

Robotik

Priv.-Doz. Dr. Thomas Wiemann
Institut für Informatik
Autonome Robotik

SoSe 2021



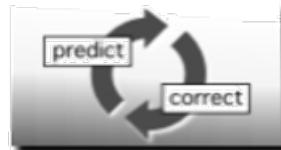
Kalman-Filter als spezieller Bayes-Filter

$$P(x_{t+1} | x_t, u_t) = \mathcal{N}(Ax_t + Bu_t, \Sigma_u)(x_{t+1})$$

$$P(z_t | x_t) = \mathcal{N}(Hx_t, \Sigma_z)(z_t)$$

n	Dimension Zustand
m	Dimension Aktion
l	Dimension Messung

- 1 Einheitsmatrix ($n \times n$)-dimensional
- A Transitionsmodell ($n \times n$)-dimensional, modelliert spontane Transitionen. Ist gleich 1, wenn keine spontanen Transitionen erfolgen
- B Aktionsmodell ($n \times m$)-dimensional, konvertiert u in den Zustandsraum
- Σ_u Kovarianzmatrix für Aktionsmodell ($n \times n$)-dimensional
- H Sensormodell, ($l \times n$)-dimensional. Konvertiert Zustand in den Messraum
- Σ_z Kovarianzmatrix für Sensormodell ($l \times l$)-dimensional



Schätzung von Mittelwert und Varianz

A-priori-Vorhersagen für $t + 1$:

$$\underline{\mathbf{x}}_{t+1} := \mathbf{A}\mu_t + \mathbf{B}\mathbf{u}$$

$$\underline{\Sigma}_{t+1} := \mathbf{A}\Sigma_t\mathbf{A}^T + \Sigma_u$$

← Vorhergesagter Zustand zu $t + 1$ nach u

← Vorhergesagte Kovarianz zu $t + 1$ nach u

Filterung / Korrektur:

$$\mu_{t+1} = \underline{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z} - \mathbf{H}\underline{\mathbf{x}}_{t+1})$$

$$\Sigma_{t+1} = (1 - \mathbf{K}_{t+1}\mathbf{H})\Sigma_{t+1}$$

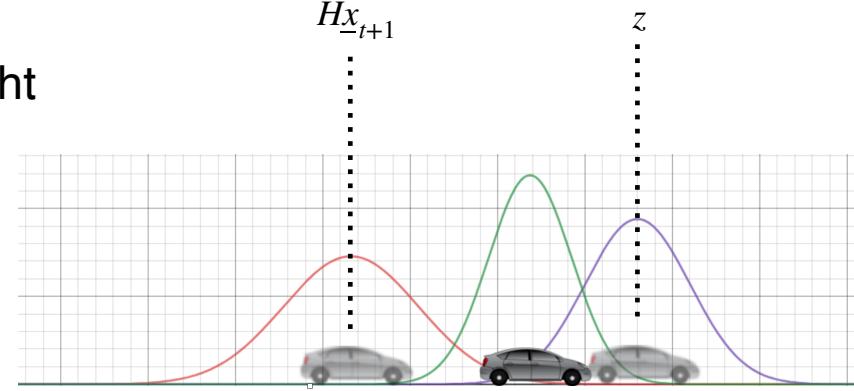
rote Kurve

grüne Kurve

$H\underline{x}_{t+1}$

z

\mathbf{K}_{t+1} sagt, wie hoch der Vorhersagefehler in neuen Wert eingeht

“Kalman Gewinnmatrix (Kalman Gain)”: 

$$\mathbf{K}_{t+1} = \Sigma_{t+1}\mathbf{H}^T(\mathbf{H}\Sigma_{t+1}\mathbf{H}^T + \Sigma_z)^{-1}$$

Beispiel: Gleichförmige Bewegung + GPS

Erinnerung:

$$\underline{x}_{t+1} := A\mu_t + Bu$$

$$\mu_{t+1} = \underline{x}_{t+1} + K_{t+1}(z - H\underline{x}_{t+1})$$

$$K_{t+1} = \Sigma_{t+1} H^T (H\Sigma_{t+1} H^T + \Sigma_z)^{-1}$$

$$\Sigma_{t+1} := A\Sigma A^T + \Sigma_u$$

$$\Sigma_{t+1} = (I - K_{t+1}H)\Sigma_{t+1}$$

Start bei $x_0 = 0$, $\sigma_0 = 0$, $\Sigma_0 = [0]$, sei $u = \Delta x = 3,2$

Sei $\sigma_u = (|\Delta x|)/10 = 0,32$

$$\Rightarrow \Sigma_u = [\sigma_u^2] = [0,32^2]$$

$$\Rightarrow \underline{x}_1 = 3,2$$

$$\Rightarrow \Sigma_1 = \Sigma_0 + \Sigma_u = [0,32^2]$$

Weiter gelten $H = [1]$ und $\Sigma_z = [\sigma_z^2] = [0,05^2]$

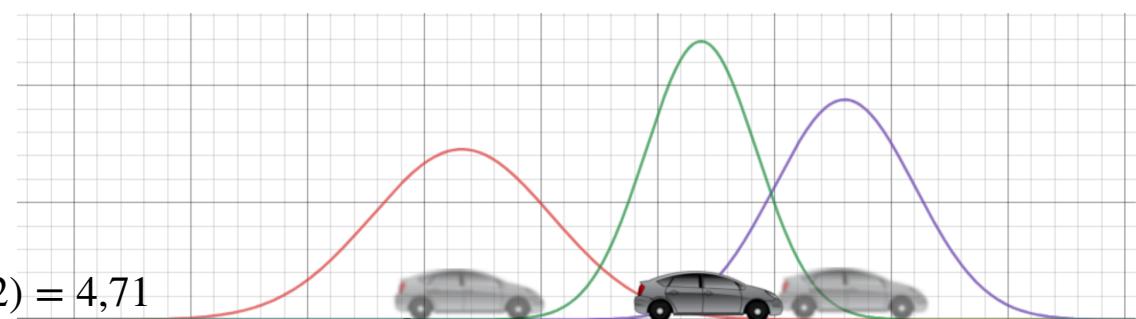
$$\Rightarrow K_1 = (0,32^2)/(0,32^2 + 0,05^2) = 0,97616$$

$$\Rightarrow \Sigma_1 = (I - K_1)\Sigma_1 = [0,00244^2]$$

Sei Messwert $z_1 = 4,75$

$$\Rightarrow \mu_1 = \underline{x}_1 + K_1(z_1 - H\underline{x}_1) = 3,2 + 0,97616(4,75 - 3,2) = 4,71$$

Messung gewinnt, da sie deutlich zuverlässiger ist:
 $\sigma_z \ll \sigma_u$



- Bring nicht-lineare Aktions- und Sensormodelle f und h in das Kalman-Filter-Verfahren ein

Ersetze Zustandsvorhersage $\underline{\mathbf{x}}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}$ durch:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u})$$

Ersetze Messvorhersage (Matrix \mathbf{H}) durch $h(\underline{\mathbf{x}})$, also die Zustandsaktualisierung, durch

$$\mathbf{x}_{t+1} = \underline{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - h(\underline{\mathbf{x}}_{t+1}))$$

- ➡ Ersetze die Matrizen $\mathbf{A}, \mathbf{B}, \mathbf{H}$ durch zeitabhängige Versionen $\mathbf{F}_t, \mathbf{H}_t$ der partiellen Ableitungen von f nach \mathbf{u} und h nach \mathbf{x} (Jacobi-Matrizen)



Gliederung

5. Fortbewegung

1. Einleitung
2. Bewegungsschätzung
- 3. Bayes- und Kalmanfilter**
- 4. Fusion von Odometriedaten**
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick



Gliederung

5. Fortbewegung

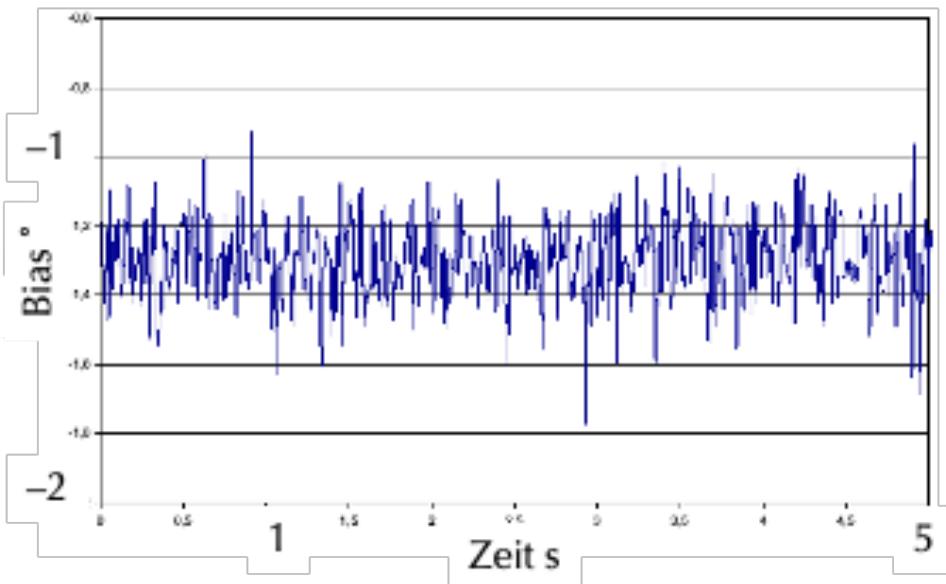
1. Einleitung
2. Bewegungsschätzung
3. Bayes- und Kalmanfilter
- 4. Fusion von Odometriedaten**
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick

Fusion von Odometriedaten

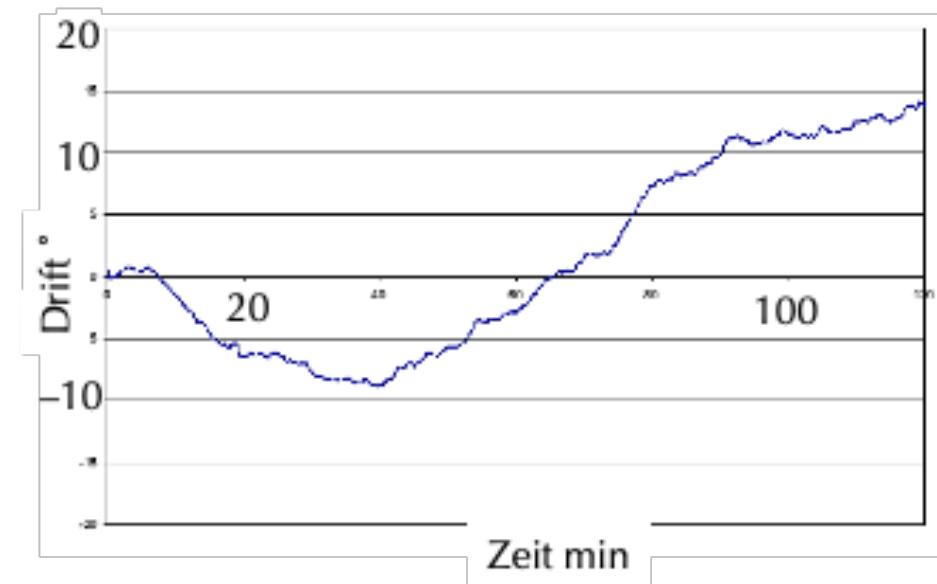
- ▶ Praktisch alle mobilen Roboter nutzen Odometriedaten („fast geschenkt!“)
- ▶ Normalerweise fusioniert mit anderen Messwerten
- ▶ Fusioniere mit solchen Messwerten, die unabhängig sind und anderen systematischen Fehlern unterliegen
- ▶ Fusionierung kann auf unterschiedliche Arten gehen (Kalman-Filter ist eine davon)
- ▶ In diesem Abschnitt Fusion von Odometriedaten mit
 1. Gyro-Daten („Gyrodometrie“)
 2. Laser-Winkelhistogrammen

Gyrofehler - Qualitativ

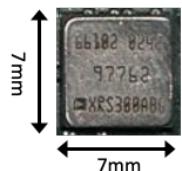
- ▶ Bias-Rauschen des ADXRS300 (Messwerte bei still stehendem Sensor)
- ▶ Bei Fahrt zusätzlich Salt & PepperNoise!



Lokal: Gauß-Rauschen
 $(\mu \approx -1.3^\circ, \sigma \approx 0.1^\circ)$

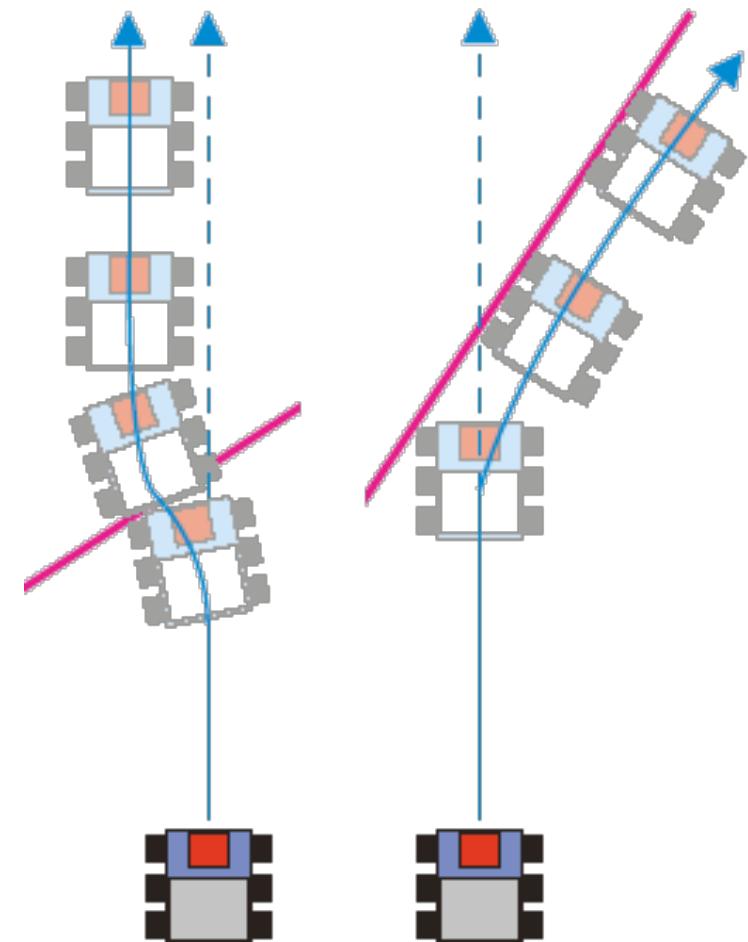


Global:



Gyrodometrie

- ▶ Gyros unterliegen systematischer Drift
- ▶ Kurzfristig sind sie verlässlich, besonders auch bei Drehung
- ▶ Odometrie erzeugt kumulative Messfehler
- ▶ Die größten Messfehler entstehen an „kritischen“ Stellen:
 - Drehungen, „Holpern“, Schlupf
- ▶ Idee: Fusioniere Odo und Gyro
 - Variante 1: Bei Differenz nimm den aktuell „vermutlich verlässlicheren“ Wert
 - Variante 2: Fusioniere Odometrie und Gyro durch Kalman-Filter



Gyrodometrie durch Kalmanfilter

- ▶ ...eigentlich komplett Pose (3D) und Gyro 2D (Messwert und Bias) in EKF
- ▶ Nichtlinearität wegen Gyro-Integration [Solda et al. IAV-2004]
- ▶ Hier vereinfacht:
 - Eindimensional: Beachte lediglich Orientierung nach Gyro (linearer Messwert)
 - Fusion von Odometrie (lineare $\Delta\theta$ -Schätzung aufgrund erfolgter Radumdrehung) mit Gyro-Messwert
 - Bezug auf feste Zeitintervalle Δt , z.B. 1s
- ▶ Entspricht strukturell dem Beispiel Auto mit GPS aus dem vorherigen Abschnitt

Gyrodometrie mit Kalmanfilter (1)

- ▶ Orientierungsschätzung Odometrie (vgl. random walk Folie 130/1):
 - Betrachtung in festen Intervallen von 1s
 - „Aktion“ $u = \Delta\theta = (v_L - v_R)/b$: Drehung innerhalb 1s
 - Transitionsmodell: $\mathbf{A} = \mathbf{B} = \mathbf{1}$
 - Varianz wäre empirisch zu ermitteln, hier „geraten“:
 - Transitions-Kovarianzmatrix:
 - $\Sigma_u = [\sigma_u^2]$, wobei $\sigma_u = (|\Delta\theta| + c)/10$ für festen Transitionsfehleranteil (Drehwinkelrauschen) c
- ▶ Orientierungsschätzung Gyro (empirisch ermittelt):
 - Feste Messfehler (z.B. Konstanter Offset von -1.3°) werden außerhalb des Filters behandelt
 - Standardabweichung $\sigma_z = 0.1^\circ = 0.00175 \text{ rad}$
 - Sensormodell: $\mathbf{H} = \mathbf{1}$
 - Sensor-Kovarianzmatrix: $\Sigma_z = [\sigma_z^2] = [0.00175^2]$

Gyrodometrie mit Kalmanfilter (2)

Erinnerung: $\underline{\mathbf{x}}_{t+1} := \mathbf{A}\mu_t + \mathbf{B}u; \underline{\Sigma}_{t+1} = \mathbf{A}\Sigma_t\mathbf{A}^\top + \Sigma_u \quad \mu_{t+1} = \underline{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z} - \mathbf{H}\underline{\mathbf{x}}_{t+1})$

$$\mathbf{K}_{t+1} = \underline{\Sigma}_{t+1}\mathbf{H}^\top \left(\mathbf{H}\underline{\Sigma}_{t+1}\mathbf{H}^\top + \Sigma_z \right)^{-1} \quad \Sigma_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\underline{\Sigma}_{t+1}$$

Start bei $(0,0,0)$, $\sigma_0 = 0$, $\Sigma_0 = [0]$, sei $u = \Delta\theta = 0.2$

Transitionsfehler $c := 0.1$, also $\sigma_u = (|\Delta\theta| + c)/10 = 0.03$

→ $\Sigma_u = [\sigma_u^2] = [0.03^2]$,

→ $\underline{x}_1 = [0.2], \underline{\Sigma}_1 = \Sigma_0 + \Sigma_u = [0.03^2]$

Weiter gelten $\mathbf{H} = [1]$ und $\Sigma_z = [\sigma_z^2] = [0.00175^2]$ (s.o.), also

→ $\mathbf{K}_1 = [(0.03^2) / (0.03^2 + 0.00175^2)] = [0.99966]$

→ $\Sigma_1 = (\mathbf{I} - \mathbf{K}_1)\underline{\Sigma}_1 = [0.0005532^2]$

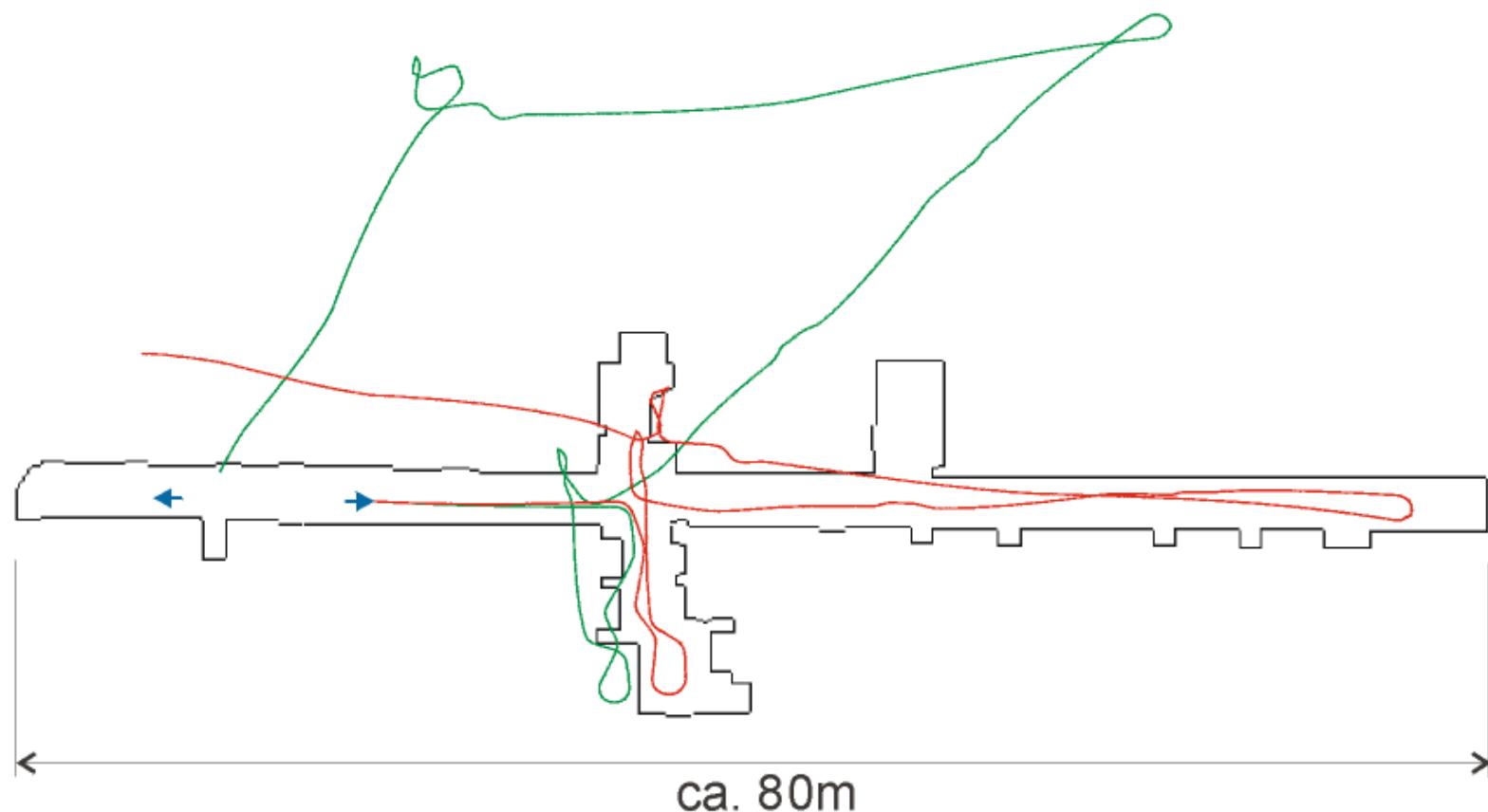
Sei Messwert $z_1 = 0.19$

→ $\mu_1 = \underline{x}_1 + \mathbf{K}_1(z_1 - \mathbf{H}\underline{x}_1) = [0.2] + [0.99966(0.19 - 0.2)] = [0.1900034]$

(Messung „gewinnt“, da sie deutlich zuverlässiger ist: $\sigma_z \ll \sigma_u$)

Beispiel: Odometrie + EKF

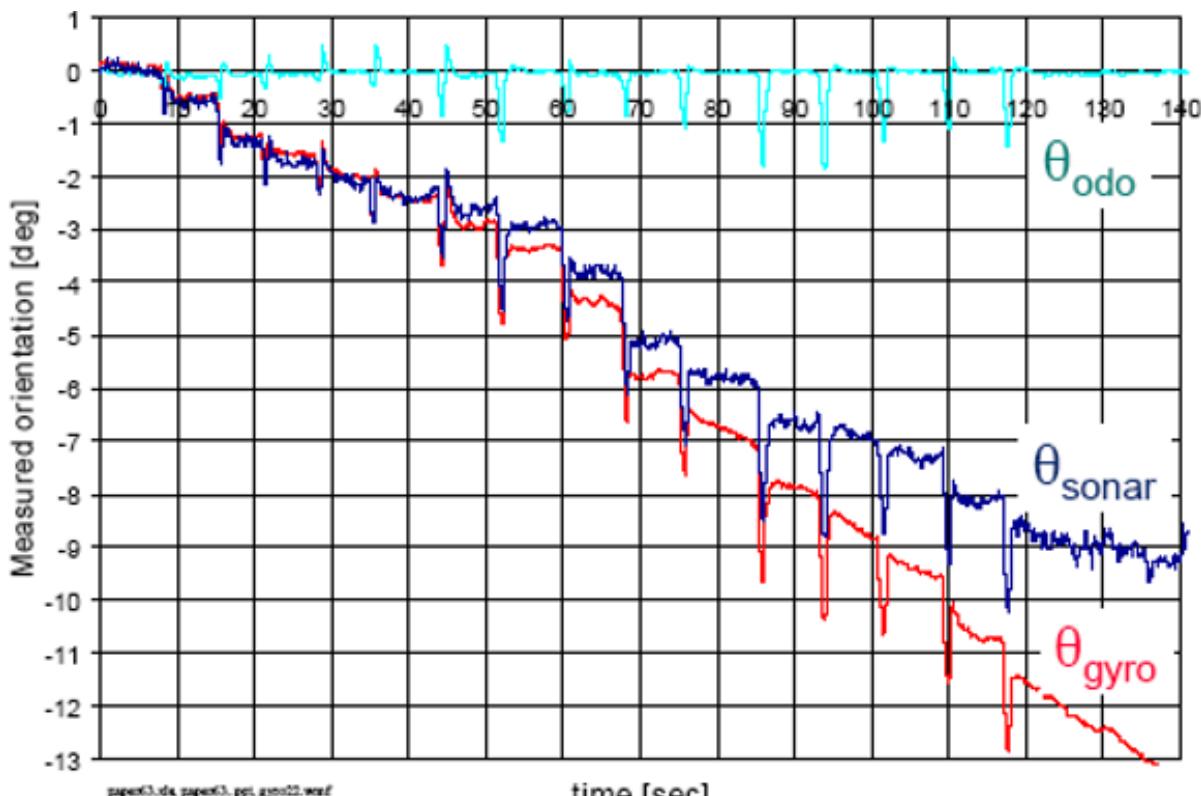
- Positionsschätzung mittels Odometrie
- Positionsschätzung mittels Fusion von Odometrie und Gyroskop



ca. 80m

Fahren über Schwellen

- ▶ 9cm-Kabel schräg zur Fahrt in gleichen Abständen gelegt



Sonar / Wandabstand als **genuine truth** bzgl. Orientierung

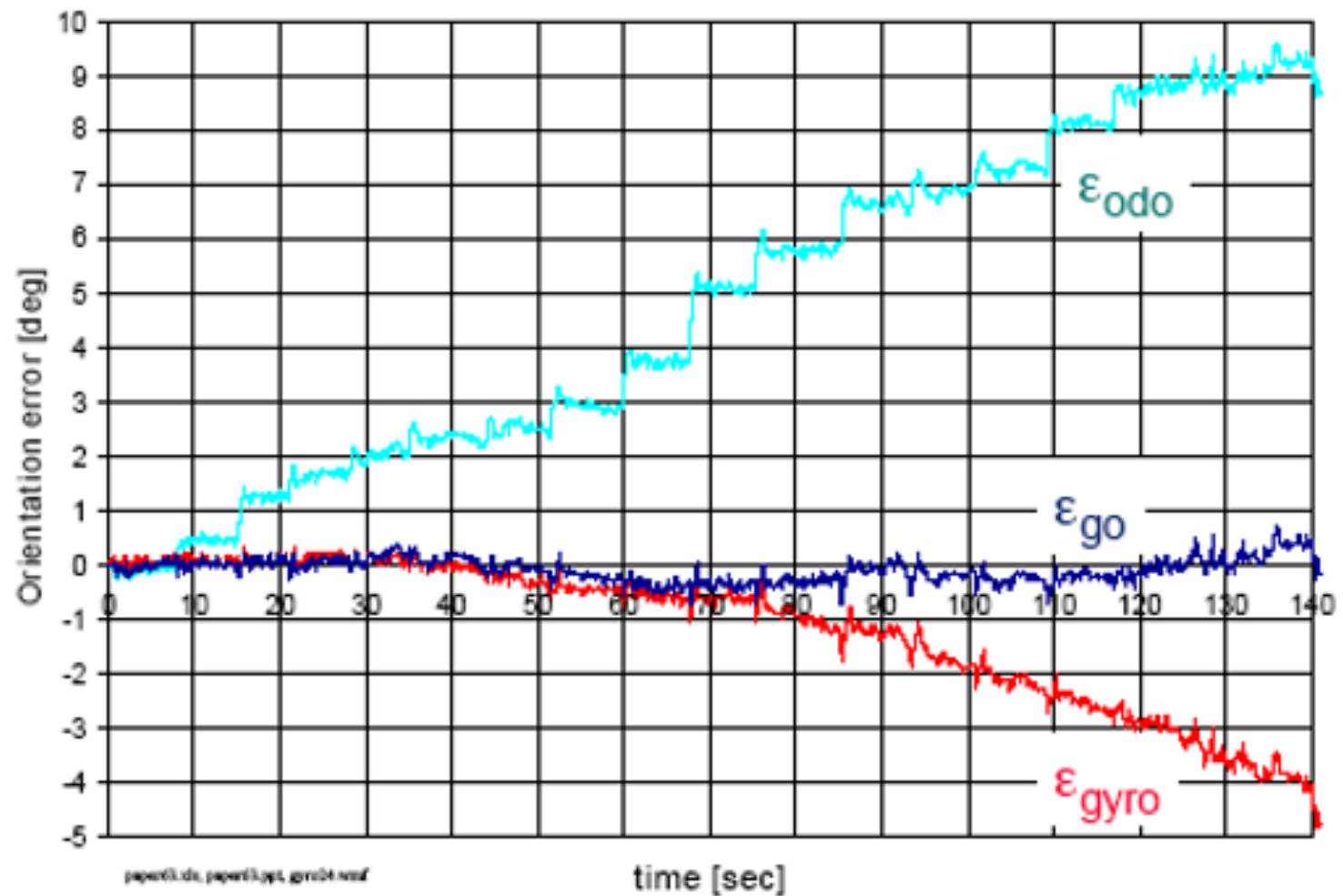
Fahrt mit 0,1m/s = 14m

Deterministische Korrektur

```

if  $(|\Delta\theta_{\text{gyro},i} - \Delta\theta_{\text{odo},i}| > \Delta\theta_{\min})$ 
then  $\theta_i := \theta_{i-1} + t\Delta\theta_{\text{gyro},i}$ 
else  $\theta_i := \theta_{i-1} + t\Delta\theta_{\text{odo},i}$ 

```



Eingabe : Rotationsschätzungen $\Delta\theta_{\text{gyro}}$, $\Delta\theta_{\text{odo}}$. Feste Schwellwerte γ_{odo} , γ_{gyro} für Odometrie resp. Gyroskop, sowie eine Aktualisierungs-Gewichtung α .

Ausgabe: Fusionierte Orientierung θ_{t+1} , Update der Schätzung von $\varepsilon_{\text{gyro}}$.

- 1: **if** $|\Delta\theta_{\text{odo}}| > \gamma_{\text{odo}}$ **or**
 $|\Delta\theta_{\text{gyro}} - \varepsilon_{\text{gyro}}| > \gamma_{\text{gyro}}$ **then** // Kurvenfahrt laut Odometrie o. Gyroskop
 - 2: $\Delta\theta_{\text{gyro}} = \Delta\theta_{\text{gyro}} - \varepsilon_{\text{gyro}}$ // Drift korrigieren
 - 3: $\theta_{t+1} = \theta_t + \Delta\theta_{\text{gyro}}$ // Winkel nach Gyro
 - 4: **else** // Fahrt relativ geradeaus
 - 5: $\varepsilon_{\text{gyro}} = \alpha \cdot \varepsilon_{\text{gyro}} + (1 - \alpha) \cdot (\Delta\theta_{\text{gyro}} - \Delta\theta_{\text{odo}})$ // Drift aktualisieren
 - 6: $\theta_{t+1} = \theta_t + \Delta\theta_{\text{odo}}$ // Winkel nach Odometrie
 - 7: **end if**
 - 8: **return** $\theta_{t+1}, \varepsilon_{\text{gyro}}$
-

Fazit Gyrodometrie

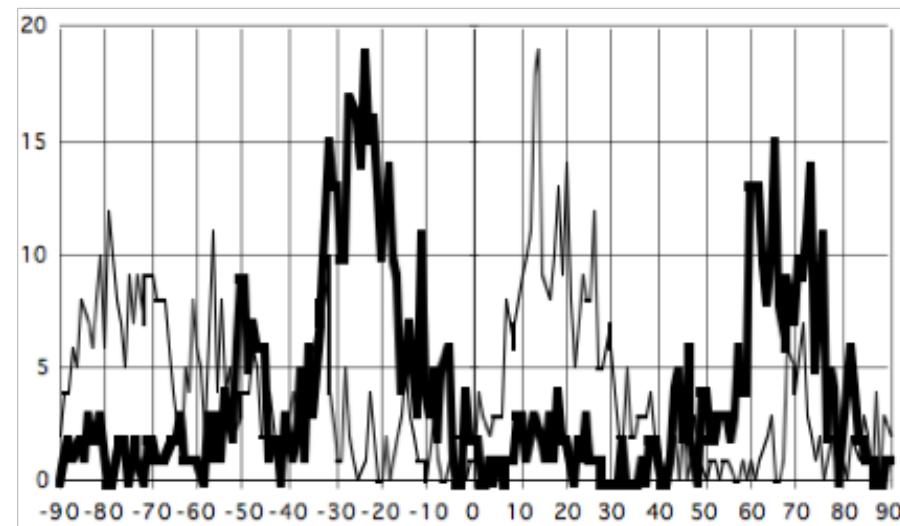
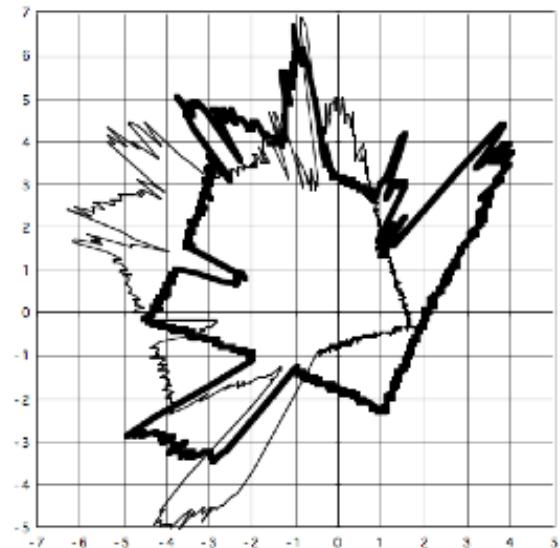
- ▶ Hoch plausible Ergänzung einfacher Odometrie
- ▶ Im deterministischen Fall auch genauso simpel
- ▶ Vorsicht bei echten Billig-Gyros (KURT): Bei Fahrt tritt unvorhersehbarer Salt-and-Pepper-Noise auf!
- ▶ Deterministische Gyrodometrie in der Literatur relativ selten aufgegriffen
- ▶ Kalman-Filterung von Odo+Gyro ist ein Standard
- ▶ Moderne IMUs liefern meist viel bessere Daten als im Rechenbeispiel hier vorausgesetzt!
- ▶ IMU-Daten sind hochgradig nicht-linear, daher in der Regel EKF

Orientierung - Laser-Winkelhistogramme

- ▶ Innerhalb und nahe bei Gebäuden dominieren Rechte Winkel von Wänden
- ▶ Ein Laserscan (kartesische Darstellung, idealisiert) enthält dann orthogonale „Hauptrichtungen“, zu denen viele Verbindungslien benachbarter Messpunkte parallel sind
- ▶ Diese Richtungen sind bewegungsinvariant
- ▶ Folglich können sie zur Korrektur von Orientierungsfehlern verwendet werden
- ▶ Fusion dieser Orientierungsinformation mit anderen über Kalman-Filter oder über deterministisches Verfahren
- ▶ („Bei klaren Hauptrichtungen und hinreichend hoher Scan-Frequenz glaube der Information aus den Winkelhistogrammen!“)

[Weiß, Wetzler & v. Putkamer, 1994]

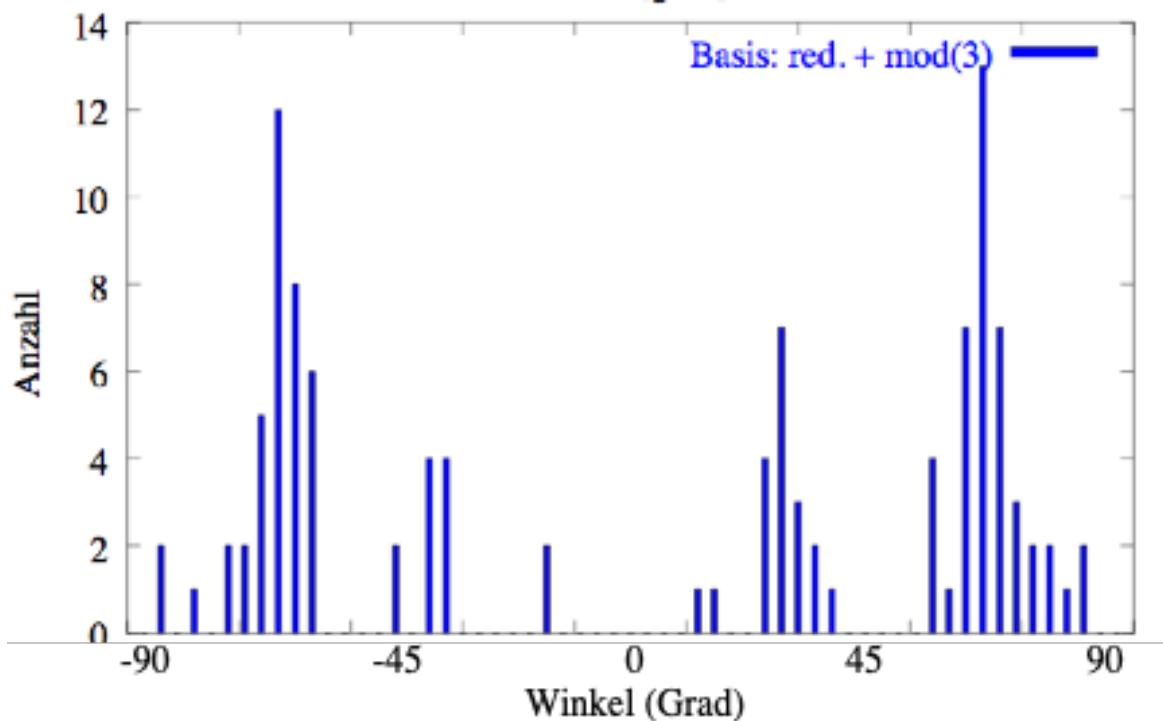
Scans und ihre Winkelhistogramme



- ▶ Je Scan verbinde benachbarte Punkte mit Linie
- ▶ Mach Winkelhistogramm durch Zählen vorkommender Orientierungen in diskreten Winkeln geschätzt im globalen Bezugssystem
- ▶ Korrigiere Orientierungsschätzung durch Matchen von Peaks

Diskrete Winkelhistogramme

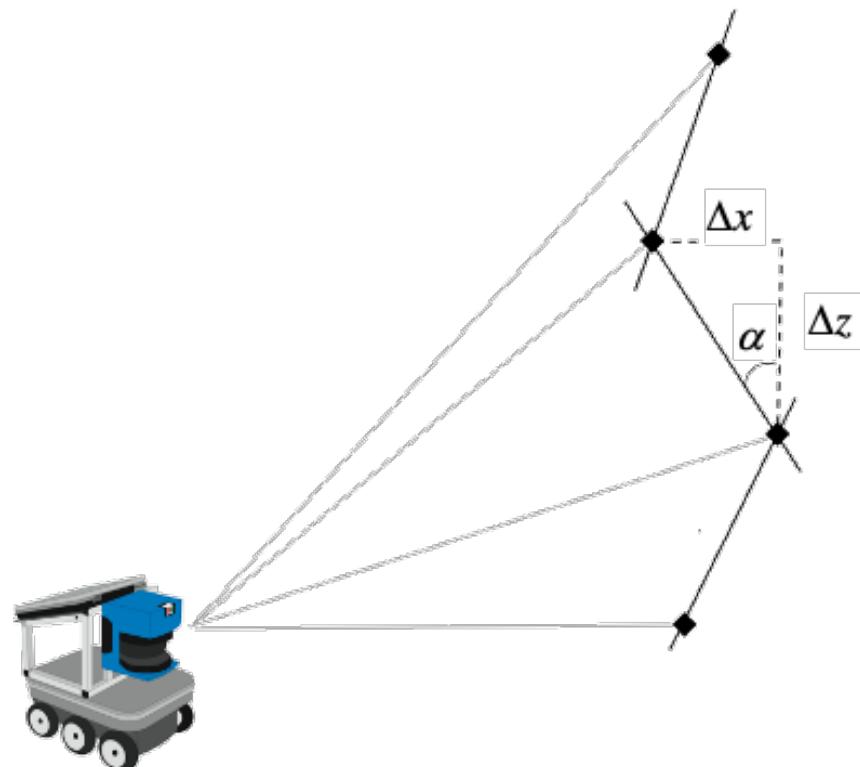
- ▶ Sammle vorgefundene Orientierungen auf in „Päckchen“ (bins) ähnlicher Winkel
 - Große bins: Klare Häufung & hoher Diskretisierungsfehler
 - Kleine bins: Schwammige Histogramme & hohe Orientierungsauflösung



180° Winkelhistogramm
mit bin size 3°

Berechnung der Winkel

- ▶ Für Winkel α_i an Punkt (x_i, z_i) gilt bzgl. (x, z) Roboter/Scanner-Bezugssystem (Winkel 0 an z -Achse, α im Uhrzeigersinn):



$$\alpha_i = \arctan\left(\frac{\Delta x_i}{\Delta z_i}\right) = \arctan\left(\frac{x_{i+1} - x_i}{z_{i+1} - z_i}\right)$$

Warum ist das so?:

Weil (s. Koordinatensysteme, aber Roboterkoordinatensystem!):

- $\tan(\alpha) = \sin(\alpha)/\cos(\alpha) = \text{Gegenkathete}/\text{Ankathete} = \Delta x_i/\Delta z_i$
- $\alpha = \arctan(\Delta x_i/\Delta z_i)$

Ermitteln der Winkelverschiebung

- ▶ Seien $\mathcal{G}, \mathcal{H}, H$ Histogramme mit identischer Diskretisierung
- ▶ $\mathcal{G}(j)$ Füllung des j -ten bins.

$$K_i(\mathcal{H}, \mathcal{G}) = \sum_{j=0}^n \mathcal{H}(j) \cdot \mathcal{G}((j+i) \bmod (n+1))$$

- ▶ Kreuzkorrelation zwischen \mathcal{H} und dem um i Bins verschobenen \mathcal{G}

Optimale Übereinstimmung ergibt sich bei Verschiebung
(in Vielfachen der *bin*-Auflösungen) von

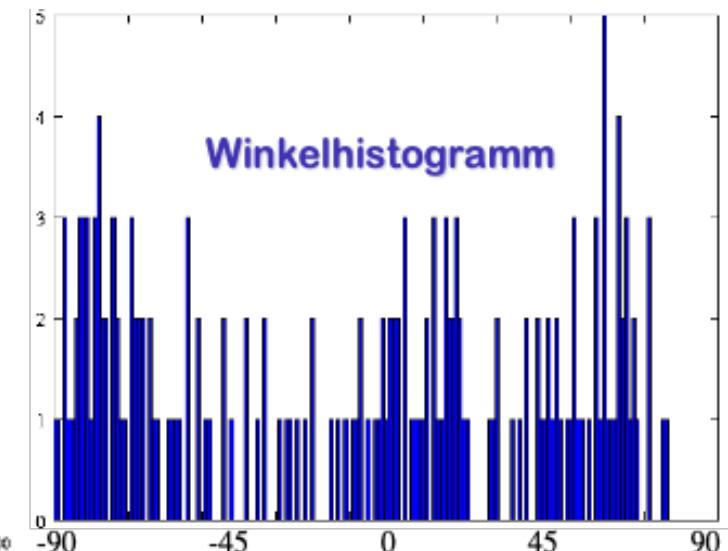
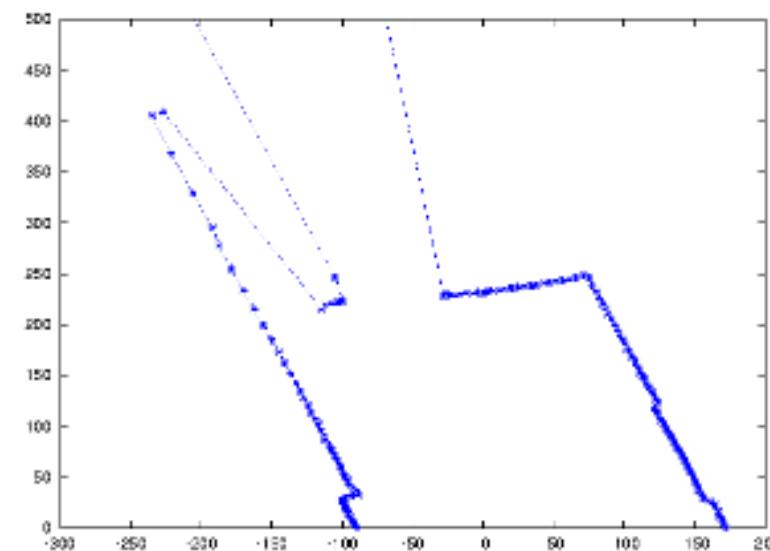
$$\Delta\theta = \arg \max_i K_i(\mathcal{H}, \mathcal{G})$$

Naive Umsetzung

Blick auf die Umgebung

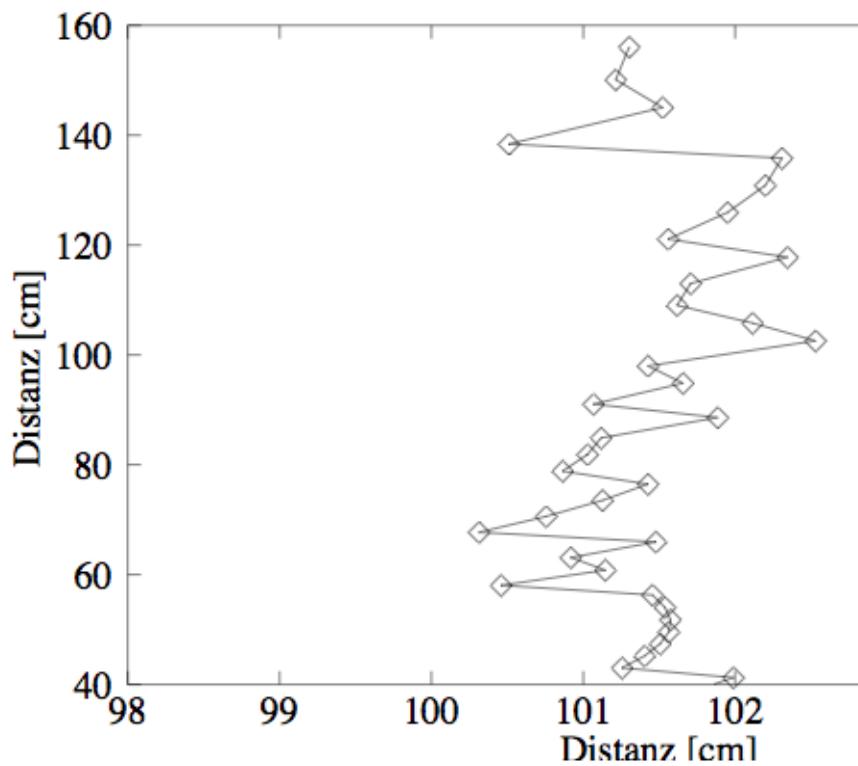


Roboterperspektive



Was ging hier schief?

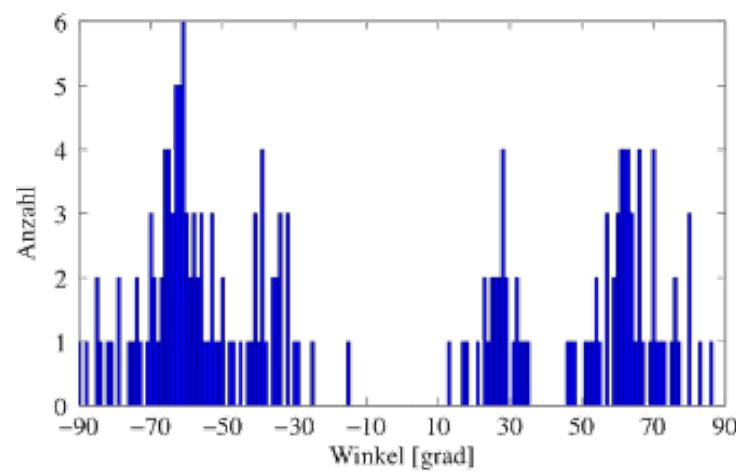
Sensorrauschen zerstört Histogramme



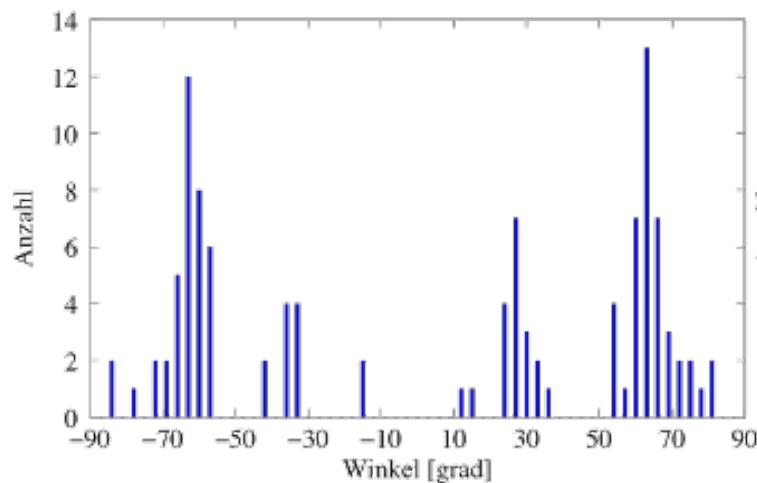
Scan einer planaren Wand

- ▶ Mögliche Abhilfen:
 - Winkelauflösung vergrößern
 - Punktmenge filtern
 - m aus n weglassen
 - Reduktions/Medianfilter
 - Linienfilter + Resampling verwenden
 - ...

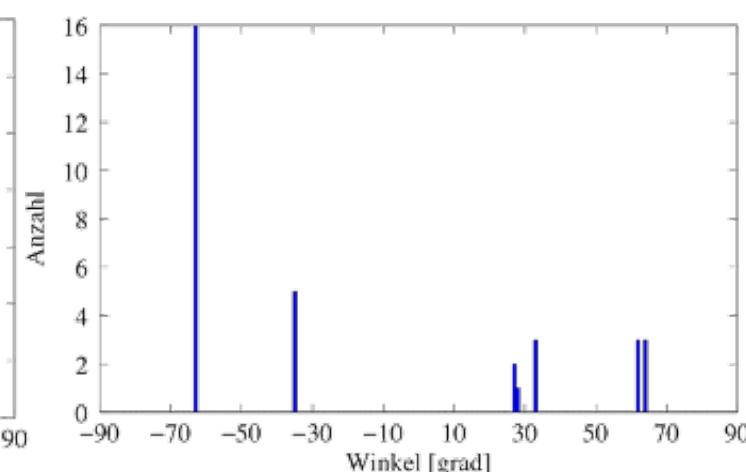
Filtern von Histogrammen



Original-Histogramm,
Winkelauflösung 1°

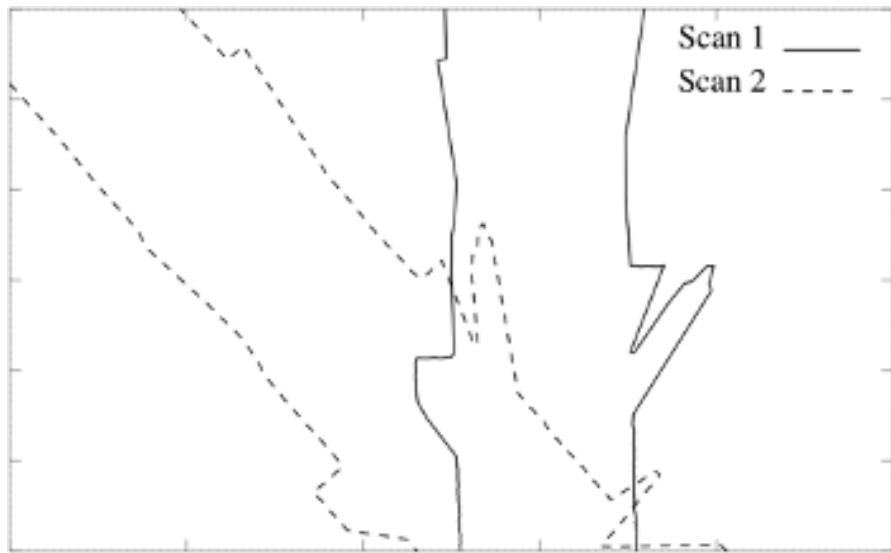


Reduktionsfilter,
Bin-Size 3°

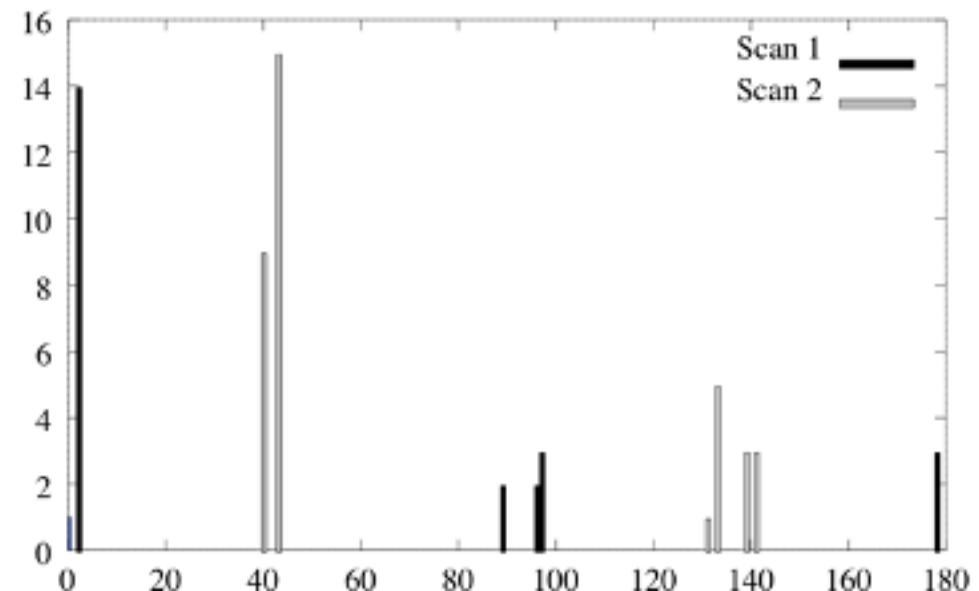


Linienfilter,
Bin-Size 3°

Beispiel



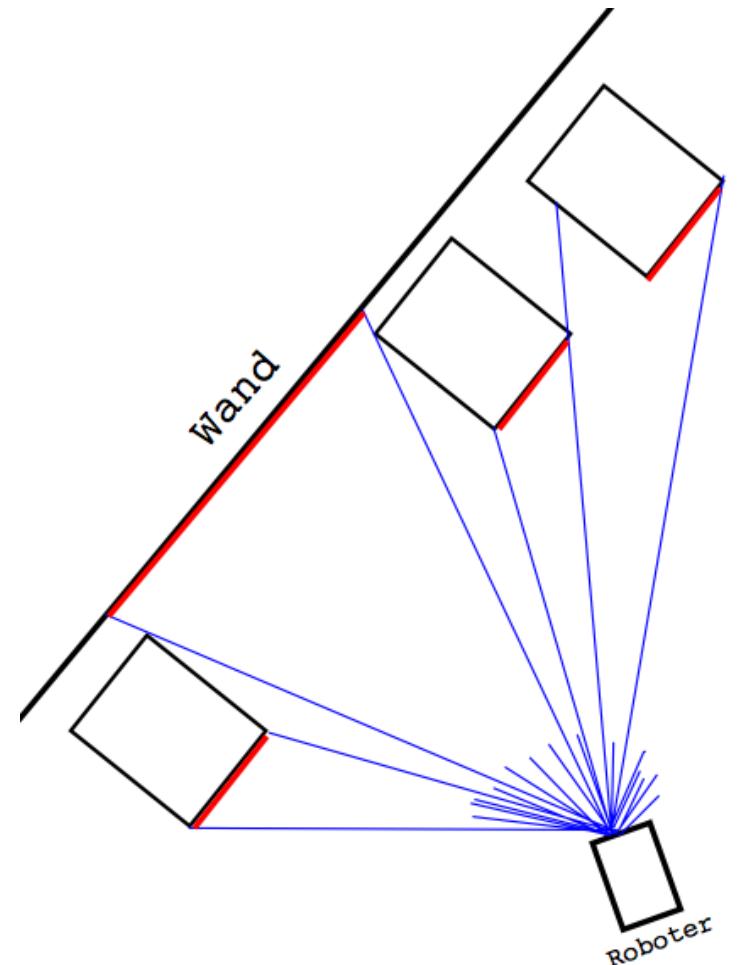
Zwei 2D-Laserscans, aufgenommen von derselben Position mit unterschiedlicher Rotation



Zugehörige Winkelhistogramme nach Reduktion und Filterung

Fazit Winkelhistogramme

- ▶ Im Umfeld von Gebäuden hoch plausible Orientierungs-Hilfe
- ▶ Fusion mit Odometrie wiederum deterministisch oder mit Kalman-Filter
- ▶ Korrigiert damit die schwächste Seite der Odometrie
- ▶ Billig zu berechnen
- ▶ Klappt auch, wenn nicht Wände, sondern wandparallele Strukturen sichtbar
- ▶ In der Literatur selten
- ▶ aufgegriffen



Ende Kapitel 5

- ▶ Schätzung der Eigenbewegung aufgrund von Vorwärts-kinematikmodell/Odometrie („Ich weiß doch, wie ich meine eigene Fortbewegung angesteuert habe!“)
- ▶ In dieser Vorlesung praktisch nur für differenzialgetriebene Radfahrzeuge
- ▶ Eigenbewegungsschätzung ist Grundlage für Tracking
- ▶ Da sie ungenau ist, fusioniere mit anderen Sensordaten
- ▶ Fusion geht mit probabilistischen Verfahren – Bayes-Filter sind Stand der Technik
- ▶ Kalman-Filter ein verbreiteter Spezialfall
- ▶ Zu fusionierende Daten kommen von Bewegungssensoren (Gyro, IMU) und/oder aus Umgebungssensoren (Laserscanner)

Inhalt



Gliederung

1. Einleitung
2. Robot Operating System
3. Sensorik
4. Sensordatenverarbeitung
5. Fortbewegung
- 6. Lokalisierung**
7. Mapping
8. Navigation
9. Ausblick



Gliederung

- 6. Karten
- 1. Karten
- 2. Triangulation
- 3. Lokalisierungsalgorithmen
- 4. Sensorik
- 5. Fortbewegung
- 6. Lokalisierung
- 7. Mapping
- 8. Navigation
- 9. Ausblick

Formen von Lokalisierung

- ▶ Triviale Lokalisierung
 - keine mobilen Einheiten ➡ kein Lokalisierungsproblem („Lokalisierung“ = momentane interne Pose: z.B. Automationsroboter)
- ▶ Keine Lokalisierung
 - Kenntnis der Pose unnötig/unerwünscht ➡ keine Lokalisierung („verhaltensbasierte“ Robotik, random-walk-Staubsauger)
- ▶ Inkrementelle Lokalisierung (relative L., Lokale L., tracking)
 - Ermittle Pose-Änderung relativ zu Startpose oder zu Zwischenposen
 - Haben wir in Kapitel 5 gemacht
- ▶ Globale Lokalisierung (Absolute L.)
 - Ermittle Pose in externem Bezugssystem („Karte“, Koordinatensystem)



Inhalt

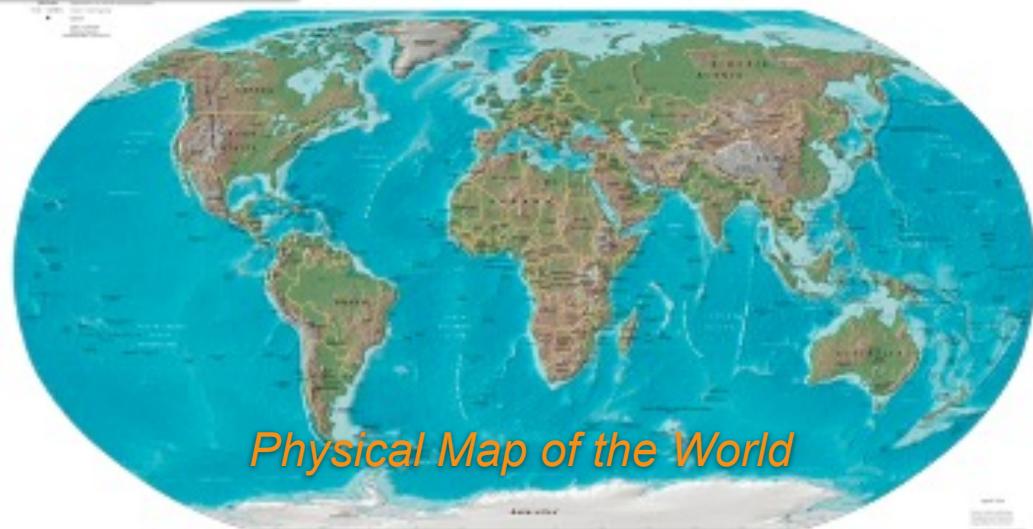


Gliederung

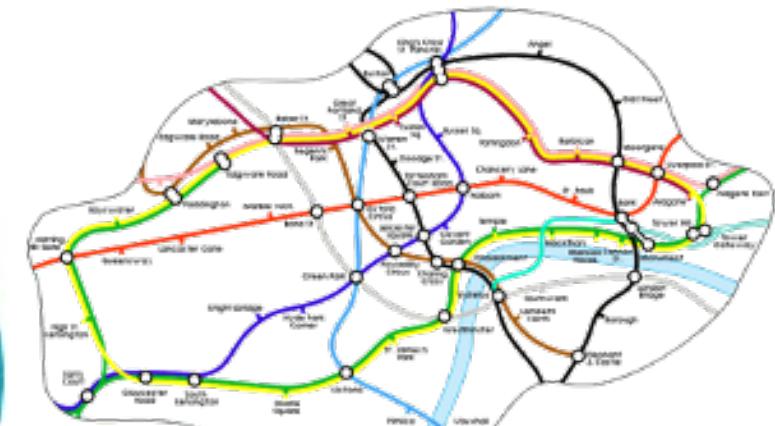
- 6. Karten
- 1. Karten
- 2. Triangulation
- 3. Lokalisierungsalgorithmen
- 4. Sensorik
- 5. Fortbewegung
- 6. Lokalisierung
- 7. Mapping
- 8. Navigation
- 9. Ausblick

Karten

Es gibt nicht die Art Karte,
sondern endlos viele davon!



Eine Roboterkarte ist eine explizite, persistente Repräsentation des Raums, die der Roboter für seine Steuerung nutzt.



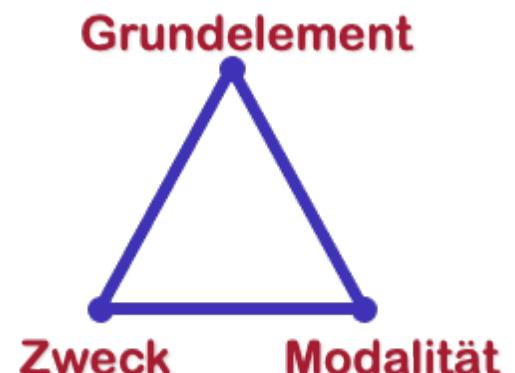
Londoner U-Bahnnetz



Ebstorfer Weltkarte

Wie und was repräsentieren Karten?

- ▶ Für Roboterkarten (2D, 3D, ... gilt):
 - Drei Charakteristika für Karten (mindestens!)
 - Konkrete Roboterkarten sollen möglichst hohen Gesamtnutzen haben
 - Oft braucht man „mehr als eine Karte“ derselben Umgebung („hybride Repräsentation“ ➔ Integrationsproblem!)
 - Raumrepräsentation (einschließlich effizienter Inferenzverfahren) in der Robotik erstaunlich schwach entwickeltes Forschungsgebiet! (➔ Arbeitsthemen!)



Beispiele für 2D-Roboterkarten

- ▶ Roboterkarten ergeben sich aus dem Kreuzprodukt Modalität / Element

Modalität Element	Sensordaten	syntaktische Merkmale	semant. Merkmale Roboterbspl. Kap.8!
metrisch kontinuierlich			
metrisch diskret			
topologisch			

Modalitäten der Raumrepräsentation

- ▶ **Metrisch kontinuierlich** (geometrische Belegungen im Raum)
 - Repräsentiere (x, z, θ) bzw. $(x, y, z, \theta_x, \theta_y, \theta_z)$ numerisch
 - Vorteil: Präzision, oft kompakte Repräsentation
 - Nachteil: Rechnen im Kontinuierlichen (z.B. W'dichten)
- ▶ **Metrisch diskret** (Raum „an sich“ und Belegungen)
 - repräsentiere (x, z) bzw. (x, y, z) diskret („Fliesen“, „Voxel“)
 - Vorteil: Uniformität, oft einfache Algorithmen
 - Nachteil: Diskretisierungsfehler
- ▶ **Topologisch** (i.d.R. diskret)
 - repräsentiere „Landmarken / Orte“ + „Wege dazwischen“
 - Vorteil: kompakte Repräsentation, einfache Algorithmen
 - Nachteil: Diskretisierungsfehler, viel Rauminfo. fehlt

Elemente der Raumrepräsentation

- ▶ **Sensordaten** (Messwerte z.B. Laser-Punkte, Bilder)
 - Vorteil: Daten unverfälscht, vielfach interpretierbar
 - Nachteil: i.d.R. viel zu hohes Datenvolumen
- ▶ **Syntaktische Merkmale** („semantikfrei“ verarbeitet, z.B. Linien)
 - Vorteil: Komprimierung gegenüber Rohdaten,
 - Standard-Vorverarbeitung spart Zeit
 - Nachteil: ggf. fällt interessante Roh-Information weg
- ▶ **Semantische Merkmale** (z.B. Objektklassen)
 - Vorteil: Extrem hohe Komprimierung,
 - Speicherung in Termini der Nutzungskategorien
 - Erlauben Verknüpfung mit Wissensrepräsentation
 - Nachteil: Originaldaten nicht im Ansatz rekonstruierbar, wenn Nutzung sich ändert

Zwecke von Karten

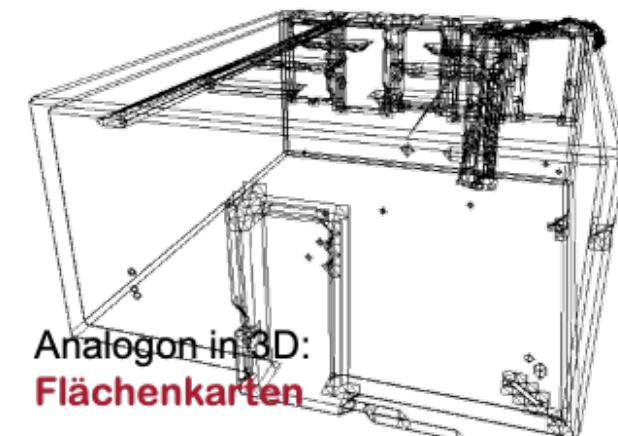
- ▶ **Lokalisierung**
 - Optimiere: Eindeutiges Matching von Sensordaten in Karte
 - Kartenelemente: „Oberflächen“ (z.B. Linien/Flächen bei Abstandssensoren)
 - Texturen / Bildfeatures bei Kameras
- ▶ **Pfadplanung** (hier Fahrpfade, allg. auch Greiftrajektorien)
 - Optimiere: Rechenzeit für Planung kollisionsfreier Trajektorien von Start- nach Zielpose
 - Kartenelemente: „Freiraum“-Regionen in 2D oder 3D
- ▶ **Diverse Anwendungszwecke**
 - Optimiere: z.B. Präsentation für Nutzer, Archivierung, Auswertung
 - Kartenelemente: Objekte/Kategorien der Anwendung
 - (z.B. ATV-Schadensklassen bei Kanalinspektion)

Linienkarten

- ▶ Lokalisierung (Kanten in Scanebene markiert)



- ▶ Metrische Lokalisierung mit Entfernung-Sensoren (Laser)
- ▶ Linien nur im Messbereich! (z.B. Scanebene)
- ▶ Voraussetzung: Messwerte effizient auf Linien matchbar



Beispiele Zwecke

- ▶ Geoinformationssysteme
- ▶ Alle raumbezogenen und nominalen Daten, die für eine Domäne relevant sind

