University of Applied Sciences
Department of
Design, Computer Science, Media
Prof. Dr. D. Richter

Transilvania University Brasov
Faculty of
Mathematics and Informatics
Dr. I. Plajer

ISIP 2019
International School on Image Processing
July 15th - 26th, 2019

detlef.richter@HS-RM.de

University of Applied Sciences
Department of
Design, Computer Science, Media
Prof. Dr. D. Richter

Lectures in Image Processing

Chapter 6
Interpolation

6. Interpolation

6. Interpolation, 6.1 Introduction

Problem :

Geometrical adjustment of images to different imaging conditions.
- Mutual adjustment of two images taken in different situations :
  - seasons (summer / winter),
  - positions (airplane – satellite)
  - orientations (lines of sight)
  - different technical sensors (visible wave lengths – IR / CT - NMR),
- Correction of single image (according to deficiencies of image sensors)

Note:
- Only two-dimensional problems considered.
- No perspective correction of three-dimensional objects, e.g. buildings etc.

6. Interpolation, 6.1 Introduction

Extension and reduction of image size : $F(g(x, y)) \rightarrow F'(g'(x', y'))$

with spatial transformation equation

$x' = f_1(x,y)$ and $y' = f_2(x,y)$ (no statement about grey value)

If constant aspect ratio is required : $x' = f_1(x,y) = f_1(x) = a * x$
$y' = f_2(x,y) = f_2(y) = a * y$

The image expands from upper left down to the right

6. Interpolation, 6.1 Introduction

In case of 512² - images and for expansion out of image center :

image center M = (255.5 / 255.5)

$$x' = a * x + (1-a) * 255.5 \qquad (Gl. 1a)$$
$$y' = a * y + (1-a) * 255.5$$

Other frequently used image sizes :
768 x 576
640 x 480

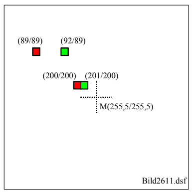
Where is the image center ?

6. Interpolation, 6.1 Introduction

Example for a = + 3 :
Consider two adjacent points :

p(200,200) → p(89, 89)
p(201,200) → p(92, 89),

The points in between p(90,89) and p(91,89) are not assigned.



Consequences of image extension

6. Interpolation, 6.1 Introduction

Resampling : Calculate positions of target pixels back into source image

$x = f_1^{-1}(x',y')$ = int ((x'-255.5)/a + 255.5 + 0.5) (1)
$y = f_2^{-1}(x',y')$ = int ((y'-255.5)/a + 255.5 + 0.5)

x'	88	89	90	91	92	93	94
xfloat	199.7	200	200.3	200.7	201	201.3	201.7
xfloat +0.5	200.2	200.5	200.8	201.2	201.5	201.8	202.2
xint (2)	200	200	200	201	201	201	202

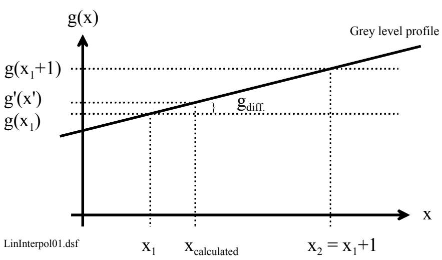
Note 1 : Correct rounding of values
Note 2 : If assigning g'(x',y') the value g(x,y), the new image shows a rough pixel structure
Note 3 : Grey values g'∈ { G* }; G* : values of source image

6. Interpolation, 6.1 Introduction, Image Expansion

Original Image

Expanded Image by Factor a = 5

6. Interpolation, 6.2 Linear Interpolation



Linear Interpolation of grey values

6. Interpolation, 6.2 Linear Interpolation

$$\frac{g(x+1) - g(x)}{1} = \frac{g_{diff.}}{x_{calculated} - x}$$

g'(x') = g(x) + $g_{diff.}$

g'(x') = g(x) + (g(x+1) - g(x))*($x_{calculated}$ - x) (Eq. 2)

g'(x') = int [(g(x) + (g(x+1) - g(x))*($x_{calculated}$ - x)) + 0.5]

For the above mentioned example follows :

g'(90) = int [(g(200) + (g(201) - g(200)) * 0.33)+0.5]

6. *Interpolation, 6.2 Linear Interpolation*

If we denote in Eq. 2

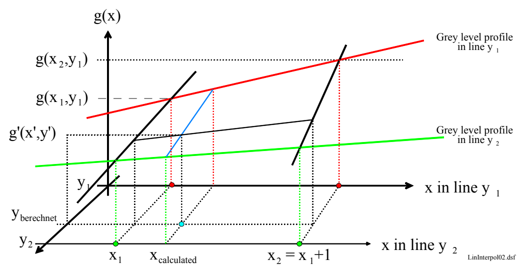
Eq.2 : $g'(x') = g(x) + (g(x+1) - g(x))*(x_{calculated} - x)$

$(x_{calculated} – x)$ by x_d, the distance of the destination pixel to the target pixel, we get :

$$g'(x') = (1 – x_d) * g(x) + x_d * g(x+1)$$

for 1-dimensional interpolation.

6. *Interpolation, 6.3 Bilinear Interpolation*



Bilinear interpolation of grey values

6. *Interpolation, 6.3 Bilinear Interpolation*

In line y :

$g'(x',y) = g(x,y) + (g(x+1,y) - g(x,y)) *(x_{calculated} - x)$

In line y+1 :

$g'(x',y+1) = g(x,y+1) + (g(x+1,y+1) - g(x,y+1)) *(x_{calculated} - x)$

$g'(x',y') = \textbf{int} [(g'(x',y)+(g'(x',y+1) - g'(x',y)) * (y_{calculated} -y))+0.5]$ (Eq.3)

6. *Interpolation, 6.3 Bilinear Interpolation*

Expanded Image without Interpolation

Expanded Image with Interpolation

6. *Interpolation, 6.3 Bilinear Interpolation*

If we denote in Eq.3
$(y_{calculated} – y)$ by y_d and
$(x_{calculated} – x)$ by x_d
the vertical resp. horizontal distance of the destination pixel to the target pixel, we get :

$$
\begin{aligned}
g'(x', y') = &(1 – x_d) * (1 - y_d) * g(x,y) \\
&+ x_d * (1 - y_d) * g(x+1,y) \\
&+ (1 – x_d) * y_d * g(x,y+1) \\
&+ x_d * y_d * g(x+1,y+1)
\end{aligned}
$$

for 2-dimensional interpolation.

6. *Interpolation, 6.4 Trilinear Interpolation*

For 3-dimensional data (volume data, voxels) e.g. DICOM data format, use tri-linear interpolation accordingly :

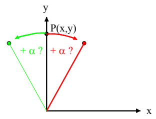
$$
\begin{aligned}
g'(x', y', z') = &(1 – x_d) * (1 - y_d) * (1 – z_d) * g(x,y,z) \\
&+ x_d * (1 - y_d) * (1 – z_d) * g(x+1,y,z) \\
&+ (1 – x_d) * y_d * (1 – z_d) * g(x,y+1,z) \\
&+ x_d * y_d * (1 – z_d) * g(x+1,y+1,z) \\
&+ (1 – x_d) * (1 - y_d) * z_d * g(x,y,z+1) \\
&+ x_d * (1 - y_d) * z_d * g(x+1,y,z+1) \\
&+ (1 – x_d) * y_d * z_d * g(x,y+1,z+1) \\
&+ x_d * y_d * z_d * g(x+1,y+1,z+1)
\end{aligned}
$$

6. Interpolation, 6.5 Rotation of Images

Rotation of a vector by rotational matrix R: $\begin{pmatrix} x' \\ y' \end{pmatrix} = R_1 * \begin{pmatrix} x \\ y \end{pmatrix}$

$R_1 = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$

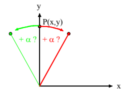
$R_2 = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$



Which matrix rotates the pixel into the red and which one into the green direction?

6. Interpolation, 6.5 Rotation of Images

Rotation of images :



Apply rotational Matrix $R = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$

R performs a rotation around the origin (0, 0).

For rotation around an arbitrary point, e.g. image center (x_m=383.5 / y_m=287.5 for 768x576 or x_m = 255.5/ y_m= 255.5) for 512² shift the co-ordinate system:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = R \cdot \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x_m \\ y_m \end{pmatrix} \right) + \begin{pmatrix} x_m \\ y_m \end{pmatrix}$$

6. Interpolation, Homogeneous Coordinates

Translation of a Vector Using Vector Addition	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$
Result of Vector Addition	x' = x + a y' = y + b
Addition of a Vector by a Matrix T using Homogeneous Coordinates	$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

6. Interpolation, Homogeneous Coordinates

Rotation of a Vector using Rotation Matrix R	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}$
Rotation of a Vector by Matrix R using Homogeneous Coordinates	$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
Rotation and Translation of a Vector using Homogeneous Coordinates	$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha & a \\ -\sin\alpha & \cos\alpha & b \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

6. Interpolation, Homogeneous Coordinates

General 2-dimensional Form	$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
General n-dimensional Form	$\begin{pmatrix} x'_1 \\ \bullet \\ x'_n \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & \bullet & r_{1n} & t_1 \\ \bullet & \bullet & \bullet & \bullet \\ r_{1n} & \bullet & r_{nn} & t_n \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x_1 \\ \bullet \\ x_n \\ 1 \end{pmatrix}$

6. Interpolation, 6.5 Rotation of Images

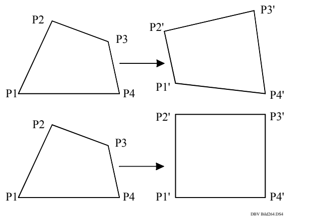
Use homogeneous coordinates and apply :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = T^{-1} * R * T * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha & -x_m\cos\alpha - y_m\sin\alpha + x_m \\ -\sin\alpha & \cos\alpha & +x_m\sin\alpha - y_m\cos\alpha + y_m \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

6. Interpolation, 6.6 Projective Collineation

Wanted: Transformation of a tetragon onto another tetragon.



In general : Transformation of a tetragon onto tetragon

Special case : Transformation of a tetragon onto rectangle

Brasov, RO
Pizzeria Venezia
Str. Appolonia Hirscher

6. Interpolation, 6.6 Projective Collineation

Points P_i und P_i' (i = 1,...,4) in homogeneous coordinates

$$P_1 = \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \ldots P_4 = \begin{pmatrix} x_4 \\ y_4 \\ 1 \end{pmatrix} \quad \text{und} \quad P_1' = \begin{pmatrix} x_1' \\ y_1' \\ 1 \end{pmatrix} \ldots P_4' = \begin{pmatrix} x_4' \\ y_4' \\ 1 \end{pmatrix}$$

Assumed, the tetragons are not degenerated
↔ neither three of the P_i nor P_i' lie onto a line,
↔ every three of the P_i or P_i' are linear independent vectors,

↔ it exists $a_i \neq 0$ and $a_i' \neq 0$ so that

$$P_4 = a_1 * P_1 + a_2 * P_2 + a_3 * P_3 \qquad \text{(Eq. 1)}$$
$$P_4' = a_1' * P_1' + a_2' * P_2' + a_3' * P_3'$$

6. Interpolation, 6.6 Projective Collineation

or that $\quad P_4 = \mathbf{P} * \mathbf{a} \qquad P_4' = \mathbf{P'} * \mathbf{a'}$, (Eq. 1a)

with matrices $\quad \mathrm{P} = (P_1 \quad P_2 \quad P_3) \qquad \mathrm{P}' = (P_1' \quad P_2' \quad P_3')$

and vectors : $\quad a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \qquad a' = \begin{pmatrix} a_1' \\ a_2' \\ a_3' \end{pmatrix}$

Multiply (Eq. 1) a from left by the inverse matrices $\mathbf{P^{-1}}$ resp. $\mathbf{P'^{-1}}$ and calculate the unknown vectors **a** resp. **a'** :

$$a = P^{-1} * P_4 \qquad a' = P'^{-1} * P_4' \qquad \text{(Eq. 1b)}$$

6. Interpolation, 6.6 Projective Collineation

Wanted : 3 X 3 - Matrix **A**, which maps the vectors P_i onto P_i' except for an unknown scaling factor.

$\mathbf{A} * \mathbf{P_i} = \lambda_i * \mathbf{P_i}'$ in which $\lambda_i \neq 0$ for i = 1, …, 4 (Eq. 2)

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} * \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11}x_i + a_{12}y_i + a_{13} \\ a_{21}x_i + a_{22}y_i + a_{23} \\ a_{31}x_i + a_{32}y_i + a_{33} \end{pmatrix} = \lambda_i \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \lambda_i * P_i'$$

6. Interpolation, 6.6 Projective Collineation

Elimination of the scaling factors $\lambda_1, \lambda_2, \lambda_3$:

$$A * P_4 \underset{(Gl.1)}{=} \sum_{i=1}^{3} A * (P_i * a_i) = \sum_{i=1}^{3} (A * P_i) * a_i \underset{(Gl.2)}{=} \sum \lambda_i * P_i' * a_i \qquad \text{(Eq. 3)}$$

$$A * P_4 \underset{(Gl.2)}{=} \lambda_4 * P_4' \underset{(Gl.1)}{=} \lambda_4 * \sum P_i' * a_i' = \sum \lambda_4 * P_i' * a_i' \qquad \text{(Eq. 4)}$$

Comparison of coefficients in the right side of Eq. 3 and 4 :

$a_i * \lambda_i = a_i' * \lambda_4$ for i = 1, 2, 3 (Eq.5)

Thus

$\lambda_i = \lambda_4 * a_i' / a_i$ for i = 1, 2, 3 with $a_i \neq 0$

6. Interpolation, 6.6 Projective Collineation

Simple procedure :

Known are the points P_i und P_i' for i = 1, 2, 3 and 4

Compose matrix **P** with i = 1, 2, 3

$$A * P = A * \begin{pmatrix} P_1 & P_2 & P_3 \end{pmatrix} \underset{(Eq.2)}{=} \begin{pmatrix} \lambda_1 P_1' & \lambda_2 P_2' & \lambda_3 P_3' \end{pmatrix} \underset{(Eq.5)}{=} \lambda_4 * \begin{pmatrix} \frac{a_1'}{a_1} P_1' & \frac{a_2'}{a_2} P_2' & \frac{a_3'}{a_3} P_3' \end{pmatrix}$$

Calculate $\mathbf{P^{-1}}$ and multiply from the right by $\mathbf{P^{-1}}$:

$$A = A * P * P^{-1} = \lambda_4 * \begin{pmatrix} \frac{a_1'}{a_1} P_1' & \frac{a_2'}{a_2} P_2' & \frac{a_3'}{a_3} P_3' \end{pmatrix} * P^{-1}$$

In which the a_i and a_i' are known from (Eq. 1b).

6. Interpolation, 6.6 Projective Collineation

The inverse matrix P^{-1} exists, because P_1, P_2 and P_3 are linear independent.

$\lambda_4 \neq 0$ is arbitrary, because the matrix A is defined except for an unknown factor.

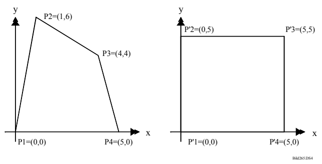
Use A * P_i = P_i' to calculate the new positions in the target image.

Note 1 : These equations were derived without resampling !
Use resampling to avoid undefined grey values in the target image.

Note 2 : Only two-dimensional problems are considered.

6. Interpolation, 6.6 Projective Collineation, Example

Example for projective collineation :
Transformation of the left tetragon onto the right square :



The vertices are known. i.e.

$$P = \begin{pmatrix} 0 & 1 & 4 \\ 0 & 6 & 4 \\ 1 & 1 & 1 \end{pmatrix} \quad P_4 = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} \quad P' = \begin{pmatrix} 0 & 0 & 5 \\ 0 & 5 & 5 \\ 1 & 1 & 1 \end{pmatrix} \quad P_4' = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix}$$

6. Interpolation, 6.6 Projective Collineation, Example

Known are the vertices

$$P = \begin{pmatrix} 0 & 1 & 4 \\ 0 & 6 & 4 \\ 1 & 1 & 1 \end{pmatrix} \quad P_4 = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} \quad P' = \begin{pmatrix} 0 & 0 & 5 \\ 0 & 5 & 5 \\ 1 & 1 & 1 \end{pmatrix} \quad P_4' = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix}$$

Calculate P^{-1} , a, a'

$$P^{-1} = \begin{pmatrix} -0,1 & -0,15 & 1 \\ -0,2 & 0,2 & 0 \\ 0,3 & -0,05 & 0 \end{pmatrix} \quad a = P^{-1} P_4 = \begin{pmatrix} 0,5 \\ -1 \\ 1,5 \end{pmatrix} \quad P'^{-1} = \begin{pmatrix} 0 & -0,2 & 1 \\ -0,2 & 0,2 & 0 \\ 0,2 & 0 & 0 \end{pmatrix} \quad a' = P'^{-1} P'_4 = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

thus A :

$$A = \lambda_4 \begin{pmatrix} \frac{a_1'}{a_1} P_1' & \frac{a_2'}{a_2} P_2' & \frac{a_3'}{a_3} P_3' \end{pmatrix} * P^{-1}$$

6. Interpolation, 6.6 Projective Collineation, Example

Choose λ_4 = 1 and get :

$$\lambda_1 = \frac{a_1'}{a_1} * \lambda_4 = 2 \qquad \lambda_2 = \frac{a_2'}{a_2} * \lambda_4 = 1 \qquad \lambda_3 = \frac{a_3'}{a_3} * \lambda_4 = 0,666...$$

Therefore get for A :

$$A = \begin{pmatrix} 0 & 0 & 3,333.. \\ 0 & 5 & 3,33.. \\ 2 & 1 & 0,66.. \end{pmatrix} * \begin{pmatrix} -0,1 & -0,15 & 1 \\ -0,2 & 0,2 & 0 \\ 0,3 & -0,05 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -0,16665 & 0 \\ 0 & 0,83335 & 0 \\ -0,2 & -0,13335 & 2 \end{pmatrix}$$

A = 1 * { } * P^{-1}

6. Interpolation, 6.6 Projective Collineation, Example

Prove for the known vertices :

$$P_1' = A * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \xrightarrow[\text{div. by } \lambda_1]{} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad P_2' = A * \begin{pmatrix} 1 \\ 6 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ 1 \end{pmatrix}$$

$$P_3' = A * \begin{pmatrix} 4 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 3,33.. \\ 3,33.. \\ 0,66.. \end{pmatrix} \xrightarrow[\text{div. by } \lambda_3]{} \begin{pmatrix} 5 \\ 5 \\ 1 \end{pmatrix} \qquad P_4' = A * \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix}$$

6. *Interpolation, 6.6 Projective Collineation, Example*

Original Image

Adjusted Image without Resampling

6. *Interpolation, 6.6 Projective Collineation, Example*

Original Image

Adjusted Image using Resampling

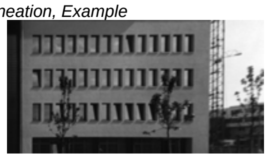
6. *Interpolation, 6.6 Projective Collineation, Example*

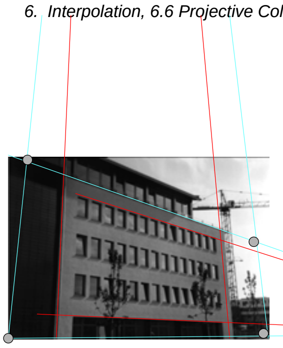
Example of rectification : Photo shot in Brasov Pizzeria, 8 Bit grey level image

6. *Interpolation, 6.6 Projective Collineation, Example*

6. *Interpolation, 6.6 Projective Collineation, Example*

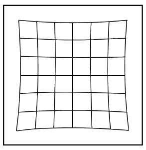
6. *Interpolation, 6.6 Projective Collineation, Example*



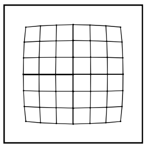


- Analyze image by Hough transform for lines.
- Calculate vanishing points (red).
- Connect these points with opposite corners of image (cyan lines).
- Calculate intersection points (grey) to define a tetragon for the projective collineation.

6. Interpolation, 6.7 Lens Distortion



DBV Bild4223.DS4

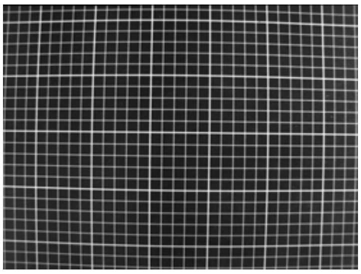


DBV Bild4225.DS4

Distortion by lens aberration :
- Pillow like distortion (above)
- Barrel like distortion (below)
- Mathematical modeling necessary

- Model :
  - „ideal" pixel image resulting from pin hole camera (geometrical optical laws !),
  - pixels with square aspect ratio (linear interpolation !)
  - radial symmetric distortion with origin in image center

6. Interpolation, 6.7 Lens Distortion



Conventional objective

Image „little_focal_length.bmp" in the folder

6. Interpolation, 6.7 Lens Distortion, Conventional Digital Camera

6. Interpolation, 6.7 Lens Distortion

$$\vec{x}_d = (1 + k_1 * r^2 + k_2 * r^4 + ...) * \vec{x}_1$$

$$\vec{x}_1 = (1 + k_1 * r^2 + k_2 * r^4)^{-1} * \vec{x}_d$$

Parameters k_1 and k_2 are unknown and specific for the lens and have to be defined by a calibrating process.

$k_1 > 0$ results in a pillow like distortion

$k_1 < 0$ results in a barrel like distortion

Radius : $r^2 = x^2 + y^2$

$\vec{x}_d = \begin{pmatrix} x_d \\ y_d \end{pmatrix}$: distorted coordinates, analog for pinhole coordinates $\vec{x}_1$

6. Interpolation, 6.7 Lens Distortion

- Image of calibration pattern
- Linear Interpolation to adjust the pixel aspect ratio
- High-pass filtering to get edges
- Hough-Transformation (Chap. 7) to find linear equations of edges
- Calculation of approximate intersection points
- Calculation in the vincinity of intersecting points suppixel precise calibation points

6. Interpolation, 6.7 Lens Distortion

- Assume image center to be optical center (according to model)
- Calculation of parameters k_1 and k_2 by overdefined equation system
- Solution by Householder-Transformation
- Result : k_1 und k_2

Application :
- Transform image data by resampling

6. Interpolation, 6.8 Correction of Distorted Images

Projective Collineation
- Gray value overflow and underflow suppressed
- Resampling necessary
- 2D interpolation

Radial symmetric distortion
- Origin of coordinate system shifted to image center, i. e.

$$\vec{x}_{target} = \left(1 + k * r^2\right) * \left(\vec{x}_{source} - \vec{x}_m\right) + \vec{x}_m$$

- Empirical choice of k
- Position data restricted to image frame
- 2D interpolation

6. Interpolation, 6.8 Correction of Distorted Images

6. Interpolation, 6.8 Correction of Distorted Images

Original Image

Corrected Color Image

Summary of Chapter 6

- Importance of Resampling
- Linear interpolation
- Bilinear interpolation
- Tri-linear interpolation
- Projective Collineation
  (Mathematical fundamentals: homogeneous coordinates, matrix inversion)
- Treatment of radial symmetric distortions
- Modeling of optical lenses
- Parameter calibration of optical lenses
- Application on images