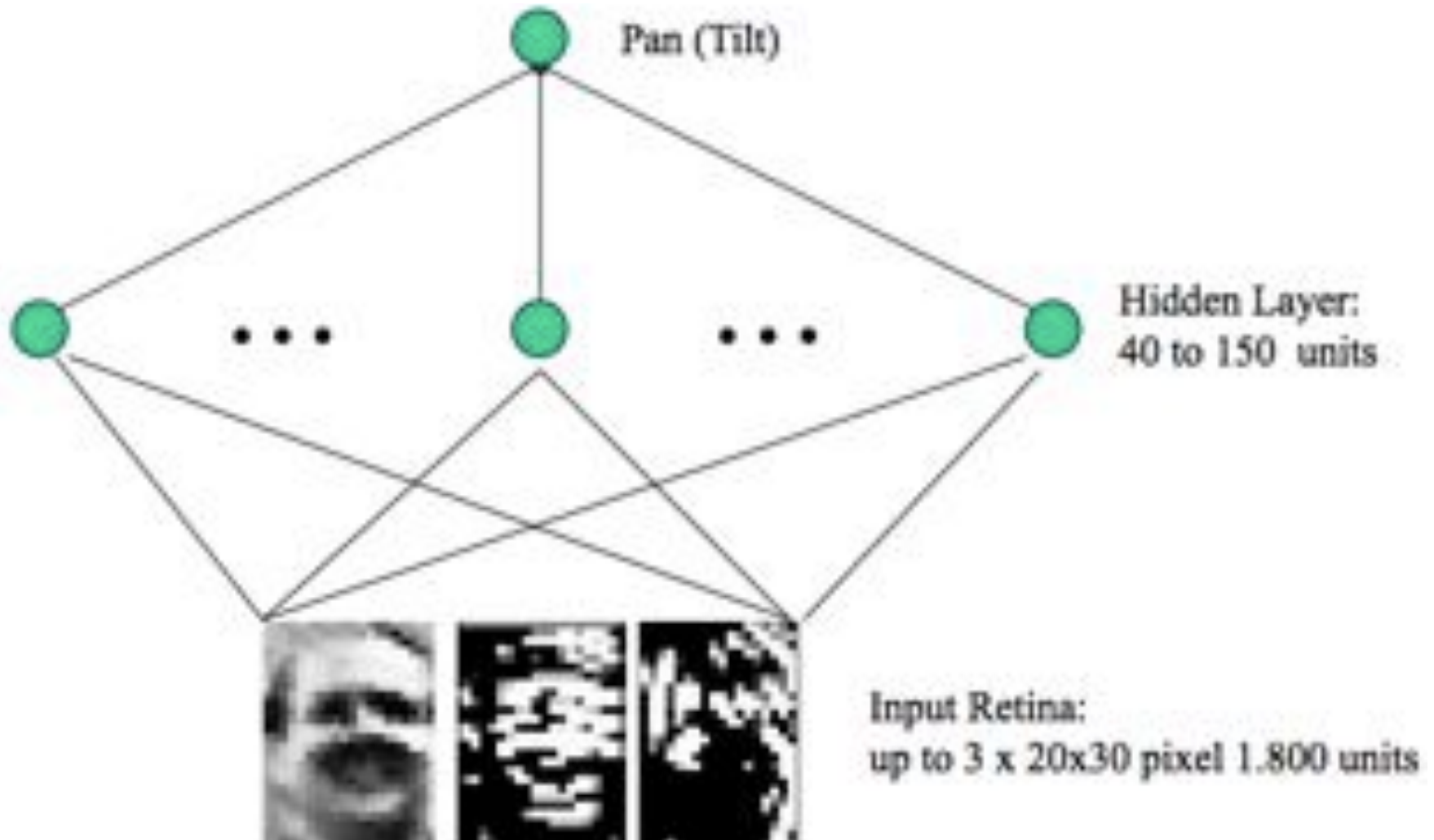
- Appearance-based approaches:
 - Estimate new pose with function approximator (such as ANN) (Beymer et al. '94, Schiele & Waibel '95, Rae & Ritter '98)
 - Use face database to encode images (Pentland et al. '94)

Estimating Head Pose with ANNs[†]

- Train neural network to estimate head orientation
- Preprocessed image of the face used as input



Estimating Head Pose with ANNs – Network Topology†



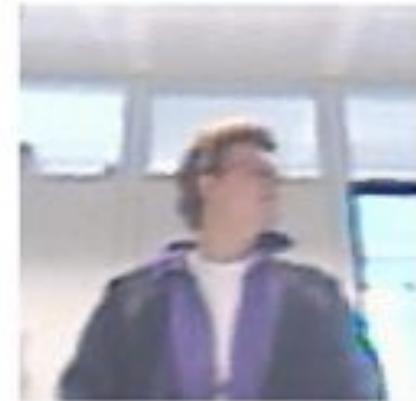
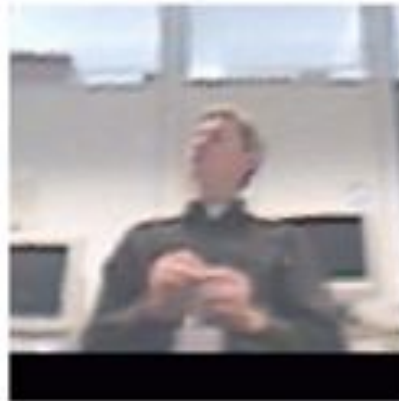
Estimating Head Pose with ANNs – Preprocessing†

- Automatic extraction of faces (using a skin-color model)
- Preprocessing
 - a) Histogram normalization of grayscale images
 - b) Extracting horizontal- and vertical edges
 - c) Down-sampling to 20x30 pixel



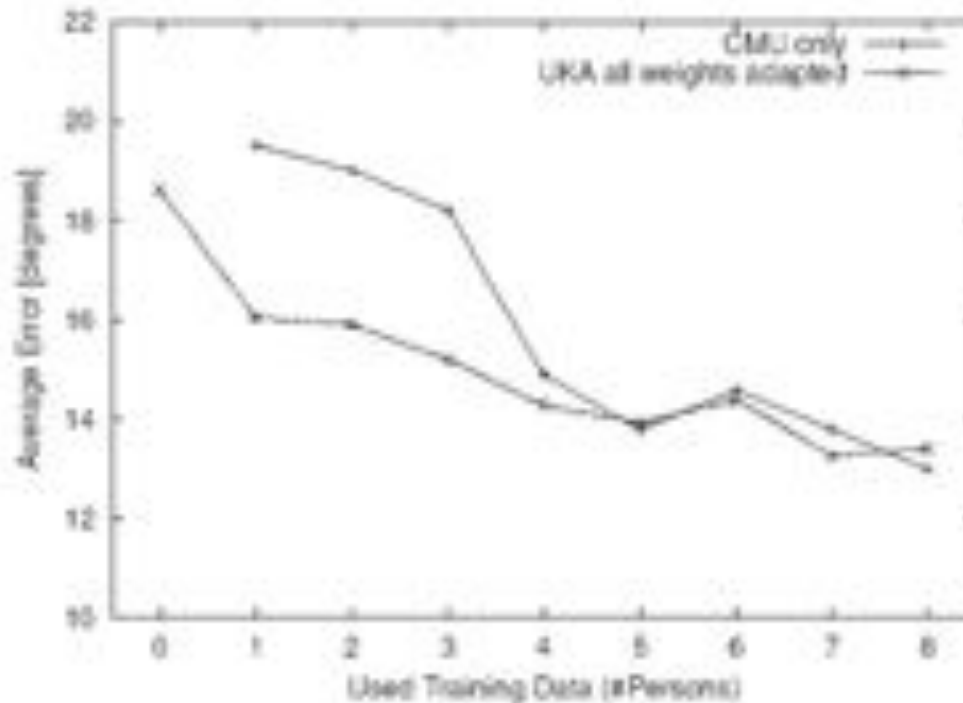
Estimating Head Pose with ANNs – Data Collection†

- Estimate head pan and tilt from the facial image
- Data Collection:
 - Perspective views of users are collected with pano-cam
 - Labels from a magnetic field pose tracker
 - Data from fourteen users, at four positions around table collected



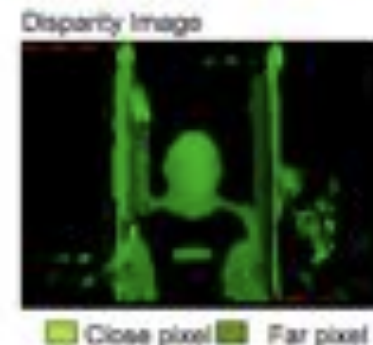
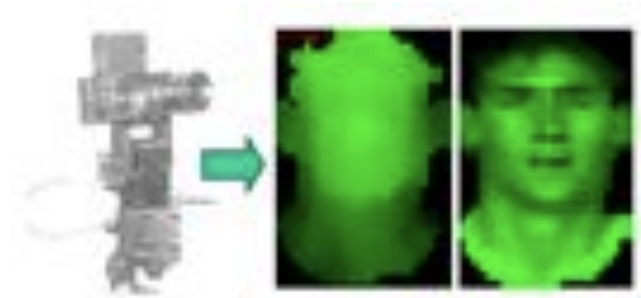
Estimating Head Pose with ANNs – Problems†

- Illumination Changes are problematic
- Retraining / Adaptation is necessary

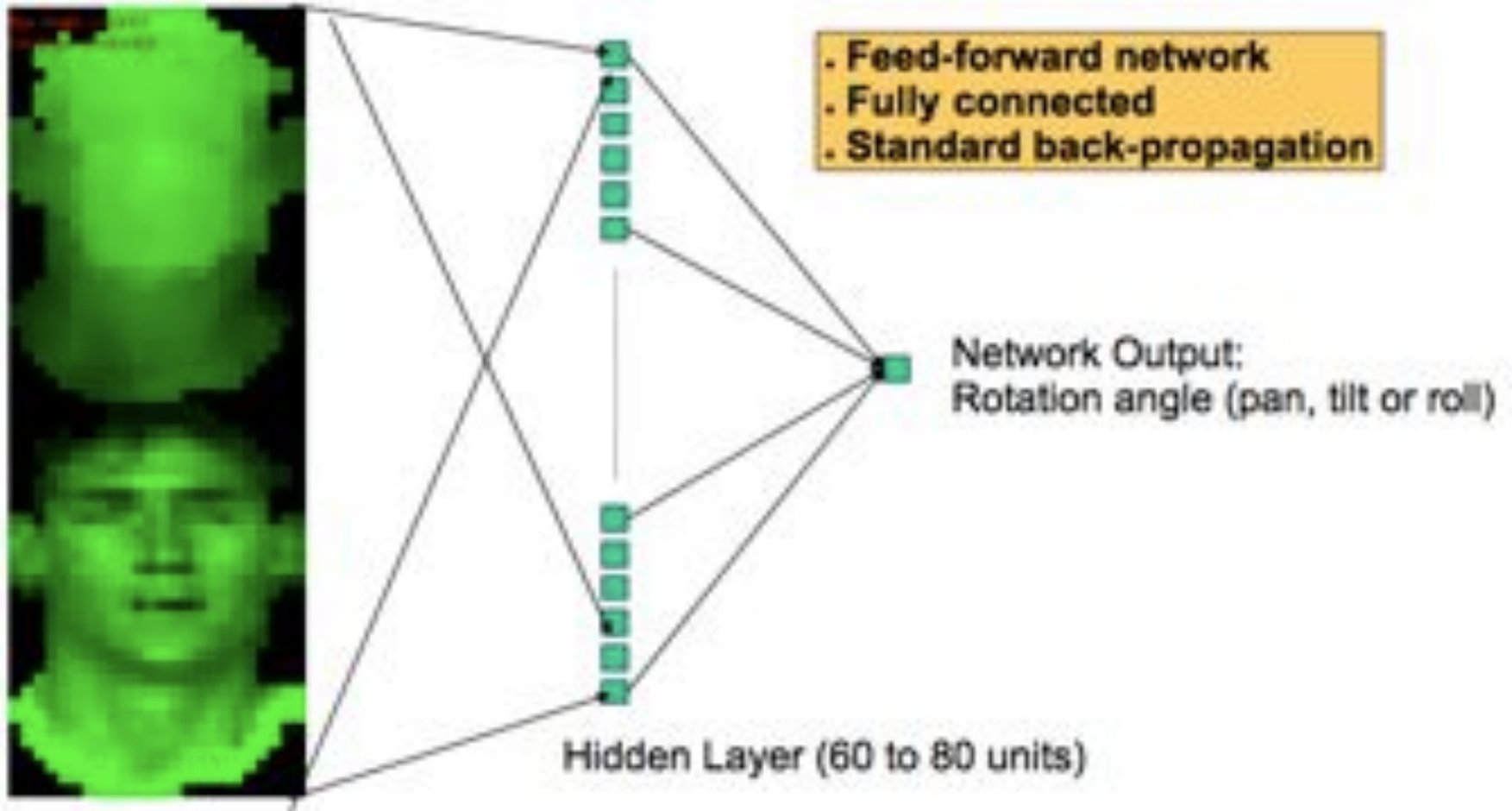


Estimating Head Pose Using Depth from Stereo†

- Using stereo cameras, the distance of a point on the object from the cameras can be computed
 - Distance r of a pixel is inversely related to the difference in projections of the point in the two camera views („disparity“)
$$r = (b \cdot f) / (dL - dR)$$
 - Disparities are computed for each pixel by finding correspondences in the two views
- Idea
 - Disparity (depth) images should be less affected by illumination changes than monocular greyscale images
 - Since both camera views share the illumination change, correspondence finding should still be robust
 - Use disparity images for pose estimation



Estimating Head Pose Network Architecture†

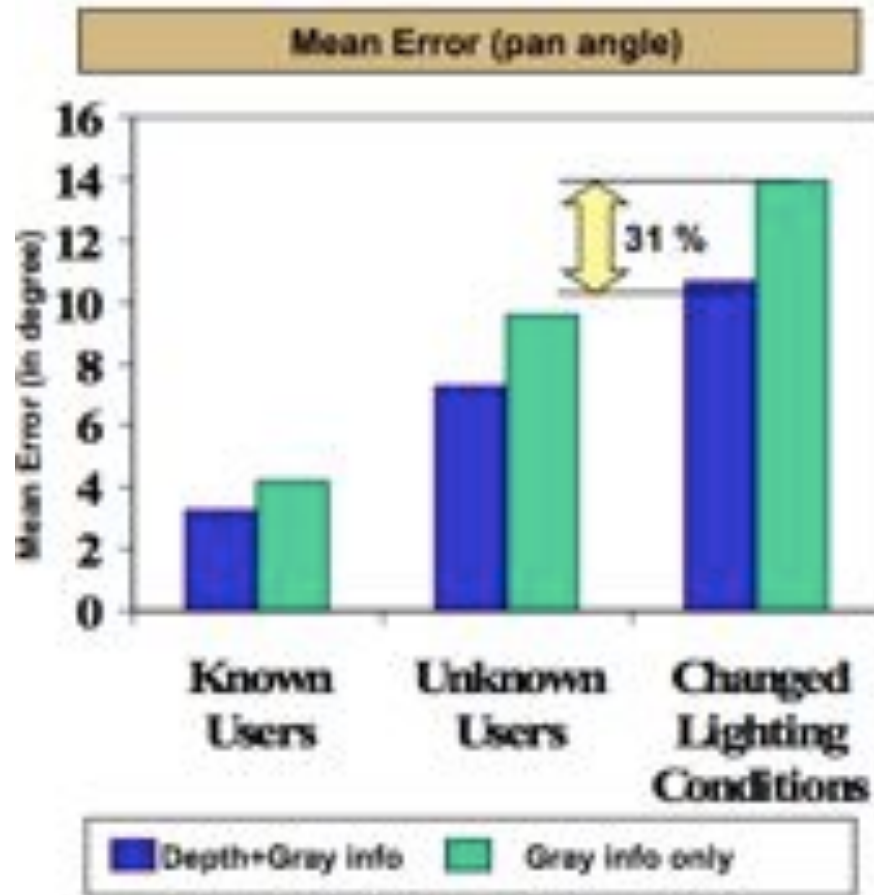


Estimating Head Pose Data Collection†

- 10 users
- 2 different lighting conditions (day light, artificial light)
- 250-500 images per person and lighting condition
- Reference angles captured with magnetic sensor (FoB)
- Resolution 640x480
- Test persons could move freely in pan, tilt and roll direction



Estimating Head Pose Results under Changed Illumination†



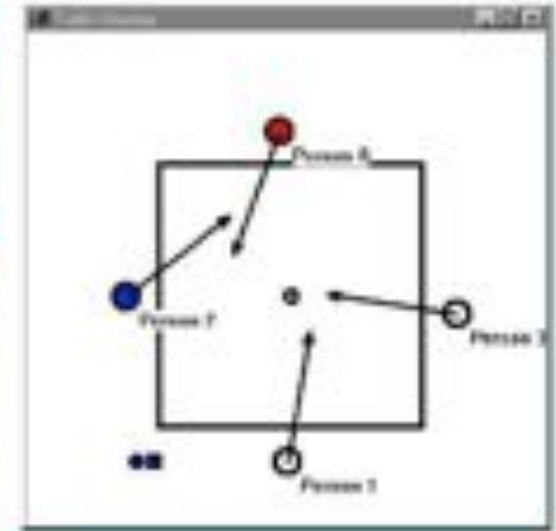
Tracking Focus of Attention (FoA) †

- **Focus of Attention tracking:**
 - To detect a person's interest
 - To know what a user is interacting with
 - To understand his actions/intentions
 - To know whether a user is aware of something
- **Human-Human Interaction:**
 - To determine the addressee of a speech act
 - To understand the dynamics of interaction
 - For meeting indexing / retrieval
- **Human-Robot Interaction:**
 - Was the robot addressed or not?
- Smart Environments, Cars, ...

FoA Tracking in meetings[†]

- Why:
 - to determine the addressee of a speech act
 - to track the participants attention
 - to analyse, who was in the center of focus
 - for meeting indexing / retrieval, ...
- How:
 1. Track all participants' faces (color)
 2. Estimate their head orientations (ANNs)
 3. Map head orientations onto likely targets (e.g. the other participants)

FoA: Detection of likely Targets[†]



- Idea: For each person, find the most likely target person T , given the observed head orientation x

Modeling Focus from Head Orientation†

- Head Orientation is a strong indicator of social attention
 - Eye-Gaze is difficult to measure
 - Estimate a persons focus of attention based on his head orientation
 - can be formulated with Bayes rule:

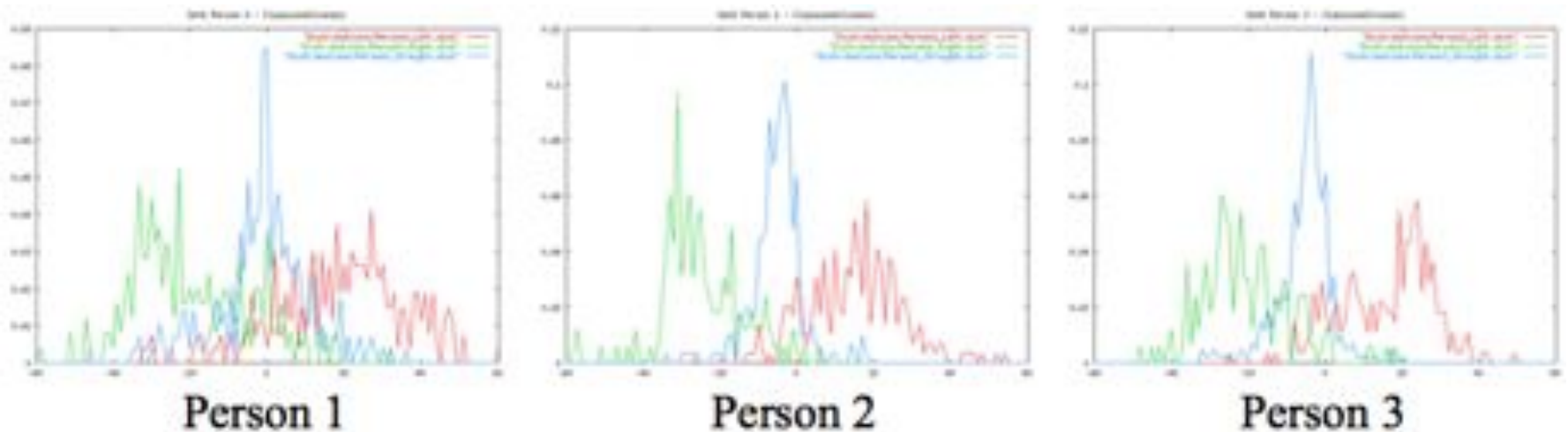
$$P(Focus_S = T | x) = \frac{p(x | Focus_S = T) \cdot P(Focus = T)}{p(x)}$$

x: head pan in degrees,

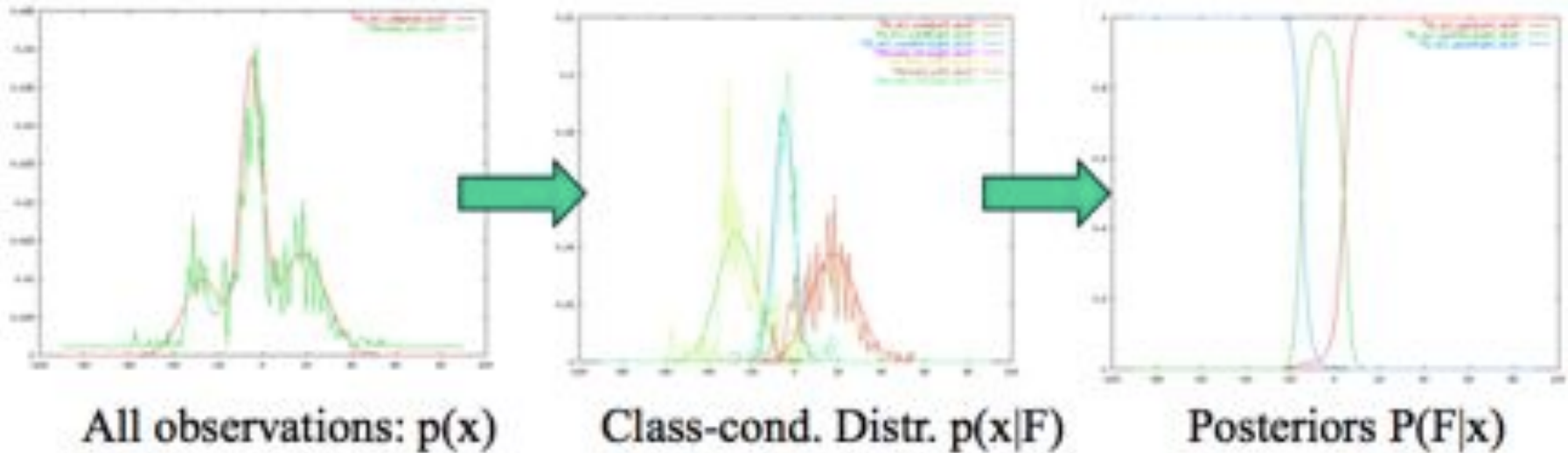
T \in {Person 1, Person 2, ..., Person M }

Head Pan Distributions $p(x|F_i)$ †

- Head pan distributions are dependent on personal head-turning “styles” and location of targets
- Distributions should be adapted for each person and each meeting



Adaptation of the Model†



- $P(x)$ is modeled as mixture of Gaussians:
$$p(x) \approx \sum_{j=1}^M p(x | j)P(j)$$
- Model parameters are found using EM-algorithm
- Individual Gaussian components are used as class-conditionals $p(x|F)$
- Priors of the mixture model $P(j)$ are used as focus prior $P(F = T)$

Focus based on Head Orientation†

- Results on four meetings
 - 4 participants in each meeting
 - Participants automatically detected and tracked
 - Unsupervised adaptation of model parameters

	P(Focus Gaze)
Meeting A (4 participants)	68.8 %
Meeting B (4 participants)	73.4 %
Meeting C (4 participants)	79.5 %
Meeting D (4 participants)	69.8 %
Avg.	72.9 %

Percentage of correctly assigned focus targets
based on computing $P(\text{Focus} | \text{Head pan})$

Facial Expression Recognition[†]

- What is Facial Expression?
 - *Facial Expressions are the facial changes in response to a person's internal emotion states, intentions, or social communications*

Role of Facial Expressions

- Almost the most powerful, natural, and immediate way (for human beings) to communicate emotions and intentions
- Face can express emotion sooner than people verbalize or realize feelings
- Faces and facial expressions are an important aspect in interpersonal communication and man-machine interfaces

Facial Expressions[†]

- Facial expression(s):
 - nonverbal communication
 - voluntary / involuntary
 - results from one or more motions or positions of the muscles of the face
 - closely associated with our emotions
- The fact:
 - Most people's success rate at reading emotions from facial expression is only a little over 50 percent.

Facial Expression Analysis vs. Emotion Analysis[†]

- Emotion analysis requires higher level knowledge, such as context information.
- Since, besides emotions, facial expressions can also express intention, cognitive processes, physical effort, etc.

Emotions conveyed by Facial Expressions†



Happiness



Surprise



Sadness



Fear



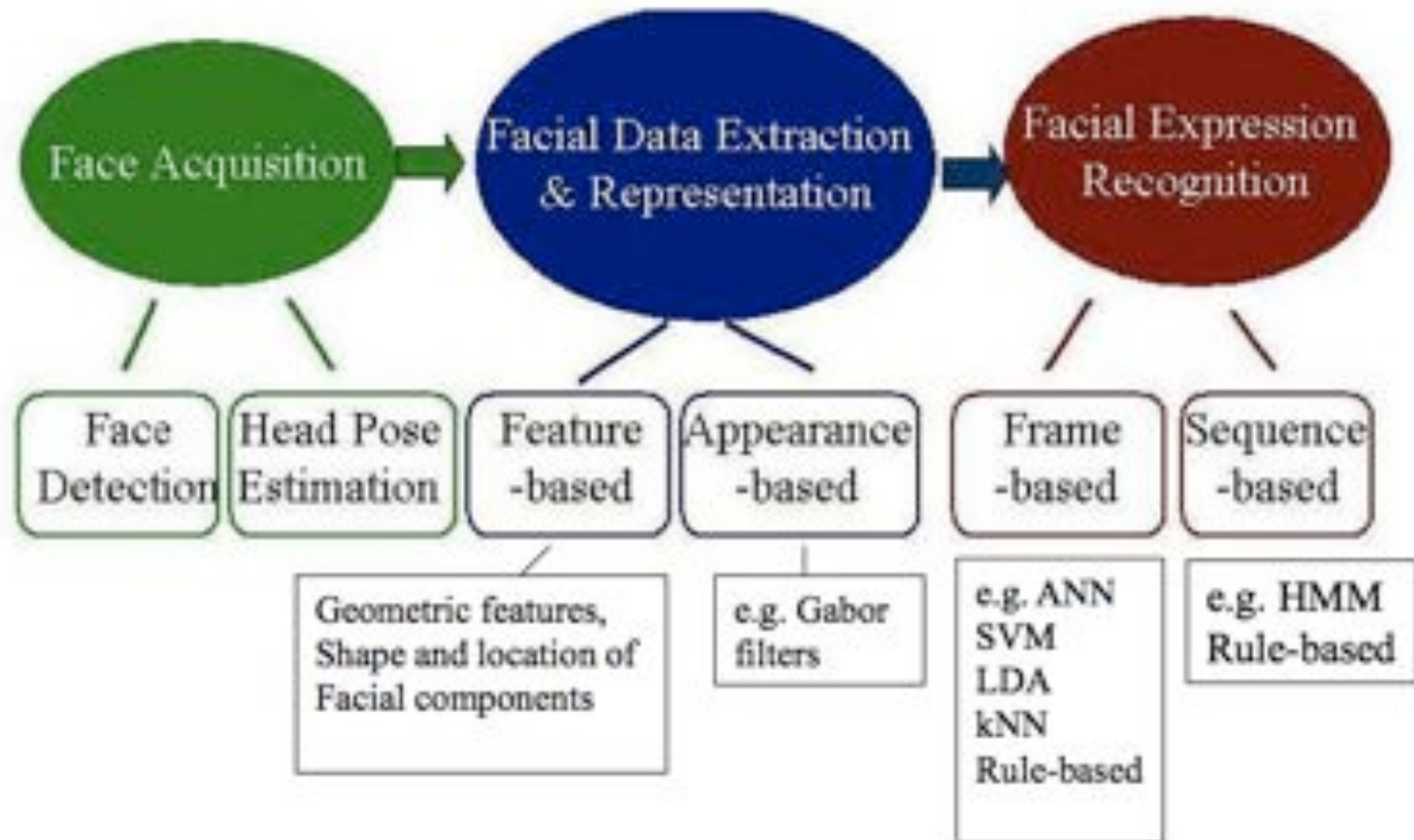
Disgust



Anger

- Six basic emotions (according to Ekman)
 - assumed to be innate

Basic Structure of Facial Expression Analysis Systems†



Facial Action Coding System (FACS)[†]

- Developed by Ekman & Friesen (1978)
- A human-observer based system designed to detect subtle changes in facial features.
- Viewing videotaped facial behavior in slow motion, trained observer can manually FACS code all possible facial displays
- These facial displays are referred to as action units (AU) and may occur individually or in combinations.













Interpretation of Facial Actions†

- Facial expressions can be linked with psychological interpretations, e.g. emotions
 - e.g Facial Action Coding System Affect Interpretation Dictionary (FACSAID) project (Ekman & Friesen)
- Facial expressions are also assumed to reveal something about sincerity or lying
 - Can be used to reveal deception (Ekman)
 - E.g. through analysis of microexpressions (very quick involuntary expression that last less than a quartersecond)
 - Detection of voluntary vs. involuntary smiles
 - insincere and voluntary smile: contraction of zygomatic major alone
 - sincere and involuntary (Duchenne) smile: contraction of zygomatic major and inferior part of orbicularis oculi
- Has also been used for analysis of depression



















Action Units (AUs) †

- There are 44 AUs
- 30 AUs related to contractions of special facial muscles :
 - 12 AUs for upper face
 - 18 AUs for lower face
- Anatomic basis of the remaining 14 is unspecified. These are referred to in FACS as miscellaneous actions.
- For action units that vary in intensity, a 5-point ordinal scale is used to measure the degree of muscle contraction.

Upper Face Action Units†

Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
					
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Check Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
					
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink

Lower Face Action Units†

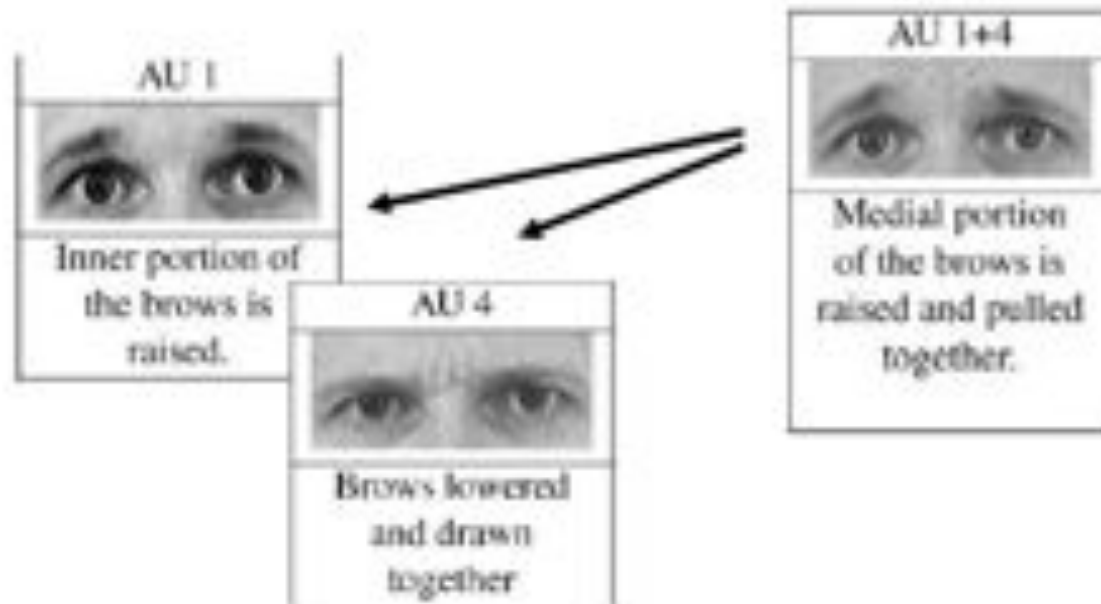
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
					
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
					
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
					
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Some Examples of Combination of FACS Action Units†



Combinations of AUs†

- More than 7000 different AU combinations have been observed.
- Additive – appearance of single AUs does not change
- Nonadditive – appearance of single AUs does change



Nonadditive AU Combination Sample†

