Computational Optical Imaging - Optique Numerique

-- Image Imperfections, Panoramas, and the Basics of 3D --

Winter 2013

Ivo Ihrke
Imperfections in Imaging
Lens Aberrations

- Spherical aberration
- Coma
- Astigmatism
- Curvature of field
- Distortion
Sharpness Related Aberrations
Chromatic Aberration

- Index of refraction varies with wavelength
- For convex lens, blue focal length is shorter
- Can correct using a two-element “achromatic doublet”, with a different glass (different $n'$) for the second lens

- Achromatic doublets only correct at two wavelengths…
- Why don’t humans see chromatic aberration?

(check [http://www.telescope-optics.net/eye_chromatism.htm](http://www.telescope-optics.net/eye_chromatism.htm) )
Chromatic Aberrations

- Longitudinal chromatic aberration (change in focus with wavelength)

![Diagram of Chromatic Aberration](image: Smith 2000)

*Figure 3.10* The undercorrected longitudinal chromatic aberration of a simple lens is due to the blue rays undergoing a greater refraction than the red rays.

image: Smith 2000

Ivo Ihrke / Winter 2013
Chromatic Aberrations

- Lateral color (change in magnification with wavelength)

Figure 3.11 Lateral color, or chromatic difference of magnification, results in different-sized images for different wavelengths.

image: Smith 2000

Ivo Ihrke / Winter 2013
Spherical Aberration

- Focus varies with position on lens.
- Depends on shape of lens
- Can correct using an aspherical lens
- Can correct for this and chromatic aberration by combining with a concave lens of a different $n'$

Images: Forsyth & Ponce and Hecht 1987
Oblique Aberrations

- Spherical and chromatic aberrations occur on the lens axis. They appear everywhere on the image.

- Oblique aberrations do not appear in center of field and get worse with increasing distance from axis.
Aberrations

- Coma
  - off-axis will focus to different locations depending on lens region
  - (magnification varies with ray height)

Figure 2.16. A typical comatic star image.

images: Smith 2000 and Hecht 1987

Ivo Ihrke / Winter 2013
Coma
Astigmatism

- The shape of the lens for an off-center point might look distorted, e.g. elliptical
  - different focus for tangential and sagittal rays

![Diagram of Astigmatism](image: Smith 2000)

Hardy & Perrin
Ivo Ihrke / Winter 2013
Astigmatism
Astigmatism
Curvature of Field

- focus “plane” is actually curved

Object

Image
Field Curvature
Field Curvature

different image distance
Bad Optics

curvature of field, coma, chromatic aberration
Aberration Correction

- By deconvolution or correcting optics
  - Correcting optics – example Hubble space telescope
    uncorrected optics  corrected optics  PFS before correction
    ![Uncorrected optics](image1)  ![Corrected optics](image2)  ![PFS before correction](image3)

- Deconvolution (later lecture for details)
  uncorrected optics  deconvolved with APEX [Carasso06]
    ![Uncorrected optics](image4)  ![Deconvolved with APEX](image5)

Ivo Ihrke / Winter 2013
Distortion
Distortion

- Ratios of lengths are no longer preserved.
Geometric distortion

- Change in magnification with image position

image: Smith 2000

Ivo Ihrke / Winter 2013
Radial Distortion
Radial distortion

Barrel distortion

Pincushion distortion
Radial distortion

- Straight lines are no longer straight
- Violation of our linear camera model
- Must be compensated for computer vision applications
- Radial distortion function can be approximated by a Taylor expansion in radial direction

\[ f(r) = 1 + \kappa_2 r + \kappa_3 r^2 + \kappa_4 r^3 + \kappa_5 r^4 + \ldots \]

- Maps radius to a new radius
- One or two coefficients are usually sufficient
Radial distortion

**Barrel distortion**

**Pincushion distortion**

\[
p = f(|p_d - c|)(p_d - c) + c
\]
Radial Distortion Correction - Example

- Checkerboard serves as target for estimation
  - Corner positions define line segments
  - Minimize curvature
Contrast Issues
Radial Falloff

- **Vignetting** – your lens is basically a long tube.

- **Cos\(^4\) falloff** – “rule of thumb”.
  - At an angle, area of aperture reduced by \(\cos(a)\)
  - \(1/r^2\): Falls off as \(1/\cos(a)^2\) (due to increased distance to lens)
  - Light falls on film plane at an angle, another \(\cos(a)\) reduction.
Vignetting - Example

- a white diffuse target

- actual photograph

White field at different f-stops

<table>
<thead>
<tr>
<th>f/2.8</th>
<th>f/2.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>f/4</td>
<td>f/8</td>
</tr>
</tbody>
</table>
Flare

- Artifacts and contrast reduction caused by stray reflections
Flare

- Artifacts and contrast reduction caused by stray reflections
- Can be reduced by antireflection coating (now universal)
Ghost Images

Figure 6.14. The formation of ghost images by light reflected from the internal surfaces in a lens.
Figure 6.13. A typical family of ghost images, formed by an uncoated high-aperture lens.
Lens Flare – Effect of Coating

coating
Several Images – Panoramas

-- extending the field of view --
Mosaics and Panoramas

- Outline
  - Perspective Panoramas
    - Hardware-Based
    - Software-Based (Multiple Photographs)
      - Image registration
      - Image blending
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°

Slide from Brown & Lowe

Ivo Ihrke / Winter 2013
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Slide from Brown & Lowe
Ivo Ihrke / Winter 2013
Single vs. Multiple Viewpoint

- Single-viewpoint
  - Necessary for creating pure perspective images.
  - Many vision algorithms assume pinhole cameras.
  - Images that aren’t perspective images look distorted.

- Multi-viewpoint
  - Cross-slit panoramas, etc.
  - necessary for scenes which cannot be captured from a single viewpoint
Omnidirectional Catadioptric Cameras

O-360

EyeSee360

catadioptric = mirror + lens system
Images of an Omnidirectional Camera

images: CAVE lab
Ivo Ihrke / Winter 2013
Catadioptric System – Full Texture

[Kuthirummal 2006]
Catadioptric System – Stereo

Camera → Mirror → Subject

Center

Object

Object’
Multi-camera, Single-viewpoint?

Immersive Media “Dodeca2000”

PointGrey Ladybug
Perspective Panoramas

Registration
Single Center of Projection

- Take a sequence of images from the same position
  - Rotate the camera about its optical center
- Compute transformation between second image and first
- Transform the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images, repeat

- ...why don’t we need the 3D geometry?
Image Reprojection

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*
A pencil of rays contains all views

Can generate any synthetic camera view as long as it has **the same center of projection**!
Image reprojection

- How to relate two images from the same camera center?

- Images contain the same information along the same ray.

- Use 2D image warp
Taxonomy of Projective Transformations

\[
\begin{pmatrix}
  x'_1 \\
  x'_2 \\
  x'_3
\end{pmatrix} =
\begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3
\end{pmatrix},
\]
Perspective Transformation

- 3D to 2D projection
  - Point in world coordinates $P(x_e, y_e, z_e)$
  - Distance center of projection – image plane $D(=f)$
  - Image coordinates $(x_s, y_s)$

\[
x_s = D \frac{x_e}{z_e}
\]
\[
y_s = D \frac{y_e}{z_e}
\]
Homogeneous Coordinates: Point Representation

\[ x = \frac{x}{w} \quad \text{and} \quad y = \frac{y}{w} \]

where \( x, y, w \) are the homogeneous coordinates of a point.
Hom. Coord.: Line Representation

\[ l = (a, b, c) \]

\[ ax + by + c = 0 \]
Hom. Coordinates: Point on Line

\[ x \cdot l = 0 \]
Hom. Coordinates: Intersection of Lines

\[ l' \times l = x \]
Hom. Coordinates: Line through 2 Points

\[ l = (a, b, c) \]

\[ x = (x, y, w) \]

\[ x' = (x', y', w') \]

\[ x' \times x = l \]
Homogeneous Coordinates for 2D

- Embedding of $R^2$ into $P^2$
  - For the time being

$$R^2 \ni \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in P^2,$$

and

$$\begin{pmatrix} X \\ Y \\ W \end{pmatrix} \rightarrow \begin{pmatrix} X/W \\ Y/W \end{pmatrix}$$

- Representation of transformations by 3x3 matrices
- Mathematical trick
  - convenient representation to express rotations and translations as matrix multiplications
  - Easy to find line through points, point-line/line-line intersections
- Easy representation of projective transformation (homography)
Projective Transformations

Projecting one plane onto another using one projection center

**Definition 1.11. Projective transformation.** A planar projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular $3 \times 3$ matrix:

$$
\begin{pmatrix}
    x'_1 \\
    x'_2 \\
    x'_3
\end{pmatrix} =
\begin{bmatrix}
    h_{11} & h_{12} & h_{13} \\
    h_{21} & h_{22} & h_{23} \\
    h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{pmatrix}
    x_1 \\
    x_2 \\
    x_3
\end{pmatrix},

(1.5)

or more briefly, $x' = Hx$. 

Ivo Ihrke / Winter 2013
Examples of Projective Transformations

- Projection between 2 images via a world plane
  ⇒ Concatenating two projective transforms gives another projective transform
- Projection between 2 images with the same camera center
  ⇒ Rotating camera or camera with varying focal length
- Shadow projection of a plane onto another plane
Taxonomy of Projective Transformations

\[
\begin{pmatrix}
    x'_1 \\
    x'_2 \\
    x'_3
\end{pmatrix} =
\begin{bmatrix}
    h_{11} & h_{12} & h_{13} \\
    h_{21} & h_{22} & h_{23} \\
    h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{pmatrix}
    x_1 \\
    x_2 \\
    x_3
\end{pmatrix}
\]
**Taxonomy of Projective Transformations**

\[
\begin{pmatrix}
  x'_1 \\
  x'_2 \\
  x'_3 \\
\end{pmatrix} =
\begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33} \\
\end{bmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
\end{pmatrix},
\]

<table>
<thead>
<tr>
<th>Group</th>
<th>Matrix</th>
<th>Distortion</th>
<th>Invariant properties</th>
</tr>
</thead>
</table>
| Projective    | \[
\begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33} \\
\end{bmatrix}
\] | ![concurrency](#) | Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps; cross ratio (ratio of ratio of lengths). |
| Affine        | \[
\begin{bmatrix}
  a_{11} & a_{12} & t_x \\
  a_{21} & a_{22} & t_y \\
  0 & 0 & 1 \\
\end{bmatrix}
\] | ![parallelism](#) | Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, \(1_\infty\). |
| Similarity    | \[
\begin{bmatrix}
  sr_{11} & sr_{12} & t_x \\
  sr_{21} & sr_{22} & t_y \\
  0 & 0 & 1 \\
\end{bmatrix}
\] | ![ratio of lengths](#) | Ratio of lengths, angle. The circular points, \(1, J\) (see section 1.7.3). |
| Euclidean     | \[
\begin{bmatrix}
  r_{11} & r_{12} & t_x \\
  r_{21} & r_{22} & t_y \\
  0 & 0 & 1 \\
\end{bmatrix}
\] | ![length, area](#) | Length, area |
Distortions under Central Projection

- **Similarity**: circle remains circle, square remains square
  \[\Rightarrow\] line orientation is preserved
- **Affine**: circle becomes ellipse, square becomes rhombus
  \[\Rightarrow\] parallel lines remain parallel
- **Projective**: imaged object size depends on distance from camera
  \[\Rightarrow\] parallel lines converge
Removing Projective Distortion

Projective transformation in inhomogeneous form

\[
\begin{pmatrix}
x'_1 \\
x'_2 \\
1
\end{pmatrix} =
\begin{bmatrix}
h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix},
\]

4 general point correspondences \((x,y \rightarrow x',y')\) on the planar facade lead to eight linear equations of the type

\[
x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}
\]
\[
y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}.
\]

Sufficient to solve for \(H\) up to multiplicative factor
The Direct Linear Transform (DLT) Algorithm

Given: 4 2D point correspondences

\[ \mathbf{x}_i = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \Leftrightarrow \quad \mathbf{x}_i' = \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} \]

Objective: estimate the projective transform matrix \( \mathbf{H} \)

\[ \mathbf{x}' = \mathbf{Hx} \]

\[
\begin{pmatrix}
  x_1' \\
  x_2' \\
  x_3'
\end{pmatrix} =
\begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3
\end{pmatrix}.
\]
The DLT Algorithm II

Estimating matrix $H$ from point correspondences is equivalent to

$$x' = Hx.$$  \hspace{1cm}  \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix}  \Leftrightarrow  \begin{pmatrix} x_i' \\ y_i' \\ w_i' \end{pmatrix}$$

$$x_i' \times Hx_i = 0$$

gives

$$x_i' \times Hx_i = \begin{pmatrix} y_i' h^3 \top x_i - w_i' h^2 \top x_i \\ w_i' h^1 \top x_i - x_i' h^3 \top x_i \\ x_i' h^2 \top x_i - y_i' h^1 \top x_i \end{pmatrix}.$$ 

Re-ording into $h$ vector

$$\begin{bmatrix} 0^\top & -w_i' x_i^\top & y_i' x_i^\top \\ w_i' x_i^\top & 0^\top & -x_i' x_i^\top \\ -y_i' x_i^\top & x_i' x_i^\top & 0^\top \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0.$$
The DLT Algorithm III

Only rows 1 and 2 are linearly independent \( \Rightarrow \) omit row 3

\[
\begin{bmatrix}
0^T & -w_i'x_i^T & y_i'x_i^T \\
w_i'x_i^T & 0^T & -x_i'x_i^T \\
-y_i'x_i^T & x_i'x_i^T & 0^T \\
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
h_3 \\
\end{bmatrix}
= 0.
\]

\[x_i = \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} \quad \Leftrightarrow \quad x_i' = \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix}
\]

Inhomogeneous solution: set one matrix entry equal to 1 (e.g. \( h_{33} \))

\[
\begin{bmatrix}
0^T & -w_i'x_i^T & y_i'x_i^T \\
w_i'x_i^T & 0^T & -x_i'x_i^T \\
0 & x_i'w_i' & y_i'w_i' \\
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
h_3 \\
\end{bmatrix}
= 0.
\]

\[A_i h = 0 \]

Solve by Gaussian elimination or least-squares techniques
Estimating Homographies

Objective

Given \( n \geq 4 \) 2D to 2D point correspondences \( \{x_i \leftrightarrow x_i'\} \), determine the 2D homography matrix \( H \) such that \( x_i' = Hx_i \).

Algorithm

(i) **Normalization of \( x \):** Compute a similarity transformation \( T \), consisting of a translation and scaling, that takes points \( x_i \) to a new set of points \( \tilde{x}_i \) such that the centroid of the points \( \tilde{x}_i \) is the coordinate origin \( (0,0)^\top \), and their average distance from the origin is \( \sqrt{2} \).

(ii) **Normalization of \( x' \):** Compute a similar transformation \( T' \) for the points in the second image, transforming points \( x'_i \) to \( \tilde{x}'_i \).

(iii) **DLT:** Apply algorithm to the correspondences \( \tilde{x}_i \leftrightarrow \tilde{x}'_i \) to obtain a homography \( \tilde{H} \).

(iv) **Denormalization:** Set \( H = T'^{-1}\tilde{H}T \).
Homography or not?

- Coincidences between 3D points at different depths are preserved
- Pure camera rotation about camera center
  $\Rightarrow$ 2D Homography

- Different depths are imaged to different image positions
- Camera rotates and translates
  $\Rightarrow$ Motion Parallax, no Homography
Panoramic Mosaicing

Rotation about camera center: homography

- choose one image as reference
- compute homography to map neighboring image to reference image plane
- projectively warp image, add to reference plane
- repeat for all images

⇒ bow tie shape
Alternative Panoramas

- Project images onto different surfaces:
  - Spherical
  - Cylindrical
  - Cubic (think of cube map)

Images www.panoguide.com
Ivo Ihrke / Winter 2013
Register left and right image to the middle one using two homographies $H_1, H_2$
Example – My former office

all images registered to the central one (2 homographies)
## Example – My former office

\[
I_{\text{out}} = \begin{cases} 
I_1 & I_1 > 0 \land \neg (I_2 > 0 \lor I_3 > 0) \\
I_2 & I_2 > 0 \land \neg (I_1 > 0 \lor I_3 > 0) \\
I_3 & I_3 > 0 \land \neg (I_1 > 0 \lor I_2 > 0) \\
(I_1 + I_2)/2 & (I_1 > 0 \land I_2 > 0) \land \neg (I_3 > 0) \\
(I_3 + I_2)/2 & (I_3 > 0 \land I_2 > 0) \land \neg (I_1 > 0)
\end{cases}
\]
Camera Models

-- relating the 3D world with the 2D image --

Slides by Thorsten Thormaehlen
Advanced camera models

- Finite cameras
- Projective cameras
- Affine cameras
- Other cameras
- Homework: Camera matrix decomposition
Finite cameras

- Start with the most simple model (pinhole camera, perspective camera, we already know) and then generalize
- The finite cameras model can represent today's real CMOS or CCD cameras but also other forms of image generation, like
  - X-ray images
  - scanned film
  - time-of-flight cameras
  - …
Perspective camera (pinhole camera)

\[ x = f \frac{X_C}{Z_C} \quad \text{and} \quad y = f \frac{Y_C}{Z_C} \]
Matrix representation (for homogeneous points)

with \( \mathbf{P}_c = (X_c, Y_c, Z_c, 1)^\top \) and \( \mathbf{p} = (\tilde{x}, \tilde{y}, \tilde{z})^\top \)

where \( x = \frac{\tilde{x}}{\tilde{z}} \) and \( y = \frac{\tilde{y}}{\tilde{z}} \)

it follows

\[
\mathbf{p} = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
P_c
\end{bmatrix}
\]
Matrix representation (for homogeneous points)

\[
p = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_c \end{bmatrix}
\]

with \( K = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \) (calibration matrix) it follows
Matrix representation (for homogeneous points)

with \( P_c = \begin{bmatrix} R & \mathbf{0} & \mathbf{C} \\ 0 & 1 \end{bmatrix} P \)

external camera orientation and position

it follows

\[
p = K \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} P
\]

\( \Leftrightarrow p = KR[I | -C]P \)

\( p = AP \)
digital cameras sometimes do not have square pixels

- PAL 4:3 with (720 x 576 pixels)
- NTSC 4:3 with (720 x 480 pixels)

- Pixel aspect ratio \( \frac{p_x}{p_y} \)
  - Pixel aspect ratio in PAL 1.06667
  - Pixel aspect ratio in NTSC 0.8888
    (or 0.9 in case of 720 x 486)

- Unit of \( p_x \) and \( p_y \) often [pixel / mm] or [pixel / 3D unit]
Finite cameras – Pixel Aspect Ratio

\[
p = \begin{bmatrix}
\frac{f}{p_x} & 0 & 0 & 0 \\
0 & \frac{f}{p_y} & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \quad P_c
\]

Calibration matrix is now

\[
k = \begin{bmatrix}
\frac{f}{p_x} & 0 & 0 \\
0 & \frac{f}{p_y} & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

area of one pixel on chip
Finite cameras – Principle Point Offset

- origin of image plane co-ordinate system and point where optical axis intersects the image plane may not be the same

\[ x = f \frac{X_c}{Z_c} + c_x \quad \text{and} \quad y = f \frac{Y_c}{Z_c} + c_y \]
Finite cameras – Principle Point Offset

\[ x = f \frac{X_c}{Z_c} + c_x \quad \text{and} \quad y = f \frac{Y_c}{Z_c} + c_y \]

\[
p = \begin{bmatrix}
f/p_x & 0 & c_x & 0 \\
0 & 1/f_y & c_y & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\[ K = \begin{bmatrix}
f/p_x & 0 & c_x \\
0 & 1/f_y & c_y \\
0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
f_x & 0 & c_x \\
0 & f_y & c_y \\
0 & 0 & 1
\end{bmatrix}
\]

- A finite camera \( A = KR [I] - C \) has 10 degree of freedom (4 + 3 rotation + 3 translation)
Finite projective camera

- To add more generality, we can consider a calibration matrix of the form

\[
K = \begin{bmatrix}
    f_x & s & c_x \\
    0 & f_y & c_y \\
    0 & 0 & 1
\end{bmatrix}
\]

- The added parameter is referred to as *skew* parameter
- is zero for most real world examples
- Such a camera \( A = KR [I] - C \) is called *finite projective*
- It has 11 degrees of freedom, the same as a general 3x4 matrix (up to arbitrary scale)
- The left 3x3 sub-matrix is non-singular
General projective camera – Finding the camera center

- The projective camera center

\[ C = (C_x, C_y, C_z, C_w)^\top \]

can be calculated by

\[ A C = 0 \]

- E.g. by singular value decomposition (basis vector of 1D null-space)
General projective camera – Decomposition

- Decomposition of the left 3x3 sub-matrix into the calibration and rotation matrix

\[ KR = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \]

- can be done by QR decomposition
The strategy of the QR algorithm is to clear out the lower half of the matrix, one value at a time, using Givens rotations

- Givens rotation = rotation in an axis-aligned plane

The R in QR decomposition stands for *right-triangular* not *rotation*, in fact, **Q is the rotation matrix** here

- QR returns *rotation* times *upper-triangular*
- Need *upper-triangular* times *rotation*, i.e. **RQ decomposition**

- 2 options: modified QR, or Givens rotations from the right
General projective camera – RQ Decomposition

\[ A' = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{ Q}_x (a'_{32} = 0) \]

\[ A'' = \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} \\ a'_{21} & a'_{22} & a'_{23} \\ a'_{31} & 0 & a_{33} \end{bmatrix} \text{ Q}_y (a''_{32} = 0; a''_{31} = 0) \]

\[ K = \begin{bmatrix} a''_{11} & a''_{12} & a''_{13} \\ a''_{21} & a''_{22} & a''_{23} \\ 0 & 0 & a_{33} \end{bmatrix} \text{ Q}_z (k_{32} = 0; k_{31} = 0; k_{21} = 0) \]

\[ R = Q_z^T Q_y^T Q_x^T \]

with the Givens rotations

\[ Q_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \]

\[ Q_y = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix} \]

\[ Q_z = \begin{bmatrix} c & -s & 0 \\ s & 0 & c \\ 0 & 0 & 1 \end{bmatrix} \]

\[ c := \cos \theta \]
\[ s := \sin \theta \]

\[ 0 = c \cdot a_{32} + s \cdot a_{33} \]
\[ 0 = \sqrt{1 - s^2} a_{32} + s a_{33} \]

\[ \Rightarrow s = \frac{a_{32}}{\sqrt{a_{32}^2 + a_{33}^2}} \]

\[ c = \frac{-a_{33}}{\sqrt{a_{32}^2 + a_{33}^2}} \quad \text{and} \quad s = \frac{a_{32}}{\sqrt{a_{32}^2 + a_{33}^2}} \]

\[ \text{with } c = \frac{a'_{33}}{\sqrt{a'_{31}^2 + a'_{33}^2}} \quad \text{and} \quad s = \frac{a'_{31}}{\sqrt{a'_{31}^2 + a'_{33}^2}} \]

\[ c = \frac{a''_{22}}{\sqrt{a''_{21}^2 + a''_{22}^2}} \quad \text{and} \quad s = \frac{-a''_{21}}{\sqrt{a''_{21}^2 + a''_{22}^2}} \]
// QR decomposition

Matrix < double > KR(3,3); // must be filled somehow
Matrix < double > Qx(3,3);
double a = KR[2][1]; double b = KR[2][2];
double c = -b/sqrt(a*a+b*b); double s = a/sqrt(a*a+b*b);
Qx[0][0] = 1.0; Qx[0][1] = 0.0; Qx[0][2] = 0.0;
Qx[1][0] = 0.0; Qx[1][1] = c; Qx[1][2] = -s;
Qx[2][0] = 0.0; Qx[2][1] = s; Qx[2][2] = c;

KR = KR*Qx;
//cout << "KR" << endl << KR << endl;

Matrix < double > Qy(3,3);
a = KR[2][0]; b = -KR[2][2];
c = -b/sqrt(a*a+b*b); s = a/sqrt(a*a+b*b);
Qy[0][0] = c; Qy[0][1] = 0.0; Qy[0][2] = s;
Qy[1][0] = 0.0; Qy[1][1] = 1.0; Qy[1][2] = 0.0;
Qy[2][0] = -s; Qy[2][1] = 0.0; Qy[2][2] = c;

KR = KR*Qy;
//cout << "KR" << endl << KR << endl;

Matrix < double > Qz(3,3);
a = -KR[1][0]; b = -KR[1][1];
c = -b/sqrt(a*a+b*b); s = a/sqrt(a*a+b*b);
Qz[0][0] = c; Qz[0][1] = -s; Qz[0][2] = 0.0;
Qz[1][0] = s; Qz[1][1] = c; Qz[1][2] = 0.0;
Qz[2][0] = 0.0; Qz[2][1] = 0.0; Qz[2][2] = 1.0;

K = KR*Qz;
//cout << "K" << endl << K << endl;

R = transpose(Qz)*transpose(Qy)*transpose(Qx);
RQ decomposition from QR

- Modified QR decomposition
  - Reason: QR decomposition is often implemented in linear algebra packages (e.g. MATLAB, Octave, …)
  - Idea: decompose $R^T K^T$
  - Problem: $K^T$ is upper triangular, $\rightarrow$ $K$ is lower-triangular
  - Solution, use row and column exchanges by permutation matrix $P$
    \[
P = \begin{bmatrix}
    0 & 0 & 1 \\
    0 & 1 & 0 \\
    1 & 0 & 0
    \end{bmatrix}
    \]
  - Compute decomposed matrices as
    \[
    \hat{R}, \hat{K} = \text{qr}((PKR)^T) \\
    R = P\hat{R}^T \\
    K = P\hat{K}^TP
    \]
Notes on the RQ decomposition:

- The decomposition is not unique – there is a sign ambiguity (e.g. multiplying the resulting K and R both by -1 does not change the product KR)

- Relates to the choice of image (2D) coordinate axes w.r.t. the world (3D) coordinate axes
Camera Models – Special Cases
Orthographic and Affine cameras

Rendering with a **perspective** camera and an **orthographic** camera from the same viewpoint
General projective camera

- To add more generality, we can consider just an arbitrary $3 \times 4$ matrix

$$ p = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34}
\end{bmatrix} \quad P_c $$

- And now even the left $3x3$ sub-matrix can be singular

- But whole matrix must have full rank of 3 (otherwise no image is formed, and projection is just a line or a point)

- The general projective camera has also 11 degrees of freedom
Orthographic camera

The z-coordinate does not matter anymore …

\[ P = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} P_c \]

\[ A_{\text{ortho}} = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

applying rotation and translation stays the same

\[ P_c = \begin{bmatrix} R & -R \overline{C} \\ 0 & 1 \end{bmatrix} P \]

and we get

\[ A_{\text{ortho}} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & -R \overline{C} \\ 0 & 1 \end{bmatrix} \]
Affine camera

- To add more generality, we can consider just an arbitrary 3 x 4 matrix, with the first 3 elements of the last row equal to zero and the last element equal to 1

\[
A_{\text{affine}} = \begin{bmatrix}
m_{11} & m_{12} & m_{13} & t_1 \\
m_{21} & m_{22} & m_{23} & t_2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- This is a general affine camera (8 degrees of freedom)

- Possible decomposition:

\[
A_{\text{affine}} = \begin{bmatrix}
f_x & s & 0 \\
0 & f_y & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
r_1^T \\
r_2^T \\
0^T
\end{bmatrix} \begin{bmatrix}
t_1 \\
t_2 \\
1
\end{bmatrix}
\]
Other cameras – Pushbroom cameras

- Camera model for satellites of planes at high altitude
- A single line image line is captured at a time

Source: [1]
Other cameras – Pushbroom cameras

- It turns out that the mapping from 3D space to the image can be described by a general 3 x 4 matrix

- However, it is transferred back differently to inhomogeneous coordinates

\[
p = A P \quad \text{with} \quad P = (\bar{x}, \bar{y}, \bar{z})^\top
\]

\[
x = \frac{\bar{x}}{\bar{z}} \quad \frac{\bar{x}}{\bar{z}}
\]

- The camera has still the full projective 11 degrees of freedom
Other cameras – Pushbroom cameras

- More complicated motion (not linear in x direction) leads to interesting image formations
- leads to slit-scan photography

Slit-scan photography

backlit artwork

foreground slit

resulting image

camera

Source: [2]

Ivo Ihrke / Winter 2013
Stargate sequence in 2001: Space Odyssey (1968)
Homework: Camera matrix decomposition

- Decompose the following camera matrix into its components

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & a_{13} & a_{14}; \\
    a_{21} & a_{22} & a_{23} & a_{24}; \\
    a_{31} & a_{32} & a_{33} & a_{34}
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
    421.0140 & 312.9683 & 219.5796 & -1376.3198; \\
    -75.2415 & 9.1790 & 484.1022 & -669.269784; \\
    0.7071 & -0.3535 & 0.6124 & -0.918565
\end{bmatrix}
\]

- Send your solution to ivo.ihrke@inria.fr before the next lecture, use the following format (3 text lines)

\[
K = \begin{bmatrix}
    k_{11} & k_{12} & k_{13}; \\
    k_{21} & k_{22} & k_{23}; \\
    k_{31} & k_{32} & k_{33}
\end{bmatrix}
\]

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

\[
C = \begin{bmatrix}
    c_1; \\
    c_2; \\
    c_3
\end{bmatrix}
\]