

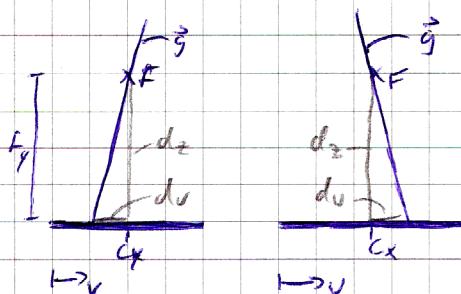
BProj: calculate Object Ray

Berechnung des Strahl, auf dem das Objekt liegt, im Kamera Koordinatensystem.

• Sensor / Bild - Koordinaten :



C : principal Point
Optischer Mittelpunkt d. Sensors



hier: $f_x = f_y = f$! $f, c_x, c_y = \text{const.}$

$$\vec{g}(u, v) = \begin{pmatrix} c_x - v \\ c_y - v \end{pmatrix} / f$$

→ für Kamerakoordinaten mit Ursprung im Brennpunkt F
↳ Koordinaten u, v, w
↳ Blickrichtung d. Kameras in w -Achse !

• Koordinatentransformation in Weltkoordinaten:

- ↳ \vec{g} muss in Weltkoordinaten rotiert werden → \vec{g}'
- ↳ Translation nicht notwendig, da objectRay als Vektorgl. in Parameterform
- ↳ Ortsvektor d. Kamera bekannt

⇒ Euler-Transformation nach Gier-Nick-Paul-Konvention
↳ Rotation um z , dann y' , dann x''

siehe auch MatLab Script in Dokumentation

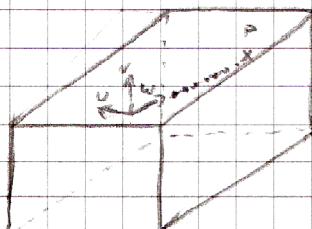
Weltkoordinatensystem x, y, z wird auf Kameraposition rotiert:

x'' ist Kamera-Blickrichtung, im weitergehenden w

y'' ist Sensorkoordinate v

z'' ist Sensorkoordinate u

↳ es wurde keine direkte Zuordnung ($x'' \leftrightarrow u$, $y'' \leftrightarrow v$, $z'' \leftrightarrow w$) gewählt, da die Euler-Rotationen so besser verständlich sind



Kameraposition und Rotation:

- Ortsvektor : \vec{o}
- Bildmittelpunkt Wand : \vec{p}
- 2. Bildpunkt an Wand : \vec{t}
- Richtungsvektor : $\vec{r} = \vec{p} - \vec{o}$
- 2. Richtungsvektor : $\vec{s} = \vec{t} - \vec{o}$

⇒ 1. Eulerwinkel = $-\arctan(\vec{r}.y, \vec{r}.x)$

⇒ 2. Eulerwinkel = $-\arcsin(\vec{r}.z / |\vec{r}|)$

⇒ 3. Eulerwinkel : Winkel zw. \vec{t}' (Normale d. x, y -Ebene)
und d. Normale d. u, v -Ebene bzw. \vec{r}, \vec{s} -Ebene

BProj: Camera Calibration

=> intrinsische Parameter:

recommended: CV-CALIB-FIX-K3

=> extrinsische Parameter mit calibrateCamera

- object points → Granzahlig als Spalten und Zeilen
 - ↳ Einheit mm/n ns Raster Schachbrett in mm
 - ↳ legt Kamera Vek., Objekt und Kamera-Koordinaten-Einheit fest

=> Rotation / Translation : Objekt in Kamera-Koordinatensystem erzeugen
Rotation um Achsen
Translation im rechten Koordinatensystem

=> mit FindExtrinsicParameters: ähnlich, nur mit festen intrinsischen Parametern

triangulatedPoints erst NACH stereoRectify !

B-Proj.: Camera Calibration + Calculated Object Ray

- Methode nach openCV:

$$S \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R & t_x \\ R & t_y \\ R & t_z \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

R : Rot. Matrix
 t : Translationvekt.
 u, v : Koordinaten Sensor
 x, y, z : Koordinaten Welt

- IST: Kamera 1: 840x680px $f=6\text{mm}$ 21BF04 1/16" / 5,6µm

$$\Rightarrow A : \begin{bmatrix} 821 & 0 & 319 \\ 0 & 821 & 239 \\ 0 & 0 & 1 \end{bmatrix} \quad \rightarrow \text{plausibel}$$

$$\Rightarrow rvec_0 : \begin{pmatrix} -1,47 \\ +1,15 \\ +0,82 \end{pmatrix} \quad tvec : \begin{pmatrix} 702 \\ -857 \\ 9590 \end{pmatrix}$$

Kamera 2: 1280x960px $f=8\text{mm}$ 41AF02 1/12" / 4,65µm

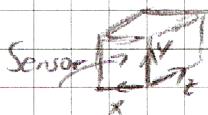
$$\Rightarrow A : \begin{bmatrix} 1354 & 0 & 690 \\ 0 & 1354 & 480 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow rvec_0 : \begin{pmatrix} -1,36 \\ +1,42 \\ +1,78 \end{pmatrix} \quad tvec : \begin{pmatrix} 1806 \\ -844 \\ 9125 \end{pmatrix}$$

- SOLL (Theorie) Rotation d. Kamera in das Weltkoord. syst.

1. -90° um X, oder ...
2. $+90^\circ$ um Z'

1. $+90^\circ$ um X, oder ...
2. $+180^\circ$ um Y'
3. -90° um Z'



x, y : Sensorkoordinaten
 \hookrightarrow Kamera „auf dem Kopf“

