

# Transformationen

Einführung in die  
Computergrafik

# Änderungshistorie

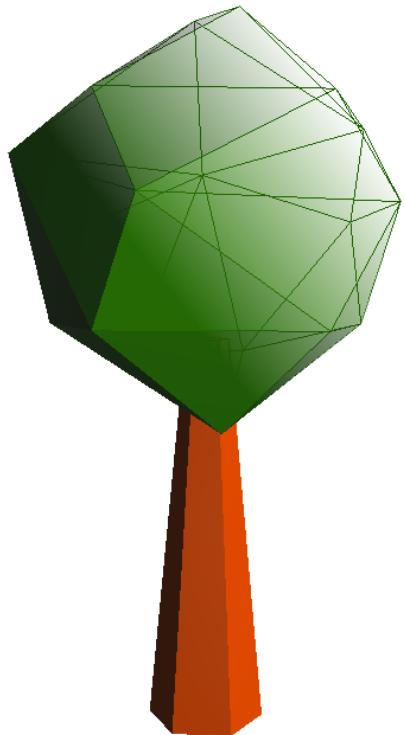
- 8.4.19
  - Learning Outcomes ergänzt

# Wiederholung

- Dreiecksnetze
- Normalen
- Lokale Beleuchtungsmodelle
- Texturen
- Nachbarschaftsdatenstrukturen
- Glättung
- Grundlagen: Matrizen

# Agenda

- Transformationen
- Kamera-Transformationen
- Tracking



# Homogene Koordinaten

# Homogene Koordinaten

- häufige Transformation:
  - Rotation + Verschiebung (Translation)
- $$T : \mathbf{R} \cdot \vec{x} + \vec{t}$$
- grundsätzlich:
  - Rotationen und Skalierungen lassen sich als Matrizen darstellen
  - Rotation eines Vektor = Vektor mit Matrix multiplizieren
  - affine Transformationen
- Translationen
  - nicht affin (keine Invarianz der Ursprungs)
  - daher: Vektoraddition
- Wunsch: alle Transformationen als Matrizen  
→ homogene Koordinaten

# Homogene Koordinaten

- Idee:
  - Einbettung in Raum mit um 1 höherer Dimension (w-Koordinate)
  - Vektoren: Koordinate mit 0 ergänzen
  - Matrizen: Zeile und Spalte mit 0en ergänzen, 1 auf der Hauptdiagonalen
- allgemein (hier 2D):
  - Vektoren:  $\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ w \end{pmatrix}$        $\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ 1 \end{pmatrix}$
  - Matrizen:  $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix}$

# Homogene Translation

- Verschiebung als „homogene“ Spalte
- Beispiel: Verschiebung im 2D:

- Verschiebungsvektor  $\begin{pmatrix} t_x \\ t_y \end{pmatrix}$
- homogene Verschiebungsmatrix:  $\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$

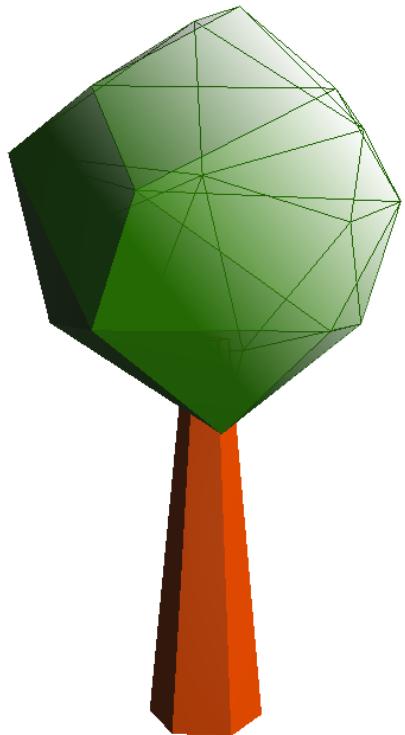
- Test:

$$\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

# Übung: Homogene Verschiebung

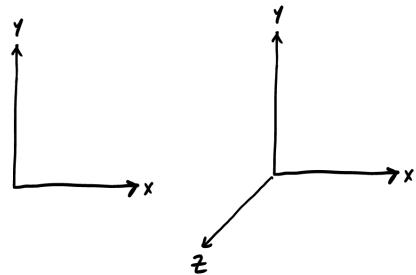
- Verwenden Sie eine homogene Repräsentation, um die Verschiebung des Punktes  $p = (2,3)$  um den Vektor  $(-1, -2)$  zu verschieben



# Basisvektoren

# Karthesisches Koordinatensystem

- Achsen: Einheitsvektoren, senkrecht zueinander, rechtshändig



- Basisvektoren:  $\vec{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \vec{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

- Beispiel:

$$\vec{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad 1 \cdot \vec{e}_0 + 2 \cdot \vec{e}_1$$

# Andere Koordinatensysteme

- Beispiel: Basisvektoren

$$\vec{b}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \vec{b}_1 = \begin{pmatrix} 0 \\ 0.5 \end{pmatrix}$$

- dann  $\vec{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

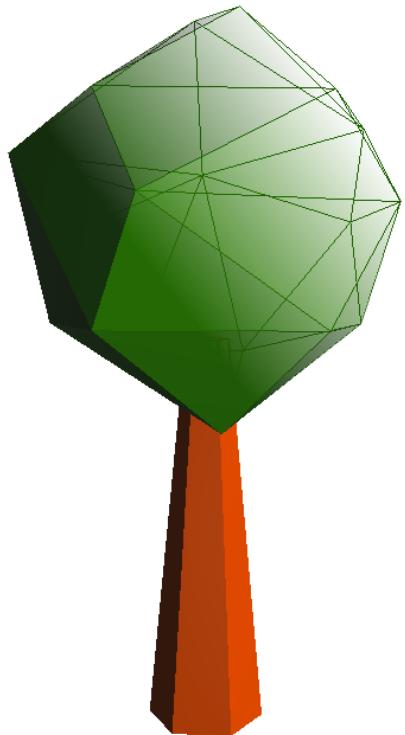
$$\begin{aligned} & 1 \cdot \vec{b}_0 + 2 \cdot \vec{b}_1 \\ = & 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 2 \cdot \begin{pmatrix} 0 \\ 0.5 \end{pmatrix} \\ = & \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \end{aligned}$$

- allgemein für Punkt in Koordinatensystem:

$$\vec{p} = \sum_{i=0}^{n-1} v_i \cdot \vec{b}_i$$

# Übung: Basisvektoren

- Geben Sie eine Basis (zwei Basisvektoren) im 2D mit den folgenden Eigenschaften an:
  - die Länge der Basisvektoren in Wurzel(2)
  - ein Basisvektor zeigt nach „Nord-Osten“
  - die Basisvektoren stehen senkrecht aufeinander



# Transformationsmatrizen

# Transformationsmatrix

- Abbildung von Basisvektoren eines Koordinatensystems auf die entsprechenden Basisvektoren eines anderen Koordinatensystems
- Kernsatz:

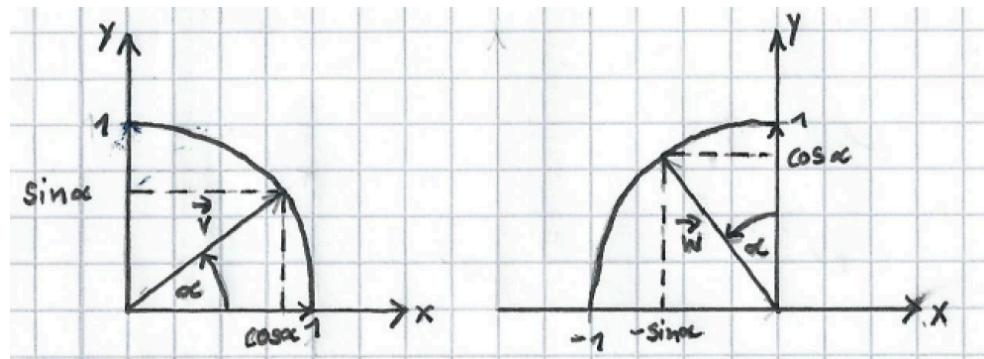
*Die Bilder der Basisvektoren stehen in den Spalten der Transformationsmatrix.*

# Rotationsmatrizen

- Rotation um den Winkel  $\phi$  im 2D:
  - Rotation der Basisvektoren des Karthesischen Koordinatensystems

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} \cos \phi \\ \sin \phi \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} -\sin \phi \\ \cos \phi \end{pmatrix}$$



- „Bilder des Basisvektoren als Spalten der Transformationsmatrix“:

$$\mathbf{R} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix}$$

- analoge Herleitung für Rotationsmatrizen um x-, y- und z-Achse im 3D

# Transformation zwischen Koordinatensystemen

- Rotationsbeispiel
  - Basis-Koordinatensystem = Karthesisches Koordinatensystem
  - Ziel-Koordinatensystem = um  $\Phi$  gedrehtes Koordinatensystem
  - Rotationsmatrix = Transformation vom Ziel-Koordinatensystem ins Karthesischen Koordinatensystem

# Transformation zwischen Koordinatensystemen

- allgemein: Transformation von Koordinatensystem  $K_1$  in Koordinatensystem  $K_2$
- Idee: Umweg über Karthesisches Koordinatensystem

$$K_1 \rightarrow K_{\text{Karthesisch}} \rightarrow K_2$$

- also

$$\vec{v}^{K_2} = T_2^{-1} \cdot T_1 \cdot \vec{v}^{K_1}$$

- mit  $T_1$  = Transformation  $K_1$  nach Karthesisch,  $T_2$  analog,  $T_2^{-1}$  = Inverse von  $T_2$

# Beispiel

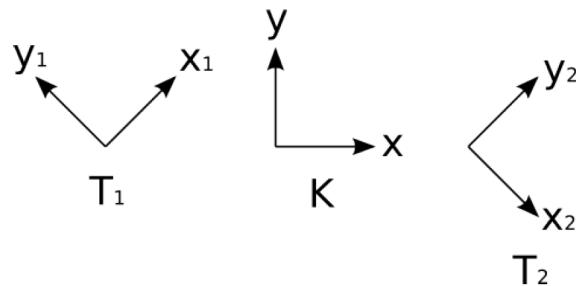
- zwei Koordinatensysteme mit  $T_1$  und  $T_2$

$$T_1 : x_1 = (1, 1), y_1 = (-1, 1)$$

$$T_2 : x_2 = (1, -1), y_2 = (1, 1)$$

$$T_1 : \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

$$T_2 : \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$



- betrachten wir  $v_1$  in  $T_1$  mit  $v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

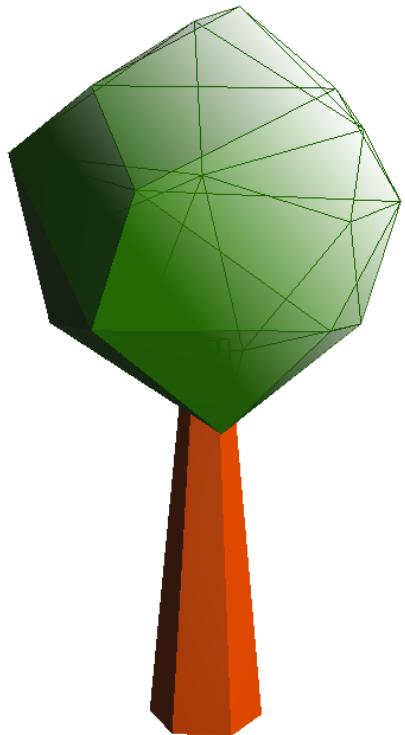
- $v_1$  im Karthesischen Koordinatensystem:  $v_K = T_1 v_1 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$

- $v_2$  im Koordinatensystem  $T_2$ :

$$v_k = T_2 v_2 = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

# Übung: Transformationsmatrizen

- Geben Sie eine Transformationsmatrix für 2D-Objekte an, die
  - die Objekte um  $10^\circ$  gegen den Uhrzeigersinn dreht
  - die Objekte um den Vektor  $(1,2)$  verschiebt



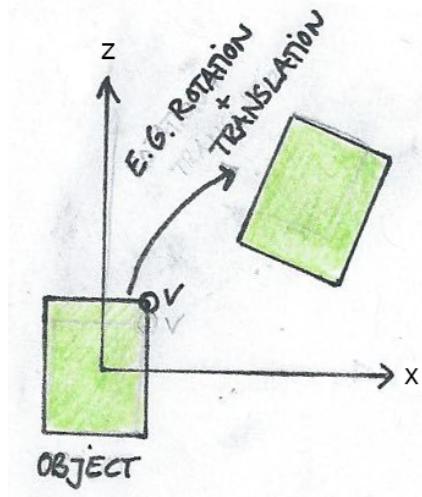
# Kamera- Transformationen

# Kamera-Transformation

- Umrechnung eines 3D-Punktes in der Szene in einen Bildpunkt (Pixel)
- basiert
  - Szenengraph
  - extrinsische Kameraparameter
  - intrinsische Kameraparameter
- Teilschritte
  - Model-Transformation
  - View-Transformation
  - Perspektivische Transformation
  - Pixel-Abbildung

# Model-Transformation

- Umsetzung der Transformationen im Szenengraphen
  - von der Wurzel bis zum Knoten, der den Vertex beinhaltet
  - zusammengefasste Transformation:  $M$



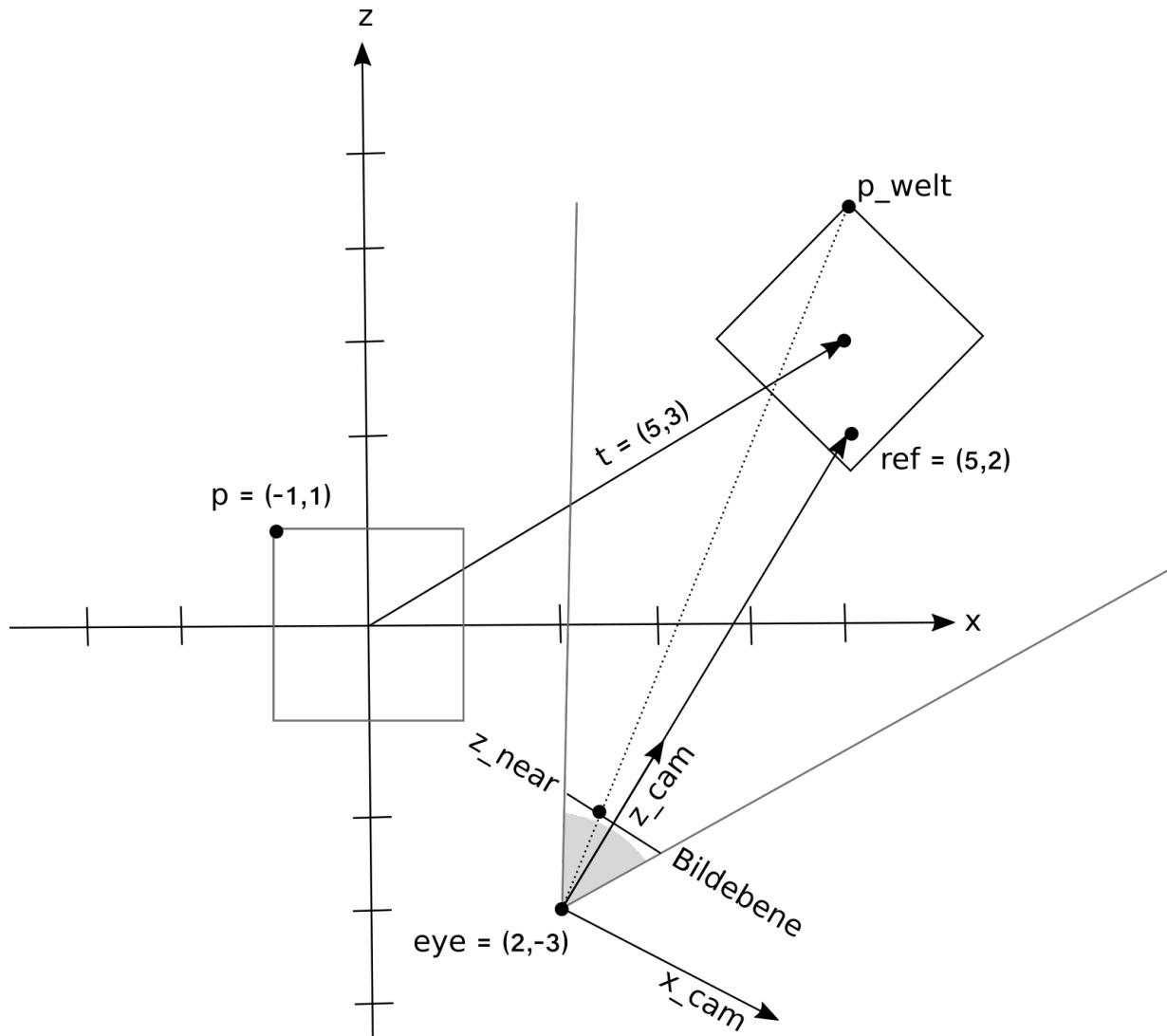
$$v' = \mathbf{M}v$$

$v$  = Vertexposition im lokalen Koordinatensystem

$v'$  = Vertexposition nach Model-Transformation  
(Weltkoordinaten)

- Hinweis: wir verwenden hier im 2D die x- und die z-Achse
  - Grund: die Kamera guckt üblicherweise entlang „ihrer“ z-Achse

# Beispiel



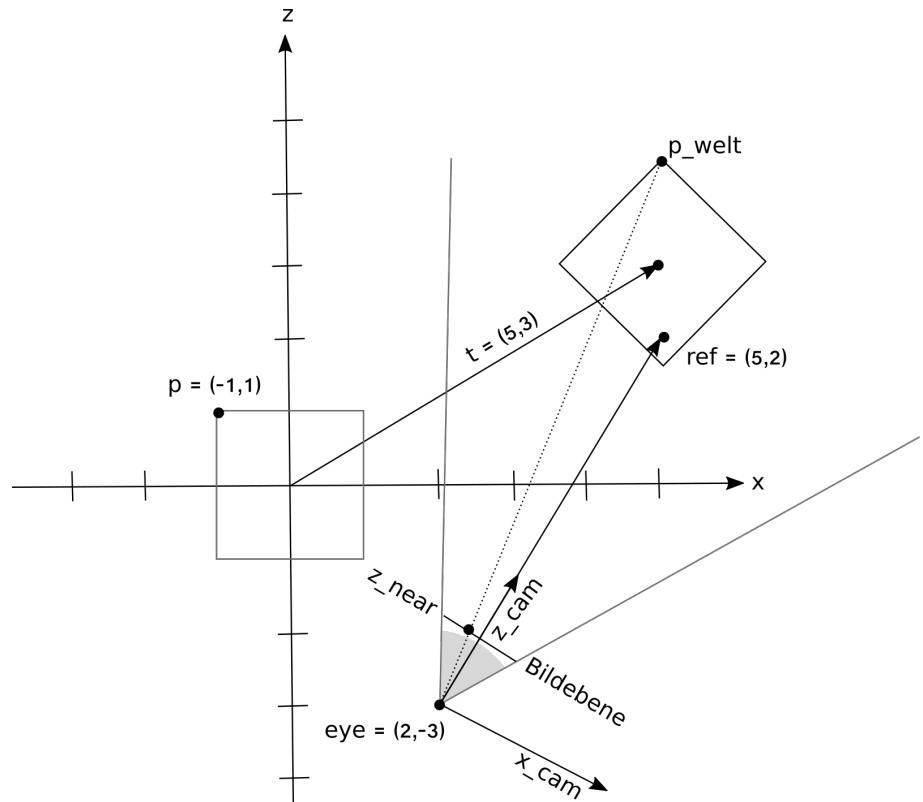
# Model-Transformation

- zwei Transformationen
  - Verschiebung um  $t = (5,3)$
  - Rotation um  $45^\circ$  im Uhrzeigersinn

$$\mathbf{M} = \begin{pmatrix} \cos(-45) & -\sin(-45) & 5 \\ \sin(-45) & \cos(-45) & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

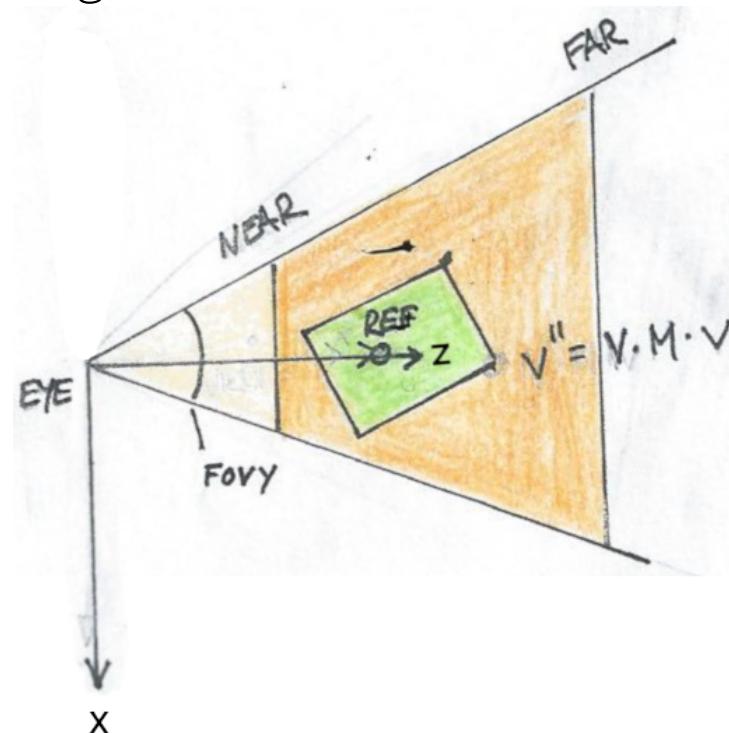
- und damit

$$\mathbf{p}_{\text{welt}} = \mathbf{M} \cdot \mathbf{p} = (5, 4.414, 1)$$



# View-Transformation

- Abbildung in das Kamera-Koordinatensystem = View Matrix
  - Ursprung = Kamera-Position
  - Achsen ergeben sich aus Position, Referenzpunkt und Oben-Vektor
  - Kamera-Blickrichtung entlang der z-Achse



# View Matrix

$$\mathbf{z} = \frac{\mathbf{ref} - \mathbf{eye}}{\|\mathbf{ref} - \mathbf{eye}\|}$$

$$\mathbf{x} = \mathbf{up} \times \mathbf{z}$$

$$\mathbf{y} = \mathbf{z} \times \mathbf{x}$$

$$\mathbf{V} = \begin{pmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{eye} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1}$$

- View Matrix
  - Rotationsanteil durch  $x, y, z$
  - Translation durch Augpunkt
  - Inverse, weil Transformation ins Kamerakoordinatensystem

# View Matrix

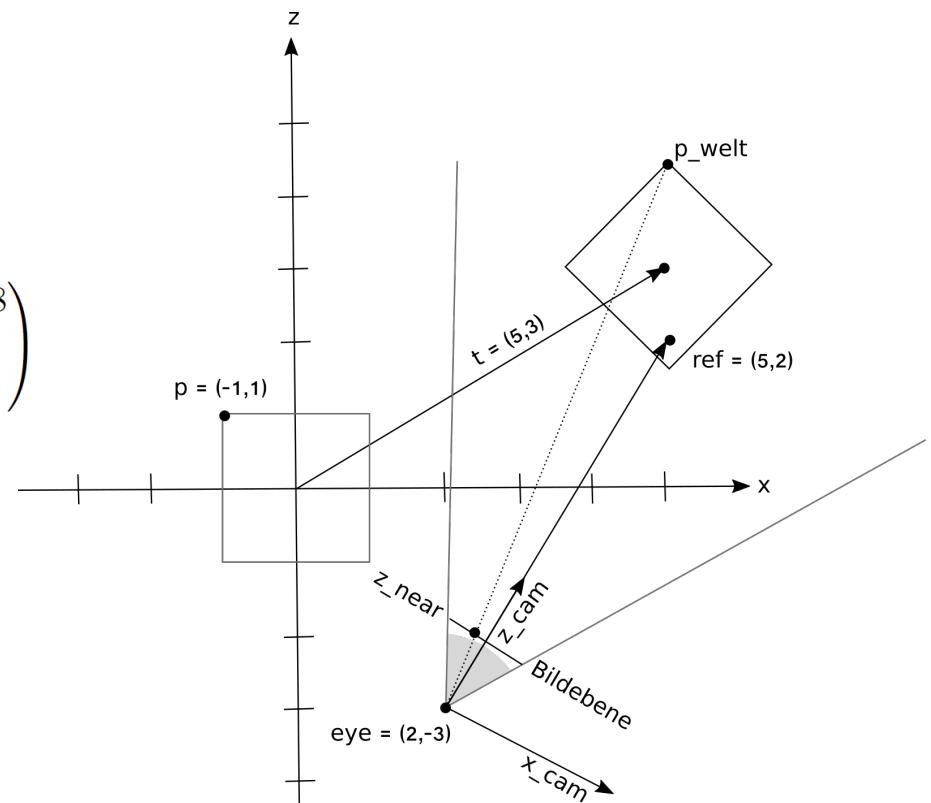
- im Beispiel (2D)
- x- und z-Achse des Kamerakoordinatensystems

$$\mathbf{z} = \frac{(5, 2) - (2, -3)}{\|(5, 2) - (2, -3)\|} = (0.514, 0.857)$$

$$\mathbf{x} = (0, 1, 0) \times \mathbf{z} = (0.857, -0.514)$$

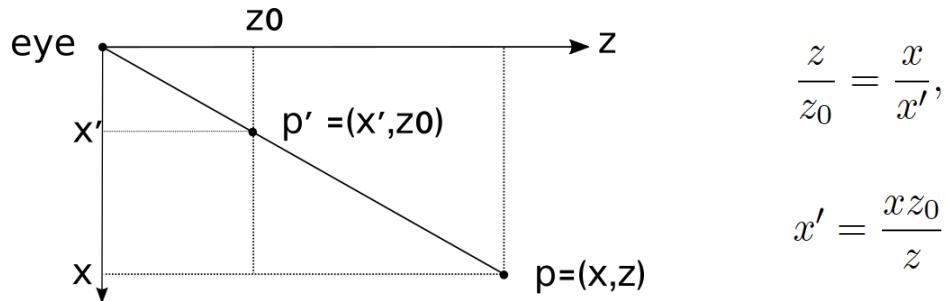
$$V = \begin{pmatrix} 0.857 & 0.514 & 2 \\ -0.514 & 0.857 & 3 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 0.857 & -0.514 & -3,258 \\ 0.514 & 0.857 & 1.543 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{p}_{\text{cam}} = \mathbf{V} \cdot \mathbf{p}_{\text{welt}} = (-1.242, 7.901)$$



# Perspektivische Projektion

- Projektion auf die Bildebene (= z-Near-Ebene)
- Herleitung: Anwendung des Strahlensatzes



- in Matrix-Notation:

$$\begin{pmatrix} x' \\ y' \\ z_0 \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} z_0 & 0 & 0 & 0 \\ 0 & z_0 & 0 & 0 \\ 0 & 0 & z_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Perspektivische Projektionsmatrix:  $\mathbf{P} = \begin{pmatrix} z_0 & 0 & 0 & 0 \\ 0 & z_0 & 0 & 0 \\ 0 & 0 & z_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{z_0} & 0 \end{pmatrix}$

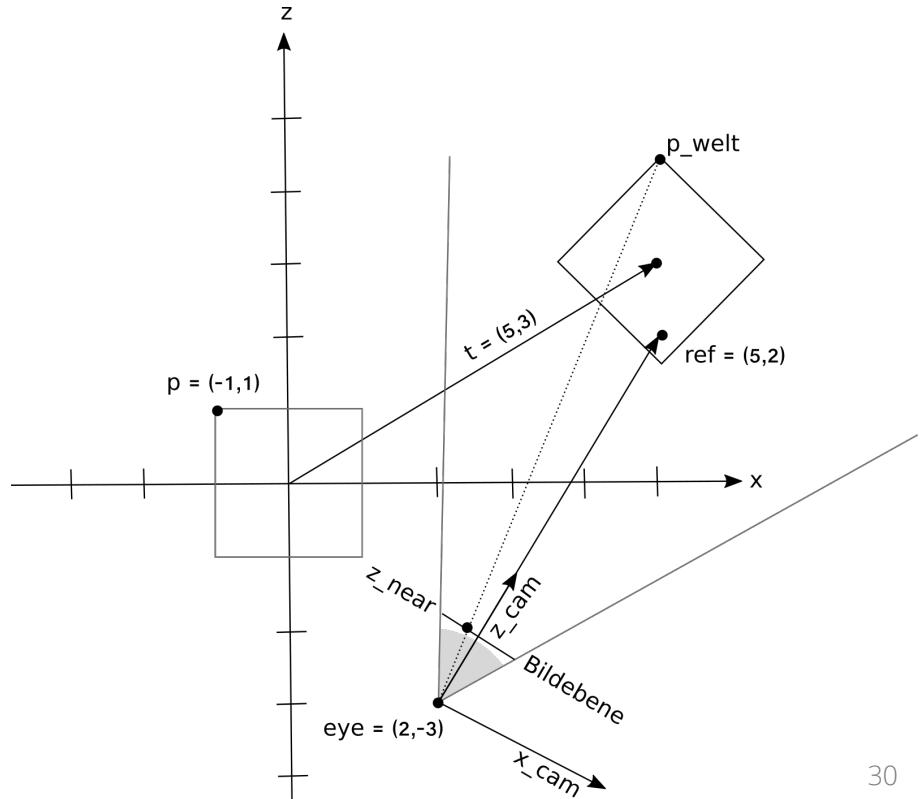
# Perspektivische Projektion

- in unserem Beispiel (homogenes Ergebnis):

$$\mathbf{p}_{\text{bild}} = \mathbf{P} \cdot \mathbf{p}_{\text{cam}} = (-1.242, 7.901, 7.901)$$

- homogene Koordinaten: wenn  $w \neq 0$ , dann Division durch  $w$ :

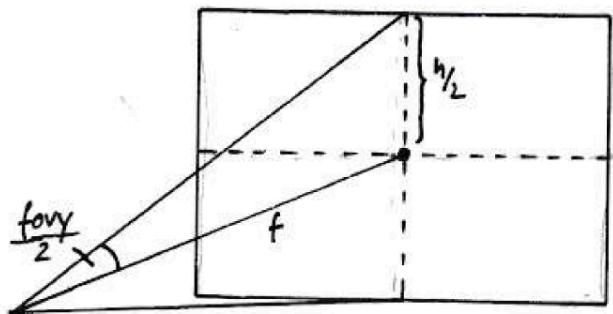
$$\mathbf{p}_{\text{bild}} = (-0.157, 1)$$



# Pixel-Transformation

- finaler Schritt: Umrechnung von der Bildebene auf Pixel-Koordinaten
- basiert auf
  - intrinsische Kameraparameter (Öffnungswinkel)
  - Auflösung der Bildebene in Pixeln ( $w \times h$ )

$$\mathbf{K} = \begin{pmatrix} f & 0 & 0 & w/2 \\ 0 & f & 0 & h/2 \end{pmatrix} \quad (w/2, h/2) = \text{optischer Mittelpunkt}$$



$\text{fov}_x$  = Öffnungswinkel in x-Richtung  
 $\text{fov}_y$  = Öffnungswinkel in y-Richtung

hier:  $f = f_x = f_y$

Bemerkung:  $K$  reduziert die Dimension auf x/y

$$f_x = \frac{w}{2 \cdot \tan(\frac{\text{fov}_x}{2})}$$

$$f_y = \frac{h}{2 \cdot \tan(\frac{\text{fov}_y}{2})}$$

# Pixel-Transformation

- im Beispiel:
  - Auflösung = 1024 x 786 (y-Richtung wird ignoriert)
  - Öffnungswinkel  $\text{fov}_x = 60^\circ$
- damit:

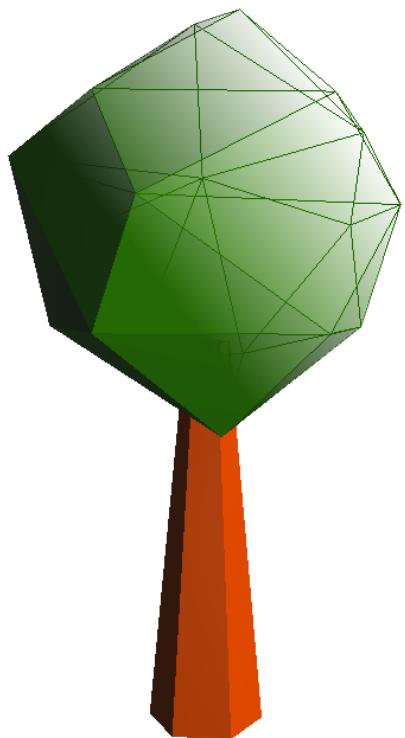
$$\mathbf{K} = \begin{pmatrix} 886.810 & 0 & 0 & 512 \\ 0 & 886.810 & 0 & 384 \end{pmatrix}$$

- und so

$$\mathbf{p}_{\text{pixel}} = \mathbf{K} \cdot \mathbf{p}_{\text{bild}} = (372, 384)$$

# Übung: Kameratransformation

- Gegeben ist der Punkt (1,2,3) im Kamerakoordinatensystem. Bestimmen Sie die Koordinaten des projizierten Punktes auf die Bildebene mit  $z_{NEAR} = 2$



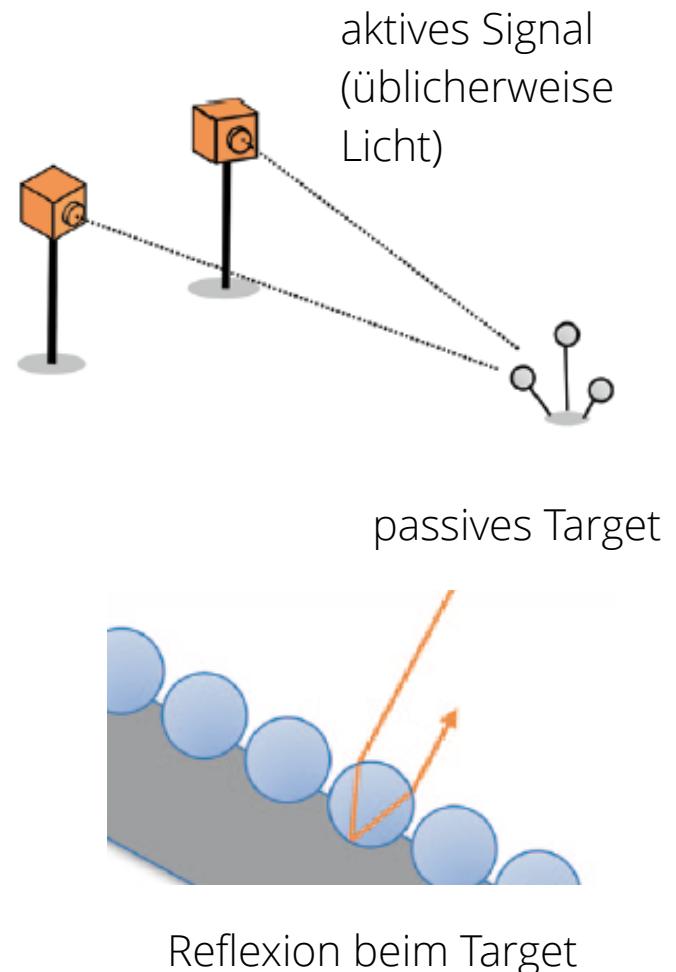
# Tracking

# Tracking

- Target-basiertes Tracking
- Marker-basiertes Tracking
- markerloses Tracking

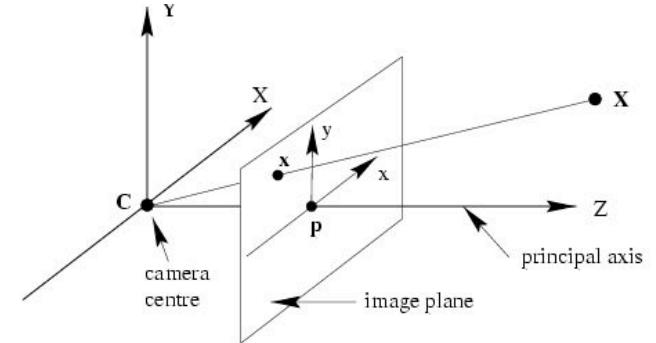
# Target-basiertes Tracking

- bekannte passive Targets
- aktive Sender
  - z.B. Infrarot
- Target reflektiert Signal zurück zum Sender
- Sender ist gleichzeitig Empfänger
  - Aufnahme von Bild mit Reflexionen
  - Positionen der Reflexionen =  $(x,y)$  Bildpunkte



# Kamera-Model

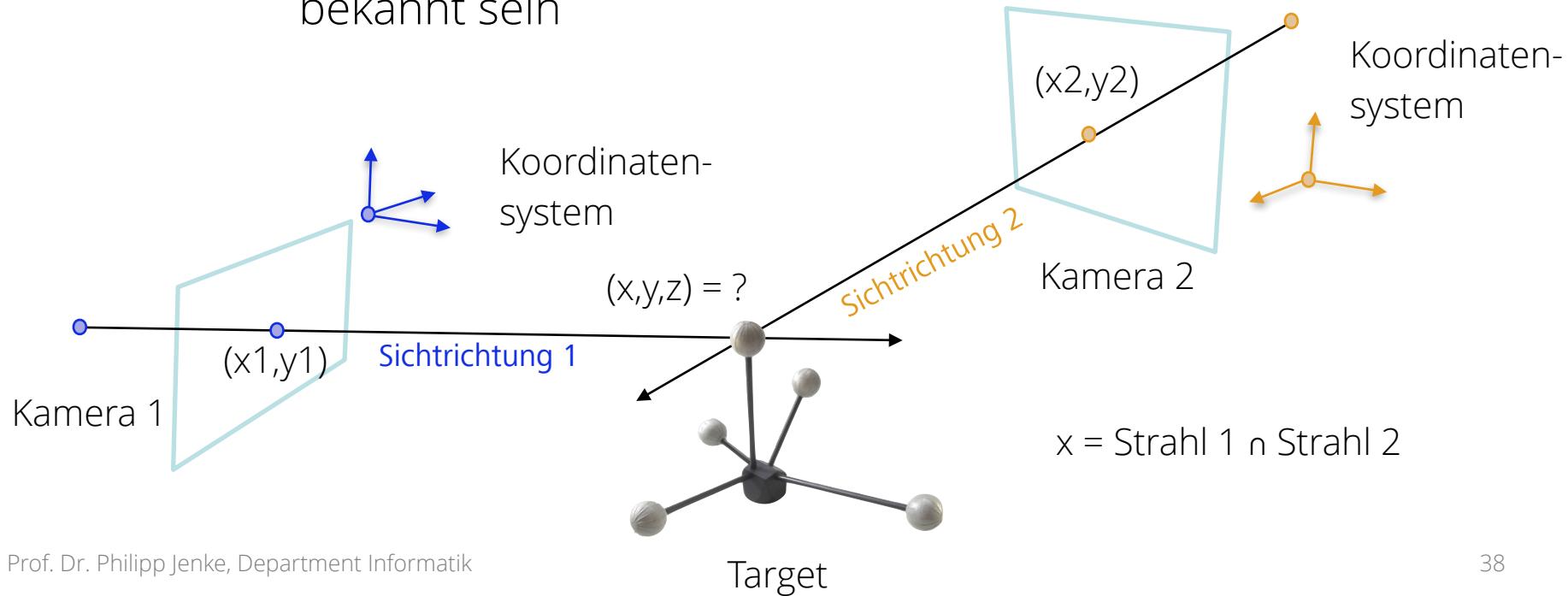
- Lochkamera-Modell
- Hauptachse = Sichtlinie
- Entfernung Auge (C) - Bildebene = focal length
- extrinsische Kameraparameter
  - Position ( $t_x, t_y, t_z$ )
  - Orientierung (Euler-Winkel)
  - extrinsische Kameramatrix
- intrinsische Kameraparameter (einfache Variante)
  - focal length ( $f_x = f_y = f$ )
  - Auflösung:  $w \times h$
  - principal point ( $c_x, c_y$ ) =  $(w/2, h/2)$
  - intrinsische Kameramatrix:



$$\mathbf{K} = \begin{pmatrix} f & 0 & 0 & w/2 \\ 0 & f & 0 & h/2 \end{pmatrix}$$

# Target-basiertes Tracking

- Rekonstruktion der Target-Position aus Reflexionsbildern
- mindestens zwei Kameras notwendig
  - je mehr desto besser
  - System muss kalibriert sein
    - intrinsische and extrinsische Kameraparameter müssen bekannt sein



# Target Position

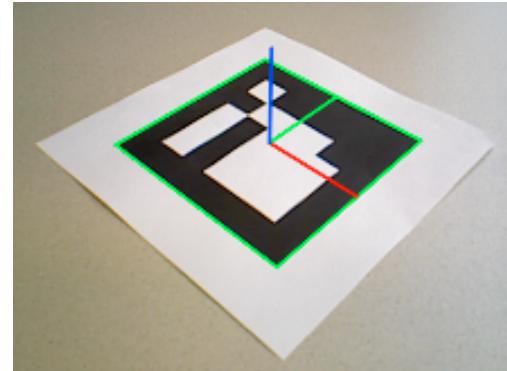
- Kamera 1: 
$$\begin{pmatrix} x_1 \\ y_1 \\ w_1 \end{pmatrix} = K_1 \cdot T_1 \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Kamera 2: 
$$\begin{pmatrix} x_2 \\ y_2 \\ w_2 \end{pmatrix} = K_2 \cdot T_2 \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Auflösen nach der unbekannten Position (x,y,z)

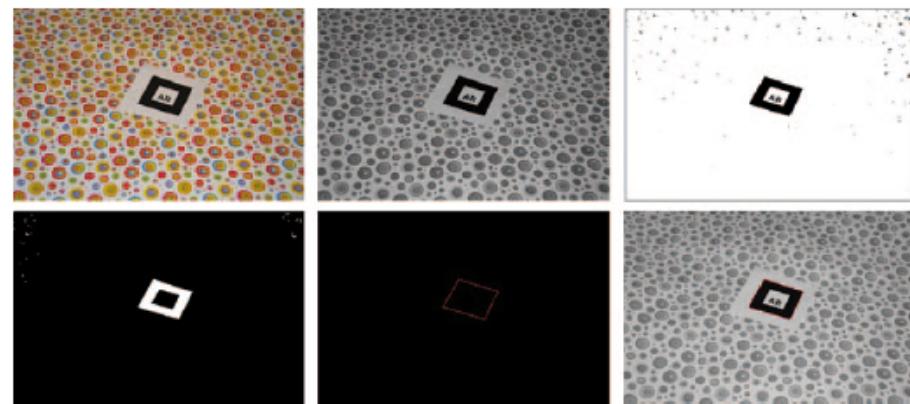
# Marker-basiertes Tracking

- Idee: positioniere Marker in der Szene
- Marker repräsentiert 6-DOF Koordinatensystem
  - Verschiebung (3-DOF)
  - Orientierung (3-DOF)



Aruco Marker

- Tracking
  - Bildaufnahme
  - Kanten- und Eckenerkennung
  - Markererkennung
  - Schätzen der Pose



# Kanten- und Eckenerkennung

- Computer Vision-Problem
- Canny-Algorithmus:
  - Vorverarbeitung: Bildfaltung mit Matrix M

$$M = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- Anwenden des Sobel-Operators S

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

- partielle Ableitung
- x- und y-Richtung
- Kantenrichtung schätzen
- Kantenstärke schätzen

$$\theta = \text{atan}_2(g_y, g_x)$$

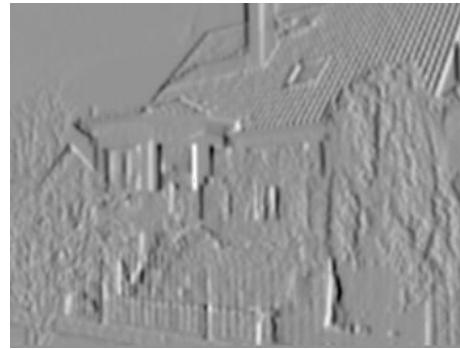
$$G(x, y) = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}$$

- Non-Maximum-Suppression: nur lokale Maxima behalten
- Hysterese: Verfolgung von Startpixel, Schwellwert

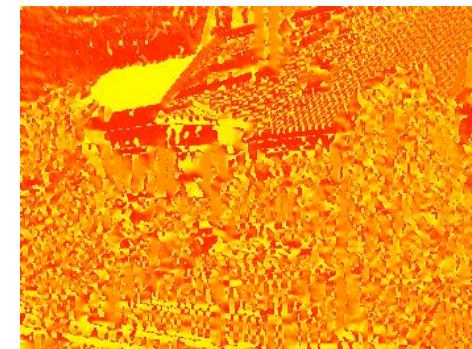
# Canny Kantenerkennung



Vorverarbeitung:  
Bildfaltung



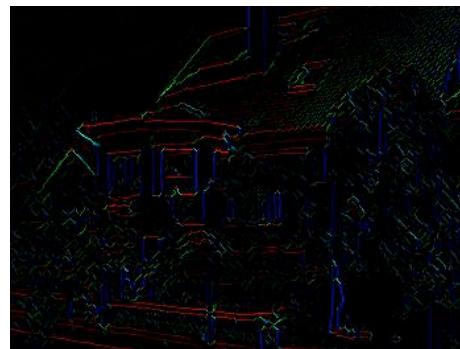
Sobel-Operator



Richtung



Kantenstärke



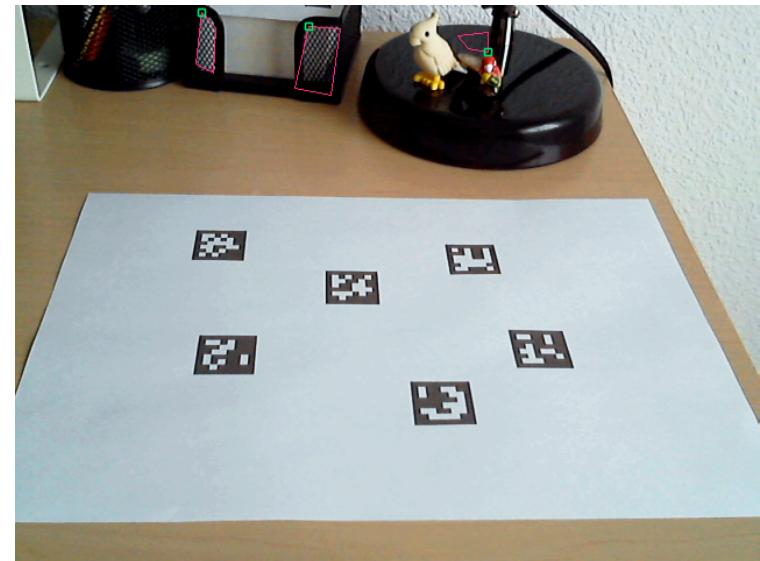
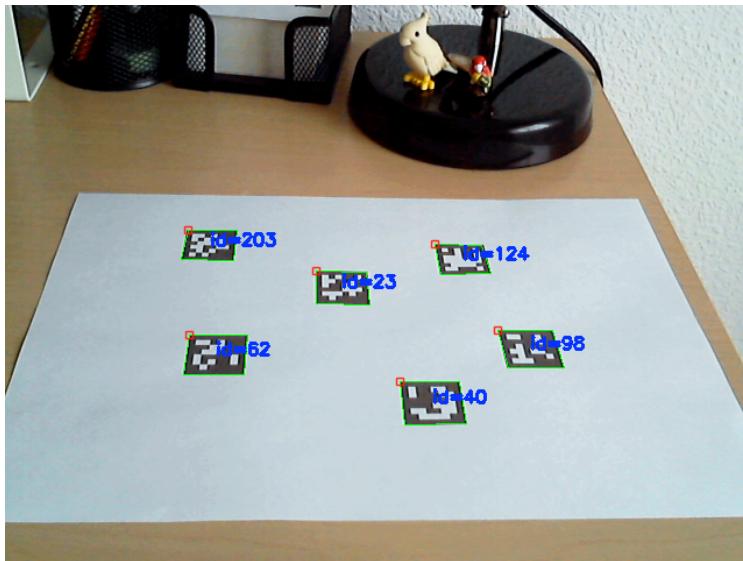
Non-Maximum-  
Suppression



Hysterese

# Erkennen von Vierecken

- Schnitt von Kanten: Ecken
- 4 durch Kanten verbundene Ecken: Kandidat für einen Marker

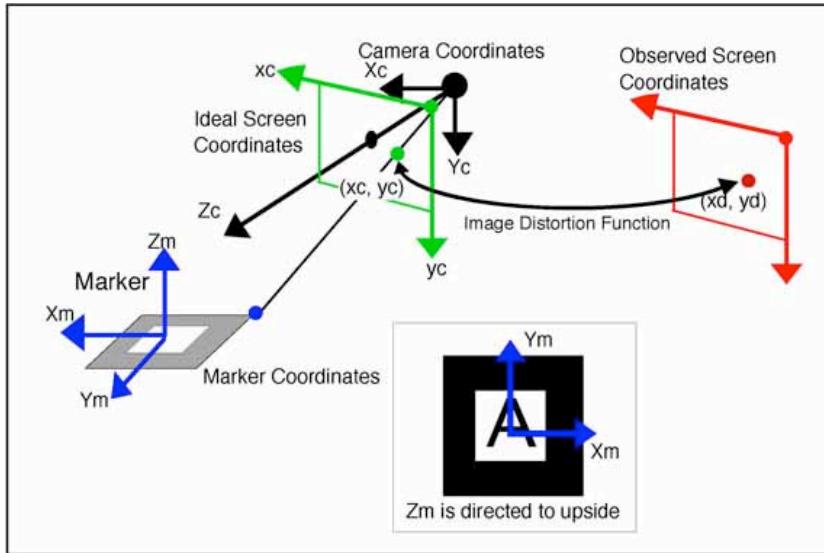


OpenCV Aruco Bibliothek

# Registrierung

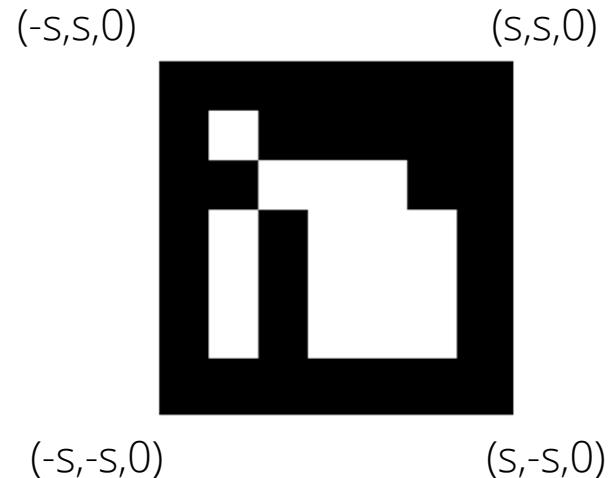
- gegeben:
  - Marker-Ecken ( $x_d, y_d$ )
  - intrinsische Kameraparameter  $K$
  - Kandidaten für Marker-Ecken
- unbekannt: Marker-Pose  $T$  (Position + Orientierung)

$$\mathbf{p}_{\text{pixel}} = \mathbf{K} \mathbf{P} \mathbf{T}^{-1} p$$



Kamera- und Marker Koordinaten-  
system (Quelle: ARToolkit)

Prof. Dr. Philipp Jenke, Department Informatik



Markergröße: 2s

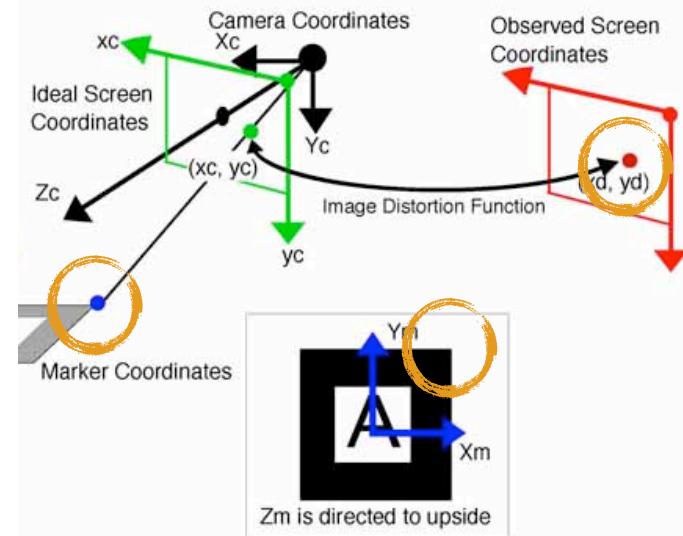
# Registrierung

- Bestimmen der Marker-Pose
- Fehlerfunktion  $f$

$$f = \sum_{i=1}^4 (x_i - \mathbf{KPT}^{-1} c_i)^2$$

- $(x_d^i, y_d^i)$ : Pixel für Ecke  $i$
- $(c_x^i, c_y^i, c_z^i)$ : Eckpunkt im Marker Koordinatensystem
- Lösung: Finden des Minimums der Fehlerfunktion für Parameter
  - Position:  $t_x, t_y, t_z$
  - Orientierung:  $\alpha, \beta, \gamma$

$$\arg \min_{t_x, t_y, t_z, \alpha, \beta, \gamma} f$$



# Markerloses Tracking

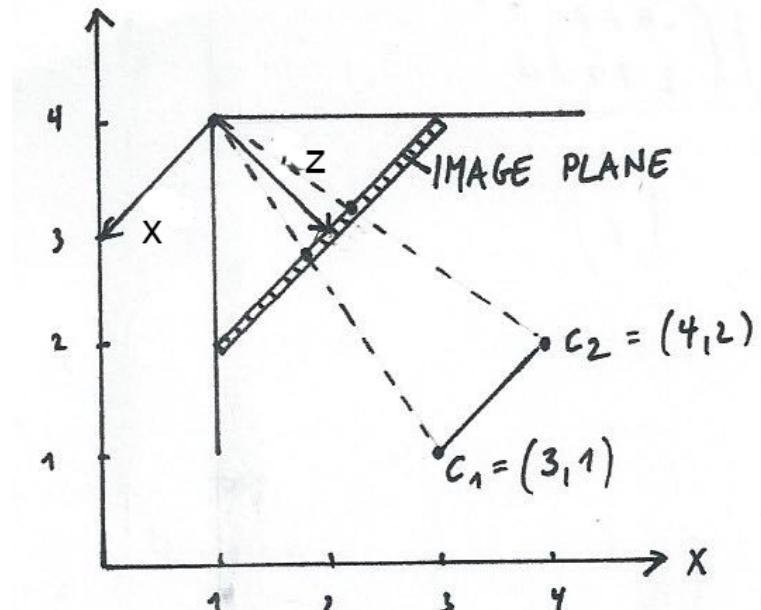
- Informationen zum getrackten Objekt muss vorher bekannt sein
- Finden von markanten Punkten auf dem Objekt (z.B. SIFT, SURF, ...)
- Finden identischer markanter Punkte im Bild, Festlegen von Korrespondenzen
- weitere Details in der einschlägigen Literatur, z.B. [1]

# Übung: Tracking

- es soll die Pose eines Markers in 2D bestimmt werden.
- der Marker hat die Eckpunkte  $(3,1)$  und  $(4,2)$
- die intrinsische Kameramatrix  $K$  lautet

$$K = \begin{pmatrix} 128 & 128 \\ 0 & 1 \end{pmatrix}$$

- die erste Schätzung für die Kamera ist
  - eye =  $(1,4)$
  - z-Richtung:  $(1,-1)$
  - x-Richtung:  $(-1,-1)$
- Wie groß ist der quadratische Fehler der Abschätzung (in Pixeln)



# Zusammenfassung

- Transformationen
- Kamera-Transformationen
- Tracking

# Literatur

- [1] Schmalstieg, Dieter und Höllerer, Tobias. (2016): Augmented Reality - Principles and Practice, Addison Wesley, 1. Auflage, 2016

# Learning Outcomes

Zum erfolgreichen Abschneiden bei der Prüfung sind Sie in der Lage ...

- Vektoren und Matrizen in homogene Darstellung und zurück zu wandeln.
- eine Translation als homogene Matrix darzustellen.
- aus beliebigen Basisvektoren die Transformationsmatrix für eine Koordinatensystem zu erstellen.
- aus den trigonometrischen Funktionen Rotationsmatrizen herzuleiten.
- die Transformationsmatrix zwischen zwei beliebigen Koordinatensystemen aufzustellen.
- die Einzelschritte der Kamera-Transformation (Model-Transformation, View-Transformation, perspektivische Transformation, Pixel-Abbildung herzuleiten).
- Target- und Marker-basiertes Tracking zu erläutern.
- Teilschritte eines Tracking-Verfahrens basierend auf Korrespondenzen durchzuführen.