



MuS2: Pose Estimation

W. Woeber

Manipulieren von erkannten Objekten

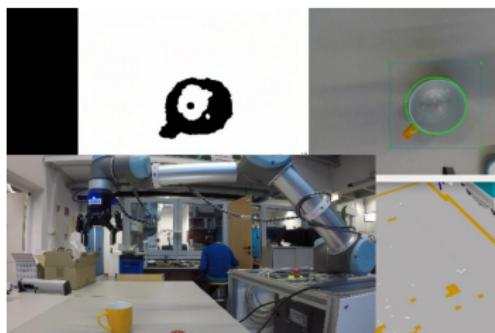
- Bisher können wir:
 - Objekte/Merkmale erkennen (SIFT oder durch machine learning)
 - Objekte/Merkmale finden (z.B.: Brute force matcher)
- Zum Manipulieren (z.B.: Greifen) **fehlt** noch
 - 3D Transformation zum Objekt (bisher nur in Bildkoordinaten)
 - Transformation von Kamera zum Objekt ${}^O_C\mathbf{T} = {}^C_O\mathbf{T}^{-1}$
 - Diese können wir für Bahnplanung nutzen
- Hier diskutieren wir die Berechnung von ${}^O_C\mathbf{T}$
 - Stereovision: Vermessung des Objekts
 - Kameramodel und Aufstellung von Gleichungssystemen
 - ${}^O_C\mathbf{T}$ Schätzung: Vereinfachung, klassische Pose Estimation, RANSAC

Ein einfaches Beispiel

Ein "klassisches" Computer Vision Ansatz

- Stereokamera liefert RGB und Sterobild
- Erkennen des Objekts bzw. Greifpunkts im linken Bild
- Mit Tiefeninformation und erkannte Features ${}^o_C\mathbf{T}$ berechnen
- Roboterbewegung durch Transformation berechnen

- Links oben: Segmentiertes Disparitätsbild
- Rechts oben: Objektdetektion
- Unten: Aufbau



Vorgehen zur Berechnung von ${}^O_C T$

Was muss vorhanden sein, was wird geschätzt?

- 3D Koordinaten am Objekt
 - Vermessene Punkt oder CAD Files
 - Achtung:** Die Punkte sind im Objekt-Koordinatensystem vermessen.
 - Nutzung von Stereovision o.ä.
- Korrespondierende 2D Koordinaten
 - Für 3D Punkte am Objekt Koordinaten am Bild
 - z.B.: SIFT Features oder Harris Ecken
 - Set an Markierungen (z.B.: Logos in bestimmter Anordnung)
- Ausgangssituation ist daher
 - 2D Abbildung der 3D Information ist bekannt
 - Kameras sind kalibriert und Bilder sind rektifiziert
- Wir berechnen: ${}^O_C T$ basierend auf Punktpaaren

Vorgehen zur Berechnung von ${}^O_C T$

Inhalt dieses Moduls

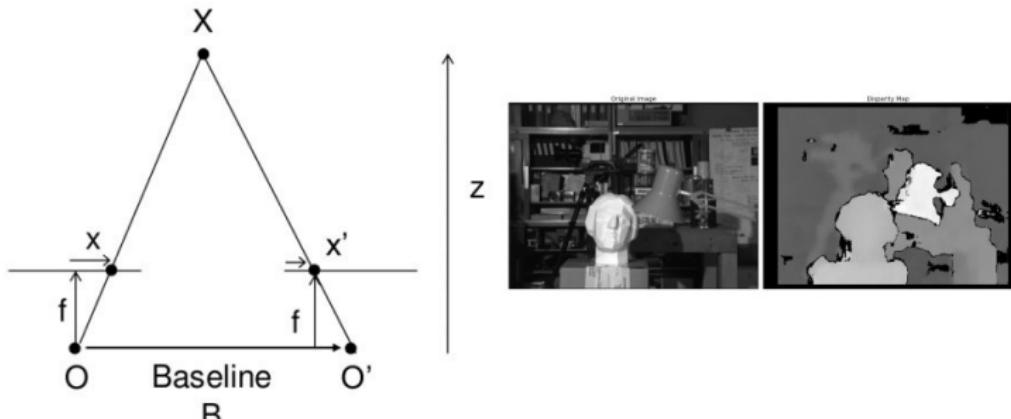
- Stereovision (Wiederholung) und ein Beispiel
- Kameramodel (Wiederholung) und Problembeschreibung
- Lösung des Pose Estimation Problems
 - Vereinfachung für "flache" und einfache Objekte
 - Aufstellen/Lösen der Gleichungssysteme (Lineare Regression)
 - RANSAC
- Beispiele

Stereo Vision

Wiederholung¹

- Ausgangslage
 - (mindestens) zwei Kameras
 - Starrer Aufbau
 - Synchronisierter Bilderstream
 - Kalibrierinformation: Extrinsisch/intrinsisch

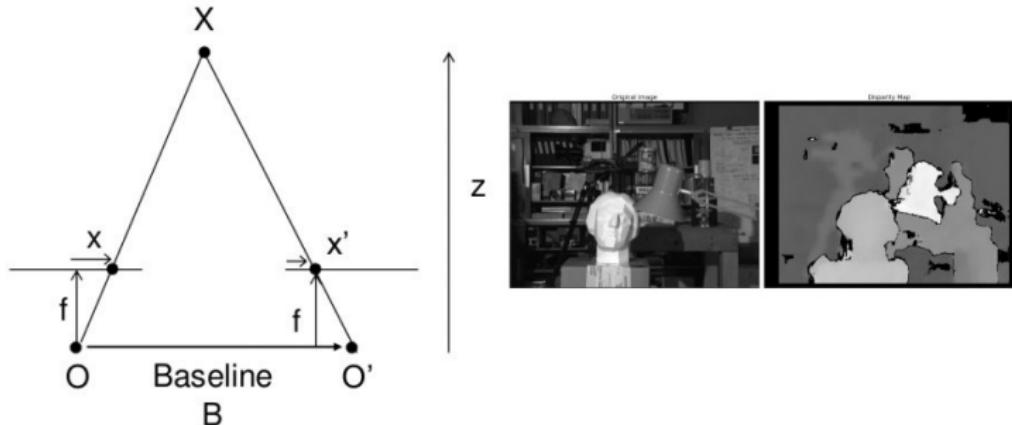
(siehe MMR1 - Advanced Sensor Systems)



Stereo Vision

Grundlagen 1/4

- Durch Kalibrierung
 - Beide Kameras Fokuslänge f
 - Baseline
- Abstand zum Objekt X ist Z
- Darstellung am Kamerachip bei x bzw. x'

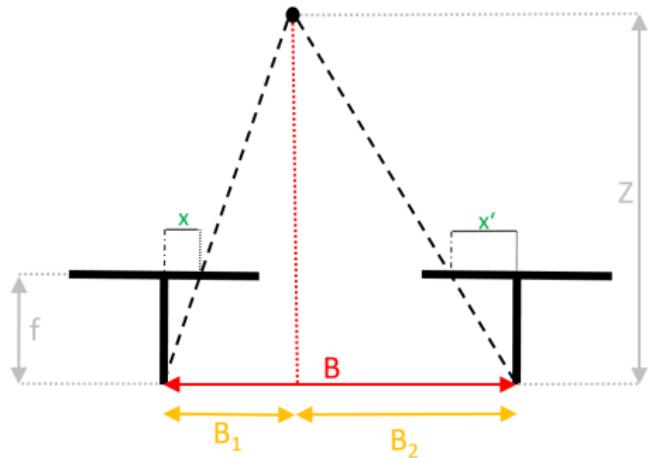


Stereo Vision

Grundlagen 2/4

Herleitung durch ähnliche Dreiecke (Ähnlichkeitssatz)

f ... Fokuslänge
 B ... Baseline
(Abstand zwischen Kameras)
 Z ... Abstand zu Punkt
 x und x' ... Punkte in Bild



Stereo Vision

Grundlagen 3/4

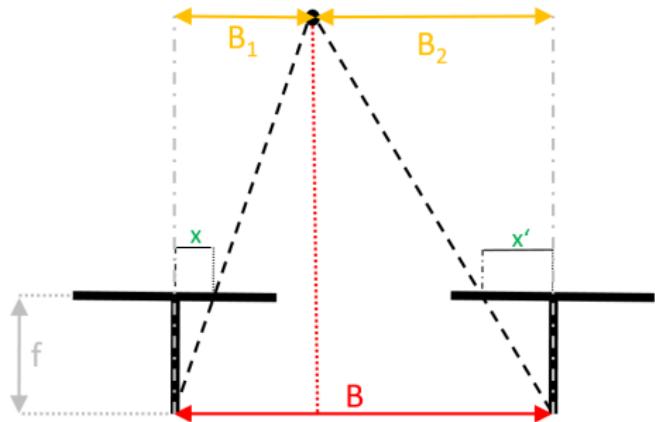
$$\frac{B_1}{Z} = \frac{x}{f}$$

$$B_1 = \frac{x}{f} Z \quad (1)$$

$$\frac{B_2}{Z} = \frac{x'}{f} = \frac{B - B_1}{Z}$$

$$B_1 = B - \frac{x'}{f} Z \quad (2)$$

f ... Fokuslänge
 B ... Baseline
 (Abstand zwischen Kameras)
 Z ... Abstand zu Punkt
 x und x' ... Punkte in Bild



Stereo Vision

Grundlagen 4/4

Demnach durch Gleichsetzen

$$\begin{aligned}
 \frac{x}{f}Z &= B - \frac{x'}{f}Z \\
 Z \left(\frac{x}{f} + \frac{x'}{f} \right) &= Z \left(\frac{x+x'}{f} \right) = B \\
 Z &= \frac{B \cdot f}{x + x'} \\
 Z &= \frac{B \cdot f}{D} \tag{3}
 \end{aligned}$$

Mit der Disparität $D = x + x'$

Achtung: Disparität manchmal anders formuliert. Wird x' vom Chipmittelpunkt berechnet ist die Disparität $D = x - x'$.

Stereo Vision

Offenen Punkte

- Kalibrierung (nicht Teil dieser ILV)
- Berechnung Disparität
 - Muss für alle Punkte im Bild gemacht werden
 - Eine Möglichkeit: Vergleich von "Patches"
 - Durch Kalibrierung: Vergleich entlang der Epipolarlinie
Abbildung von Punkt X ist in beiden Bildern in der selben Zeile anzutreffen.

→ Simple Blob Matcher



Meistens bezieht sich die Stereovision auf das linke Kamerabild (siehe linkes Bild in der Abbildung). Rechts ist das Disparity-Bild dargestellt. Versuchen Sie folgende Frage zu beantworten: Warum sind nahe Objekte hell und entfernte Objekte dunkel.

Simple Blob Matcher

Idee

- Für Punkte im linken Bild suchen wir Korrespondenz im rechten
 - Hilfe: Epipolarlinie
 - Definieren einen "Blob"
 - Größe des Blobs in Pixel
Große "Blobs" bewirken ein glatteres Disparitätsbild **aber**
Information geht verloren. Kleine "Blobs" bewirken verrausachte
Disparitätsbilder **aber** mehr Information.
 - Suchbereich im rechten Bild: minimale Disparität und
Disparitätsbereich. Diese Werte definieren den Suchraum
entlang der Epipolarlinie.
- Nachbearbeitung ("smoothing") des Disparitätsbild wird
immer durchgeführt

Simple Blob Matcher

Visualisierung



Roter Kreis: beliebiger Punkte im Bild (links) und korrespondierender Punkt (rechts)

Schwarzer Kreis: Linke Bildposition im rechten Bild

Weiße Linie: **Disparität**

Gelbe Linie: Epipolarlinie

Blauer Kasten: Suchbereich: minimale Disparität bzw. Disparitätsbereich

Simple Blob Matcher

Berechnung Disparitätsbild

- Für jedes Pixel suchen des korrespondierenden Punktes
Aufgrund der nicht kompletten Überlappung nicht für jedes Pixel möglich.
- Berechnung der Pixeldifferenz
- Nicht überall möglich
 - Konturarme Flächen
 - Verdeckungen



Simple Blob Matcher²

Beispiel Verdeckung

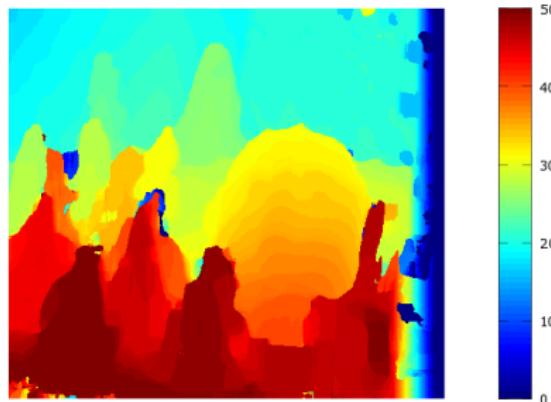


Anmerkung: Die schwarzen streifen im Disparitätsbild (oben rechts) basieren auf den begrenzten Überlappungsbereichen der Bilder bzw. den definierten "Suchfenster" Dimensionen.

² Abbildungen aus <https://www.intelrealsense.com/stereo-depth-vision-basics/>

Simple Blob Matcher³

Disparitätsbild



- Umrechnung in Tiefe mit $Z = \frac{Bf}{D}$
- Typisch: Arbeit mit Disparity Bild oder 3D *point clouds*

³ Bild übernommen aus <http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/>

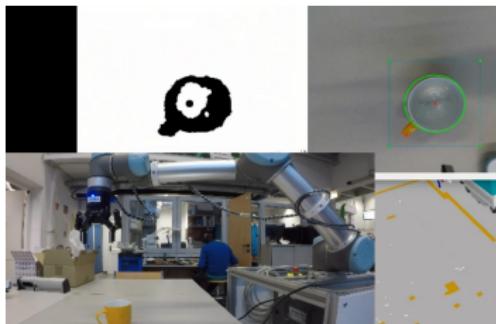
Stereo Vision: Beispiel

1. Detektion des Bechers via CNN (YOLO v.3)
2. Roboter über Becher bewegen
3. Stereovision und Kreiserkennung für Berechnung des Greifpunkts (Erinnern sie sich an Hough Circles)

Pose Estimation wurde vereinfacht (leider nicht immer möglich)

- Becher parallel zu Kamerachip
- Rotation via Griff des Bechers berechnet
- Distanz via Stereovision

In dieser Applikation gab es einige Vereinfachungen. Der Becher steht auf einer Fläche parallel zum Welt-Koordinatensystem. Die Tischhöhe ist bekannt. Ein Tiefenschnitt des Bechers ist ein Kreis - was sehr einfach zum Detektieren.



Herleitung Pose Estimation und allgemeines Vorgehen

- Herleitung des Problems: das Kameramodell
- Lösungsansatz 1: Vereinfachen
- Lösungsansatz 2: Aufstellen eines Gleichungssystems
- Lösungsansatz 3: RANSAC

Das Lochkameramodell

Grundbegriffe

- Berechnung der 3D Transformation: Projektion auf Bildebene
- ${}^O_C T$ von Kamera zu Objekt
(bzw. ${}^C_O T = {}^O_C T^{-1}$)
- Ausgangslage
 - Punkte am Objekt vermessen
 - Korrespondierende Punkte im Bild

Wichtige Kenngrößen:

$(u \quad v)$... Punkt auf Bildebene

$(x \quad y \quad z)$... Punkt in Weltkoordinatensystem

Typischerweise bezieht man sich hier auf einen Punkt am Objekt

Z ... Tiefe von Kamera-Projektionszentrum zu Punkt

f ... Fokuslänge

$(c_x \quad c_y)$... Chipmittelpunkt

Das Lochkameramodell

Kalibrierung ⁴

■ Intrinsische Kalibrierung

Bestimmung der Parameter zur Linsenentzerrung und montagebedingter Kenngrößen

- Linsenverzerrung: radial $k_{1,2,3}$ und tangential $p_{1,2}$
- Fokuslänge: f_x und f_y
- Bildhauptpunkte (Chipmittelpunkt): c_x und c_y

Fokuslänge und Bildhauptpunkte müssen wegen bedingter Genauigkeit der Montage (Chip verkleben) berechnet werden.

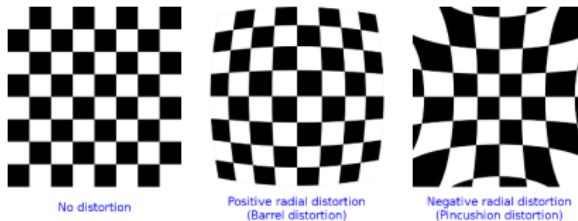
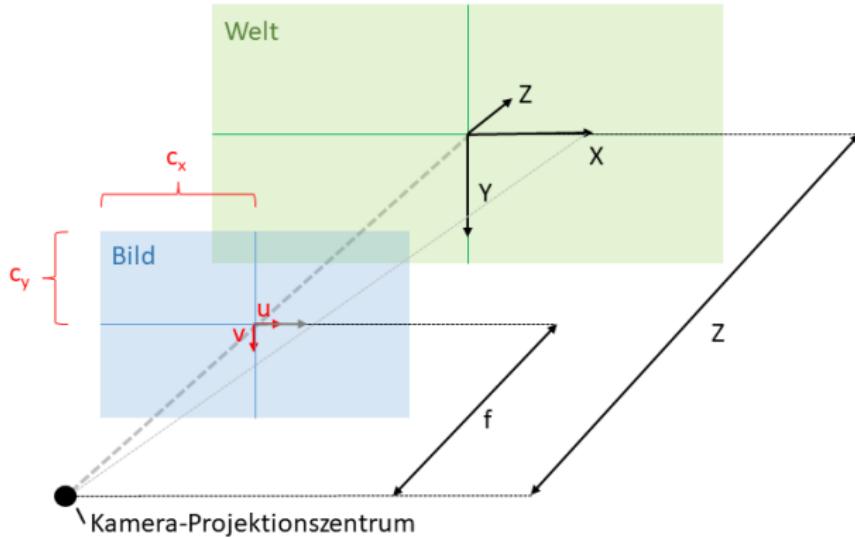


Abbildung aus https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

⁴ Siehe MMR1: Advanced Sensor Systems

Das Lochkameramodell

Visualisierung des Models



Projektion von Punkten

Von Koordinaten im Bild zu realen Koordinaten

Keine Verzerrung und Chip optimal platziert
Punkt am Chip berechenbar durch:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} x \\ y \end{pmatrix}$$
$$\begin{pmatrix} \frac{u}{f} \\ \frac{v}{f} \end{pmatrix} = \begin{pmatrix} \frac{x}{Z} \\ \frac{y}{Z} \end{pmatrix} \quad (4)$$

Basis: Ähnliche Dreiecke

Leider müssen wir den durch Kalibrierung bekannten Chipversatz und demnach die neue Fokulänge einbinden.

Projektion von Punkten

Einbeziehen von Chipversatz

Versatz durch $(c_x \ c_y)^T$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \underbrace{\begin{pmatrix} c_x \\ c_y \end{pmatrix}}_{\text{"Schieben" zum Mittelpunkt Ähnliche Dreiecke}} \underbrace{\frac{1}{z_c} \begin{pmatrix} f_x x_c \\ f_y y_c \end{pmatrix}}_{(5)}$$

und in matrixschreibweise:

$$\begin{pmatrix} uz_c \\ vz_c \\ z_c \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \quad (6)$$

z_c ... Abstand von Projektionszentrum zu Chip

Durch das Problem der Mehdeutigkeit: Skalierungsfaktor s (siehe
MMR1: Advanced Sensor Systems)

Projektion von Punkten

Darstellung mit Transformationsmatrix

Daraus ergibt sich folgende Gleichung zur Projektion eines Punkts:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{smallmatrix} O \\ C \end{smallmatrix} \mathbf{T} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} O \\ C \end{smallmatrix} \mathbf{R} \begin{bmatrix} O \\ C \end{smallmatrix} \vec{t} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (7)$$

Wir suchen erhalten $\begin{smallmatrix} O \\ C \end{smallmatrix} \mathbf{T}$!

Sie sehen, dass prinzipiell auch \mathbf{K} berechnet werden kann.

Lösung durch Vereinfachung 1/3

Anpassung des Problems

Folgende Annahmen können wir treffen

- Objektkoordinatensystem ist **nicht** rotiert
- Translation vom Kamera- zum Objektkoordinatensys. in X und Y ist 0.
- Objekt ist flach und somit sind die Z Koordinaten am Objekt immer 0
- Kamera ist kalibriert: $c_x = c_y = 0$ und Fokalängen sind bekannt

$$\begin{aligned}
 s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} &= \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{o} \\ \mathbf{R} \\ \vec{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \\
 \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \\ \vec{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} &= \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I}_x \mathbf{I}_y \mathbf{I}_z \\ \vec{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (8)
 \end{aligned}$$

Lösung durch Vereinfachung 2/3

Anpassung des Problems

$$\begin{aligned}
 \dots &= \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \\
 &= \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{bmatrix} f_x & 0 & 0 & f_x t_x \\ 0 & f_y & 0 & f_y t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f_x X + f_x t_x \\ f_y Y + f_y t_y \\ t_z \end{bmatrix} \quad (9)
 \end{aligned}$$

Wegen $c_x = c_y = 0$ und $t_x = t_y = 0$ folgt

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} f_x X \\ f_y Y \\ t_z \end{pmatrix} \quad (10)$$

Lösung durch Vereinfachung 3/3

Anpassung des Problems

Sie sehen nun, dass wir basierend auf den Annahmen ein Verhältnis gefunden haben: (Das gleiche gilt für Y)

$$s = t_z$$

$t_z u = f_x X$... wir dividieren

$$\frac{u}{f_x} = \frac{X}{t_z} \tag{11}$$

Wir kommen also wieder zur Gleichung für ähnliche Dreiecke!

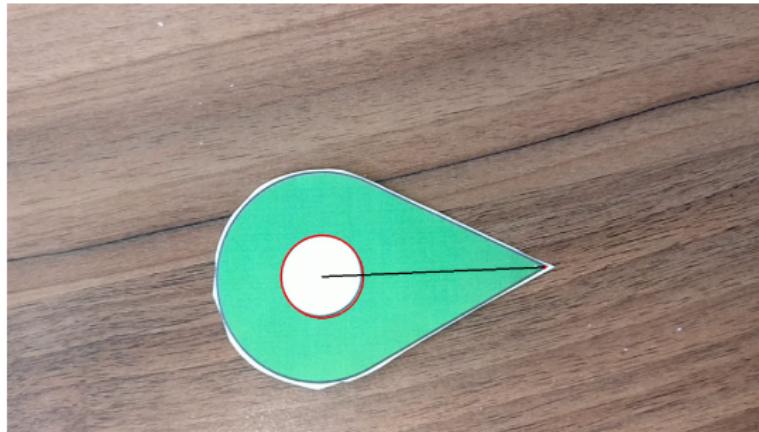
Bedenken Sie die Randbedingungen, die wir angenommen haben. Die halten in der Realität leider nicht immer. Je nach Randbedingung, die in der Realität "fällt" müssen komplexe Methoden das Gleichungssystem für die Projektion lösen.

Tiefenschätzung für flache Objekte Intuitive Lösung: Geometrische Betrachtung $$\frac{d_l}{f} = \frac{D_l}{Z}$$ $$\frac{d_r}{f} = \frac{D_r}{Z}$$ Demnach: $$\frac{d_l Z - D_l f}{f Z} = \frac{d_r Z - D_r f}{f Z}$$ $$\frac{d_l - d_r}{f} = \frac{D_l - D_r}{Z} \quad (12)$$ $$\frac{\text{Breite}_{\text{Bild}}}{f} = \frac{\text{Breite}_{\text{real}}}{Z}$$ The diagram shows a horizontal blue bar representing an object of width D_l at distance D_l from the camera. The camera is represented by a vertical line with a lens at the bottom. The image of the object is a smaller blue bar of width d_l at distance f from the lens. The image distance f is indicated by a vertical double-headed arrow on the right. The object distance D_l is indicated by a horizontal double-headed arrow below the object. The object width D_l is indicated by a horizontal double-headed arrow above the object. The image width d_l is indicated by a horizontal double-headed arrow below the image. The diagram illustrates that the image width d_l is proportional to the object width D_l and inversely proportional to the object distance D_l . Warum nur bei flachen, zur Kamera parallelen Objekten? Hier sind die selben Annahmen wie bei der bereits besprochenen Vereinfachung getroffen worden. Überlegen Sie was sich in den Gleichungen ändert, wenn einzelne Annahmen nicht halten. © 2019 FH Technikum Wien, Wilfried Woeber 28

Tiefenschätzung für flache Objekte

Geometrische Betrachtung 1/3

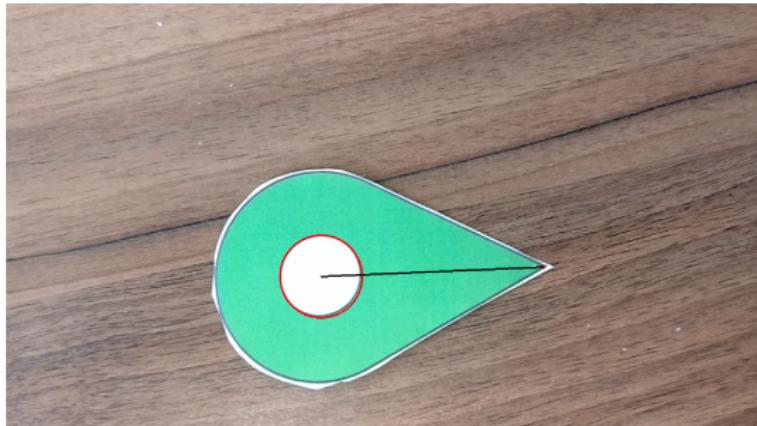
- Gegeben: Radius innerer Kreis, Fokusröße, Objektlänge
- Gesucht: X,Y Koordinate der Greifpunkte (linke Ecke)



Tiefenschätzung für flache Objekte

Geometrische Betrachtung 2/3

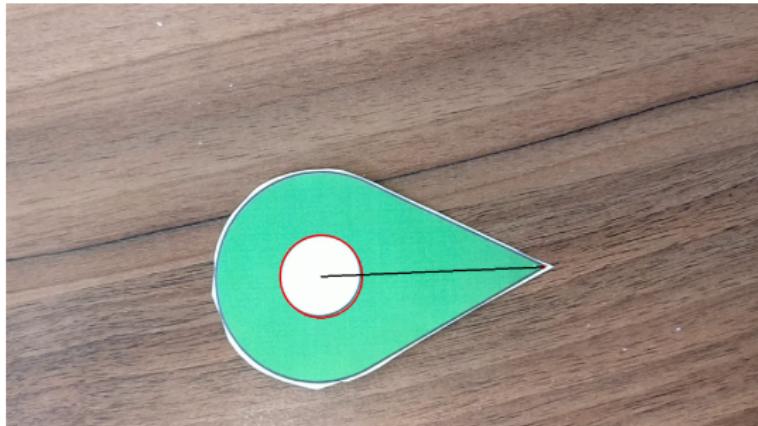
- Schätzen von Z durch Formel 12
- Hough Circle liefert Kreispunkt und Radius
- Ecke erkennen
 - Reihensumme: Y Koordinate (Extremwert)
 - Spaltensumme: X Koordinate Ecke
 - Alternative: Konturen-Toolbox



Tiefenschätzung für flache Objekte

Geometrische Betrachtung 3/3

- Rückrechnung Eckkoordinate in 3D Koordinaten
(siehe Formel 12)
- Aufpassen:
 - Einige Vorgaben (Hough Circle, ExG)
 - Begrenzte Genauigkeit
 - Sehr schnell



Klassische Pose Estimation

Vorgehen für eine allgemeine Lösung

- Was wenn keine Vereinfachungen möglich sind?
Wir haben Vereinfachungen bezüglich Orientierung und Translation des Koordinatensystems angenommen. Das wollen wir nun nicht mehr machen.
- In der Praxis wollen/können wir auf keine Vereinfachungen bauen
- Jetzt beschäftigen wir uns mit Lösungsansatz 2 und 3
 - Ausgangslage: Set an bekannten Punkten
Projektion von 3D Punkt auf 2D Bild
 - Das Gleichungssystem hält für jeden Punkt
 - Ziel: Parameter finden (das beinhaltet die Transformation)
 - Ansatz: Formulierung eines Gleichungssystem für alle bekannte Punkt

Klassische Pose Estimation

Formulierung Problemstellung

Folgende Gleichung **bleibt** bestehen:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} {}^o \mathbf{T} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Wir haben korrespondierende Punkte:

$$\left\{ \left[\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} \right], \dots, \left[\begin{pmatrix} u_n \\ v_n \end{pmatrix}, \begin{pmatrix} X_n \\ Y_n \\ Z_n \end{pmatrix} \right] \right\}$$

Was hat das für Auswirkungen auf das Gleichungssystem?

Klassische Pose Estimation

Direct Linear Transform 1/2

Transponieren wir beide Seiten:

$$(su \quad sv \quad s \quad 0) = (X \quad Y \quad Z \quad 1) (\mathbf{K}_C^O \mathbf{T})^T \quad (13)$$

Demnach für alle Punkte:

$$\begin{bmatrix} su_1 & sv_1 & s & 0 \\ \vdots & & & \\ su_n & sv_n & s & 0 \end{bmatrix} = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 \\ \vdots & & & \\ X_n & Y_n & Z_n & n \end{pmatrix} (\mathbf{K}_C^O \mathbf{T})^T \quad (14)$$

Das ist lineare Regression! Das können wir:

Wiederholung aus Modul 3

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\Theta \quad (\text{Für mehrdimensionalem Output}) \\ \rightarrow \Theta &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} \end{aligned} \quad (15)$$

Pose Estimation

Direct Linear Transform 2/2

Wir können $\mathbf{K}_C^O \mathbf{T}$ aus den korrelierenden Punkten berechnen
→ Das Verfahren nennt man "Direct Linear Transform" (DLT)
Probleme damit:

- (fast) lineare Optimierung: kein Kontext
- **Wir** wissen wo '0' hin müssen: DLT optimiert blind
- Wir wollen Regression mit Kontext

Neuer Ansatz:

- Kontext: $\mathbf{K}_C^W \mathbf{R}$ hat nur drei Freiheitsgrade - nicht neun
 - Iterative Projektion auf Bild
- Levenberg-Marquardt Optimization (LMO)

RANSAC: Die Alternative zu DLT und LMO

Idee, Implementierung und Anwendung

- **RAN**dom **S**ample **C**onsensus
- Tool zur Auffindung von Parameter
(siehe Gleichungssystem der korrelierenden Punkte)
- **Idee: wenig Daten und dadurch schnell**
Gegenteil von machine learning oder Optimierung wie DLR oder L-M Optimierung

RANSAC

- Minimale Anzahl an Samples
z.B.: 2 für $kx + d$
- Definition von Toleranz ϵ
- Zählen: Inlier (in Toleranz)
und Outlier
- N -mal probieren

Machine Learning

- Nutzen aller Samples
Aufteilen in Train/Test/Eval Sets
- Fehler minimieren
- Modelparameter "lernen"
- Geschlossen lösbar

RANSAC

Definitionen ⁵

Basis ist wieder die Projektion

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K}_C^O \mathbf{T} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} {}_C^O \mathbf{R} \vec{t} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (16)$$

Idee

- Zufällige n Punkte wählen
- Berechnung: $\mathbf{K}_C^O \mathbf{T}$ oder ${}_C^O \mathbf{R} \vec{t}$
- Dann alle Punkte evaluieren

- Definition von N (Versuche)
- Definition einer Toleranz ϵ
- Definition eines Thresholds τ (Consensus)

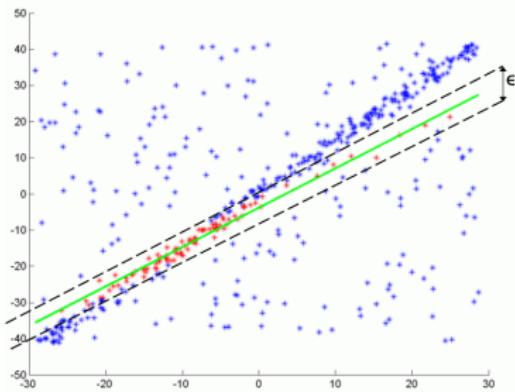
⁵ Mehr Details: http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf

RANSAC

Ein simples 2D Problem

- Model: $y = kx + t$
- Nötige Parameter: k, d
- Definition von ϵ, τ
- Nötige Punkte pro Runde: 2
- Pro Zyklus:
 1. Zufällige Wahl von 2 Samples
 2. Parameter fitten
 3. Samples in ϵ (Inliers)
 4. Berechnen $\frac{\text{Inlier}}{\#Samples}$
 5. Bruch $> \tau$: Abbruch
 6. Ansonsten Weiter bei (1)

Nach Abbruch oder Ablauf der N -Zyklen werden die Parameter mit allen Inliers für bestes Model neu berechnet.



RANSAC

Ein simples 2D Problem: Visualisierung

Vorteile

- Schnell
- Gute Initialisierung (z.B.: für DLT)
- Kann gut mit Noise umgehen

Nachteile

- Muss keine gute Lösung sein
- Abhängig von Noise

solvePnP: Use Cases 1/3

Wie, Was, Warum

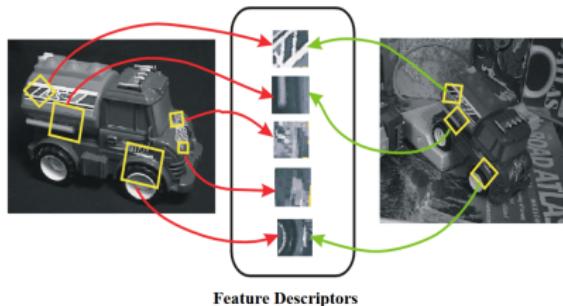
- Initialer Datensatz: Muss aufgenommen werden
 - Set an 3D Punkte: $(x \ y \ z)^T$
 - Set an korrespondierende Descriptoren: $\vec{\mathcal{D}}$

$$Train = \left\{ \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_1, \vec{\mathcal{D}}_1 \right], \dots, \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_n, \vec{\mathcal{D}}_n \right] \right\} \quad (17)$$

- Aus neuem Bild
 - Descriptoren im Bild: $\vec{\mathcal{D}}^*$
 - Korrespondierende Position im Bild: $(u \ v)^T$
- Descriptoren können gematched werden

solvePnP: Use Cases 2/3

Deskriptoren matchen



- Ergebnis: Set an zusammengehörenden Deskriptoren
- Demnach auch: zusammengehörende 3D und Bildkoordinaten
Aus Trainingsset
- Ausgangslage für DLT oder RANSAC

solvePnP: Use Cases 3/3

Transformation berechnen

- Set an 3D und Bildkoordinaten

$$\left\{ \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_1, \begin{pmatrix} u \\ v \end{pmatrix}_1 \right], \dots, \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_k, \begin{pmatrix} u \\ v \end{pmatrix}_k \right] \right\} \quad (18)$$

- Transformation berechnen (z.B.: RANSAC)
- 3D Rückprojektion (z.B.: Bounding Box) mit ${}^O_C \mathbf{T} = {}^C_O \mathbf{T}^{-1}$

