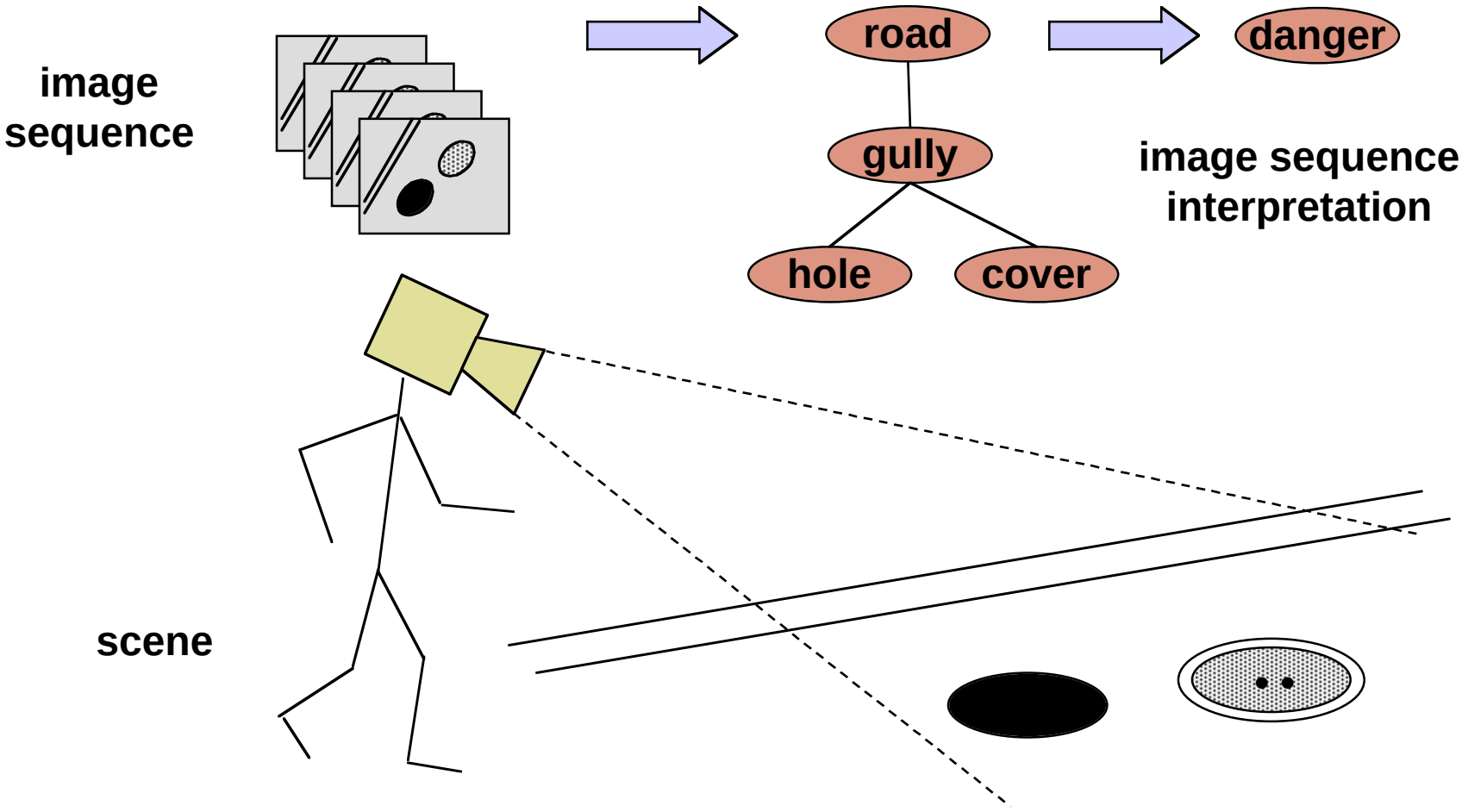
# Definition of Image Understanding

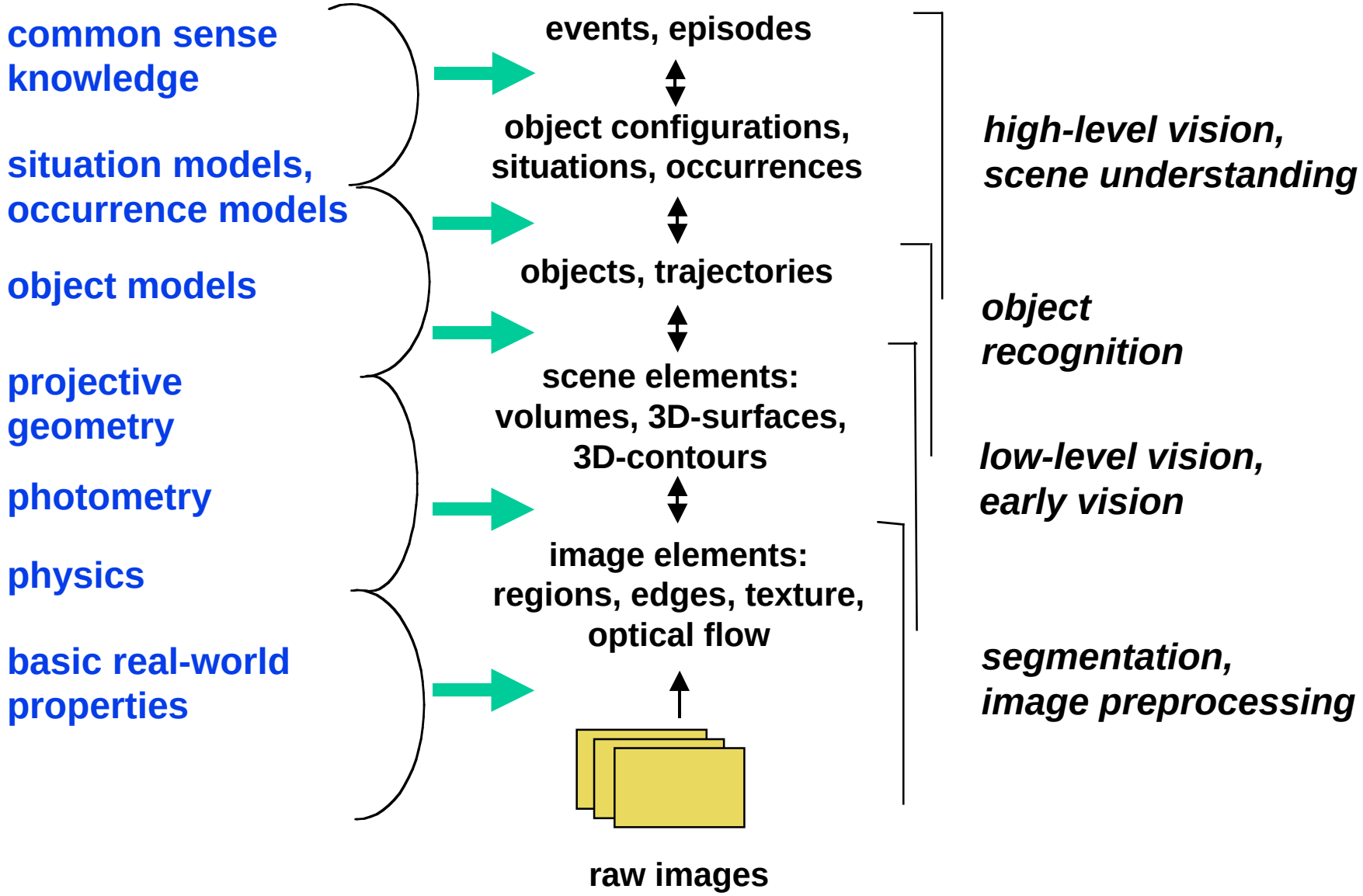
**Image understanding is the task-oriented reconstruction and interpretation of a scene by means of images**

| | |
|---|---|
| **scene:** | **section of the real world**<br>**stationary (3D) or moving (4D)** |
| **image:** | **view of a scene**<br>**projection, density image (2D)**<br>**depth image (2 1/2D)**<br>**image sequence (3D)** |
| **reconstruction and interpretation:** | **computer-internal scene description**<br>**quantitative + qualitative + symbolic** |
| **task-oriented:** | **for a purpose, to fulfill a particular task**<br>**context-dependent, supporting actions of an agent** |

# Illustration of Image Understanding



# Image Understanding as a Knowledge-based Process



# Abstraction Levels for the Description of Computer Vision Systems

**Knowledge level**

*What knowledge or information enters a process? What knowledge or information is obtained by a process?*

*What are the laws and constraints which determine the behavior of a process?*

---

**Algorithmic level**

*How is the relevant information represented?*

*What algorithms are used to process the information?*

---

**Implementation level**

*What programming language is used?*

*What computer hardware is used?*

# Example for Knowledge-level Analysis

*What knowledge or information enters a process? What knowledge or information is obtained by a process?*

*What are the laws and constraints which determine the behavior of a process?*

**Consider shape-from-shading:**

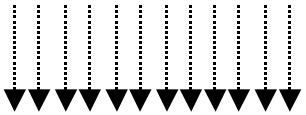
**In order to obtain the 3D shape of an object, it is necessary to**

- **state what knowledge is available (greyvalues, surface properties, illumination direction, etc.)**
- **state what information is desired (eg. qualitative vs. quantitative)**
- **exploit knowledge about the physics of image formation**

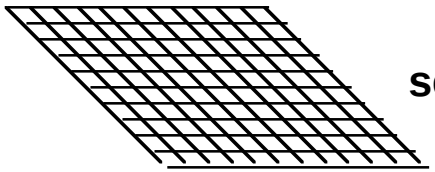
# Image Formation

**Images can be generated by various processes:**

- **illumination of surfaces, measurement of reflections** ← **"natural images"**
- **illumination of translucent material, measurement of irradiation**
- **measurement of heat (infrared) radiation**
- **X-ray of material, computation of density map**
- **measurement of any features by means of a sensory array**

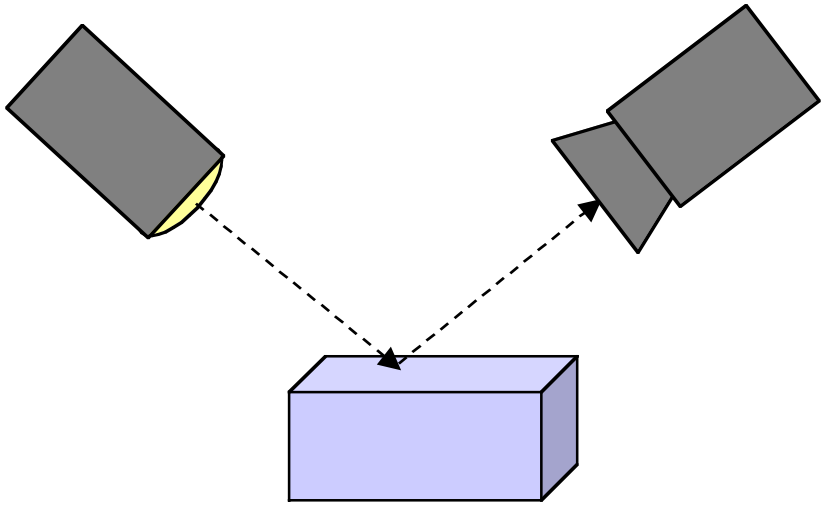


**physical signal**



**sensory array**

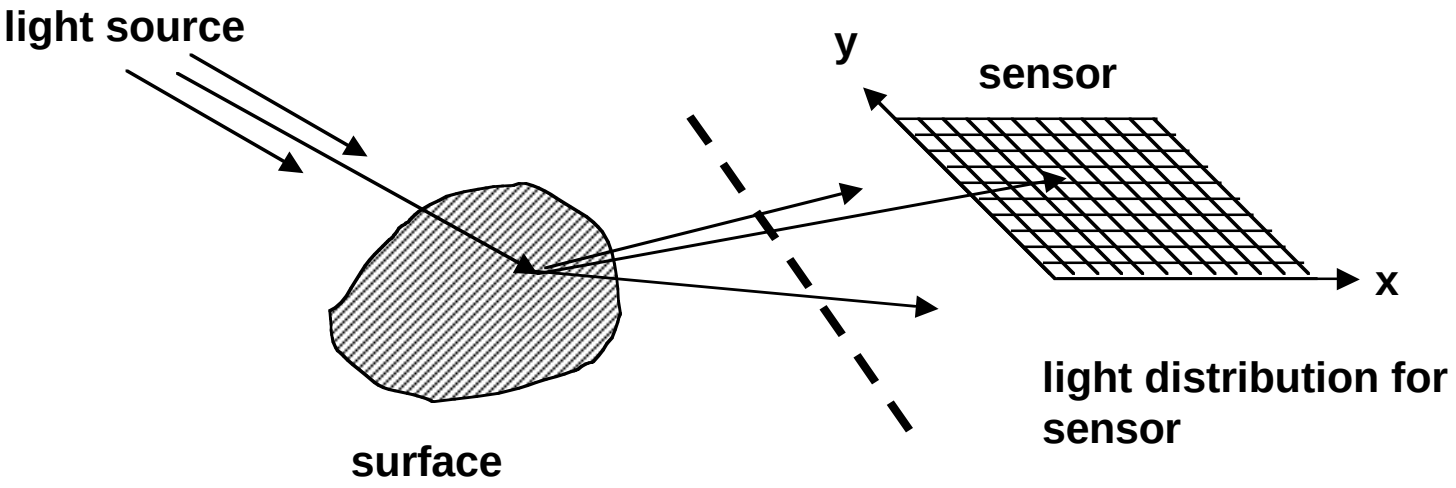
# Formation of Natural Images



**Intensity (brightness) of a pixel depends on**

1. **illumination (spectral energy, secondary illumination)**
2. **object surface properties (reflectivity)**
3. **sensor properties**
4. **geometry of light-source, object and sensor constellation (angles, distances)**
5. **transparency of irradiated medium (mistiness, dustiness)**

# Intensity of Sensor Signals




**Intensities of sensor signals depend on**
- **location x, y on sensor plane**
- **instance of time t**
- **frequency of incoming light wave λ**
- **spectral sensitivity of sensor**

$$f(x,y,t) = \int_0^\infty C(x,y,t,\lambda)S(\lambda)d\lambda$$

**sensitivity function of sensor**

**spectral energy distribution**

# Multispectral Images

**Sensors with different spectral sensitivities generate multispectral images:**

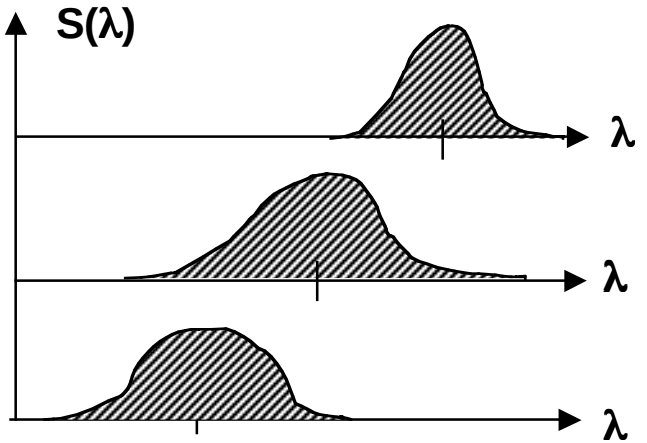
$$f_1(x,y,t) = \int_0^\infty C(x,y,t,\lambda)S_1(\lambda)d\lambda$$

$$f_2(x,y,t) = \int_0^\infty C(x,y,t,\lambda)S_2(\lambda)d\lambda$$

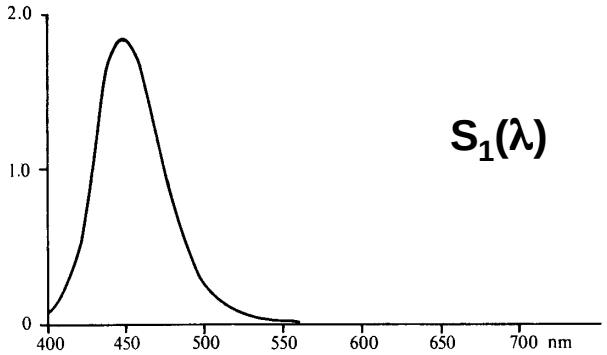
$$f_3(x,y,t) = \int_0^\infty C(x,y,t,\lambda)S_3(\lambda)d\lambda$$

**Example:**

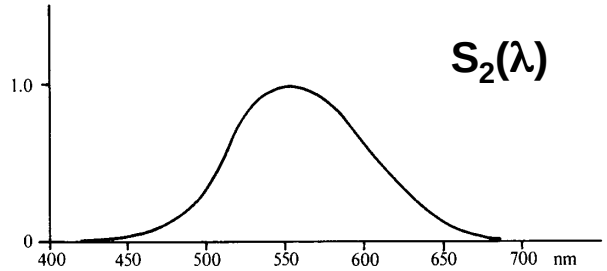
| | |
|---|---|
| **R (red)** | **650 nm center frequency** |
| **G (green)** | **530 nm center frequency** |
| **B (blue)** | **410 nm center frequency** |



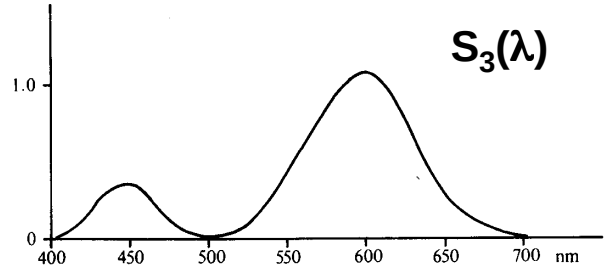
# Spectral Sensitivity of Human Eyes




$S_1(\lambda)$ "red"




$S_2(\lambda)$ "green"



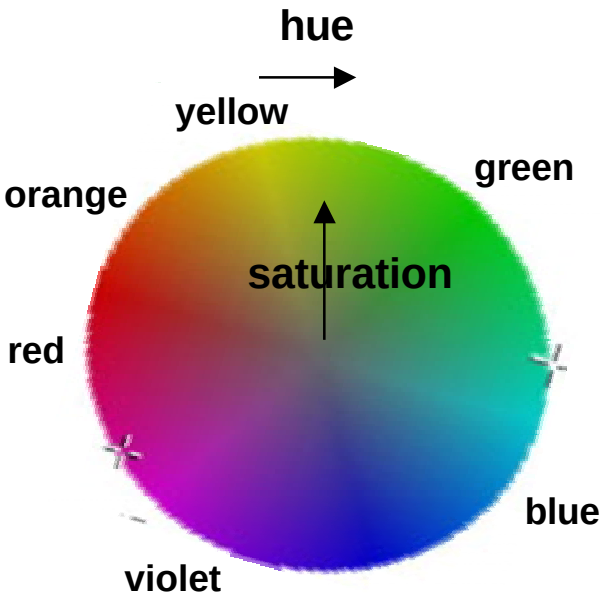

$S_3(\lambda)$ "blue"

**Note:**
**Spectral distributions which lead to identical sensor responses cannot be distinguished**

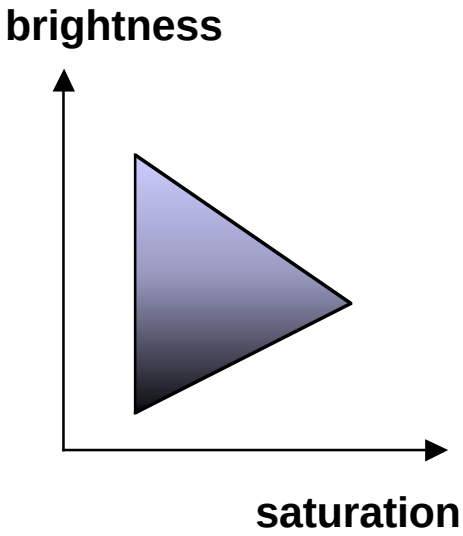
# Dimensions of Colour

**Human perception of colour distinguishes between 3 dimensions:**

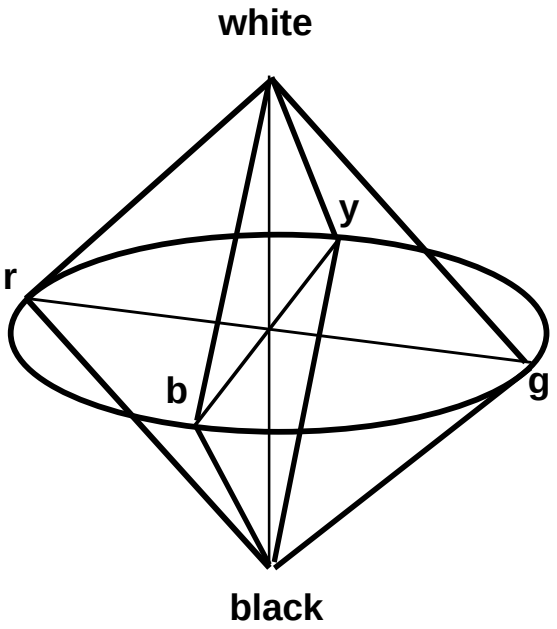
- **hue**
- **brightness**
- **saturation (chromaticness)**




**colour circle**




**NCS* colour spindle**




* Swedish Natural Colour System

# RGB Images of a Natural Scene

**Here, single colour images are rendered as greyvalue intensity images:**

**stronger spectral intensity = more brightness**

**R+G+B** **R** **G** **B**

# Discretization of Images

**Image functions must be discretized for computer processing:**

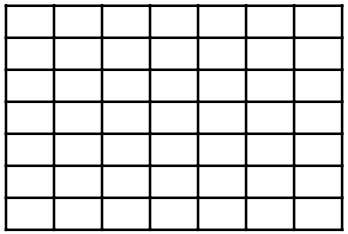
- **spatial quantization**
  **the image plane is represented by a 2D array of picture cells**
- **greyvalue quantization**
  **each greyvalue is taken from a discrete value range**
- **temporal quantization**
  **greyvalues are taken at discrete time intervals**

$$f(x,y,t) \Rightarrow \{ f_s(x_1, y_1, t_1), f_s(x_2, y_2, t_1), f_s(x_3, y_3, t_1), \ldots$$
$$f_s(x_1, y_1, t_2), f_s(x_2, y_2, t_2), f_s(x_3, y_3, t_2), \ldots$$
$$f_s(x_1, y_1, t_3), f_s(x_2, y_2, t_3), f_s(x_3, y_3, t_3), \ldots \}$$

**A single value of the discretized image function is called a pixel (picture element).**

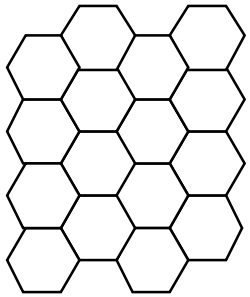
# Spatial Quantization

**Rectangular grid**



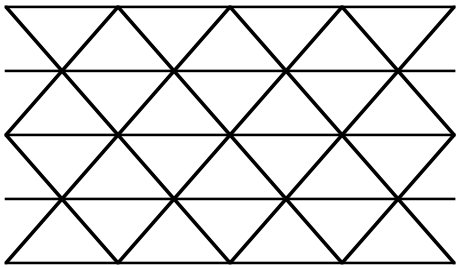
**Greyvalues represent the quantized value of the signal power falling into a grid cell.**

**Hexagonal grid**



**Note that samples of a hexagonal grid are equally spaced along rows, with successive rows shifted by half a sampling interval.**

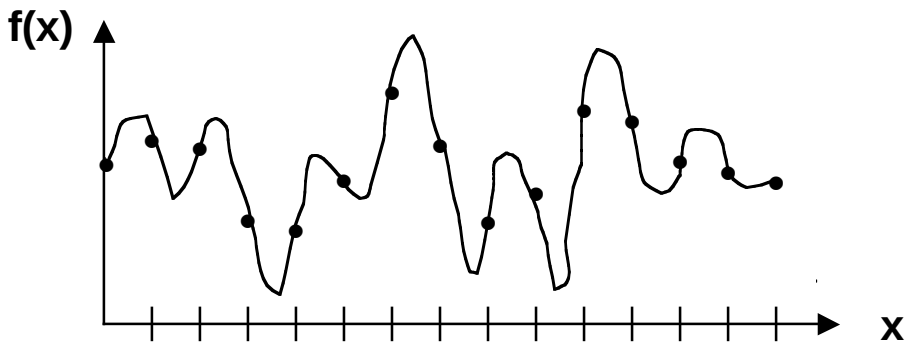
**Triangular grid**



# Reconstruction from Samples

**Under what conditions can the original (continuous) signal be reconstructed from its sampled version?**

**Consider a 1-dimensional function f(x):**




**Reconstruction is only possible, if "variability" of function is restricted.**

# Sampling Theorem

**Shannon´s Sampling Theorem:**

**A bandlimited function with bandwidth W can be exactly reconstructed from equally spaced samples, if the sampling distance is not larger than** $\frac{1}{2W}$

**bandwidth = largest frequency contained in signal**

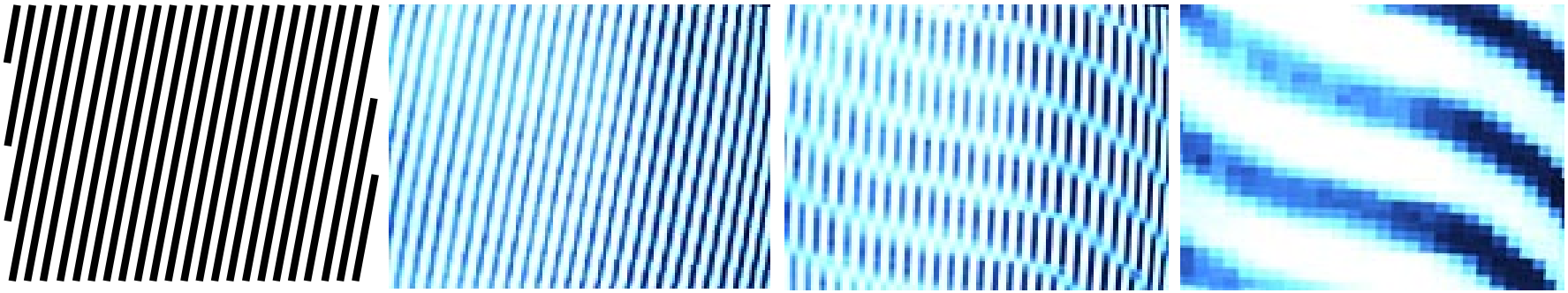
**(=> Fourier decomposition of a signal)**

**Analogous theorem holds for 2D signals with limited spatial frequencies $W_x$ and $W_y$**

# Aliasing

**Sampling an image with fewer samples than required by the sampling theorem may cause "aliasing" (artificial structures).**

**Example:**



**original** **143 x 128** **71 x 64** **35 x 32**

**To avoid aliasing, bandwidth of image must by reduced prior to sampling.**

**(=> low-pass filtering)**

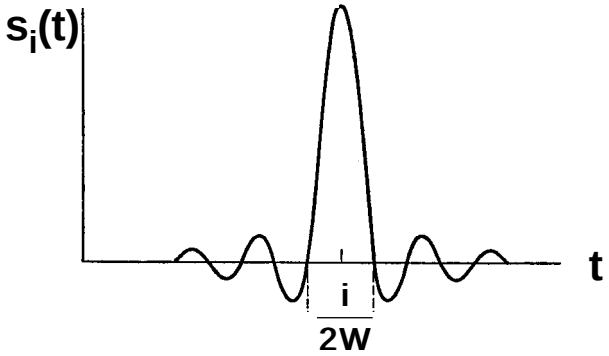
# Reconstructing the Image Function from Samples

**Formally, a continuous function f(t) with bandwidth W can be exactly reconstructed using sampling functions $s_i(t)$:**

$$s_i(t) = \sqrt{2W}\,\frac{\sin 2\pi W\left[t - i/(2W)\right]}{2\pi W\left[t - i/(2W)\right]}$$

$$x(t) = \sum_{i=-\infty}^{\infty} \sqrt{\frac{1}{2W}}\; x\left(\frac{i}{2W}\right) S_i(t)$$

**sample values**



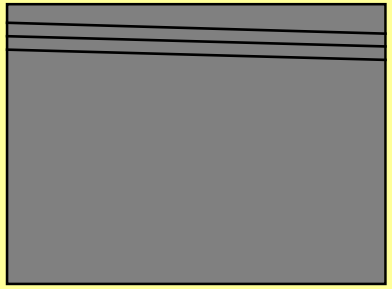

**An analogous equation holds for 2D.**

**In practice, image functions are generated from samples by interpolation.**

# Sampling TV Signals

**PAL standard:**

- **picture format 3 : 4**
- **25 full frames (50 half frames) per second**
- **interlaced rows: 1, 3, 5, ... , 2, 4, 6, ...**
- **625 rows per full frame, 576 visible**
- **64 μs per row, 52 μs visible**
- **5 MHz bandwidth**



**Only 1D sampling is required because of fixed row structure.**

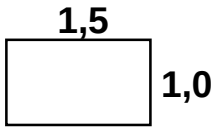
**Sampling intervals of $\Delta t = 1/(2W) = 10^{-7}$s = 100 ns give maximal possible resolution.**

**With $\Delta t$ = 100 ns, a row of duration 52 μs gives rise to 520 samples.**

**In practice, one often chooses <u>512 pixels per TV row</u>.**

**=> 576 x 512 = 294912 pixels per full frame**

**=> rectangular pixel size with width/height =$\left(\frac{4}{512}\right)$ / $\left(\frac{3}{576}\right)$ = 1,5**



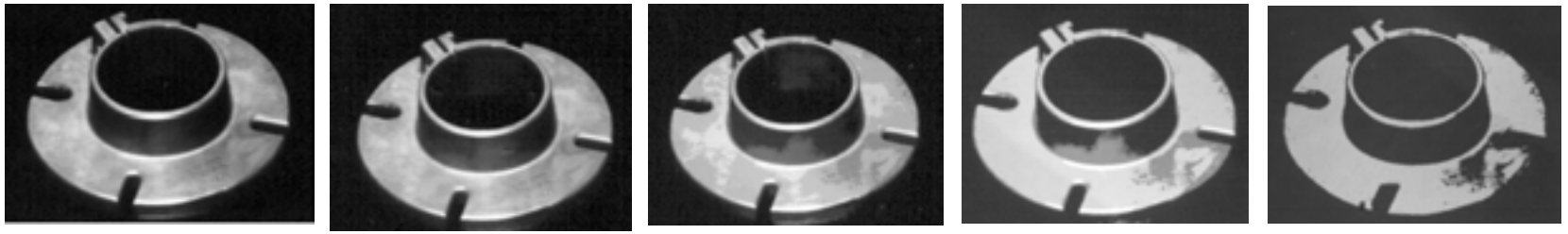
# Quantization of Greyvalues

**Quantization of greyvalues transforms continuous values of a sampled image function into digital quantities.**

**Typically 2 ... $2^{10}$ quantization levels are used, depending on task.**

**Less than $2^9$ quantization levels may cause artificial contours for human perception.**

**<u>Example</u>:**

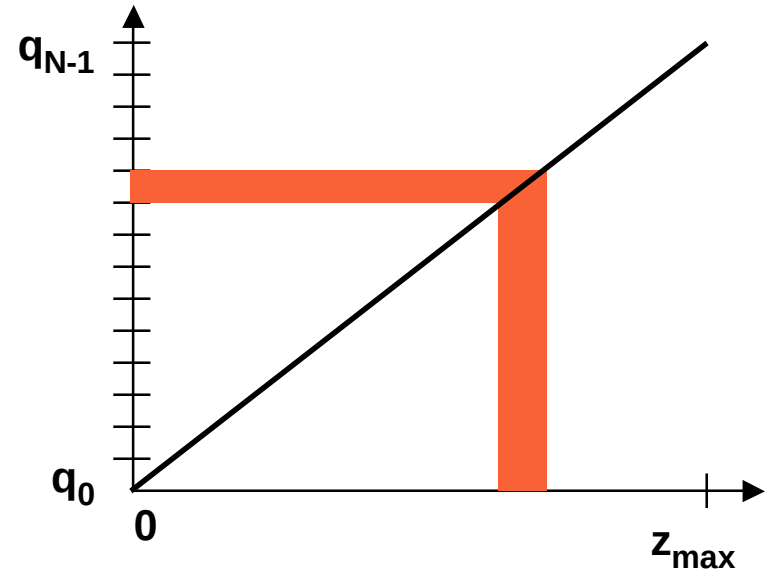


**256** **16** **8** **4** **2**

# Uniform Quantization

**Equally spaced discrete values $q_0 \ldots q_{N-1}$ represent equal-width greyvalue intervals of the continuous signal.**

**Typically $N = 2^K$ for $K = 1 \ldots 10$**

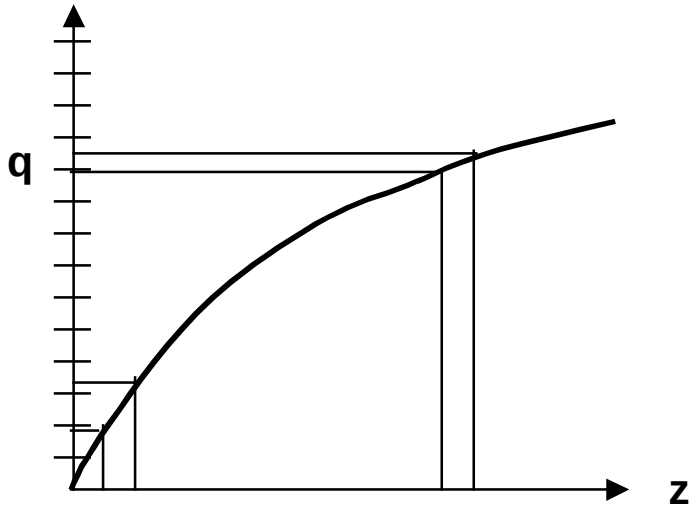



**Uniform quantization may "waste" quantization levels, if greyvalues are not equally distributed.**

# Nonlinear Quantization Curves

**E.g. fine resolution for "interesting" greyvalue ranges, coarse resolution for less interesting greyvalue ranges.**

**Example:**
**Low greyvalues are mapped into more quantization levels than high greyvalues.**




**Note:**

**Subjective brightness in human perception depends (roughly) logarithmically on the actual (measurable) brightness.**

**To let the computer see brightness as humans, use a logarithmic quantization curve.**

# Optimal Quantization (1)

**Assumption:**

**Probability density p(z) for continuous greyvalues and number of quantization levels N are known.**

**Goal:**

**Minimize mean quadratic quantization error $d_q$ by choosing optimal interval boundaries $z_n$ and optimal discrete representatives $q_n$.**

$$d_q^2 = \sum_{n=0}^{N-1} \int_{z_n}^{z_{n+1}} (z - q_n)^2 p(z) dz$$

**Minimizing by setting the derivatives zero:**

$$\frac{\delta}{\delta z_n} d_q^2 = (z_n - q_{n-1})^2 p(z_n) - (z_n - q_n)^2 p(z_n) = 0 \quad \text{for } n = 1 \ldots N-1$$

$$\frac{\delta}{\delta q_n} d_q^2 = -2 \int_{z_n}^{z_{n+1}} (z - q_n) p(z) dz = 0 \quad \text{for } n = 0 \ldots N-1$$

# Optimal Quantization (2)

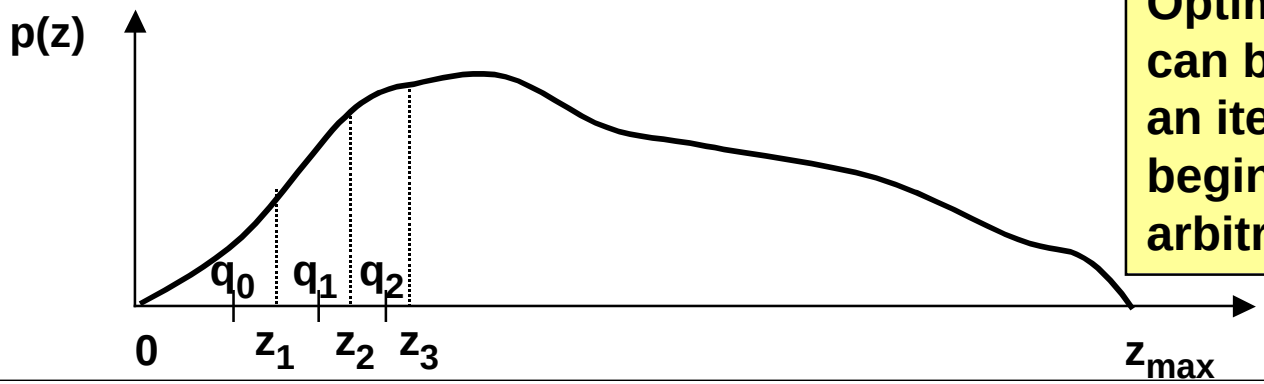
**Solution for optimal quantization:**

$$z_n = \frac{1}{2}(q_{n-1} + q_n) \quad \text{for } n = 1 \ldots N-1 \text{ when } p(z_n) > 0$$

**Each interval boundary must be in the middle of the corresponding quantization levels.**

$$q_n = \frac{\int_{z_n}^{z_{n+1}} z p(z) dz}{\int_{z_n}^{z_{n+1}} p(z) dz} \quad \text{for } n = 0 \ \ldots \ N-1$$

**Each quantization level is the center-of-gravity coordinate of the corresponding probability density area.**




**Optimal quantization can be determined by an iterative algorithm beginning with an arbitrary choice of $z_1$**

# Binarization

**For many applications it is convenient to distinguish only between 2 greyvalues, "black" and "white", or "1" and "0".**

**<u>Example</u>: Separate object from background**

**Binarization = transforming an image function into a binary image**

**<u>Thresholding</u>:**

$$g(x, y) \Rightarrow \begin{cases} 0 & \text{if } g(x, y) < T \\ 1 & \text{if } g(x, y) \geq T \end{cases}$$

**T is threshold**

**Thresholding is often applied to digital images in order to isolate parts of the image, e.g. edge areas.**

# Threshold Selection by Trial and Error

**A threshold which perfectly isolates an image component must not always exist.**

**<u>Selection by trial and error</u>:**

**Select threshold until some image property is fulfilled, e.g.**

$$q = \frac{\#\ \text{white pixels}}{\#\ \text{black pixels}} \Rightarrow q_0$$

**line strength $\Rightarrow$ $d_0$**

**number of connected components $\Rightarrow$ $n_0$**

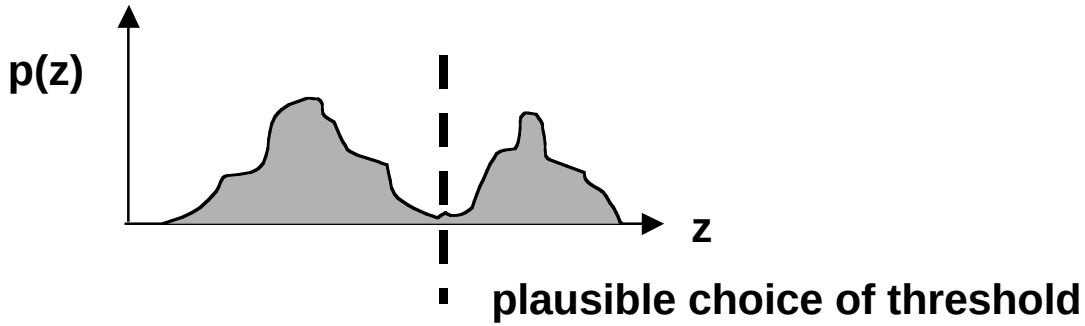
**Number of trials may be small if logarithmic search can be used.**

**<u>Example</u>:**

**At most 8 trials are needed to select a threshold $0 \leq T \leq 255$ which best approximates a given $q_0$.**

# Distribution-based Threshold Selection

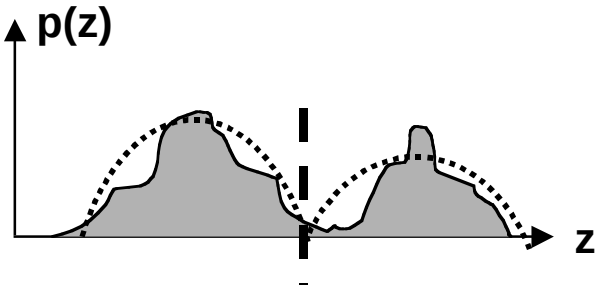
**The greyvalue distribution of the image function may show a bimodality:**



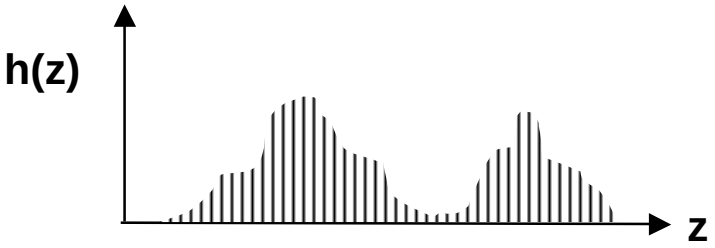

**Two methods for finding a plausible threshold:**

**1. Find "valley" between two "hills"**

**2. Fit hill templates and compute intersection**



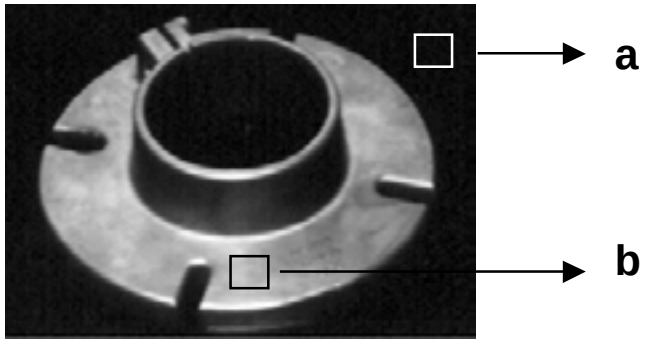

**Typically, computations are based on histograms which provide a discrete approximation of a distribution.**



# Threshold Selection Based on Reference Positions

**In many applicatons, the approximate position of image components is known a priori. These positions may provide useful reference greyvalues.**

**Example:**




**possible choice of threshold:**

$$T = \frac{a + b}{2}$$

**Threshold selection and binarization may be decisively facilitated by a good choice of illumination and image capturing techniques.**

# Image Capturing for Thresholding

**If the image capturing process can be controlled, thresholding can be facilitated by a suitable choice of**

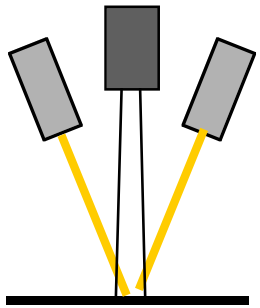
- **illumination**
- **camera position**
- **object placement**
- **background greyvalue or colour**
- **preprocessing**

**Example: Backlighting**

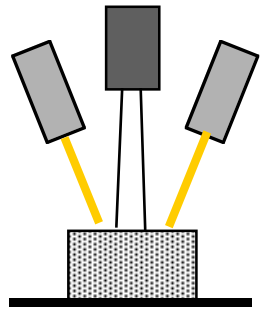
**Illumination from the rear gives bright background and shadowed object**

**Example: Slit illumination**

**On a conveyor belt illuminated by a light slit at an angle, elevations give rise to displacements which can be recognized by a camera.**



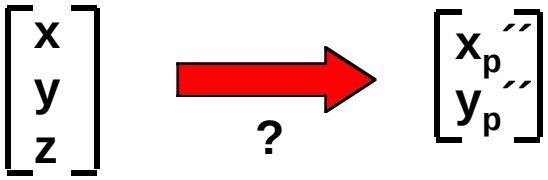
**empty**



**object present**

# Perspective Projection Transformation

**Where does a point of a scene appear in an image?**



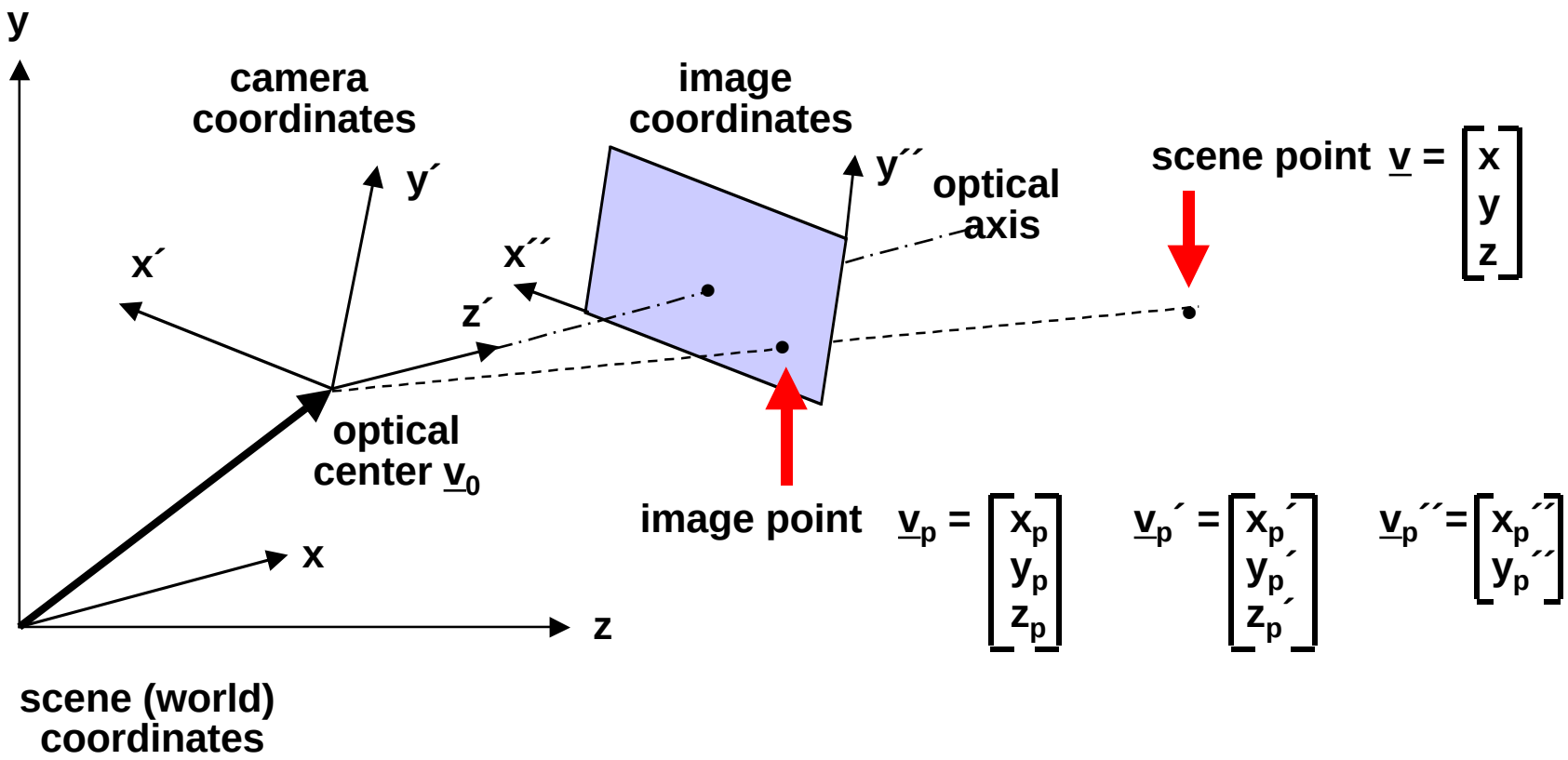

**Transformation in 3 steps:**

1. **scene coordinates => camera coordinates**
2. **projection of camera coordinates into image plane**
3. **camera coordinates => image coordinates**

**Perspective projection equations are essential for Computer Graphics. For Image Understanding we will need the inverse: What are possible scene coordinates of a point visible in the image? This will follow later.**

# Perspective Projection in Independent Coordinate Systems

**It is often useful to describe real-world points, camera geometry and image points in separate coordinate systems. The formal description of projection involves transformations between these coordinate systems.**



# 3D Coordinate Transformation (1)

**The new coordinate system is specified by a translation and rotation with respect to the old coordinate system:**

$\underline{v}' = R\,(\underline{v} - \underline{v}_0)$

$\underline{v}_0$ is displacement vector
$R$ is rotation matrix

**R may be decomposed into 3 rotations about the coordinate axes:**

$R = R_x\, R_y\, R_z$

$\alpha$ = rotation angle about x-axis
$\beta$ = rotation angle about y-axis
$\gamma$ = rotation angle about z-axis

("nick angle", "pan angle", and "tilt angle" for the camera coordinate assignment shown before)

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 3D Coordinate Transformation (2)

**By multiplying the 3 matrices $R_x$, $R_y$ and $R_z$, one gets**

$$R = \begin{bmatrix} \cos\beta\cos\gamma & \cos\beta\sin\gamma & \sin\beta \\ -\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & -\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & -\cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{bmatrix}$$

**For formula manipulations, one tries to avoid the trigonometric functions and takes**

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

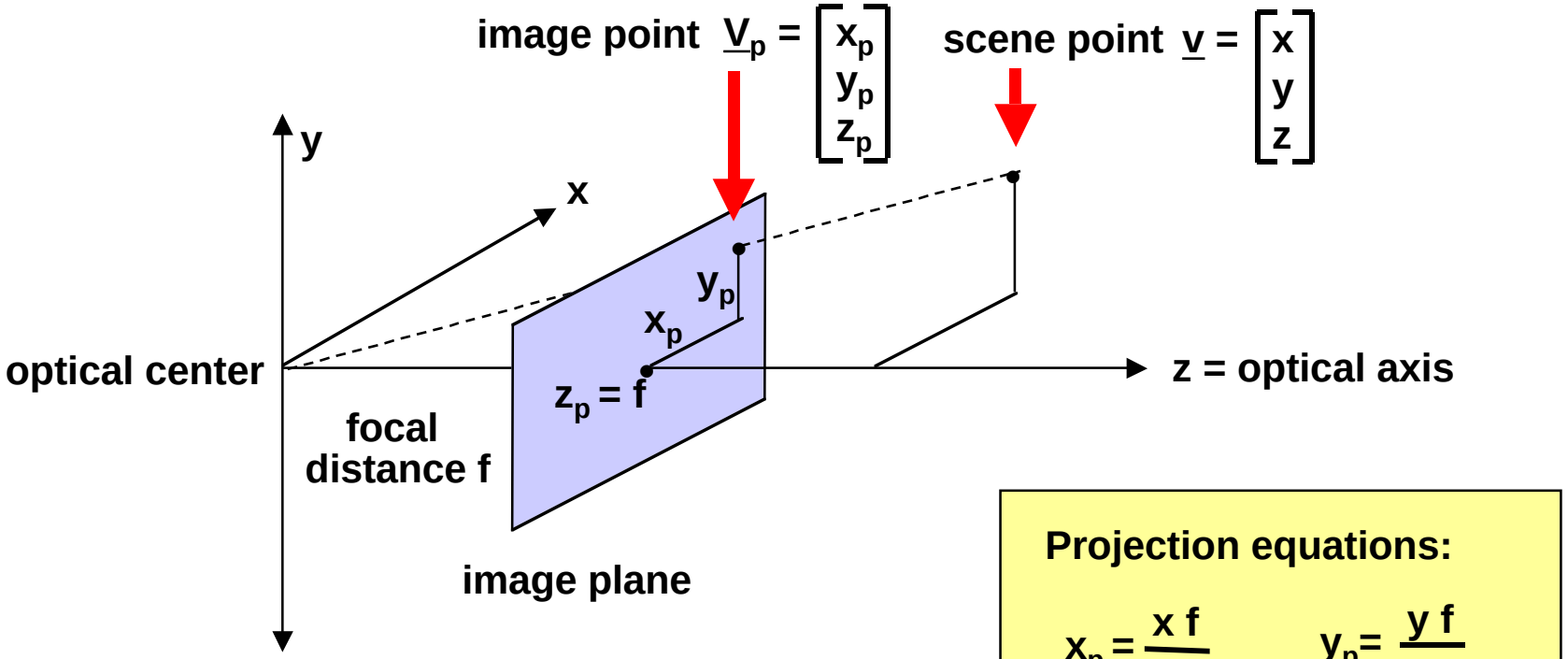
**Note that the coefficients of R are constrained:**
**A rotation matrix is <u>orthonormal</u>:**

$$R\,R^T = I \quad \text{(unit matrix)}$$

# Perspective Projection Geometry

**Projective geometry relates the coordinates of a point in a scene to the coordinates of its projection onto an image plane.**

**Perspective projection is an adequate model for most cameras.**




**Projection equations:**

$$x_p = \frac{x\,f}{z} \qquad y_p = \frac{y\,f}{z}$$

# Perspective and Orthographic Projection

**Within the camera coordinate system the perspective projection of a scene point onto the image plane is described by**

$$x_p' = \frac{x'f}{z'} \qquad y_p' = \frac{y'f}{z'} \qquad z_p' = f \qquad \text{(f = focal distance)}$$

- **nonlinear transformation**
- **loss of information**

**If all objects are far away (large z´), f/z´ is approximately constant**

**=> orthographic projection**

$$x_p' = s\,x' \qquad y_p' = s\,y' \qquad \text{(s = scaling factor)}$$

**Orthographic projection can be viewed as projection with parallel rays + scaling**

# From Camera Coordinates to Image Coordinates

**Transform may be necessary because**

- **optical axis may not penetrate image plane at origin of desired coordinate system**
- **transition to discrete coordinates may require scaling**

$x_p'' = (x_p' - x_{p0}') \, a$ **a, b scaling parameters**

$y_p'' = (y_p' - y_{p0}') \, b$ $x_{p0}'$, $y_{p0}'$ **origin of image coordinate system**

**Example:**

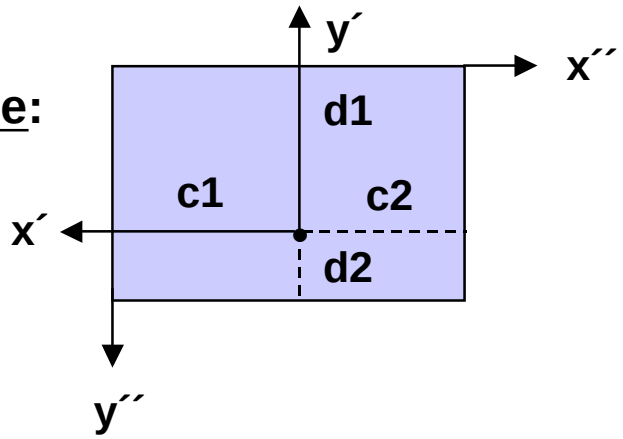



**Image boundaries in camera coordinates:**

$x'_{max} = c1 \quad x'_{min} = c2$

$y'_{max} = d1 \quad y'_{min} = d2$

**Discrete image coordinates:**

$x'' = 0 \,..\, 511 \quad y'' = 0 \,..\, 575$

**Transformation parameters:**

$x_{p0}' = c1 \quad y_{p0}' = d1 \quad a = 512 / (c2 - c1) \quad b = 576 / (d2 - d1)$

# Complete Perspective Projection Equation

**We combine the 3 transformation steps:**

1. **scene coordinates => camera coordinates**
2. **projection of camera coordinates into image plane**
3. **camera coordinates => image coordinates**

$$x_p'' = \{ f/z' [\cos\beta \cos\gamma \ (x - x_0) + \cos\beta \sin\gamma \ (y - y_0) + \sin\beta \ (z - z_0)] - x_{p0} \} \ a$$

$$y_p'' = \{ f/z' [ (- \sin\alpha \sin\beta \cos\gamma - \cos\alpha \sin\gamma ) \ (x - x_0) + (- \sin\alpha \sin\beta \sin\gamma + \cos\alpha \cos\gamma ) (y - y_0) + \sin\alpha \cos\beta \ (z - z_0)] - y_{p0} \} \ b$$

$$\text{with } z' = (- \cos\alpha \sin\beta \cos\gamma + \sin\alpha \sin\gamma ) (x - x_0) + ( - \cos\alpha \sin\beta \sin\gamma - \sin\alpha \cos\gamma ) (y - y_0) + \cos\alpha \cos\beta \ (z - z_0)$$

# Homogeneous Coordinates (1)

**4D notation for 3D coordinates which allows to express nonlinear 3D transformations as linear 4D transformations.**

**Normal:** $\underline{v}' = R\,(\underline{v} - \underline{v}_0)$

**Homogeneous coordinates:** $\underline{\boldsymbol{v}}' = \boldsymbol{A}\,\underline{\boldsymbol{v}}$

***(note italics for homogeneous coordinates)***

$$\boldsymbol{A} = \boldsymbol{R}\,\boldsymbol{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Transition to homogeneous coordinates:**

$\underline{v}^T = [x\ y\ z] \Rightarrow \underline{\boldsymbol{v}}^T = [wx\ wy\ wz\ w]$ $\quad w \neq 0$ **is arbitrary constant**

**Return to normal coordinates:**

1. **Divide components 1- 3 by 4th component**
2. **Omit 4th component**

# Homogeneous Coordinates (2)

**Perspective projection in homogeneous coordinates:**

$$\underline{v}_p' = P\,\underline{v}' \quad \text{with } P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \quad \text{and } \underline{v}' = \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix} \quad \text{gives } \underline{v}_p' = \begin{bmatrix} wx \\ wy \\ wz \\ wz/f \end{bmatrix}$$

Returning to normal coordinates gives $\underline{v}_p' = \begin{bmatrix} xf/z \\ yf/z \\ f \end{bmatrix}$

**compare with earlier slide**

**Transformation from camera into image coordinates:**

$$\underline{v}_p'' = B\,\underline{v}_p' \quad \text{with } B = \begin{bmatrix} a & 0 & 0 & -x_0 a \\ 0 & b & 0 & -y_0 b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and } \underline{v}_p' = \begin{bmatrix} wx_p \\ wy_p \\ 0 \\ w \end{bmatrix} \quad \text{gives } \underline{v}_p'' = \begin{bmatrix} wa(x_p - x_0) \\ wb(y_p - y_0) \\ 0 \\ w \end{bmatrix}$$

# Homogeneous Coordinates (3)

**Perspective projection can be completely described in terms of a linear transformation in homogeneous coordinates:**

$$\underline{v}_p'' = B\ P\ R\ T\ \underline{v}$$

***B P R T* may be combined into a single 4 x 4 matrix *C* :**

$$\underline{v}_p'' = C\ \underline{v}$$

**In the literature the parameters of these equations may vary because of different choices of coordinate systems, different order of translation and rotation, different camera models, etc.**

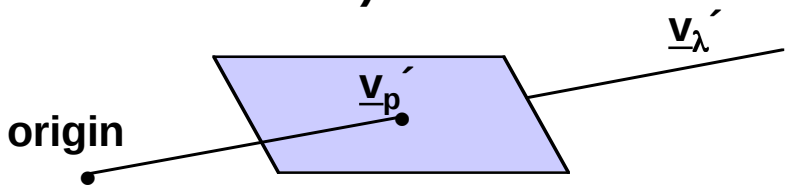
# Inverse Perspective Equations

**Which points in a scene correspond to a point in the image?**

$$\begin{bmatrix} x_p'' \\ y_p'' \end{bmatrix} \xrightarrow{?} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

**Each image point defines a projection ray as the locus of possible scene points (for simplicity in camera coordinates):**

$\underline{v}_p' \Rightarrow \underline{v}_\lambda' = \lambda \underline{v}_p'$



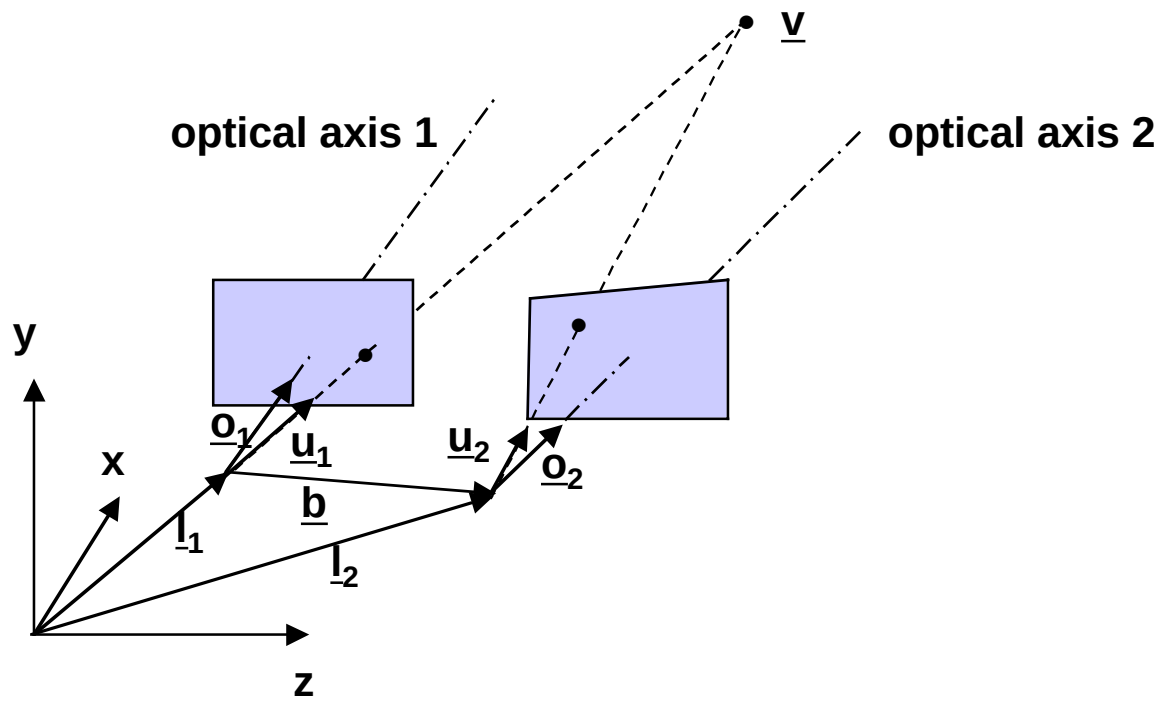

$$\underline{v} = \underline{v}_0 + R^T \lambda \underline{v}_p'$$

**3 equations with the 4 unknowns x, y, z, λ and camera parameters R and $\underline{v}_0$**

**Applications of inverse perspective mapping for e.g.**

- **distance measurements**
- **binocular stereo**
- **camera calibration**
- **motion stereo**

# Binocular Stereo (1)




| | |
|---|---|
| $\underline{l}_1$, $\underline{l}_2$ | camera positions (optical center) |
| $\underline{b}$ | stereo base |
| $\underline{o}_1$, $\underline{o}_2$ | camera orientations (unit vectors) |
| $f_1$, $f_2$ | focal distances |
| $\underline{v}$ | scene point |
| $\underline{u}_1$, $\underline{u}_2$ | projection rays of scene point (unit vectors) |

# Binocular Stereo (2)

Determine distance to $\underline{v}$ by measuring $\underline{u}_1$ and $\underline{u}_2$

Formally: $\alpha\, \underline{u}_1 = \underline{b} + \beta\, \underline{u}_2 \quad \Rightarrow \quad \underline{v} = \alpha\, \underline{u}_1 + \underline{l}_1$

$\alpha$ and $\beta$ are overconstrained by the vector equation. In practice, measurements are inexact, no exact solution exists (rays do not intersect).

Better approach: Solve for the point of closest approximation of both rays:

$$\underline{v} = \frac{\alpha_0\, \underline{u}_1 + (\underline{b} + \beta_0\, \underline{u}_2)}{2} + \underline{l}_1 \quad \Rightarrow \quad \text{minimize} \quad \| \alpha\, \underline{u}_1 - (\underline{b} + \beta\, \underline{u}_2) \|^2$$

Solution:

$$\alpha_0 = \frac{\underline{u}_1^T \underline{b} - (\underline{u}_1^T \underline{u}_2)\, (\underline{u}_2^T \underline{b})}{1 - (\underline{u}_1^T \underline{u}_2)^2}$$

$$\beta_0 = \frac{(\underline{u}_1^T \underline{u}_2)\, (\underline{u}_1^T \underline{b}) - (\underline{u}_2^T \underline{b})}{1 - (\underline{u}_1^T \underline{u}_2)^2}$$

# Distance in Digital Images

**Intuitive concepts of continuous images do not always carry over to digital images.**

**Several methods for measuring distance between pixels:**

**Eucledian distance**

$$D_E((i,j),(h,k)) = \sqrt{(i-h)^2 + (j-k)^2}$$

**costly computation of square root, can be avoided for distance comparisons**

**City block distance**

$$D_4((i,j)(h,k)) = |i-h| + |j-k|$$

**number of horizontal and vertical steps in a rectangular grid**

**Chessboard distance**

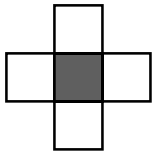
$$D_8((i,j)(h,k)) = \max\{|i-h|, |j-k|\}$$

**number of steps in a rectangular grid if diagonal steps are allowed (number of moves of a king on a chessboard)**

# Connectivity in Digital Images

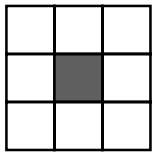
**Connectivity is an important property of subsets of pixels. It is based on adjacency (or neighbourhood):**

**Pixels are 4-neighbours if their distance is $D_4 = 1$**



**all 4-neighbours of center pixel**

**Pixels are 8-neighbours if their distance is $D_8 = 1$**



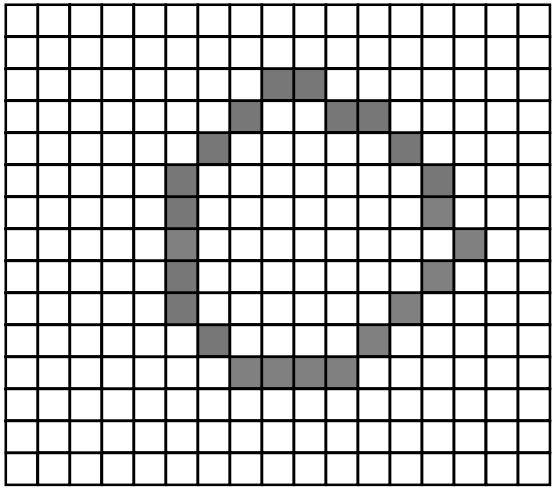
**all 8-neighbours of center pixel**

**A path from pixel P to pixel Q is a sequence of pixels beginning at Q and ending at P, where consecutive pixels are neighbours.**

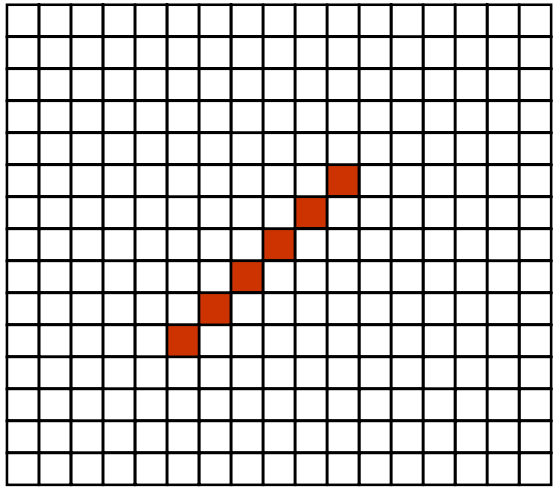
**In a set of pixels, two pixels P and Q are connected, if there is a path between P and Q with pixels belonging to the set.**

**A region is a set of pixels where each pair of pixels is connected.**

# Closed Curve Paradoxon



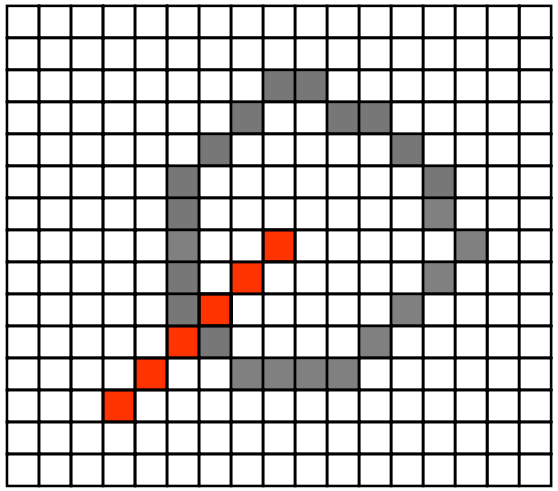
line 1



line 2

solid lines if 8-neighbourhood is used

a similar paradoxon arises if 4-neighbourhoods are used



line 2 does not intersect line 1 although it crosses from the outside to the inside

# Geometric Transformations

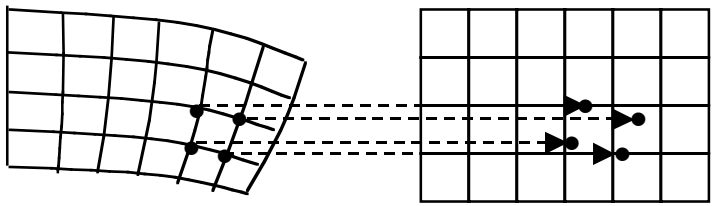
**Various applications:**

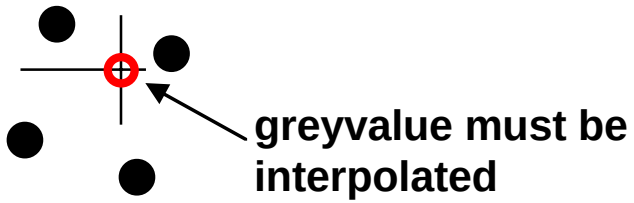
- **change of view point**
- **elimination of geometric distortions from image capturing**
- **registration of corresponding images**
- **artificial distortions, Computer Graphics applications**

**Step 1: Determine mapping $\underline{T}(x, y)$ from old to new coordinate system**

**Step 2: Compute new coordinates $(x', y')$ for $(x, y)$**

**Step 3: Interpolate greyvalues at grid positions from greyvalues at transformed positions**





# Polynomial Coordinate Transformations

**General format of transformation:**

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

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

- Assume polynomial mapping between (x, y) and (x´, y´) of degree m
- Determine corresponding points
- a) Solve linear equations for $a_{rk}$, $b_{rk}$ (r, k = 1 ... m)
  b) Minimize mean square error (MSE) for point correspondences

**Approximation by biquadratic transformation:**

$$x' = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2$$
$$y' = b_{00} + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2$$

at least 6 corresponding pairs needed

**Approximation by affine transformation:**

$$x' = a_{00} + a_{10}x + a_{01}y$$
$$y' = b_{00} + b_{10}x + b_{01}y$$

at least 3 corresponding pairs needed

# Translation, Rotation, Scaling, Skewing

**Translation** by vector $\underline{t}$:

$$\underline{v}' = \underline{v} + \underline{t} \qquad \text{with} \quad \underline{v}' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad \underline{v} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \underline{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

**Rotation** of image coordinates by angle $\alpha$:

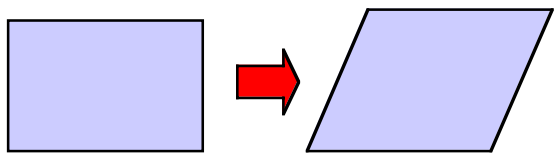
$$\underline{v}' = R\,\underline{v} \qquad \text{with} \quad R = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

**Scaling** by factor a in x-direction and factor b in y-direction:

$$\underline{v}' = S\,\underline{v} \qquad \text{with} \quad S = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

**Skewing** by angle $\beta$:

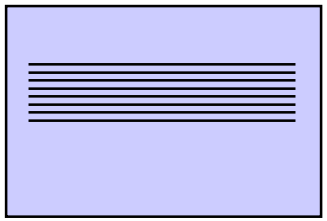
$$\underline{v}' = W\,\underline{v} \qquad \text{with} \quad W = \begin{bmatrix} 1 & \tan\beta \\ 0 & 1 \end{bmatrix}$$



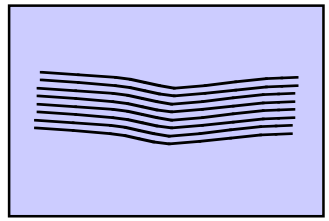
# Example of Geometry Correction by Scaling

Distortions of electron-tube cameras may be

1 - 2 % => more than 5 lines for TV images



ideal image



actual image

Correction procedure may be based on

- fiducial marks engraved into optical system
- a test image with regularly spaced marks

Ideal mark positions:

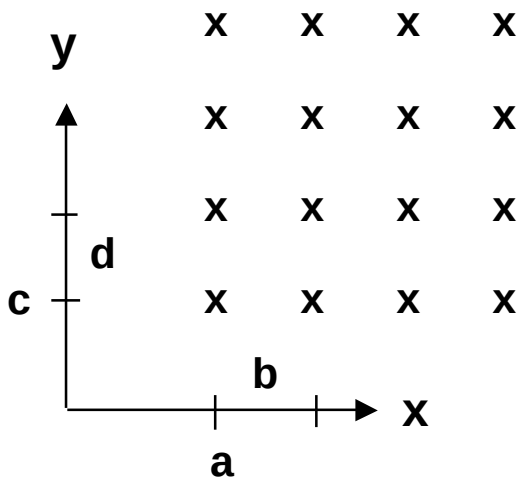
$x_{mn} = a + mb,\ y_{mn} = c + nd$

$m = 0 \ldots M-1$

$n = 0 \ldots N-1$

Actual mark positions:

$x'_{mn}, y'_{mn}$




**Determine a, b, c, d such that MSE (mean square error) of deviations is minimized**

# Minimizing the MSE

Minimize $$E = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (x_{mn} - x'_{mn})^2 + (y_{mn} - y'_{mn})^2$$

$$= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (a + mb - x'_{mn})^2 + (c + nd - y'_{mn})^2$$

From $\delta E/\delta a = \delta E/\delta b = \delta E/\delta c = \delta E/\delta d = 0$ we get:

$$a = \frac{2}{MN(M+1)} \sum_m \sum_n (2M - 1 - 3m)\, x'_{mn}$$

$$b = \frac{6}{MN(M^2-1)} \sum_m \sum_n (2m - M + 1) x'_{mn}$$

$$c = \frac{2}{MN(N+1)} \sum_m \sum_n (2N - 1 - 3n)\, y'_{mn}$$

$$d = \frac{6}{MN(N^2-1)} \sum_m \sum_n (2n - N + 1) y'_{mn}$$

**Special case M=N=2:**

$a = 1/2\ (x'_{00} + x'_{01})$

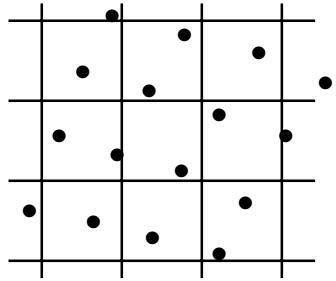
$b = 1/2\ (x'_{10} - x'_{00} + x'_{11} - x'_{01})$

$c = 1/2\ (y'_{00} + y'_{01})$

$d = 1/2\ (y'_{01} - y'_{00} + y'_{11} - y'_{10})$

# Principle of Greyvalue Interpolation

**Greyvalue interpolation = computation of unknown greyvalues at locations (u´v´) from known greyvalues at locations (x´y´)**



**Two ways of viewing interpolation in the context of geometric transformations:**

**A** **Greyvalues at grid locations (x y) in old image are placed at corresponding locations (x´y´) in new image: g(x´y´) = g(T(x y))**

**=> interpolation in new image**

**B** **Grid locations (u´v´) in new image are transformed into corresponding locations (u v) in old image: g(u v) = g($T^{-1}$(u´v´))**

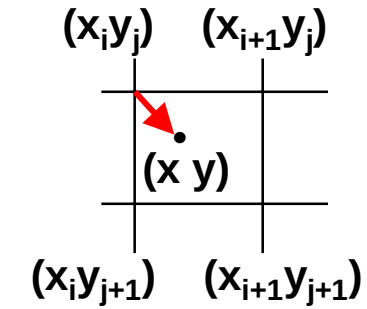
**=> interpolation in old image**

**We will take view B:**

**Compute greyvalues between grid from greyvalues at grid locations.**

# Nearest Neighbour Greyvalue Interpolation

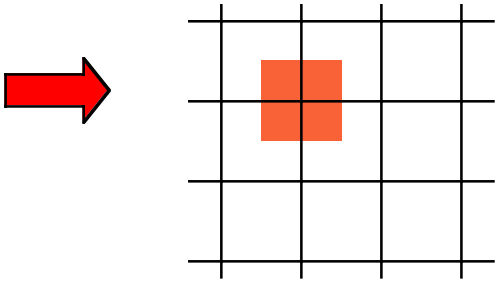
**Assign to (x y) greyvalue of nearest grid location**



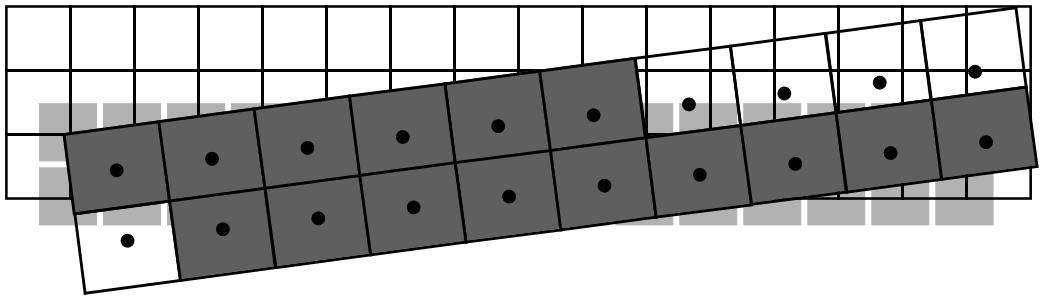

$(x_i y_j)$ $(x_{i+1} y_j)$ $(x_i y_{j+1})$ $(x_{i+1} y_{j+1})$ grid locations

(x y) location between grid with $x_i \le x \le x_{i+1}$, $y_j \le y \le y_{j+1}$

**Each grid location represents the greyvalues in a rectangle centered around this location:**



**Straight lines or edges may appear step-like after this transformation:**



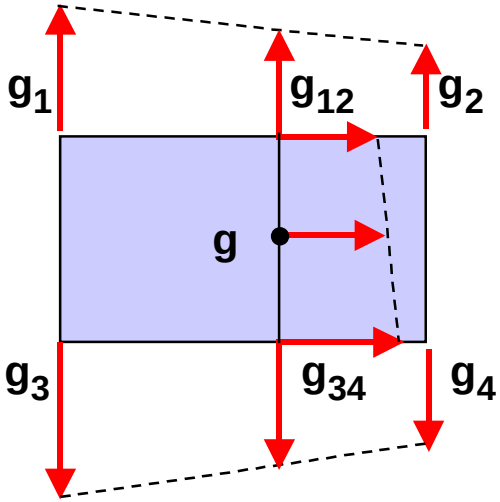
# Bilinear Greyvalue Interpolation

The greyvalue at location (x y) between 4 grid points $(x_i y_j)$ $(x_{i+1} y_j)$ $(x_i y_{j+1})$ $(x_{i+1} y_{j+1})$ is computed by linear interpolation in both directions:

$$g(x,y) = \frac{1}{(x_{i+1} - x_i)(y_{j+1} - y_i)} \Big\{ (x_{i+1} - x)(y_{j+1} - y) g(x_i y_j) + (x - x_i)(y_{j+1} - y) g(x_{i+1} y_j) + (x_{i+1} - x)(y - y_j) g(x_i y_{j+1}) + (x - x_i)(y - y_j) g(x_{i+1} y_{j+1}) \Big\}$$

Simple idea behind long formula:

1. Compute $g_{12}$ = linear interpolation of $g_1$ and $g_2$
2. Compute $g_{34}$ = linear interpolation of $g_3$ and $g_4$
3. Compute g = linear interpolation of $g_{12}$ and $g_{34}$

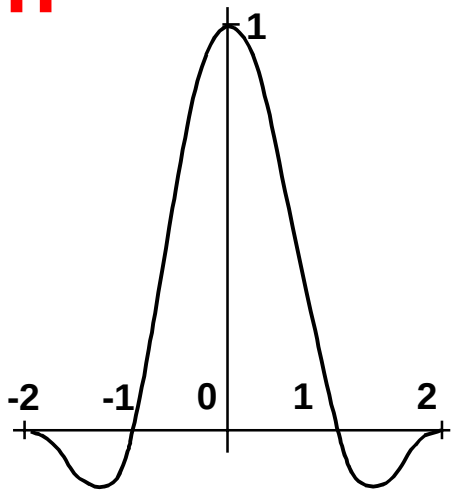



The step-like boundary effect is reduced.
But bilear interpolation may blur sharp edges.

# Bicubic Interpolation

Each greyvalue at a grid point is taken to represent the center value of a local bicubic interpolation surface with cross section $h_3$.

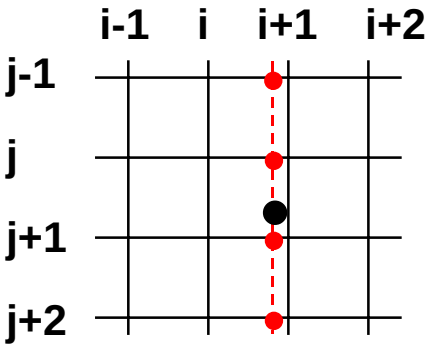
$$h_3 = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{for } 0 < |x| < 1 \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{for } 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$




cross section of interpolation kernel

The greyvalue at an arbitrary point [u, v] (black dot in figure) can be computed by

- 4 horizontal interpolations to obtain greyvalues at points [u, j-1] ... [u, j+2] (red dots), followed by
- 1 vertical interpolation (between red dots) to obtain greyvalue at [u, v].




<u>Note</u>:
For an image with constant geyvalues $g_0$ the interpolated greyvalues at all points between the grid lines are also $g_0$.