Kamerageometrie und Perspektive

Institut für Informatik
Mustererkennung und Bioinformatik

Angewandte Bildverarbeitung, SS 2007

Überblick

Die Ursprünge

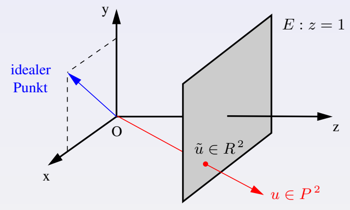
- italienische Renaissance:
  realistische Darstellung der 3D-Welt
- nicht alle Effekte mit euklidischen Räumen R^n erklärbar,
  z.B. der scheinbare Schnitt zweier Parallelen
  $\Rightarrow$ projektive Räume P^n
- P^n erweitert den R^n um *ideale* Punkte/Geraden/Ebenen,
  um Effekte im Unendlichen zu beschreiben
- Grundlage: homogene Koordinaten

Homogene Koordinaten I

- Haupteigenschaft: Invarianz gegenüber Skalierungen

$$u = (x, y, z)^T \sim \lambda (x, y, z)^T \;,\; \lambda \neq 0 \;,\; \lambda \in \mathbb{R}$$

- Punkte im P^n entsprechen $(n+1)$-dim. Vektoren
- Nullvektoren $\vec{0}$ sind ausgeschlossen



Homogene Koordinaten II

- Punkte mit $z = 0$ heissen *ideale Punkte*
- für nicht-ideale Punkte ist Umrechnung zwischen R^n und P^n möglich:

$$
\begin{aligned}
f_{PR}: \quad u = & \ (x, y, z)^T \rightarrow \tilde{u} = (x/z, y/z)^T \\
& \text{mit } (x, y, z)^T \sim (x/z, y/z, 1)^T \\
f_{RP}: \quad \tilde{u} = & \ (x, y)^T \rightarrow u = (x, y, 1)^T
\end{aligned}
$$

Homogene Koordinaten III

- homogene Koordinaten vereinfachen verschiedene mathematische Konzepte, z.B. Abbildungen:

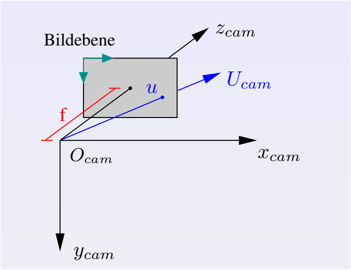
$$A(\tilde{u}) = R \cdot \tilde{u} + \vec{t} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot \tilde{u} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

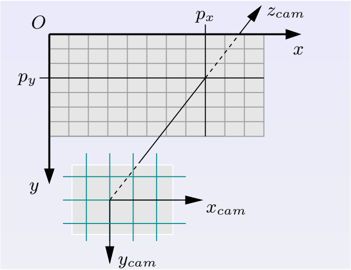
- in homogenen Koordinaten resultiert:

$$A(u) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & t_x \\ \sin(\alpha) & \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot u = \begin{bmatrix} R & \vec{t} \\ \vec{0}^{\,T} & 1 \end{bmatrix} \cdot u$$

Endliche, projektive Kameras I

- Modellierung mit dem Lochkameramodell





- Abbildung eines 3D-Punktes auf die Bildebene:

$$u = K\,[\,I\,|\,\vec{0}\,]\cdot U_{cam}\ ,\quad K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}\ ,\quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Endliche, projektive Kameras II

- Aber:
  - Koordinaten von Bildebene und Kamera nicht achsenparallel
  - Zellen des CCD-Arrays (auf der Bildebene) nicht quadratisch
- Daher:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

  - $x_0 = p_x \cdot m_x \quad , \quad y_0 = p_y \cdot m_y$
  - $\alpha_x = f \cdot m_x \quad , \quad \alpha_y = f \cdot m_y$
  - m_x, m_y sind Anzahlen an CCD-Chips pro Einheitslänge im Kamera-Koordinatensystem
  - s quantifiziert Abweichung des Koordinatensystems von Orthogonalität (s ist meist 0)

Endliche, projektive Kameras III

- Ansatz geht davon aus, dass Kamera- und Welt-Koordinatensystem identisch sind
- wenn mehrere Bilder in Beziehung gesetzt werden sollen, muss das Welt-KS explizit modelliert werden

$$u = K\,[\,I\,|\,\vec{0}\,]\cdot U_{cam} = P\cdot U = K\,R\,[\,I\,|-\tilde{C}\,]\cdot U.$$

  - $\tilde{C}$ ist das Zentrum der Kamera im Welt-KS
  - R beschreibt die Rotation zwischen Welt- und Kamera-KS

- P wird als Kameramatrix bezeichnet:

$$P = K\,[\,I\,|\,\vec{0}\,]\cdot\begin{bmatrix} R & -R\tilde{C} \\ \vec{0}^{\,T} & 1 \end{bmatrix} = K\,R\,[\,I\,|-\tilde{C}\,]$$

Projektive Transformationen

- gegeben zwei Bilder, die mit einer stationären Kamera aufgenommen wurden:

$$P = KR\,[\,I\,|-\tilde{C}\,]\,,\; P' = K'R'\,[\,I\,|-\tilde{C}\,]$$

$$\Rightarrow P \;= (KR)\cdot(K'R')^{-1}\cdot P'$$

- Abbildung eines 3D-Punktes in beide Bilder:

$$u \;=\; P\cdot U \;=\; (KR)\cdot(K'R')^{-1}\cdot P'\cdot U \;=\; (KR)\cdot(K'R')^{-1}\cdot u'$$

- Alternativ:
  Beschreibung durch eine nicht-singuläre 3×3-Matrix

$$u \;=\; (KR)\cdot(K'R')^{-1}\cdot u' = H\cdot u' \quad , \quad H:\mathbb{P}^2\rightarrow\mathbb{P}^2$$

- *H* heisst *projektive Transformation* oder *Homographie*

Rektifizierung I

- Homographien stellen eine Abbildung zwischen 2 Bildern dar
- Aufbau von Homographien:

$$H = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ c_1 & c_2 & 1 \end{bmatrix}$$

- eine Homographie hat 8 Parameter
  $\Rightarrow$ jedes Punktepaar ergibt zwei Constraints
  $\Rightarrow$ vier Punktepaare legen eindeutig eine Homographie fest

$$u'_x = \frac{a_{11} u_x + a_{12} u_y + b_1}{c_1 u_x + c_2 u_y + 1} \quad , \quad u'_y = \frac{a_{21} u_x + a_{22} u_y + b_2}{c_1 u_x + c_2 u_y + 1}$$

Rektifizierung II

- Entzerrung von Bildern:
  transformiere ein beliebiges Viereck in ein Rechteck
- Algorithmus (zur Code-Entzerrung):
  1. bestimme die Eckpunkte p_1, p_2, p_3 und p_4 des Codes
  2. schätze Homographie für die Punktepaare

     $$(p_1, [0,0]^T) \;,\; (p_2, [1,0]^T) \;,\; (p_3, [1,1]^T) \;,\; (p_4, [0,1]^T)$$

     $\Rightarrow$ lineares Gleichungssystem, direkt lösbar mit Gauss
  3. wende die Homographie auf die Coderegion an

Bild-Transformationen

- projektiv (8 DoF):

$$H = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ c_1 & c_2 & 1 \end{bmatrix}$$

- affin (6 DoF):

$$H = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix}$$

- Ähnlichkeit (4 DoF):

$$H = \begin{bmatrix} sr_{11} & sr_{12} & b_1 \\ sr_{21} & sr_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix}$$

- euklidsch (3 DoF):

$$H = \begin{bmatrix} r_{11} & r_{12} & b_1 \\ r_{21} & r_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$\Rightarrow$ wähle Transformation nach Aufnahmebedingungen aus

Kamerakalibrierung

- für Bildtransformationen werden weder intrinsische noch extrinsische Parameter gebraucht
  $\Rightarrow$ Kalibrierung aber für zielgerichtete Bewegungen!
- Ansätze für Kalibrierung:
  - explizit offline, mit Kalibriermuster
  - implizit online, z.B. aus Kamerabewegungen
- explizite Kalibrierung ist im Allgemeinen robuster
- Ziel: Rekonstruktion von P und K

Golden Standard Algorithmus

- geg. mind. 6 Punktkorrespondenzen $\{X_i \leftrightarrow x_i\}$
- Punktnormalisierung:
  mittelwertfrei, mittlerer Abstand vom Ursprung $\sqrt{2}$
- stelle lineares Gleichungssystem $Ap = 0$ auf,
  schätze initiales P mit SVD
- iterative Minimierung des geometrischen Fehlers

  $$\sum_i d(\tilde{x}_i, \tilde{P}\tilde{X}_i)^2$$

  mit Levenberg-Marquardt-Optimierung
- Denormalisierung von P: $P = T^{-1}\tilde{P}U$,
  wobei T und U die Normalisierungsmatrizen sind

Golden Standard Algorithmus II

- die intrinsischen und extrinsischen Parameter resultieren aus einer Zerlegung von P
  - Kamerazentrum:
$$PC = 0$$
  - Orientierung und intrinsische Parameter:
$$P = K[R| - RC] = [M| - MC]$$

$\Rightarrow$ zerlege $M = KR$ mittels RQ-Decomposition

Praxis der Kalibrierung

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

- s ist in der Regel 0
- x_0 und y_0 werden oft ins Zentrum des Bildes gelegt
- α_x und α_y sind bei modernen Kameras meist gleich
- Einheit der Einträge: Pixel

  $\Rightarrow$ Berechnung von Rotationswinkeln
  gemäß Strahlensatz
- Achtung beim Downsamplen!!!

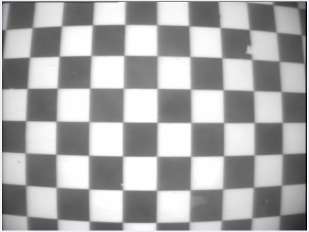
  $\Rightarrow$ Auflösungsreduktion entspricht Brennweitenerhöhung

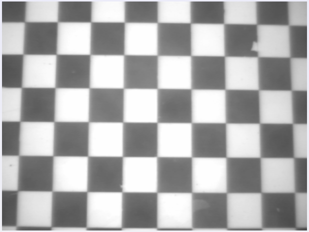
Linsenverzerrungen

- letztes Problem: Unterschied zwischen Theorie und Praxis

$$u = K[I|\vec{0}] \cdot U_{cam} = P \cdot U = K R [I | -\tilde{C}] \cdot U.$$

$\Rightarrow$ Linsen bilden nicht alles linear ab!





- daher oftmals Korrektur der Verzerrungen nötig

Linsenverzerrungen - Modellierung

- Linsenverzerrungen verursachen eine zusätzliche Bildtransformation
- 2 Arten von Verzerrungen:
  a) radiale Verzerrungen:
  $$x' = x \cdot (1 + \kappa_2 r^2 + \kappa_4 r^4 + \dots)$$
  b) tangentiale Verzerrungen:
  $$x' = 2p_1 xy + p_2(r^2 + 2x^2)$$
  - $r^2 = x^2 + y^2$ (Abstand vom Verzerrungszentrum)
- in der Praxis tangentiale Verzerrungen meist ignoriert (Anteil gering und Schätzung numerisch instabil)

Linsenverzerrungen - Korrektur

- 3 prinzipielle Vorgehensweisen:
  - explizit, mit Kalibriermuster
  - implizit, aus Bildstrukturen
  - implizit, aus Bewegung
- auch hier gilt: explizite Kalibrierung bevorzugt (aber manchmal geht das nicht. . .)
- Aufwand der Kalibrierung hängt vom Modell ab
- bei zu vielen Parametern Gefahr des Overfitting

  $\Rightarrow$ zumeist maximal 2 Parameter

Explizite Kalibrierung

- Augangspunkt: Kalibriermuster
  - Eckendetektion
  - Identifikation von Geraden aus Eckpunkten
  - Schätzung einer Transformation, die die Geraden / das Gitter ”gerade” biegt
- Ansatz führt auf nicht-lineare Gleichungssysteme

  $\Rightarrow$ Lösung mit nicht-linearer, iterativer Optimierung

Implizite Kalibrierung aus Bildstrukturen

- Augangspunkt:

  finde spezifische Bildstrukturen, die entzerrt werden sollen (gerade Linien, Kreise, Ellipsen)
  - detektiere Bildprimitiva
  - Schätzung einer Transformation,
    die die Geraden / das Gitter "gerade" biegt
- Ansatz führt auf nicht-lineare Gleichungssysteme

  $\Rightarrow$ Lösung mit nicht-linearer, iterativer Optimierung
- Detektion der Bildprimitiva oft manuell

  $\Rightarrow$ Woher weiss man, ob ein Liniensegment
  wirklich gerade sein muss. . . ?

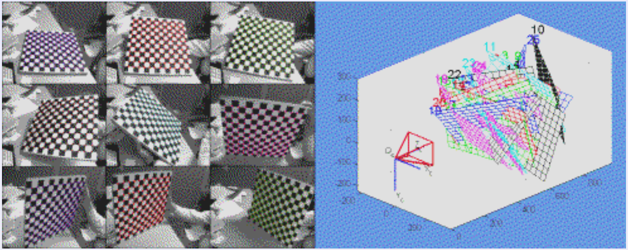
Implizite Kalibrierung

- keine Annahmen über Szenenstruktur
- Aber: Bewegung der Kamera
- Ansatz:
  schätze Bewegung (Homographie) **und** Verzerrung
- brandheisses Forschungsthema, weil ...
  - es viele Anwendungsbereiche gibt
  - es keine offensichtliche Lösung gibt
  - Homographien und Verzerrungen nicht zwingend klar trennbar sind

Fazit

Wenn möglich, explizit kalibrieren, nur wenn nötig implizit!

Camera Calibration Toolbox for Matlab



- freie Erweiterung für Matlab
- integrierte Kalibrierung *aller* Parameter
- Eingangsdaten sind Bilder eines Schachbretts
- manchmal manuelle Eckendetektion notwendig

Werbung in eigener Sache. . . Projektarbeit

- Funktionalität der Matlab-Toolbox steckt auch in der OpenCV

  $\Rightarrow$ Kalibrierungstool unter Linux / Qt
- Ziele / Aufgaben:
  - graphische Oberfläche für die Funktionen der OpenCV
  - Visualisierung von (Zwischen-) Ergebnissen
  - (möglichst) vollautomatische Kalibrierung

    $\Rightarrow$ Verbesserung / Flexibilisierung der Eckendetektion
  - ggf. Erstellung eines Kalibriermusters
- Beginn: ab sofort möglich