

Advanced Robot Perception

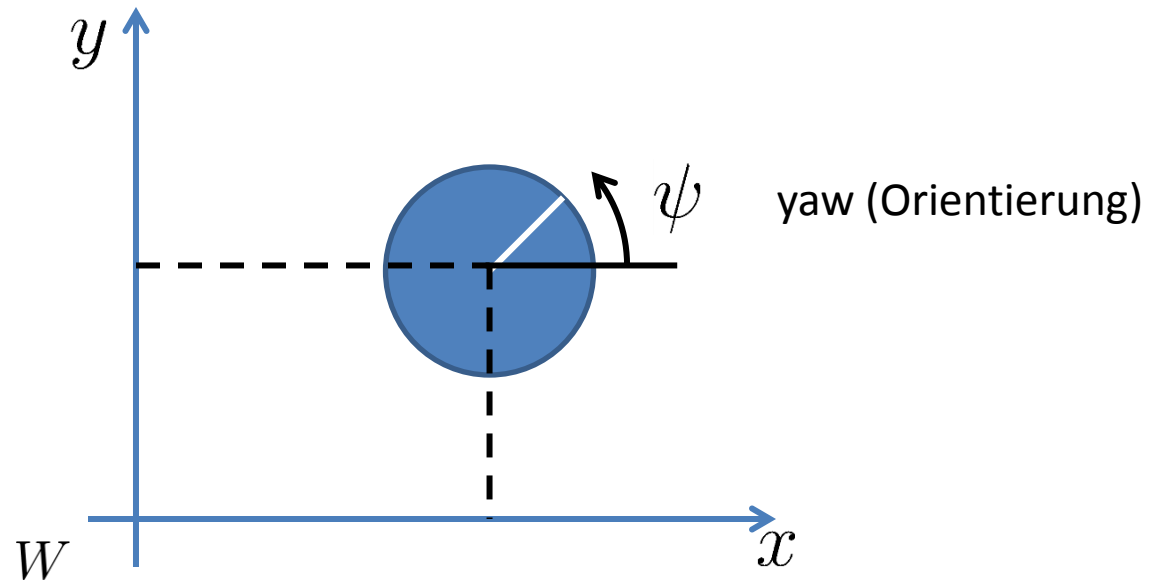
Fortgeschrittene Konzepte der Wahrnehmung für
Robotersysteme

Georg von Wichert, Siemens Corporate Technology

BEISPIEL: VERWENDUNG EUKLIDISCHER TRANSFORMATIONEN IN DER ROBOTIK

Koordinatensysteme

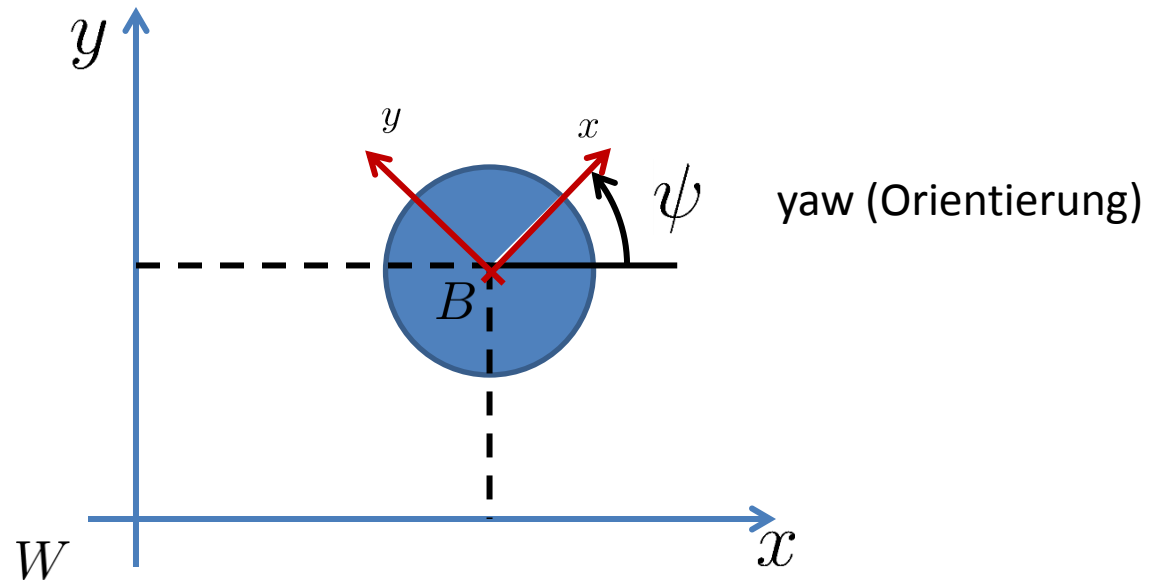
- Ein Roboter irgendwo in der Ebene



Koordinatensysteme

- Ein Roboter irgendwo in der Ebene

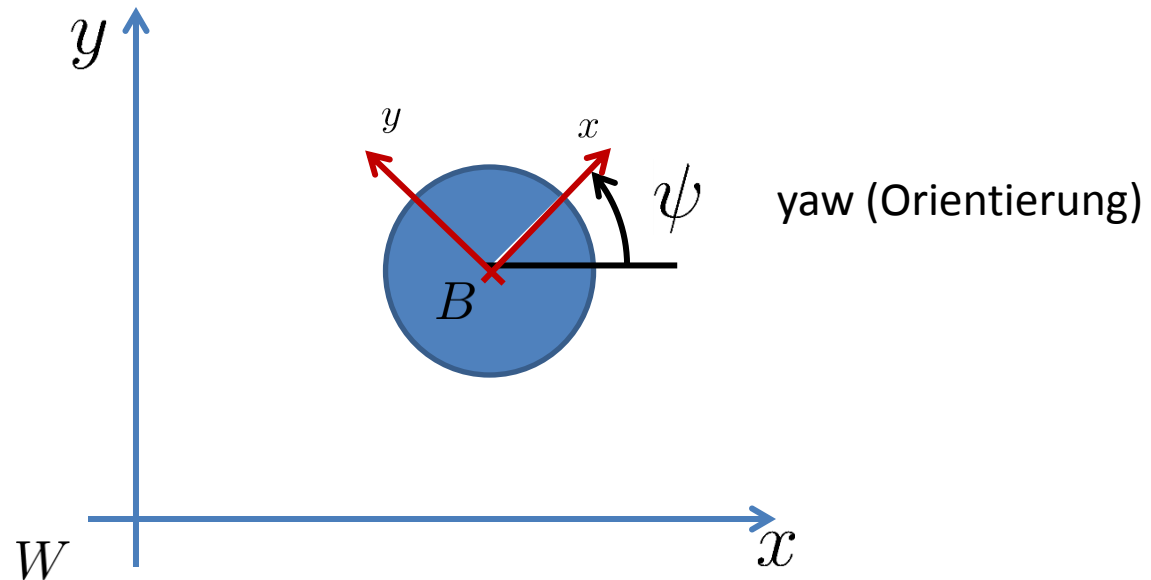
$${}^W T_B = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos \psi & -\sin \psi & x \\ \sin \psi & \cos \psi & y \\ 0 & 0 & 1 \end{pmatrix} \in \text{SE}(2) \subset \mathbb{R}^{3 \times 3}$$



Koordinatensysteme

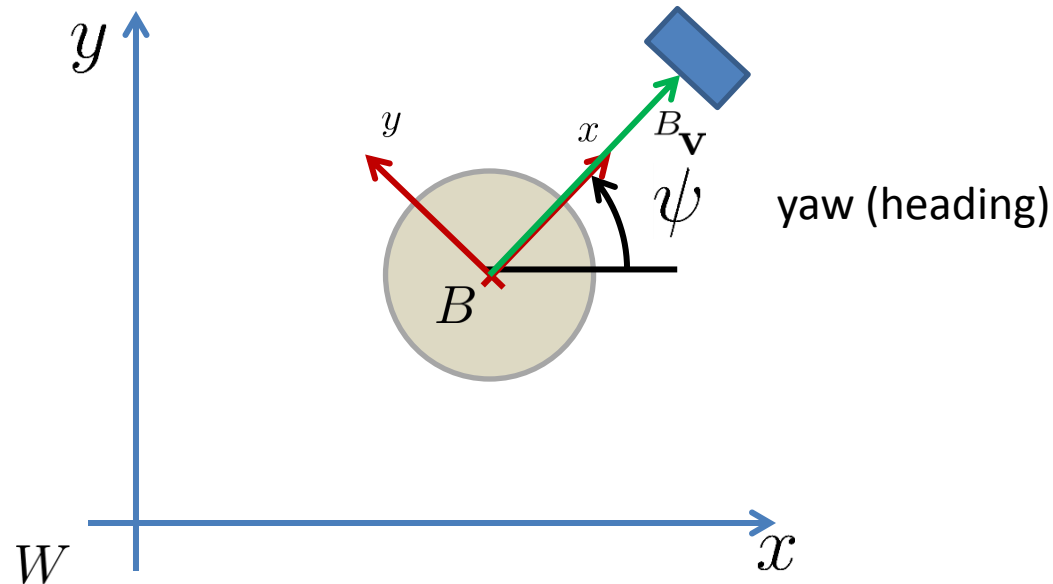
- Roboter an der Stelle $x=0.7, y=0.5, \text{yaw}=45\text{deg}$

$${}^W T_B = \begin{pmatrix} \cos 45 & -\sin 45 & 0.7 \\ \sin 45 & \cos 45 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$$



Koordinatentransformation

- Roboter an der Stelle $x=0.7$, $y=0.5$, $\text{yaw}=45^\circ$
- Roboter hat einen Sensorausleger von 1m Länge
- Was ist die Position des Sensors in W -Koordinaten?



Koordinatentransformation

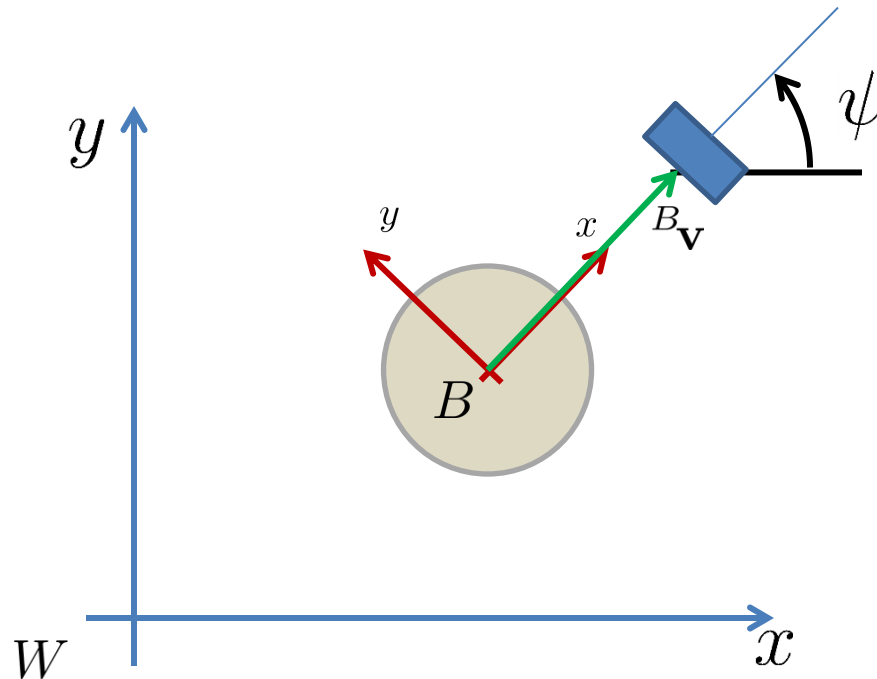
- Roboter an der Stelle $x=0.7$, $y=0.5$, $\text{yaw}=45^\circ$
- Roboter hat einen Sensorausleger von 1m Länge

Inhomogene Koordinaten

$$B_{\mathbf{V}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Homogene Koordinaten

$$B_{\tilde{\mathbf{V}}} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$



Koordinatentransformation

- Roboter an der Stelle $x=0.7$, $y=0.5$, $\text{yaw}=45\text{deg}$
- Roboter detektiert Objekt 1m voraus

$${}^W\tilde{\mathbf{v}} = {}^WT_B {}^B\tilde{\mathbf{v}}$$

Koordinatentransformation

- Roboter an der Stelle $x=0.7$, $y=0.5$, $\text{yaw}=45^\circ$
- Roboter hat einen Sensorausleger von 1m Länge

$$\begin{aligned} {}^W\tilde{\mathbf{v}} &= {}^WT_B {}^B\tilde{\mathbf{v}} \\ &= \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

Koordinatentransformation

- Roboter an der Stelle $x=0.7$, $y=0.5$, $\text{yaw}=45^\circ$
- Roboter hat einen Sensorausleger von 1m Länge

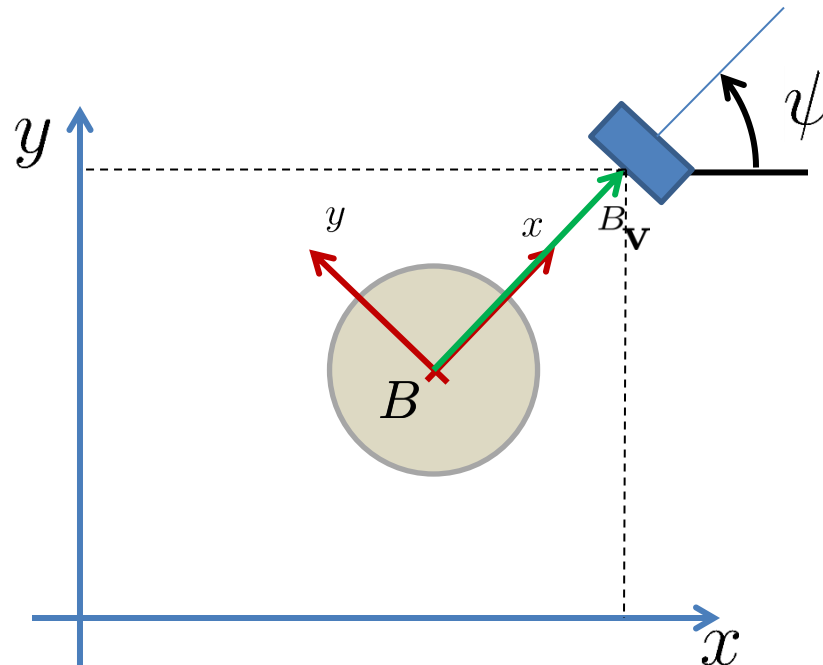
$$\begin{aligned} {}^W\tilde{\mathbf{v}} &= {}^WT_B {}^B\tilde{\mathbf{v}} \\ &= \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1.41 \\ 1.21 \\ 1 \end{pmatrix} \end{aligned}$$

Koordinatenransformation

- Roboter an der Stelle $x=0.7$, $y=0.5$, $\text{yaw}=45^\circ$
- Roboter hat einen Sensorausleger von 1m Länge

$${}^B\tilde{\mathbf{v}} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$${}^W\tilde{\mathbf{v}} = \begin{pmatrix} 1.41 \\ 1.21 \\ 1 \end{pmatrix}$$



Inverse Transformation

- Wir haben Roboter- in Weltkoordinaten transformiert
- Manchmal muss man das Umgekehrte tun
- Wie transformiert man Welt- in Roboterkoordinaten?

$${}^W\tilde{\mathbf{v}} = {}^WT_B {}^B\tilde{\mathbf{v}}$$

Inverse Transformation

- Wir haben Roboter- in Weltkoordinaten transformiert
- Manchmal muss man das Umgekehrte tun
- Wie transformiert man Welt- in Roboterkoordinaten?

$${}^W\tilde{\mathbf{v}} = {}^WT_B {}^B\tilde{\mathbf{v}}$$

$${}^B\tilde{\mathbf{v}} = {}^BT_W {}^W\tilde{\mathbf{v}}$$

Inverse Transformation

- Wir haben Roboter- in Weltkoordinaten transformiert
- Manchmal muss man das Umgekehrte tun
- Wie transformiert man Welt- in Roboterkoordinaten?

$${}^W\tilde{\mathbf{v}} = {}^WT_B {}^B\tilde{\mathbf{v}}$$

$${}^B\tilde{\mathbf{v}} = {}^BT_W {}^W\tilde{\mathbf{v}}$$

$${}^B\tilde{\mathbf{v}} = \left({}^WT_B\right)^{-1} {}^W\tilde{\mathbf{v}}$$

Inverse Transformation

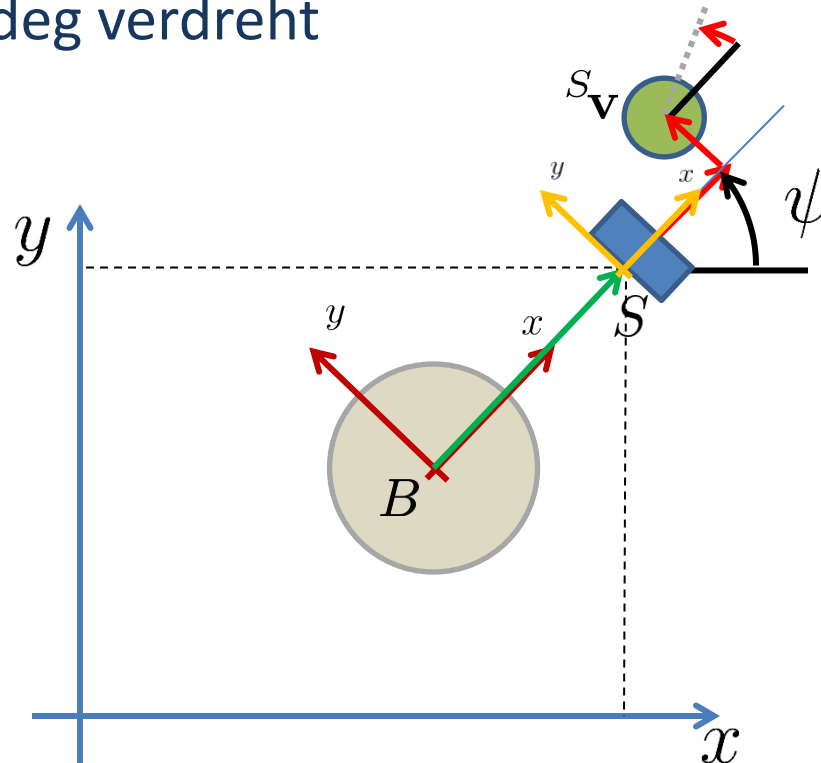
- Wir haben Roboter- in Weltkoordinaten transformiert
- Manchmal muss man das Umgekehrte tun
- Wie transformiert man Welt- in Roboterkoordinaten?

$${}^W\tilde{\mathbf{v}} = {}^WT_B {}^B\tilde{\mathbf{v}} = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} {}^B\tilde{\mathbf{v}}$$

$${}^B\tilde{\mathbf{v}} = ({}^WT_B)^{-1} {}^W\tilde{\mathbf{v}} = \begin{pmatrix} R^\top & -R^\top \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} {}^W\tilde{\mathbf{v}}$$

Verkettung von Transformationen

- Gegeben: Objekt wird von Sensor gesehen
 - Position relativ zum Sensor: 0.2m in x-Richtung, 0.1m in y-Richtung, 10deg verdreht



Verkettung von Transformationen

- Position relativ zum Sensor:
0.2m in x-Richtung, 0.1m in y-Richtung, 10deg
verdreht

$${}^B T_S = \begin{pmatrix} \cos 10 & -\sin 10 & 0.2 \\ \sin 10 & \cos 10 & 0.1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.98 & -0.17 & 0.2 \\ 0.17 & 0.98 & 0.1 \\ 0 & 0 & 1 \end{pmatrix}$$

Verkettung von Transformationen

- Die Position des Objekts in W-Koordinaten?

$$\begin{aligned} {}^W T_S &= {}^W T_B {}^B T_S = \\ &= \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.98 & -0.17 & 0.2 \\ 0.17 & 0.98 & 0.1 \\ 0 & 0 & 1 \end{pmatrix} = \dots \end{aligned}$$

Verkettung von Transformationen

Beachte: Auf die Reihenfolge kommt es an!!!

- 1m gehen, 90 Grad drehen
- 90 Grad drehen, 1m gehen

$$AB \neq BA$$

3D-Transformationen

- Translation


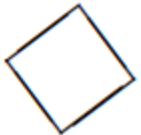
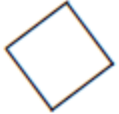

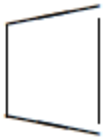
$$\bar{\mathbf{x}}' = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}}_{4 \times 4} \bar{\mathbf{x}}$$

- Euklidische Transformation (Translation + Rotation),

$$\bar{\mathbf{x}}' = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \bar{\mathbf{x}}$$

- Ähnlichkeitstransformation, Affine Transformation, ...

3D-Transformationen

| Transformation | Matrix | # DoF | Preserves | Icon |
|-------------------|---|-------|----------------|---|
| translation | $\left[\begin{array}{c c} \mathbf{I} & \mathbf{t} \end{array} \right]_{3 \times 4}$ | 3 | orientation |  |
| rigid (Euclidean) | $\left[\begin{array}{c c} \mathbf{R} & \mathbf{t} \end{array} \right]_{3 \times 4}$ | 6 | lengths |  |
| similarity | $\left[\begin{array}{c c} s\mathbf{R} & \mathbf{t} \end{array} \right]_{3 \times 4}$ | 7 | angles |  |
| affine | $\left[\begin{array}{c} \mathbf{A} \end{array} \right]_{3 \times 4}$ | 12 | parallelism |  |
| projective | $\left[\begin{array}{c} \tilde{\mathbf{H}} \end{array} \right]_{4 \times 4}$ | 15 | straight lines |  |

Euklidische Transformation in 3D

- Translation \mathbf{t} hat 3 Freiheitsgrade
- Rotation R hat 3 Freiheitsgrade

$$X = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D-Rotationen

- Rotationsmatrix
- Euler-Winkel
- Rodriguez-Darstellung (Drehachse / Winkel)
- Einheitsquaternionen

Rotationsmatrix

- Orthonormale 3x3 Matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

- Spaltenvektoren entsprechen den Koordinatenachsen

Rotationsmatrix

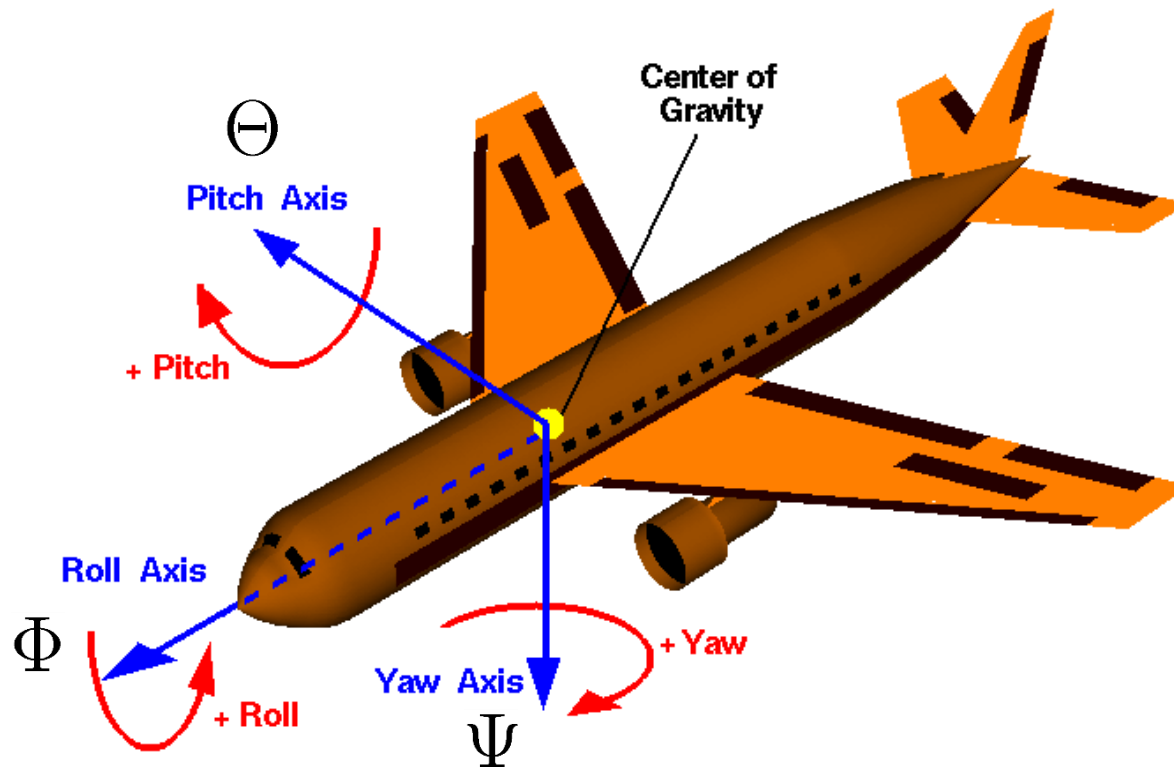
- Orthonormale 3x3 Matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

- Vorteil: Einfache Verkettung
- Nachteil: Überparametrisiert (9 Parameter statt 3)

Euler-Winkel

- Rotation durch Verkettung von 3 Achsrotationen (z.B., um X-Y-Z Achsen)
- Aus der Luftfahrt: Roll-Pitch-Yaw Konvention



Roll-Pitch-Yaw Konvention

- Yaw Ψ , Pitch Θ , Roll Φ in Rotationsmatrix umrechnen

$$\begin{aligned}
 R &= R_Z(\Psi)R_Y(\Theta)R_X(\Phi) \\
 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \Phi & \sin \Phi \\ 0 & -\sin \Phi & \cos \Phi \end{pmatrix} \begin{pmatrix} \cos \Theta & 0 & -\sin \Theta \\ 0 & 1 & 0 \\ \sin \Theta & 0 & \cos \Theta \end{pmatrix} \begin{pmatrix} \cos \Psi & \sin \Psi & 0 \\ -\sin \Psi & \cos \Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} \cos \Theta \cos \Psi & \cos \Theta \sin \Psi & -\sin \Theta \\ \sin \Phi \sin \Theta \cos \Psi - \cos \Phi \sin \Psi & \sin \Phi \sin \Theta \sin \Psi + \cos \Phi \cos \Psi & \sin \Phi \cos \Theta \\ \cos \Phi \sin \Theta \cos \Psi + \sin \Phi \sin \Psi & \cos \Phi \sin \Theta \sin \Psi - \sin \Phi \cos \Psi & \cos \Phi \cos \Theta \end{pmatrix}
 \end{aligned}$$

- Rotation matrix nach Yaw-Pitch-Roll

$$\phi = \text{Atan2} \left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2} \right)$$

$$\psi = -\text{Atan2} \left(\frac{r_{21}}{\cos(\phi)}, \frac{r_{11}}{\cos(\phi)} \right)$$

$$\theta = \text{Atan2} \left(\frac{r_{32}}{\cos(\phi)}, \frac{r_{33}}{\cos(\phi)} \right)$$

Euler-Winkel

- Vorteil:
 - Minimale Repräsentation (3 Parameter)
 - “Einfach” interpretierbar
- Nachteile:
 - Es gibt 24 “alternative” Euler-Winkel-Darstellungen (XYZ, ZXZ, ZYX, ...)
 - Schwierig zu Verketteten
 - Singularitäten (sog. “gimbal lock”)

Euler-Winkel

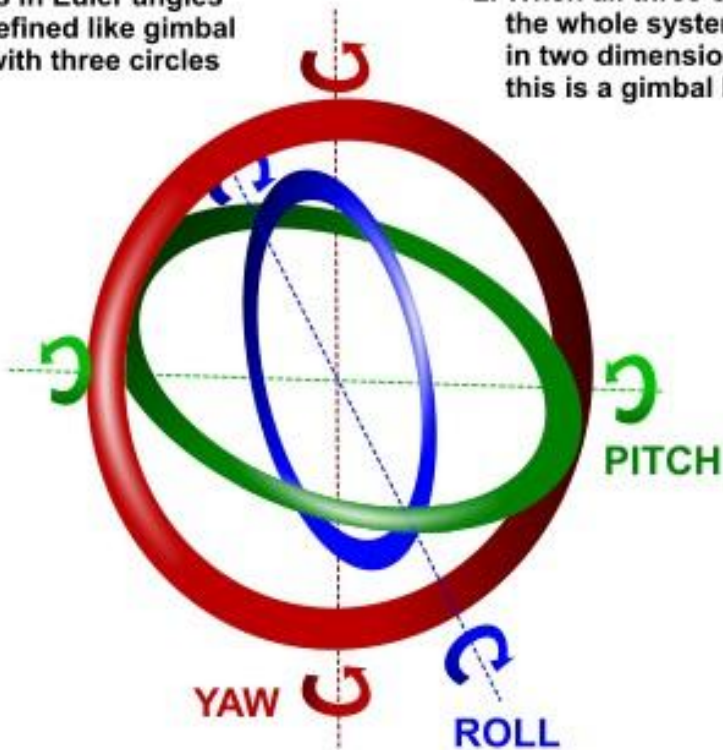
- Verkettung: Konversion in Rotationsmatrix, Multiplikation, Re-Konversion
- Invertierung: Konversion in Rotationsmatrix, Matrixinversion, Re-Konversion

$$R_Z(\psi_1)R_Y(\theta_1)R_X(\phi_1) \cdot R_Z(\psi_2)R_Y(\theta_2)R_X(\phi_2) \\ \neq R_Z(\psi_1 + \psi_2)R_Y(\theta_1 + \theta_2)R_X(\phi_1 + \phi_2)$$

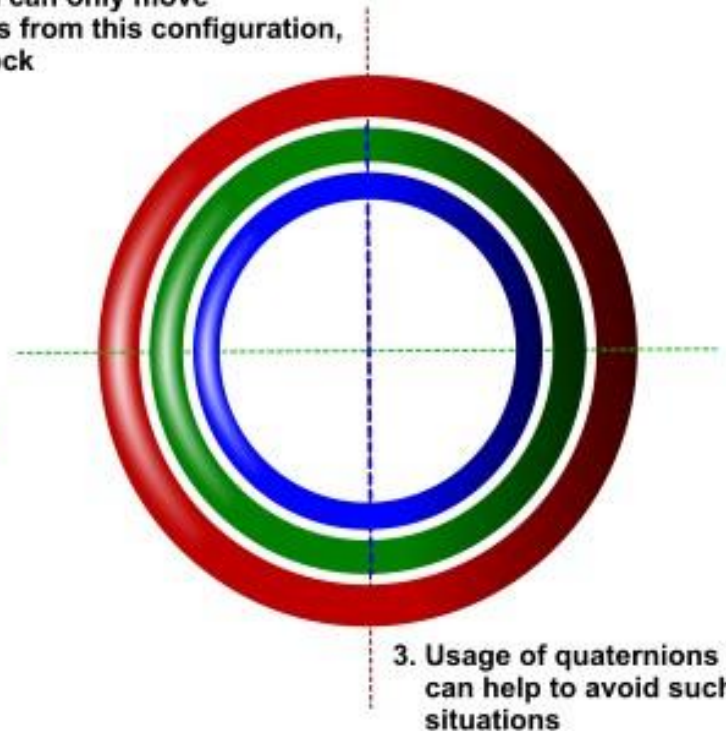
Singularitäten

- Verlust eines Freiheitsgrades

1. Rotations in Euler angles can be defined like gimbal system with three circles



2. When all three circles are lined up, the whole system can only move in two dimensions from this configuration, this is a gimbal lock

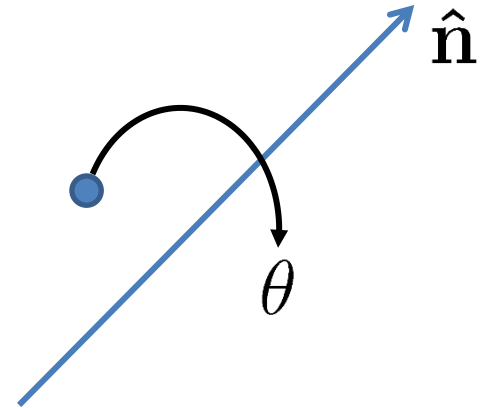


3. Usage of quaternions can help to avoid such situations

Rodriguez-Darstellung

(Drehachse / Winkel)

- Representiert Rotation durch
 - Drehachse $\hat{\mathbf{n}}$ und
 - Drehwinkel θ
- 4 Parameter $(\hat{\mathbf{n}}, \theta)$
- 3 Parameter $\boldsymbol{\omega} = \theta \hat{\mathbf{n}}$
 - Länge kodiert Winkel
 - Minimal aber nicht eindeutig (Warum?)



Konversion

- Rodriguez-Formel

$$R(\hat{\mathbf{n}}, \theta) = I + \sin \theta [\hat{\mathbf{n}}]_{\times} + (1 - \cos \theta) [\hat{\mathbf{n}}]_{\times}^2$$

- Inverse

$$\theta = \cos^{-1} \left(\frac{\text{trace}(R) - 1}{2} \right), \hat{\mathbf{n}} = \frac{1}{2 \sin \theta} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

- see: An Invitation to 3D Vision, Y. Ma, S. Soatto, J. Kosecka, S. Sastry, Chapter 2 (available online)

Einheitsquaternionen

- Quaternion $\mathbf{q} = (q_x, q_y, q_z, q_w)^\top \in \mathbb{R}^4$
- Einheitsquaternionen $\|\mathbf{q}\| = 1$
- Entgegengesetzte Quaternionen repräsentieren dieselbe Rotation $\mathbf{q} = -\mathbf{q}$
- Ansonsten eindeutig

“Komplexe Zahl mit 3 Imaginärteilen”

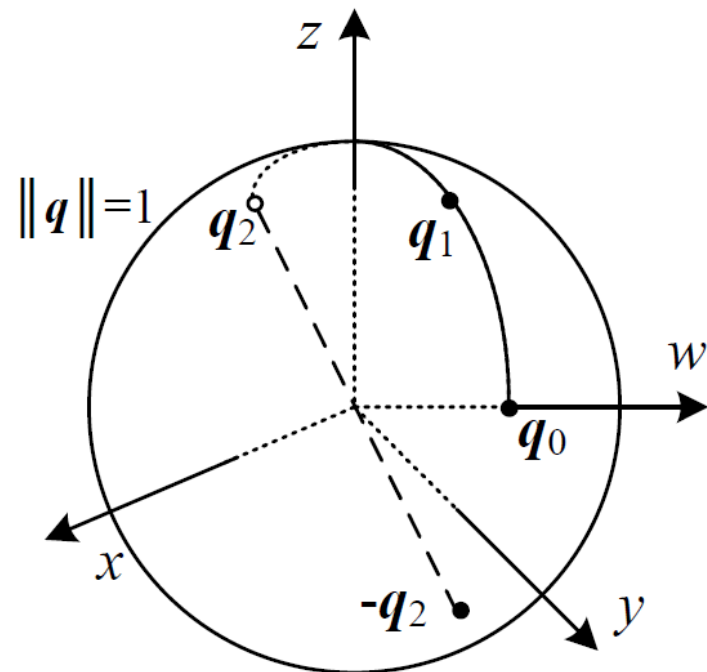
$$\mathbf{q} = iq_x + jq_y + kq_z + q_w$$

$$i^2 = j^2 = k^2 = ijk = -1$$

$$ij = k, \quad ji = -k,$$

$$jk = i, \quad kj = -i,$$

$$ki = j, \quad ik = -j,$$



Einheitsquaternionen

- Vorteil: Operationen für Multiplikation und Inversion sind effizient
- Quaternion-Quaternion Multiplikation

$$\begin{aligned}\mathbf{q}_0 \mathbf{q}_1 &= (\mathbf{v}_0, w_0)(\mathbf{v}_1, w_1) \\ &= (\mathbf{v}_0 \times \mathbf{v}_1 + w_0 \mathbf{v}_1 + w_1 \mathbf{v}_0, w_0 w_1 - \mathbf{v}_0 \mathbf{v}_1)\end{aligned}$$

- Inverse (Vorzeichenwechsel bei v oder w)

$$\begin{aligned}\mathbf{q}^{-1} &= (\mathbf{v}, w)^{-1} \\ &= (\mathbf{v}, -w)\end{aligned}$$

Vereinfachte Schreibweise

$$\mathbf{q} = \underbrace{ix + jy + kz}_{\mathbf{v}} + w$$
$$\mathbf{v} = (x, y, z)^\top$$

Einheitsquaternionen

- Quaternion-Vektor Multiplikation (rotiert Punkt p mit Rotation q)

$$\mathbf{p}' = \mathbf{q}\bar{\mathbf{p}}\mathbf{q}^{-1}$$

mit $\bar{\mathbf{p}} = (x, y, z, 0)^\top$

- Beziehung zu Rodriguez-Darstellung

$$\mathbf{q} = (\mathbf{v}, w) = \left(\sin \frac{\theta}{2} \hat{\mathbf{n}}, \cos \frac{\theta}{2} \right)$$

3D Orientierung im Allgemeinen

- **Beachte:** “Lesen von Rotationen” im Allgemeinen schwierig, egal in welcher Darstellung
- **Beobachtung:** Rotationen sind einfach zu visualisieren und dann intuitiv verständlich
- **Rat:** Zum Debugging immer visualisieren!! Es gibt viele gute 3D Visualisierungstools

C++ Libraries für Lin. Alg./Geometry

- Es gibt viele C/C++ libraries für lineare Algebra und 3D- Geometry
- Beispiele:
 - C arrays, `std::vector` (no linear alg. functions)
 - gsl (gnu scientific library, umfangreich, plain C)
 - `boost::array` (ROS messages)
 - Bullet library (3D-Geometry und Dynamik, ROS tf)
 - Eigen (Sowohl lineare Algebra als auch Geometry, guter Tipp)

WAHRNEHMUNG MIT PUNKTEWOLKEN

Wahrnehmungsaufgabe

- Aufgabe: Extraktion von Fakten aus unstrukturierten Sensordaten, z.B.
 - Für den Roboter relevante Objekte in der Umgebung
 - Lage / Pose, Klasse, Instanz



Beispiel:

Objekterkennung und -lokalisierung

- Drei wesentliche Verarbeitungsschritte
 - Segmentierung
 - Gegebenenfalls Erkennung / Vermessung
 - Lagebestimmung



Beispiel: Objekterkennung und -lokalisierung

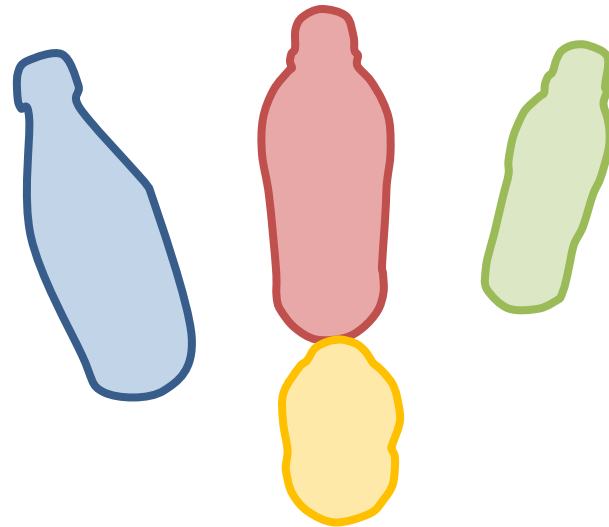
- Drei wesentliche Verarbeitungsschritte
 - Segmentierung
 - Gegebenenfalls Erkennung / Vermessung
 - Lagebestimmung



Beispiel:

Objekterkennung und -lokalisierung

- Drei wesentliche Verarbeitungsschritte
 - Segmentierung
 - Gegebenenfalls Erkennung / Vermessung
 - Lagebestimmung



Segmentierung

- Aufteilung der Daten in inhaltlich zusammenhängende Teilmengen
 - Was ist ein Segment? Kommt darauf an!
 - Segmente entsprechen beispielsweise Objekten
 - Definition (und Lösung) aufgabenabhängig



Binarisierung mit Schwellwert

Segmentierung

- Aufteilung der Daten in inhaltlich zusammenhängende Teilmengen
 - Was ist ein Segment? Kommt darauf an!
 - Segmente entsprechen beispielsweise Objekten
 - Definition (und Lösung) aufgabenabhängig



Binarisierung mit Schwellwert

Segmentierung

- Aufteilung der Daten in inhaltlich zusammenhängende Teilmengen
 - Was ist ein Segment? Kommt darauf an!
 - Segmente entsprechen beispielsweise Objekten
 - Definition (und Lösung) aufgabenabhängig



Binarisierung mit Schwellwert

Segmentierung

- Aufteilung der Daten in inhaltlich zusammenhängende Teilmengen
 - Was ist ein Segment? Kommt darauf an!
 - Segmente entsprechen beispielsweise Objekten
 - Definition (und Lösung) aufgabenabhängig



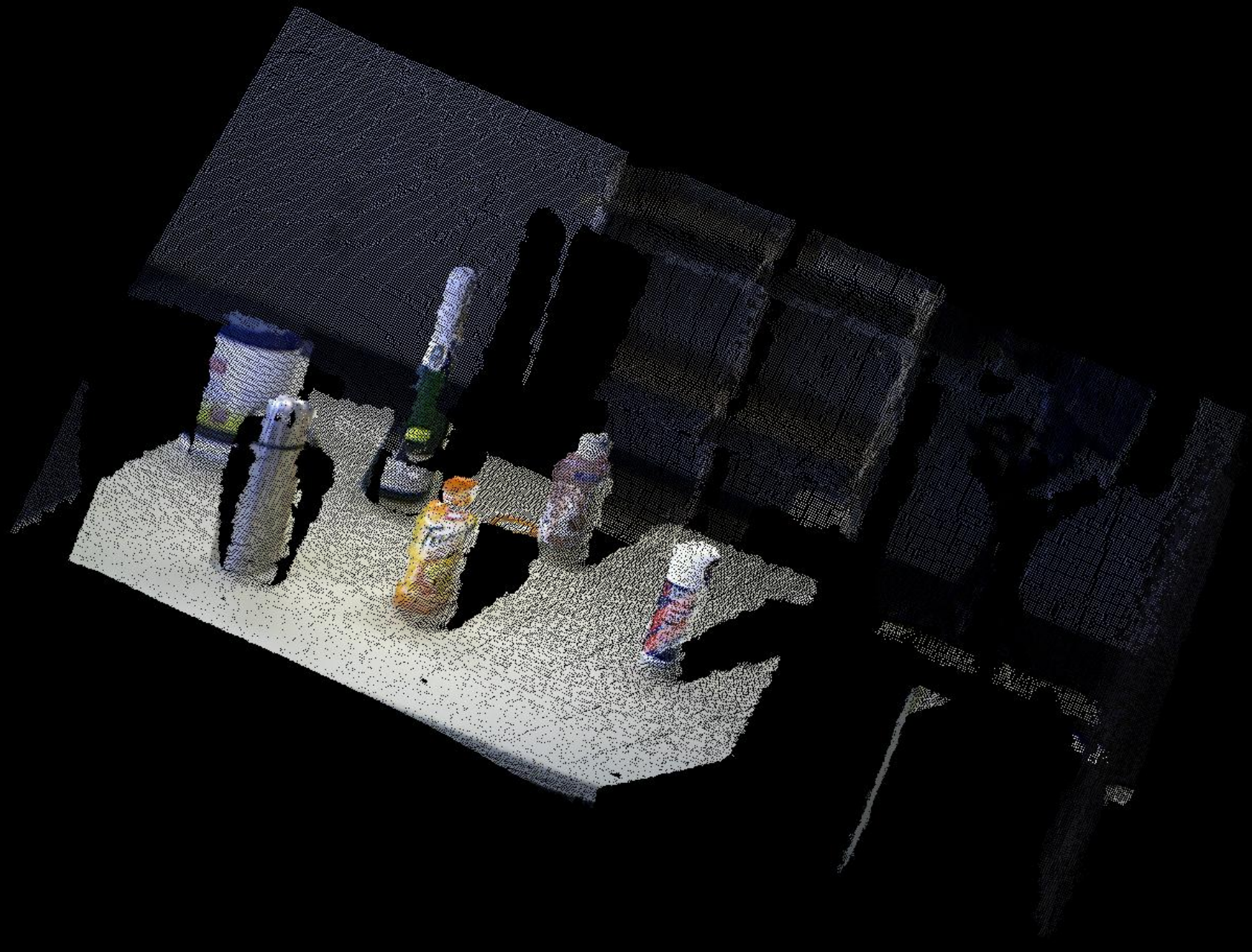
Binarisierung mit Schwellwert

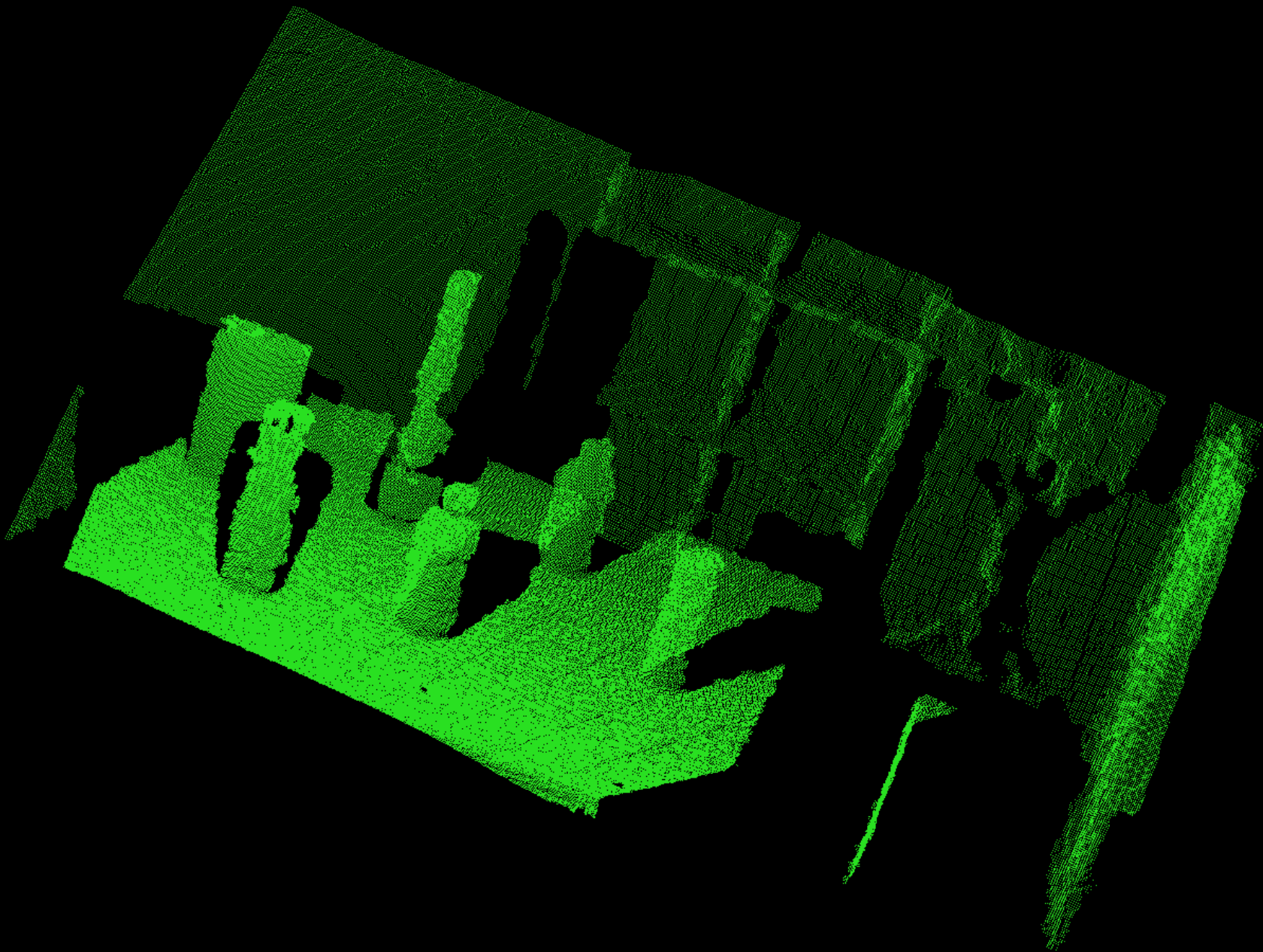




Segmentierung in Punktwolke

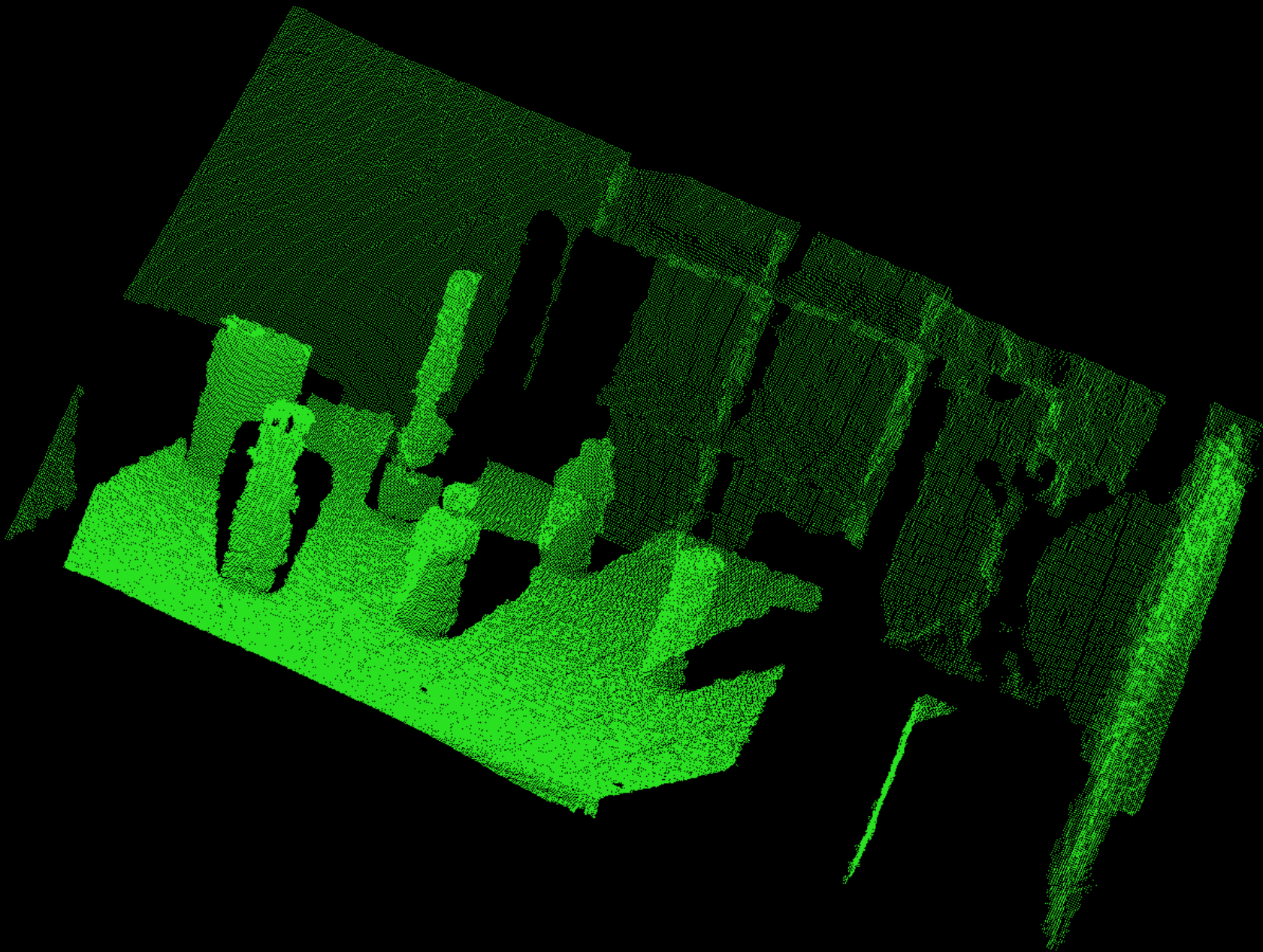
- Vorteil: Nutzen von räumlicher Struktur
 - z.B. Objekte stehen auf Tisch





Segmentierung in Punktwolke

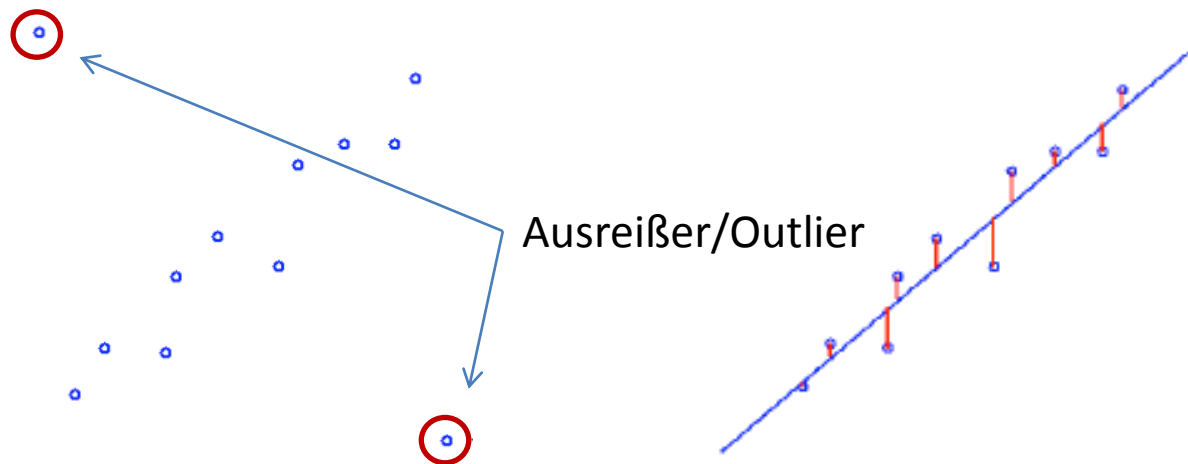
- Vorteil: Nutzen von räumlicher Struktur
 - z.B. Objekte stehen auf Tisch
- Grundsätzliches Vorgehen
 - Detektion und Vermessung der Tischebene (Höhe und Normalenrichtung)
 - Objekte sind zusammenhängende Teilpunktwolken oberhalb des Tisches
 - Bestimmung der jeweiligen Objektgeometrie



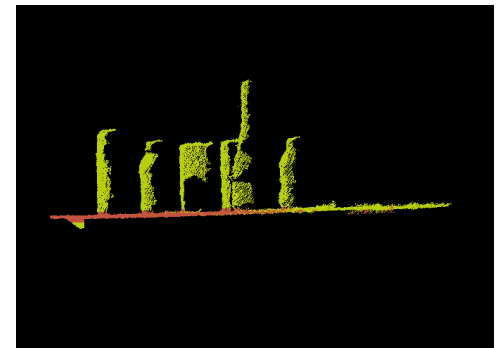
Detektion und Vermessung der Tischebene

- Ziel: Eine Ebene in die Punktwolke einpassen
 - Minimierung des mittleren quadratischen Fehlers

$$\theta^* = \arg \min_{\theta} \sum_i d^2(x_i, \theta)$$



- Problem hier bei uns: Objekte stören

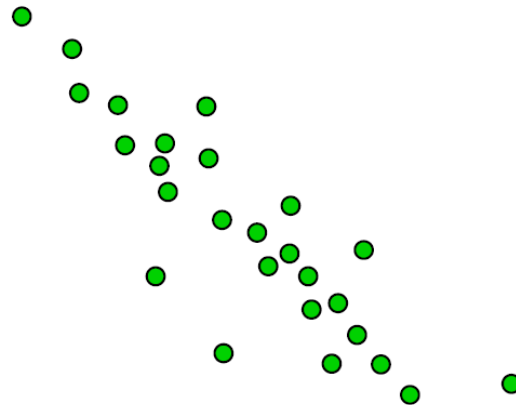


Robuste Parameterschätzung

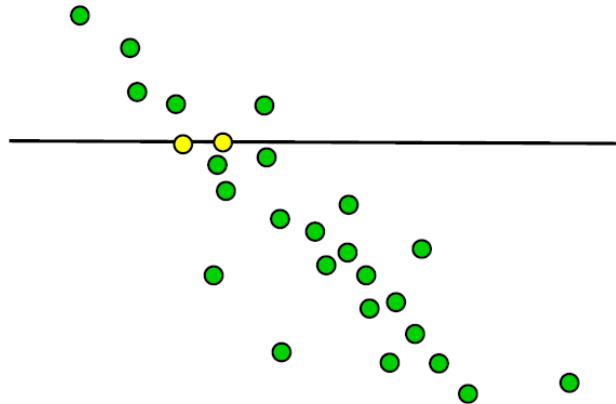
- Allgemeiner Ansatz zur robusten Modellanpassung (Model Fit)
 - Aufteilen der Punkte in Inlier und Outlier
 - Modellanpassung nur mit den Inliern durchführen
- RANSAC: Random Sample Consensus
 - **Zufällig** aus den Datenpunkten die **minimal** für das Modell **erforderliche** Anzahl von Punkten **auswählen**
 - Nachprüfen, wieviele Punkte zu dem geschätzten Modell passen
 - Das Modell mit den meisten unterstützenden Datenpunkten auswählen
 - Alle nicht unterstützenden Daten ignorieren und Modellanpassung mit den passenden Daten durchführen

M. A. Fischler and R. C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Comm. of the ACM* **24**: 381--395.

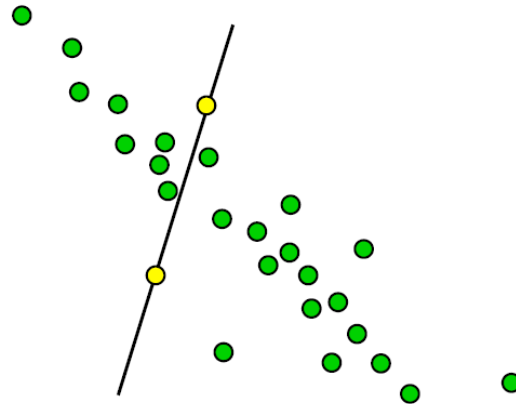
RANSAC: Random Sample Consensus



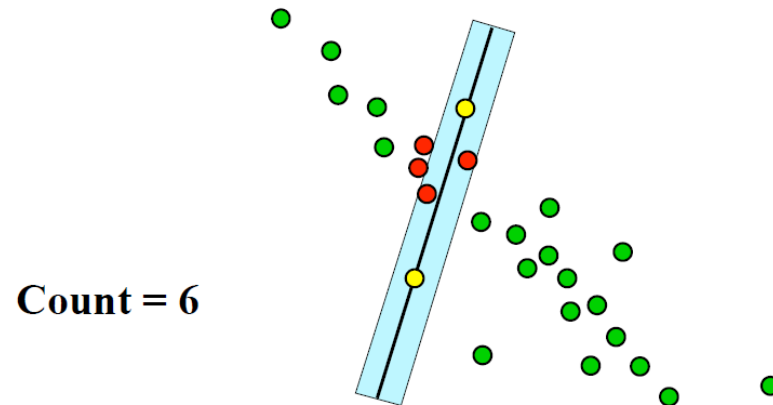
RANSAC: Random Sample Consensus



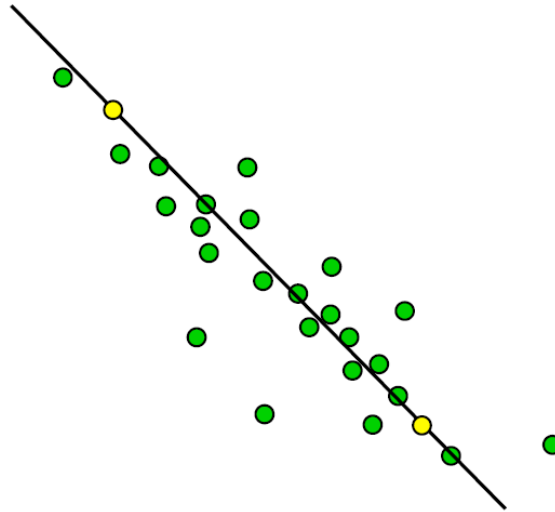
RANSAC: Random Sample Consensus



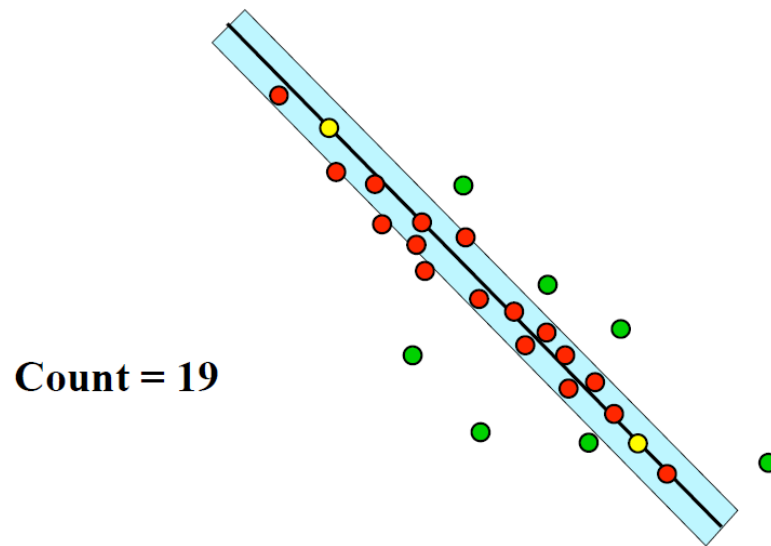
RANSAC: Random Sample Consensus



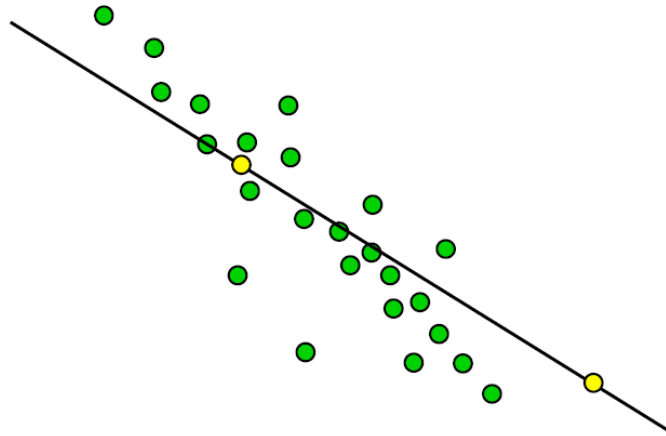
RANSAC: Random Sample Consensus



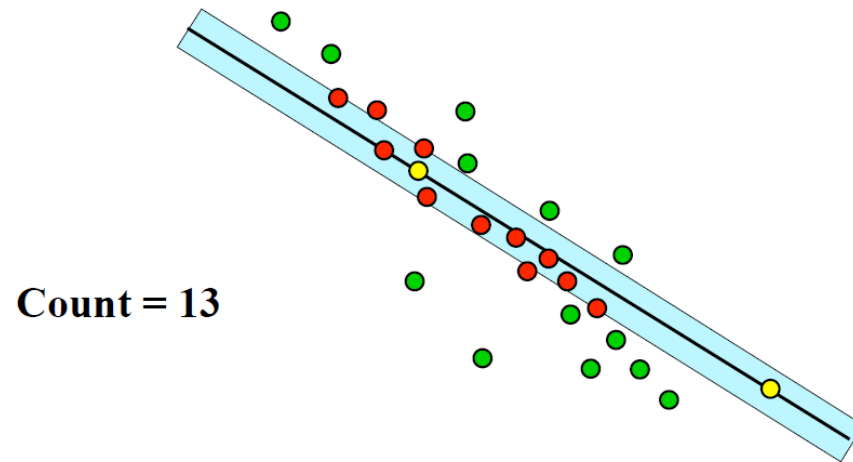
RANSAC: Random Sample Consensus



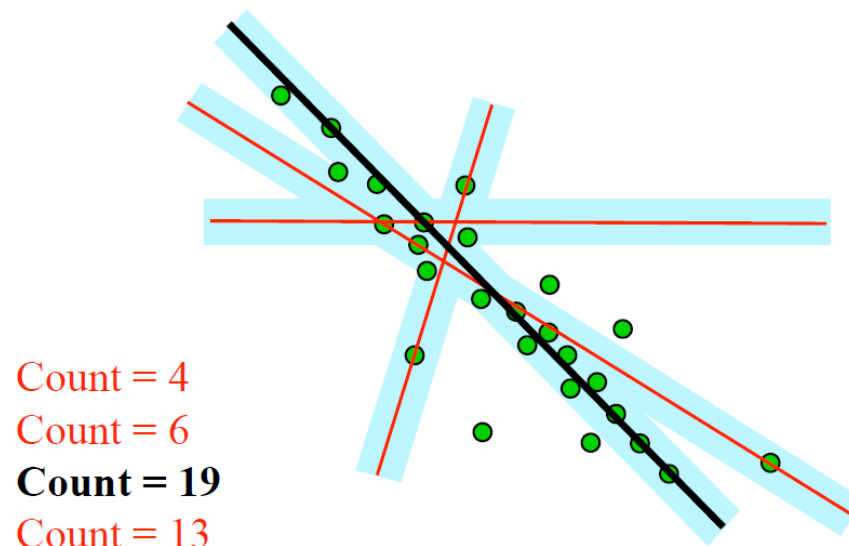
RANSAC: Random Sample Consensus



RANSAC: Random Sample Consensus



RANSAC: Random Sample Consensus



RANSAC: Random Sample Consensus

- Frage, wie oft müssen wir Samples ziehen?
 - e: Wahrscheinlichkeit dass ein Punkt ein Outlier ist
 - s: Anzahl (minimal) erforderlicher Punkte im Sample
 - N: Anzahl der Samples (wollen wir ausrechnen)
 - p: Gewünschte Wahrscheinlichkeit, dass wir ein gutes Sample bekommen

- Antwort:
$$1 - (1 - (1 - e)^s)^N = p$$

RANSAC: Random Sample Consensus

- Frage, wie oft müssen wir Samples ziehen?
 - e: Wahrscheinlichkeit dass ein Punkt ein Outlier ist
 - s: Anzahl (minimal) erforderlicher Punkte im Sample
 - N: Anzahl der Samples (wollen wir ausrechnen)
 - p: Gewünschte Wahrscheinlichkeit, dass wir ein gutes Sample bekommen

$$1 - (1 - (\underbrace{1 - e}_\text{Wahrscheinlichkeit einen Inlier zu ziehen})^s)^N = p$$

Wahrscheinlichkeit einen Inlier zu ziehen

RANSAC: Random Sample Consensus

- Frage, wie oft müssen wir Samples ziehen?
 - e: Wahrscheinlichkeit dass ein Punkt ein Outlier ist
 - s: Anzahl (minimal) erforderlicher Punkte im Sample
 - N: Anzahl der Samples (wollen wir ausrechnen)
 - p: Gewünschte Wahrscheinlichkeit, dass wir ein gutes Sample bekommen

$$1 - \underbrace{(1 - (1 - e)^s)}^{\text{Wahrscheinlichkeit s Inlier nacheinander zu ziehen}}^N = p$$

Wahrscheinlichkeit s Inlier nacheinander zu ziehen

RANSAC: Random Sample Consensus

- Frage, wie oft müssen wir Samples ziehen?
 - e: Wahrscheinlichkeit dass ein Punkt ein Outlier ist
 - s: Anzahl (minimal) erforderlicher Punkte im Sample
 - N: Anzahl der Samples (wollen wir ausrechnen)
 - p: Gewünschte Wahrscheinlichkeit, dass wir ein gutes Sample bekommen

$$1 - \underbrace{(1 - (1 - e)^s)}^{\text{Wahrscheinlichkeit mindestens einen Outlier zu erwischen}}^N = p$$

Wahrscheinlichkeit mindestens einen Outlier zu erwischen: Sample ist “verschmutzt”.

RANSAC: Random Sample Consensus

- Frage, wie oft müssen wir Samples ziehen?
 - e: Wahrscheinlichkeit dass ein Punkt ein Outlier ist
 - s: Anzahl (minimal) erforderlicher Punkte im Sample
 - N: Anzahl der Samples (wollen wir ausrechnen)
 - p: Gewünschte Wahrscheinlichkeit, dass wir ein gutes Sample bekommen

$$1 - \underbrace{(1 - (1 - e)^s)^N}_{\text{Wahrscheinlichkeit N verschmutzter Samples}} = p$$

Wahrscheinlichkeit N verschmutzter Samples

RANSAC: Random Sample Consensus

- Frage, wie oft müssen wir Samples ziehen?
 - e: Wahrscheinlichkeit dass ein Punkt ein Outlier ist
 - s: Anzahl (minimal) erforderlicher Punkte im Sample
 - N: Anzahl der Samples (wollen wir ausrechnen)
 - p: Gewünschte Wahrscheinlichkeit, dass wir ein gutes Sample bekommen

$$\underbrace{1 - (1 - (1 - e)^s)^N}_{\text{Wahrscheinlichkeit mindestens ein reines Sample zu erhalten}} = p$$

Wahrscheinlichkeit mindestens ein reines Sample zu erhalten

RANSAC: Random Sample Consensus

Wahl von N so, dass mit $p=99\%$ mindestens 1 Sample keine Outlier enthält.

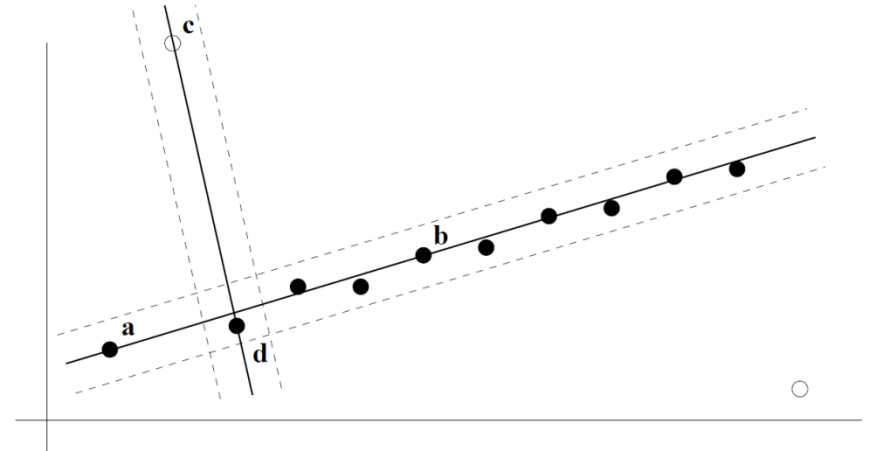
$$(1 - (1 - e)^s)^N = 1 - p$$

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

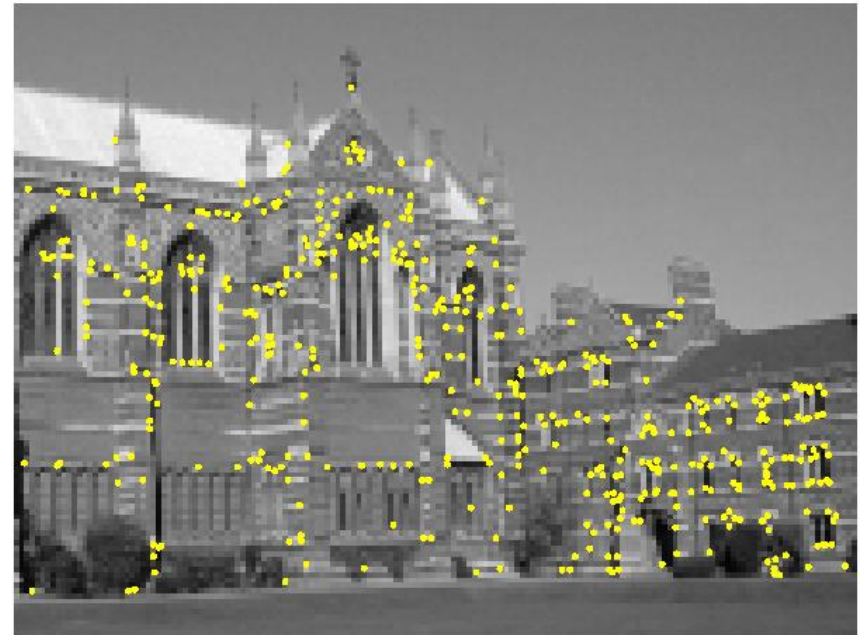
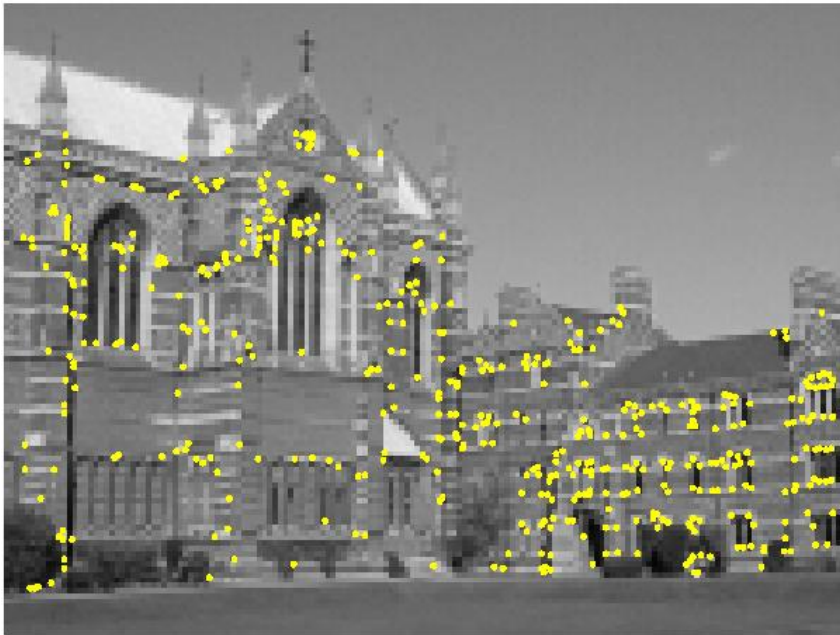
| proportion of outliers e | | | | | | | |
|----------------------------|----|-----|-----|-----|-----|-----|------|
| s | 5% | 10% | 20% | 25% | 30% | 40% | 50% |
| 2 | 2 | 3 | 5 | 6 | 7 | 11 | 17 |
| 3 | 3 | 4 | 7 | 9 | 11 | 19 | 35 |
| 4 | 3 | 5 | 9 | 13 | 17 | 34 | 72 |
| 5 | 4 | 6 | 12 | 17 | 26 | 57 | 146 |
| 6 | 4 | 7 | 16 | 24 | 37 | 97 | 293 |
| 7 | 4 | 8 | 20 | 33 | 54 | 163 | 588 |
| 8 | 5 | 9 | 26 | 44 | 78 | 272 | 1177 |

RANSAC: Random Sample Consensus

- 12 Punkte
- Samplegröße $s = 2$
- Ausreißer: 2 $\rightarrow e = 1/6$
- $N=5$: $p=99\%$ dass mindestens ein reines Sample gezogen wird
- Bei 66 Punktpaaren
- Auch andere Abbruchkriterien denkbar (viel Literatur)

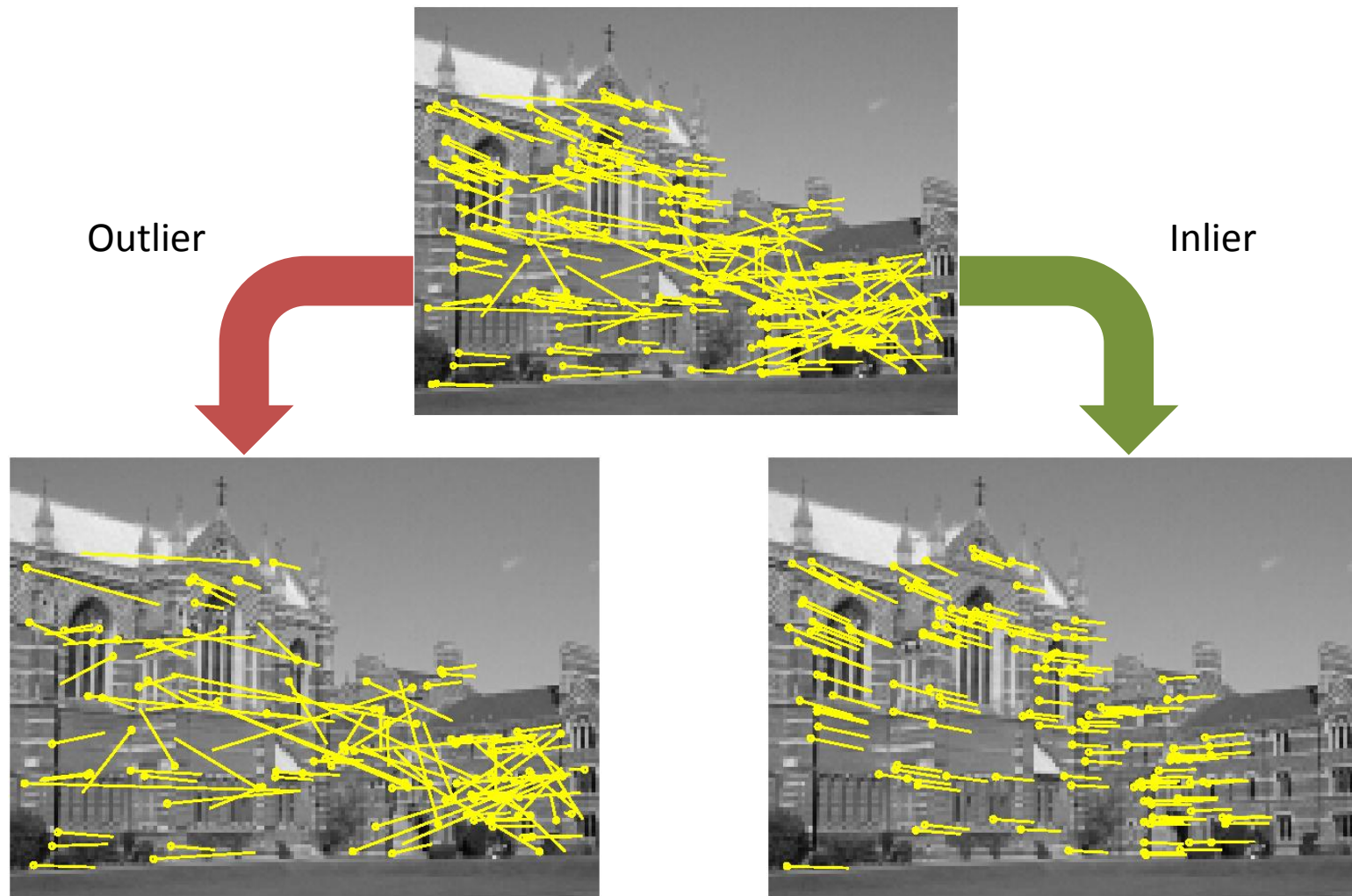


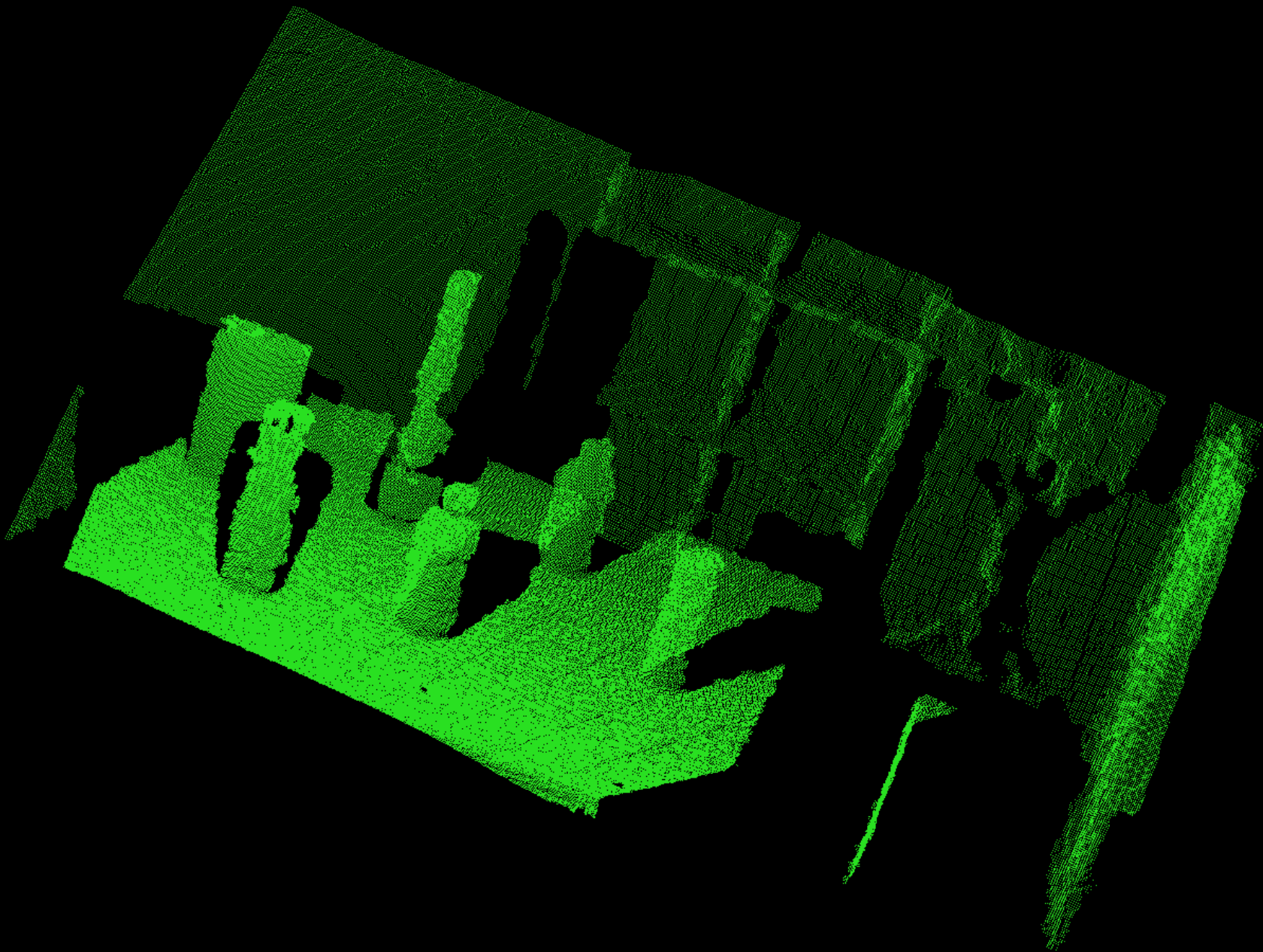
RANSAC für Bildkorrespondenzen



Eckendetektion: Finde Punkte, die sich gut vergleichen lassen

RANSAC für Bildkorrespondenzen

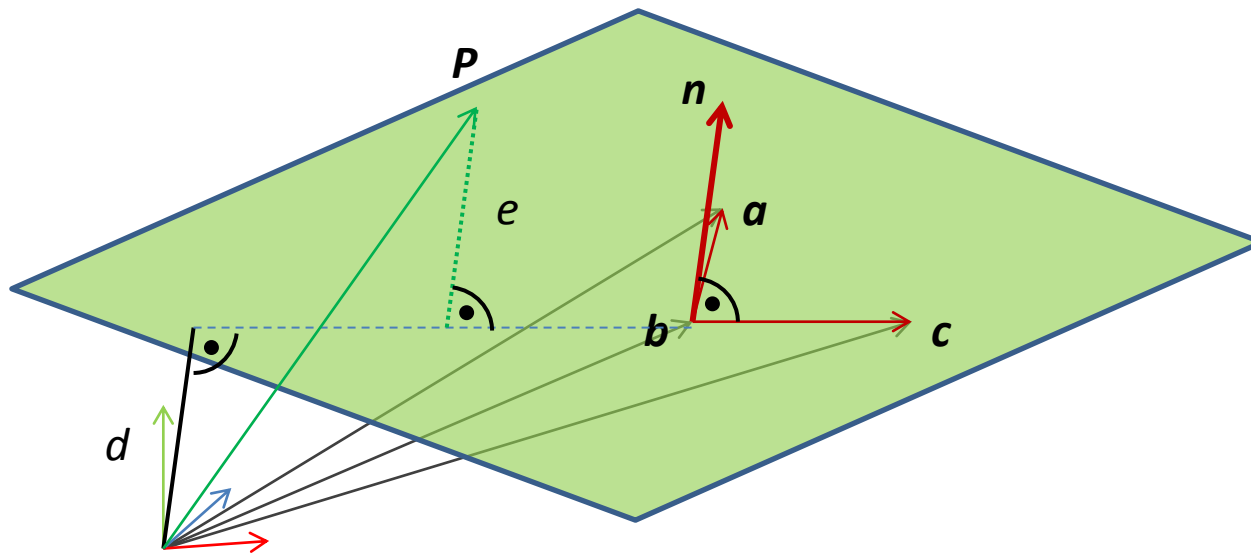




Ebenenfit mit RANSAC

- Eine Ebene ist gegeben durch 3 Punkte: ***a, b, c***
 - Ebenennormale ***n***, Abstand der Ebene vom Ursprung ***d***
 - Abstand des Punktes ***P*** von der Ebene

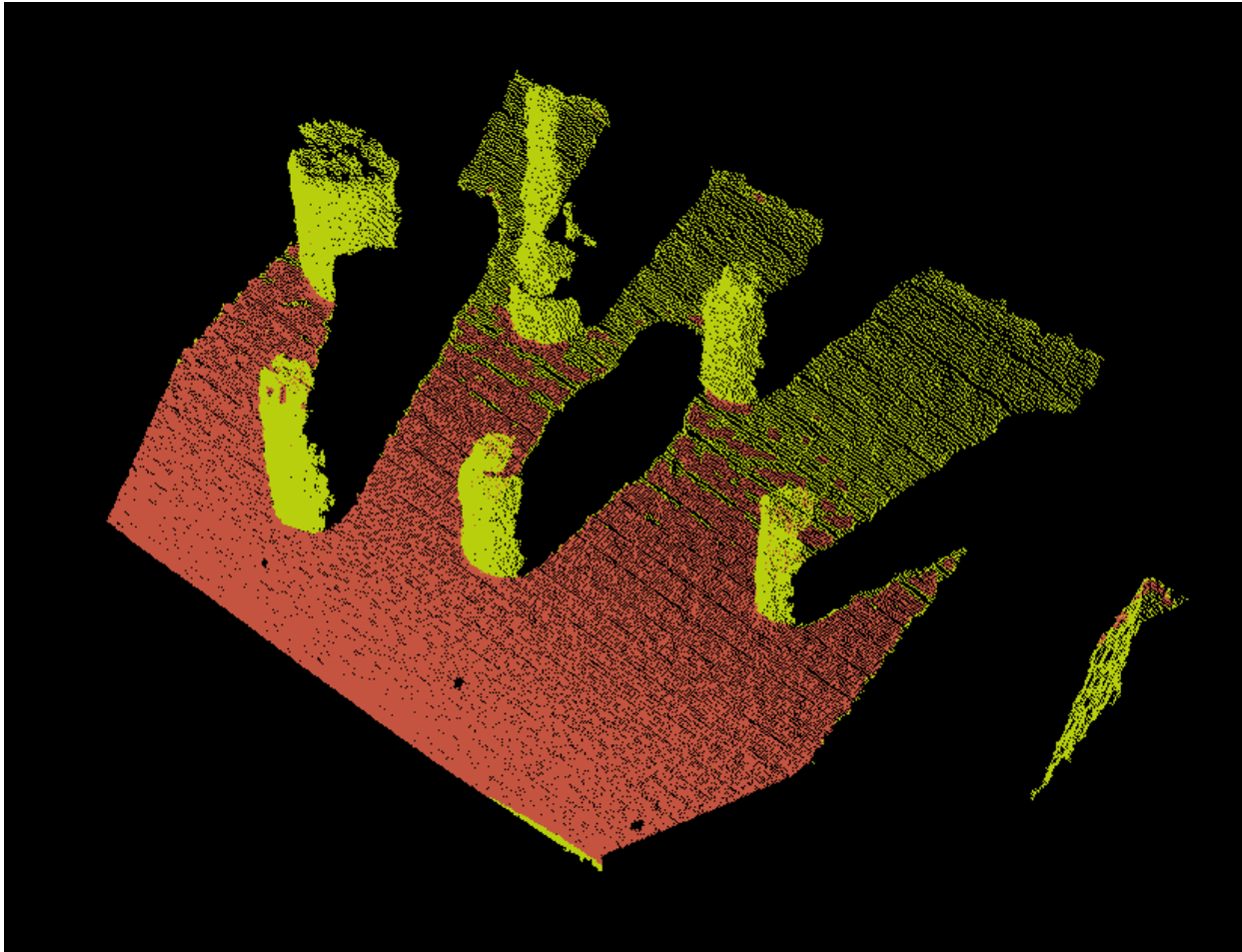
$$e = \frac{\mathbf{P} \cdot \mathbf{n}}{|\mathbf{n}|} - d$$



$$\mathbf{n} = (\mathbf{c} - \mathbf{b}) \times (\mathbf{a} - \mathbf{b})$$

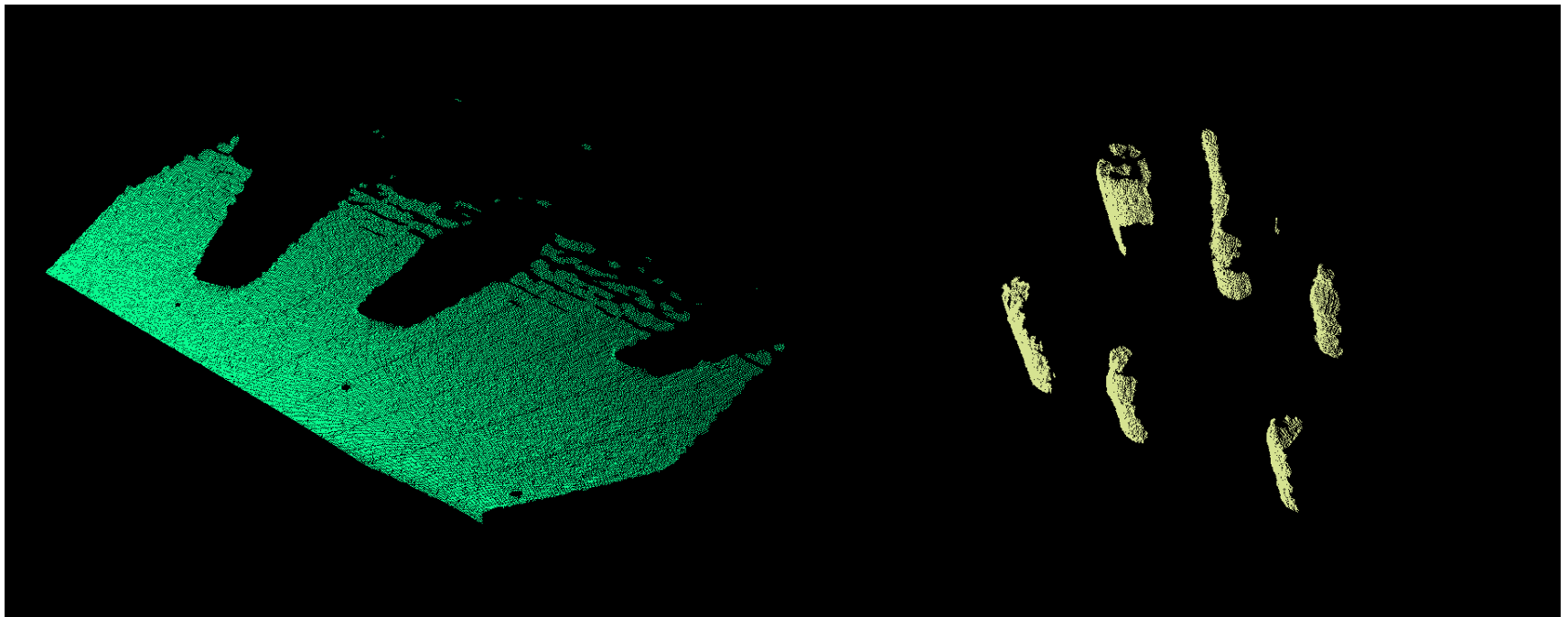
$$d = \frac{\mathbf{a} \cdot \mathbf{n}}{|\mathbf{n}|}$$

Ebenenfit mit RANSAC

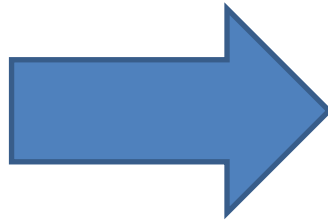


Trennung von Tisch und Objekten

- Subtraktion aller Punkte in und unter der Tischebene von der Gesamtpunktwolke
 - Es bleibt eine Punktwolke, die alle Objektpunkte enthält



Extraktion der Einzelobjekte



- Pro Objekt gibt es eine zusammenhängende Teil-Punktewolke
 - Punkte, deren minimaler euklidischer Abstand untereinander kleiner als der zu den Nachbarobjekten ist
- Segmentierung mit dem sogenannten „Region growing“
 - Andere Möglichkeit z.B. k-Means

Segmentierung der Objekte

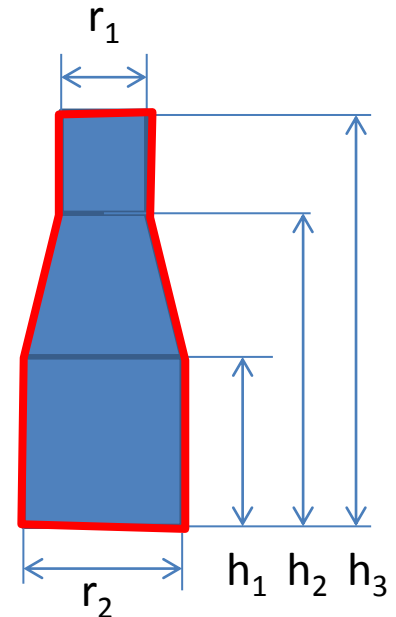
Extraktion euklidischer Cluster (engl. „Haufen“) durch „Region Growing“:

1. Ausgangspunkt ist eine Menge von Punkten P
2. Erzeuge eine leere Liste von Clustern C und eine Liste von zu überprüfenden Punkte Q
3. Für jeden Punkt $p_i \in P$ führe die folgenden Schritte aus
 - Füge den Punkt p_i in die Liste zu überprüfender Punkte Q
 - Für jeden Punkt $p_j \in Q$
 - Suche nach der Menge aller Nachbarn N_{jk} von p_j , deren Abstand zu p_j kleiner ist als t
 - Für jeden Nachbar $n \in N_{jk}$ prüfe, ob dieser schon verarbeitet wurde, falls nicht, füge ihn zu Q hinzu
 - Wenn alle Punkte in Q bearbeitet wurden, füge Q der Liste der Cluster C hinzu und dann Q zurück

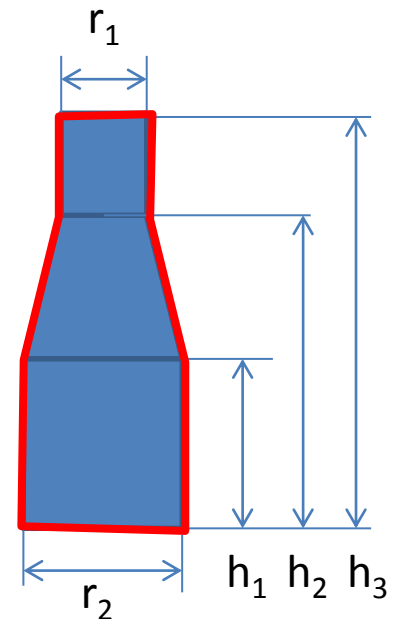
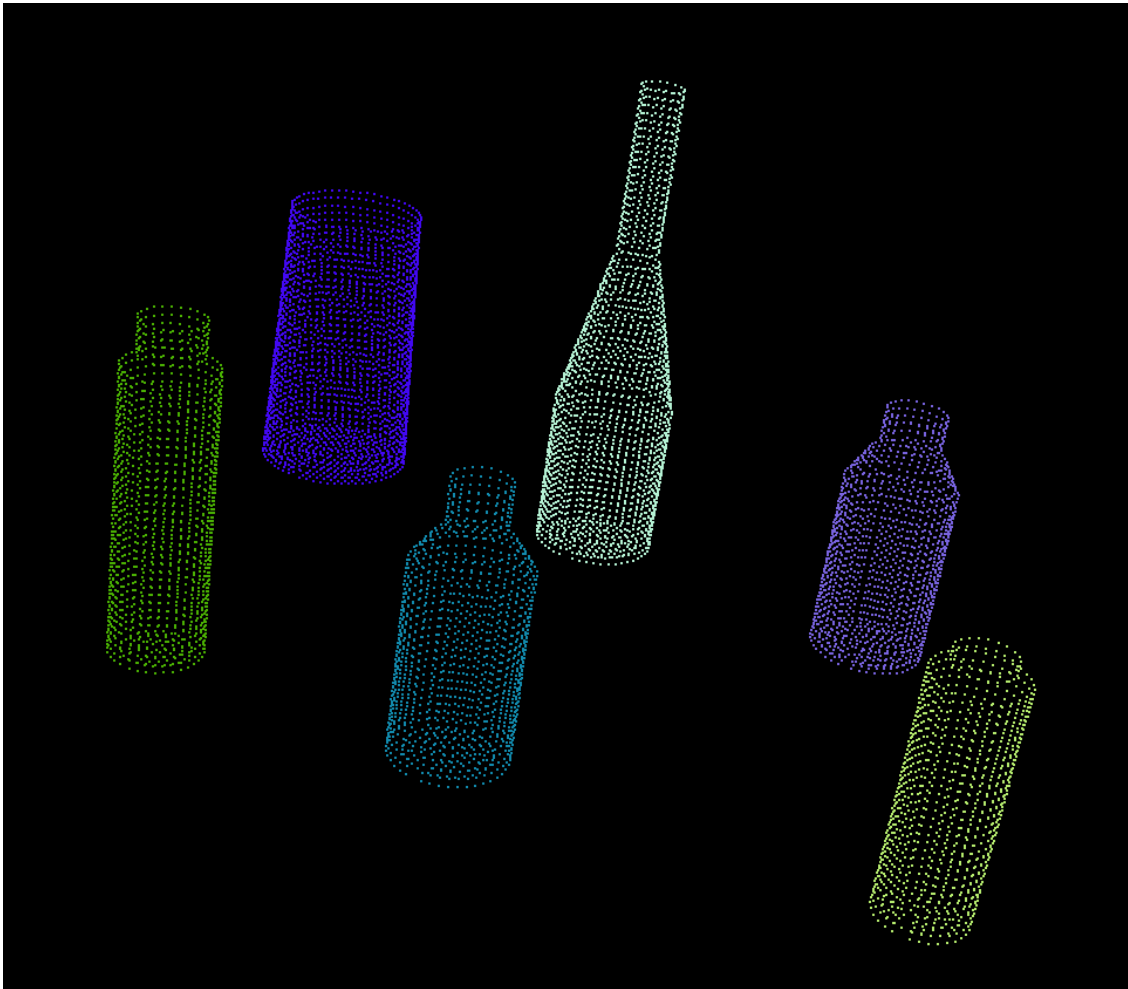
Der Algorithmus terminiert wenn alle Punkte $p_i \in P$ bearbeitet wurden und jetzt Teil eines der Cluster C sind

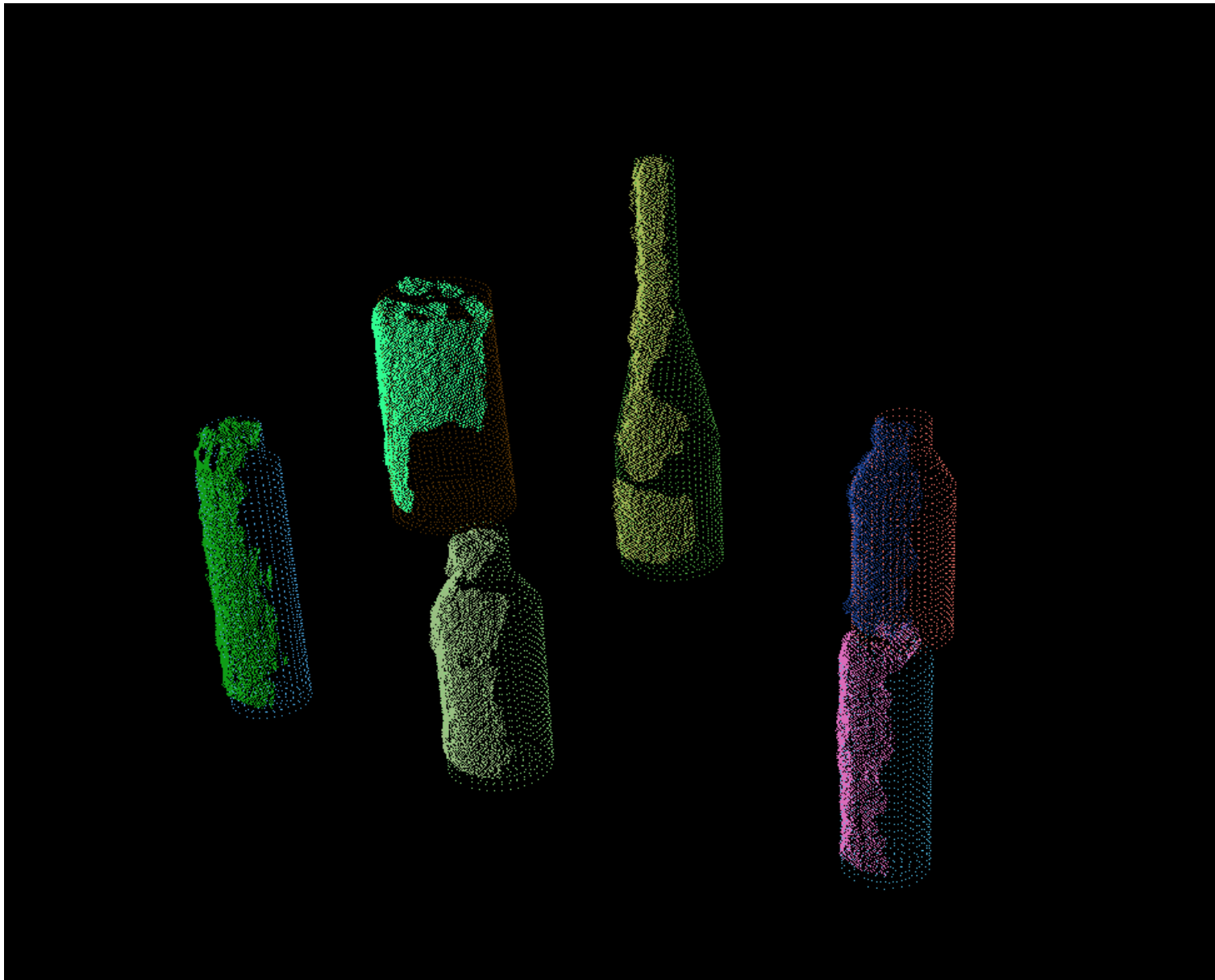
Vermessung und Lagebestimmung

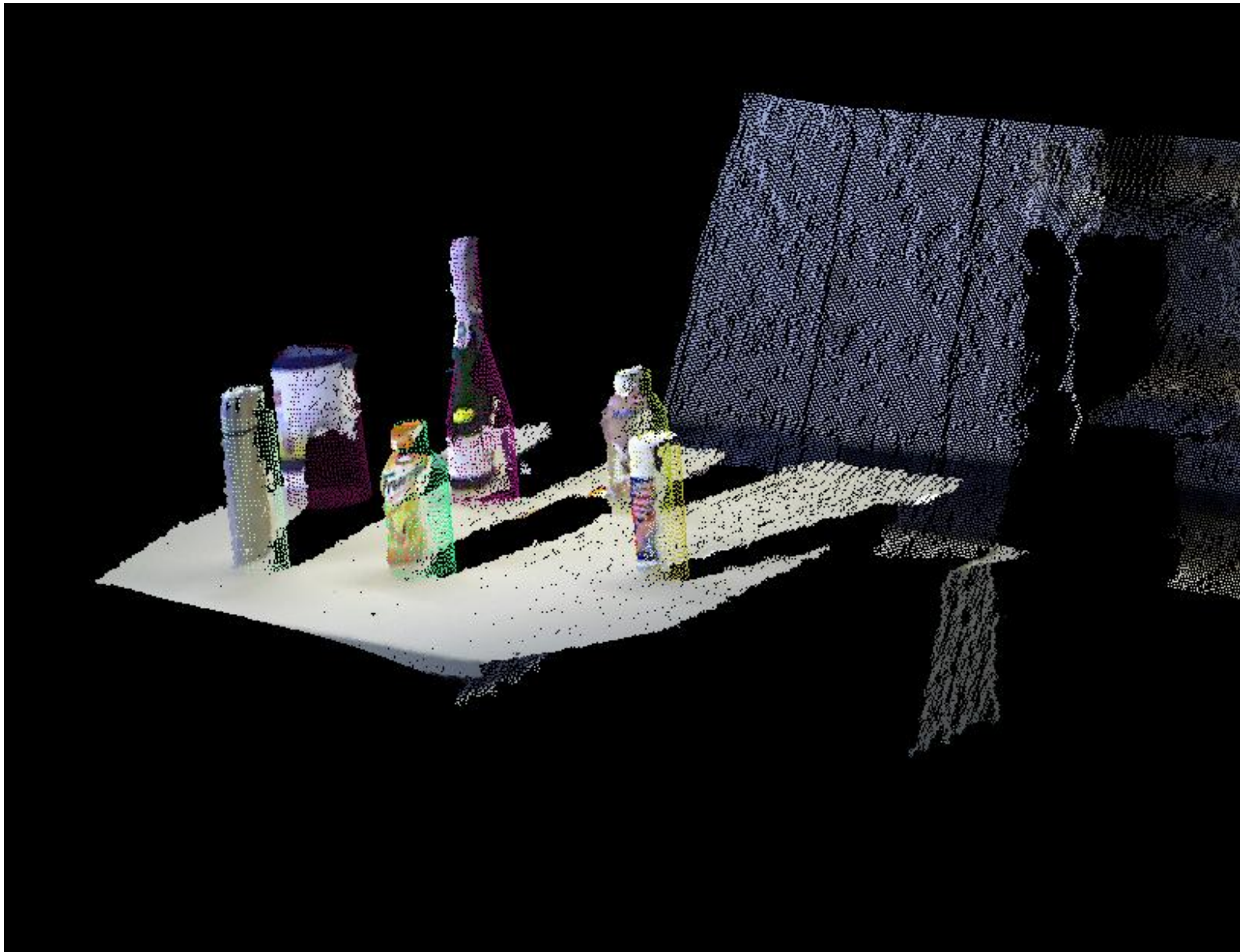
- 2D Position auf dem Tisch: x, y
 - Relativ zur Tischebene
 - Rotationssymmetrie
- Vereinfachte Objektgeometrie
 - 2 Radien, 3 Höhen
- Anpassung an Objektpunktwolke
 - Kleinste Quadrate
 - Optimierung über alle 7 Parameter



Vermessung und Lagebestimmung

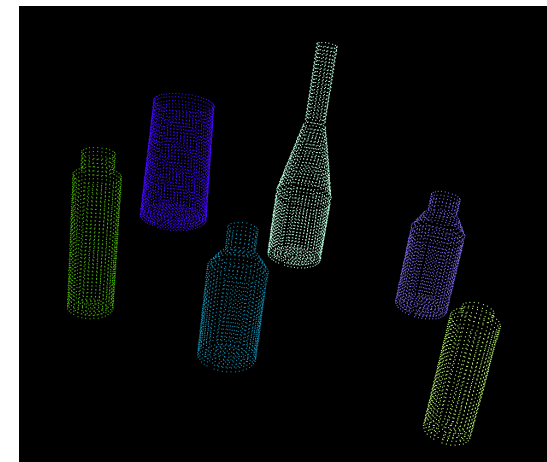






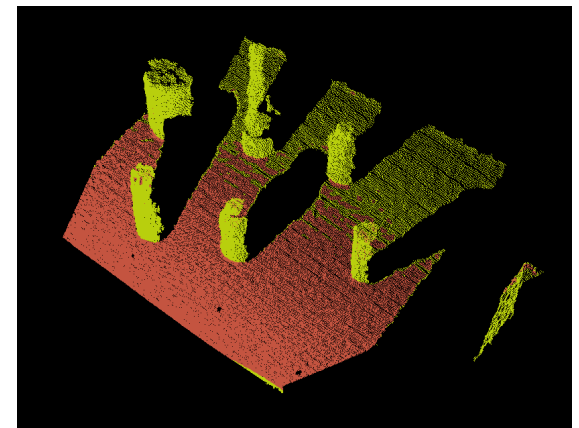
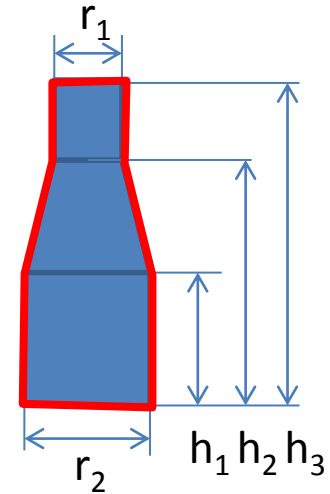
Was lernen wir jetzt daraus?

- Wahrnehmung: Extrahiert strukturierte Information aus unstrukturierten Sensordaten
 - Anzahl vorhandener Objekte
 - Position und Formparameter für jedes Objekt
- Das beschriebene Vorgehen ist nur eine von vielen Möglichkeiten, aber recht typisch
 - Modellannahmen: Einzelne Objekte auf Tischebene aufrechtstehend, rotationssymmetrisch, „flaschenförmig“



Was lernen wir jetzt daraus?

- Starke Dimensionsreduktion durch Wahl der Objekt- und Szenenmodellierung
 - Objektlage (senkrecht auf Tisch): 6D Pose -> 3D Pose
 - Rotationssymmetrie: 3D Pose -> 2D Pose
 - Abstrahiertes Flaschenmodell: Form 5D
- Schätzung der verbliebenen 7D Parametervektoren mit verhältnismäßig vielen Sensormessungen (3D Punkte) -> erhöht die Genauigkeit
 - Siehe auch Tischebene



Was lernen wir jetzt daraus?

- Nutzt Wissen über die Szene -> funktioniert nicht / schlecht bei Abweichungen von den Annahmen
 - z.B. sich (fast) berührende Flaschen stören die Segmentierung
- Aber auch wenn alles klappt: Fehler bleiben! Immer!
 - Messfehler: z.B. Objektradius ist niemals exakt sein
 - Strukturfehler: z.B. Dose bekommt keine Struktur
 - [redacted]
 - [redacted] (beim Zylinder über dem Zylinder für Minimierung des Gesamtwertes)
 - Folge der Modellannahmen

