Kapitel 3 Sensordatenverarbeitung

Worum geht es hier?

- Sensordaten sind i.A. fehlerhaft und (zu) viele
- Verfahren zur direkten Daten- und Fehlerreduktion
- Geringer Rechenaufwand Voraussetzung (online-fähig)
  - Rechenintensive Verfahren auf großen/riesengroßen Datensätzen in off-line post-processing (später ab Kap.6)
- im Groben zwei Klassen:
  - **Filter** (primär Fehlerkorrektur): Werfen bewusst Daten weg!
  - **Merkmalsdetektoren** (primär Datenreduktion): Aggregieren Daten
- Verwende später (gefilterte) Originaldaten und/oder aggregierte Merkmale
- Hier: Entfernungsdaten (Laserscans), Bilddaten

3.1 Entfernungsdaten

Reduktionsfilter:

Solange für Sequenz von Messwerten $a_i,\ldots,a_j$ gilt: $\left\| a_i \;\; a_j \right\| \qquad j \quad i$
ersetze $a_i,\ldots,a_j$ durch

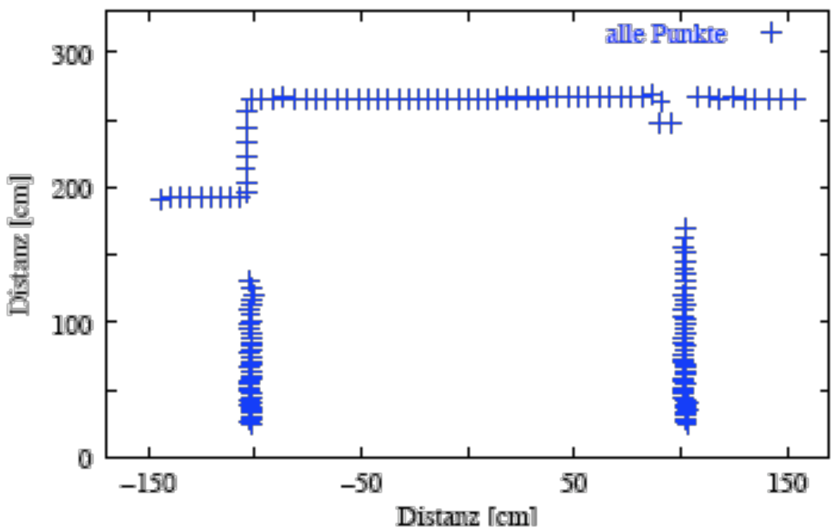
Ersetze eine Schar nah aneinander liegende Messwerte durch ihren Mittelwert

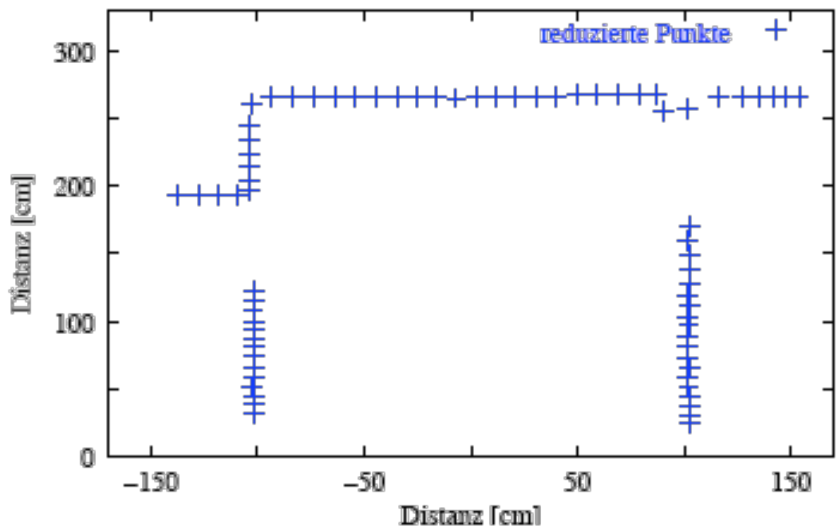
Voraussetzung:

Messwerte geordnet (wie bei Laserscanner), Metrik ll.,.ll auf Messwerten; δ Schranke (hängt mögl. ab von Punkzahl)

Eigenschaften des Reduktionsfilters

Beispiel





- Falls N Messwerte geordnet vorliegen (Laserscanner!), Zeitkomplexität $O(N)$
- Reduktionsfilter glättet als Nebeneffekt Messwertrauschen
- Ausreißer bleiben erhalten

Medianfilter

Ersetze jeden Messwert durch den Median seiner Umgebung aus k Werten

Medianfilter:

Ersetze für alle i den Wert a_i durch

$$\text{Median}\left(a_{i \quad k/2}, \ldots, a_i, \ldots, a_{i+ \; k/2}\right)$$

Voraussetzung:

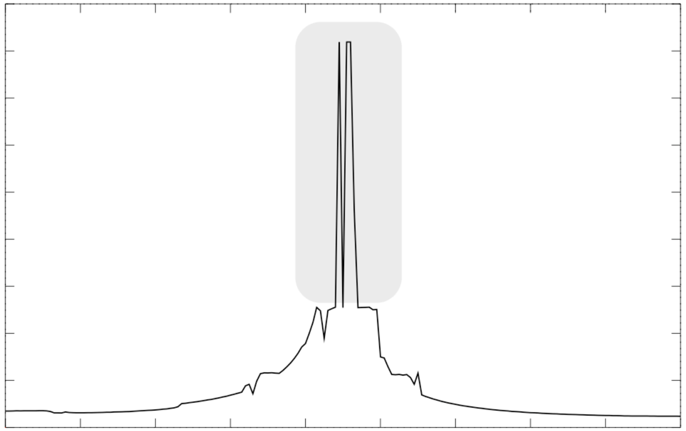
N Messwerte; Metrik auf Messwerten (z.B. Euklidische Distanz)

- Für N Messwerte Zeitkomplexität $O(N \cdot k \cdot \log k)$
  (wg Sortierung nach Distanz)
- Medianfilter verfälscht „echte“ Sprünge in $<k$ Messwerten!

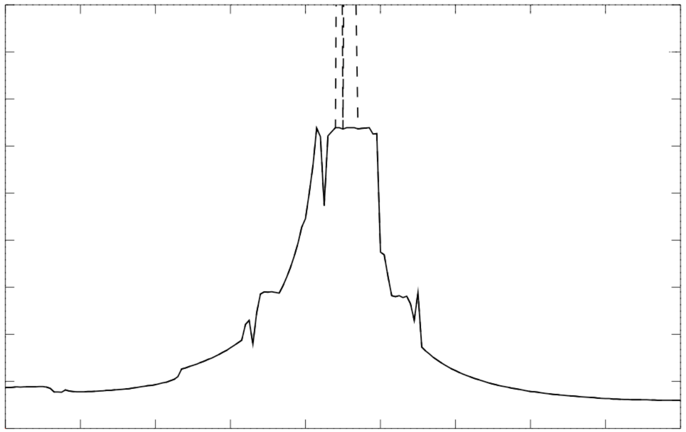
Beispiel Medianfilter

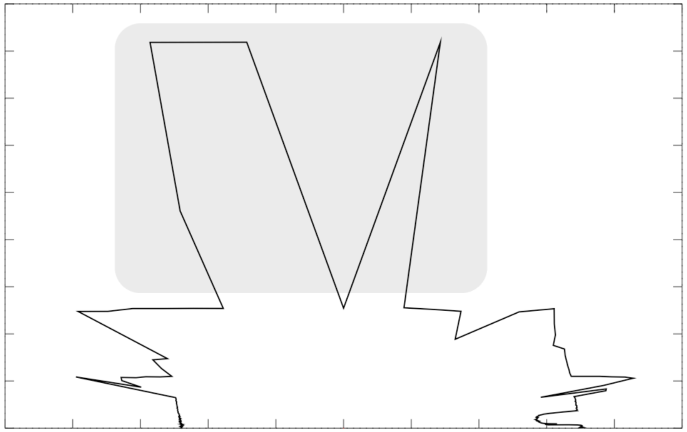
Originaler Laserscan

Scan nach Medianfilterung

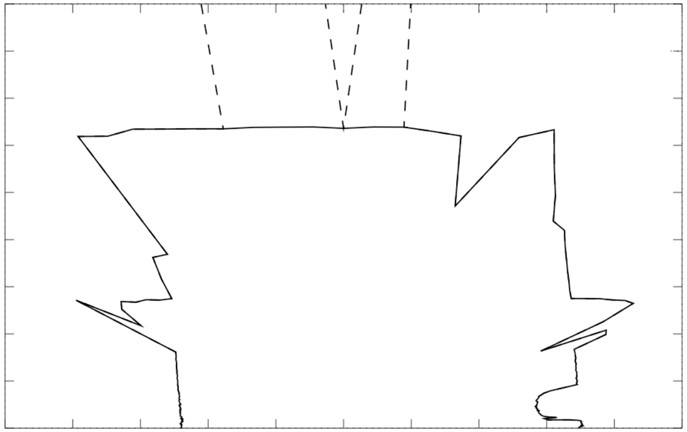


polar



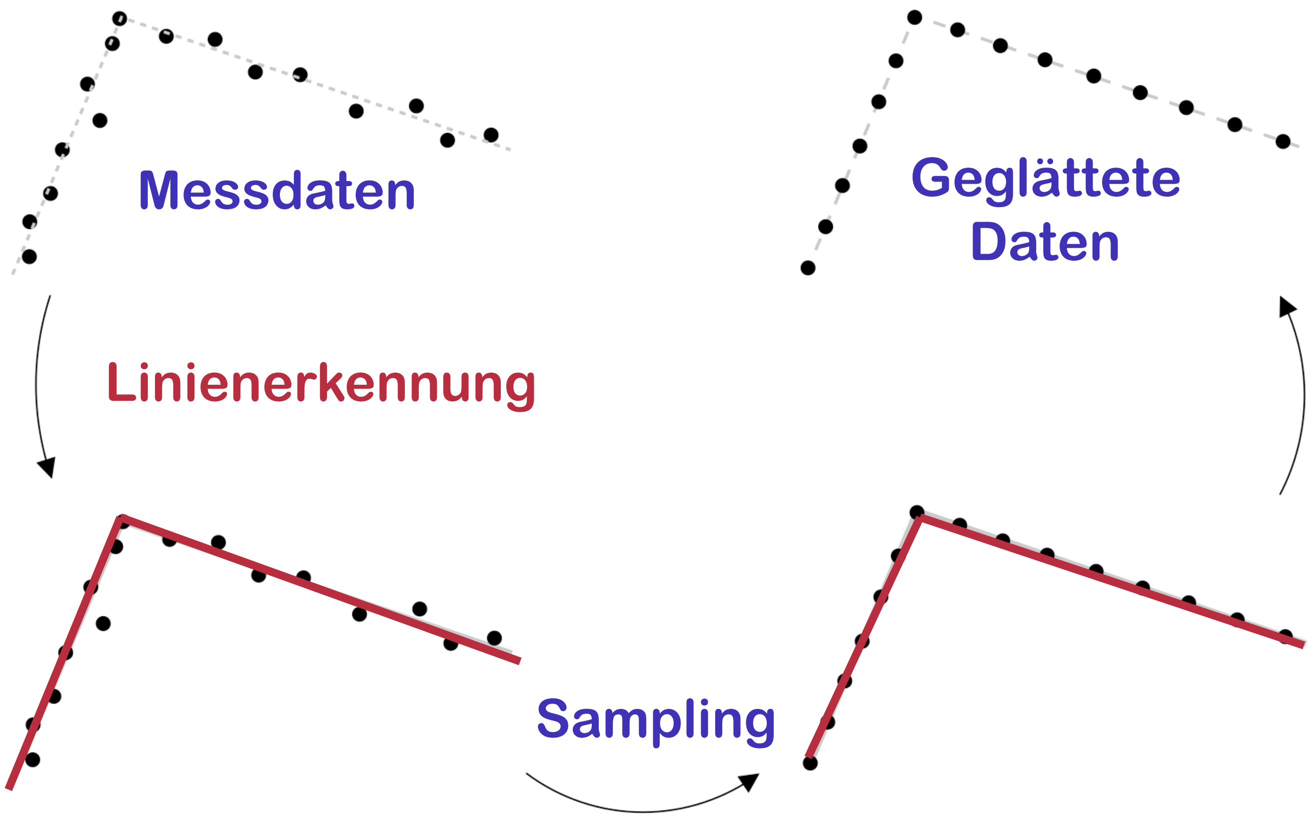


kartesisch



Merkmale auf Entfernungsdaten: Linien

Linien können als Merkmale dienen und außerdem als Rauschfilter verwendet werden!



Linien #1: Online-Linienkonstruktion

Idee für einen online-Linienfinder:

- „online" in dem Sinn:
  Punkte brauchen zu Beginn noch nicht alle vorzuliegen
- Für Punkte eines Scans prüfe in Scanreihenfolge, ob sie mit max. ε Abweichung auf Linie mit Vorgängern liegen
- (wendet man zuvor Reduktionsfilter an, ist Messrauschen reduziert!)
- Wenn ja, verlängere Linie um neuen Punkt;
  wenn nein, schließe vorige Linie und mach eine neue auf
- Linie gilt als gefunden, wenn $\geq n$ Punkte auf Linie (z.B. $n=3$)

Lokale Kriterien für Kollinearität

||…||=eukl. Abstand
$1 > \varepsilon,\ \varepsilon(k) > 0$

Sei $a_j, \ldots, a_k$ die bisher aktuell konstruierte Linie …

- Mit neuem Punkt darf Streckensumme nur wenig von Luftlinie abweichen

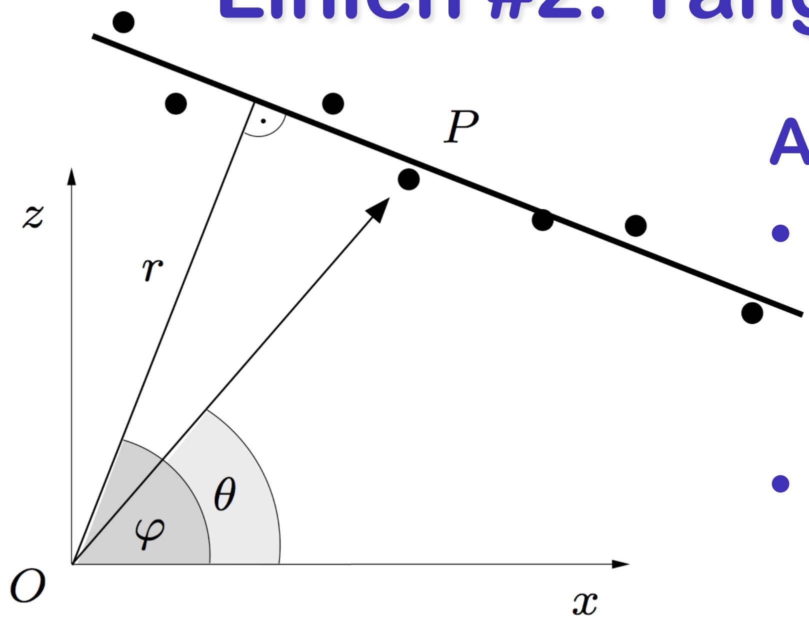
$$[1 \geq] \frac{\left\| a_j, a_{k+1} \right\|}{\sum_{i=j}^{k} \left\| a_i, a_{i+1} \right\|} \geq 1 - \varepsilon(k)$$

⊕ „Luftlinien-Argument" lokal am Ort der Linienerweiterung (typisch $\varepsilon \approx 0.2$)

$$[1 \geq] \frac{\left\| a_{k-1}, a_{k+1} \right\|}{\left\| a_{k-1}, a_k \right\| + \left\| a_k, a_{k+1} \right\|} \geq 1 - \varepsilon$$

⊕ Direkter Abstand $\left\| a_k, a_{k+1} \right\|$ und/oder Abstand $\left\| a_{k-1}, a_{k+1} \right\|$ darf festes Maximum nicht überschreiten
(„Keine Linie durchs Nichts!")

Linien #2: Tangentiallinien nach Lu/Milios

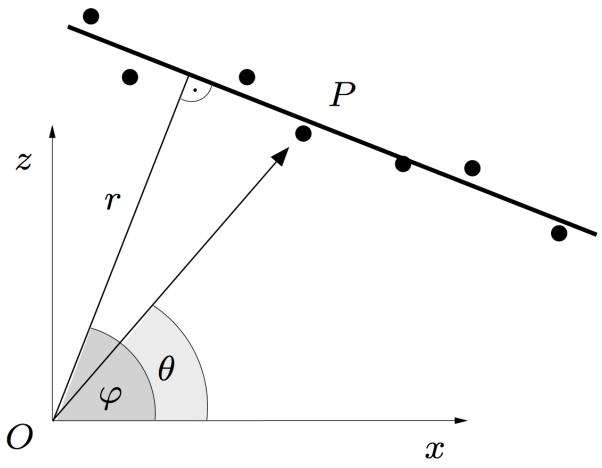


Ausgangslage:

- Für Punkt P unter Winkel θ suche **Regressionsgerade** durch je $(k–1)/2$ Nachbarpunkte rechts und links
- Normalen-Distanz auf die Linie ist r; $|\theta–\varphi|$ ist Winkeldifferenz zwischen Strecke OP und Normaler

Kriterien:

- $|\theta–\varphi|$ darf Maximalwert nicht überschreiten (andernfalls „flacher" Scanwinkel oder Entfernungssprung in Daten)
- Summe der Distanzquadrate zwischen den k Punkten darf Maximalwert nicht überschreiten



Linienlösung nach Lu/Milios

Lemma (Lu/Milios): Die gesuchte Reg.-Gerade ist die, welche folgenden Fehlerterm über die k Nachbarpunkte (x_i,z_i) von P minimiert:

$$E_{\text{fit}}(r,\varphi) = \sum_{i=1}^{k} \left(x_i \cos\varphi + z_i \sin\varphi - r\right)^2$$

Und es existiert eine geschlossene Lösung für $\min_{(r,\varphi)} E_{\text{fit}}$:

$$\varphi = \tfrac{1}{2} \arctan \frac{-2S_{xz}}{S_{zz} - S_{xx}}$$

$$r = \bar{x}\cos\varphi + \bar{z}\sin\varphi$$

Dabei sind:

$$\bar{x} = \frac{1}{k}\sum_i x_i \qquad \bar{z} = \frac{1}{k}\sum_i z_i$$

$$S_{xx} = \sum_i (x_i - \bar{x})^2 \qquad S_{zz} = \sum_i (z_i - \bar{z})^2$$

$$S_{xz} = \sum_i (x_i - \bar{x})(z_i - \bar{z})$$

und es gilt:

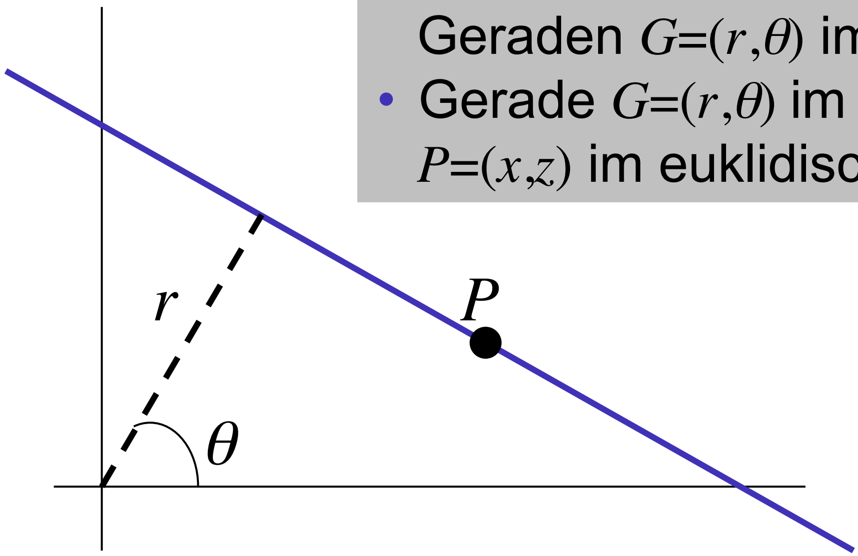
$$\min_{(r,\varphi)} E_{\text{fit}} = \frac{1}{2}\left(S_{xx} + S_{zz} - \sqrt{4S_{xz}^2 + \left(S_{zz} - S_{xx}\right)^2}\right)$$

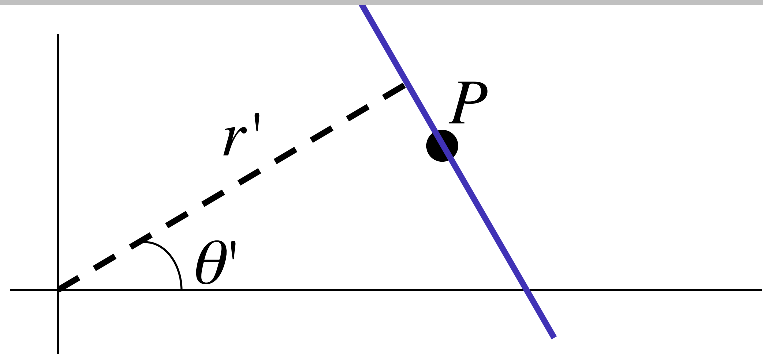
Linien #3: Hough-Transformation

sprich: [Haff]

- Parametrisierung von Geraden üblicherweise: $z = mx + b$
  (Steigung m, Achsabschnitt b)
- allgemeiner (Vertikalen!) Darstellung im **Hough-Raum**:
  $r = x \cos\,\theta + z \sin\,\theta$ (s. eben bei Lu/Milios!)
  r im Abstand (Normale) der Geraden zum Ursprung,
  θ Winkel zwischen x-Achse und Normaler

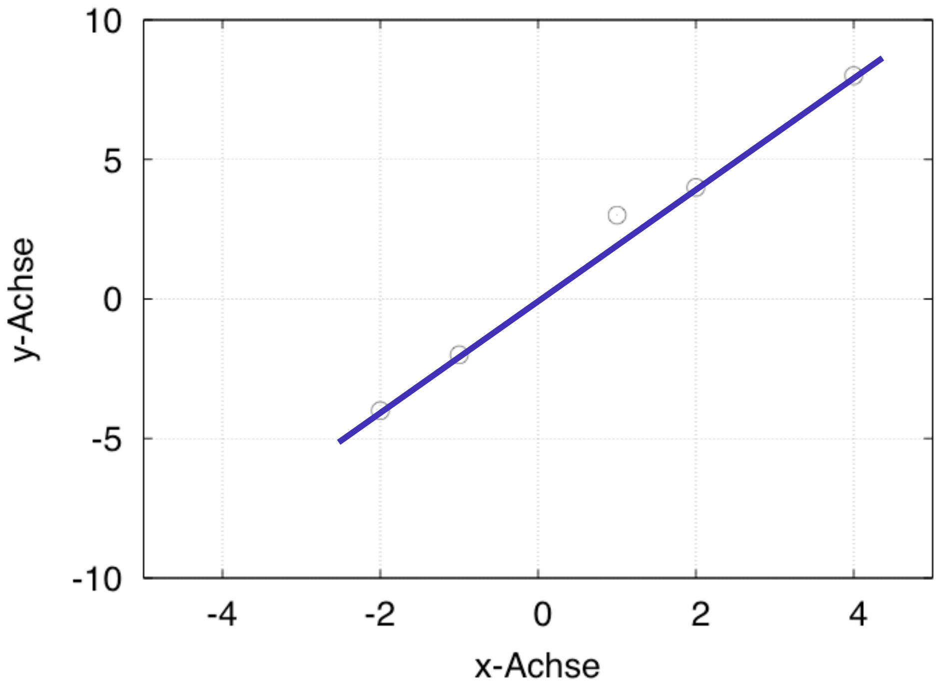
- Punkt $P=(x,z)$ im euklidischen Raum repräsentiert alle Geraden $G=(r,\theta)$ im Hough-Raum, die ihn durchlaufen
- Gerade $G=(r,\theta)$ im Hough-Raum repräsentiert alle Punkte $P=(x,z)$ im euklidischen Raum, die auf ihr liegen

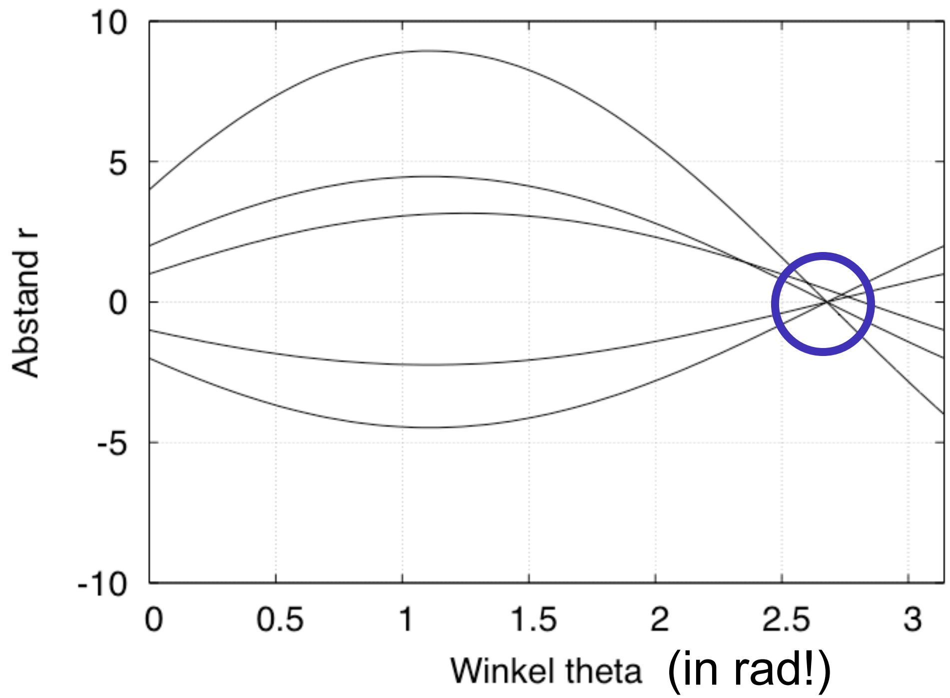




Regressionsgerade und Hough-Raum

- Punkte auf 1 Geraden im Euklidischen schneiden sich in 1 Punkt im Hough-Raum
- „fast kollineare“ Punkte im Euklidischen „schneiden sich fast“ im Hough-Raum

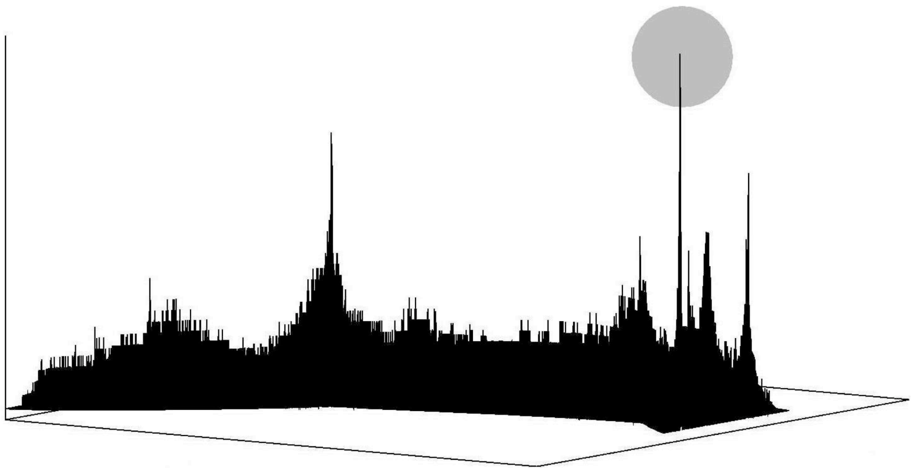


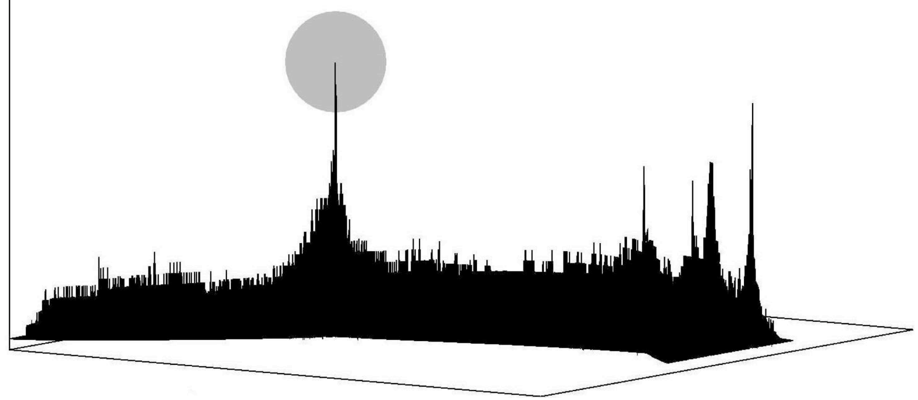


(in rad!)

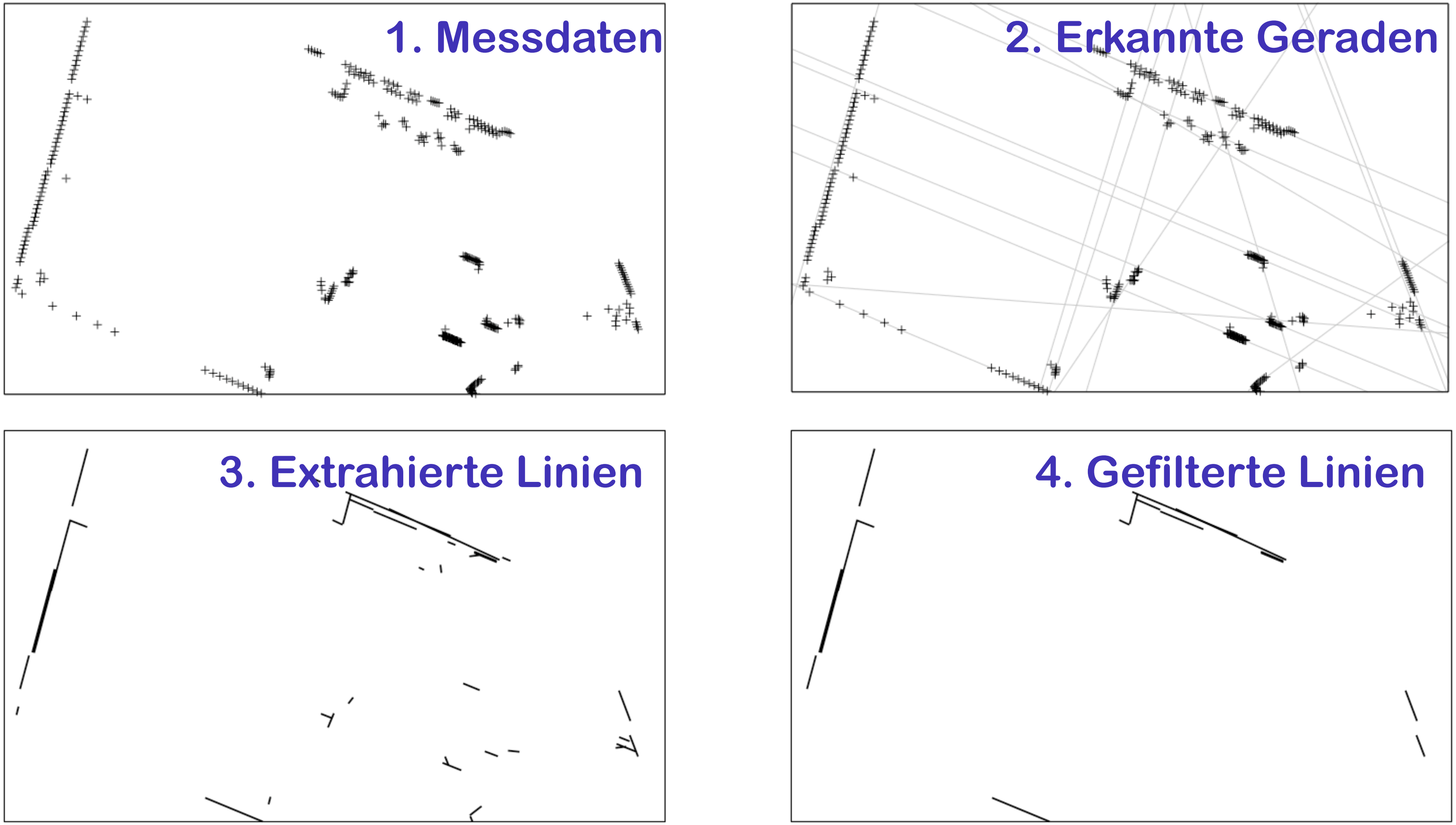
Euklidische Linien im Hough-Raum finden

1. Transformiere alle Messpunkte in den Hough-Raum; zähle für jedes Paar (r_i, θ_j), wie oft es in der Transformation vorkommt („Akkumulator-Array“)
2. Der höchste/„beste“ (r, θ)-Peak entspricht der „besten“ Geraden im Euklidischen; der zweithöchste (r, θ)-Peak entspricht der „zweitbesten“ Geraden im Euklidischen, … etc.





Hough-Linienfindeergebnisse, Beispiel



Funktioniert entsprechend im n-Dimensionalen (z.B. Ebenen finden)

3.2 Bildmerkmale

- **Kernel** in der Bildverarbeitung: kleine Maske (meist wenige Pixel), die lokal auf alle Pixel des Bildes angewendet wird
- dient der Filterung und/oder Merkmalsextraktion

Die pixelweise Anwendung eines Kernels auf ein Bild heißt **Faltung** (*convolution*) des Bildes.

Für Bild $\mathbf{I}=(I(u,v))$ mit Kernel H der Größe $(\delta_i+1) \times (\delta_j+1)$ (δ_i, δ_j gerade) berechne Faltung $\mathbf{R}=(R(u,v))$ (i,j laufen „um 0“):

$$R(u,v) = \sum_{i=-\delta_i/2}^{+\delta_i/2} \sum_{j=-\delta_j/2}^{+\delta_j/2} H(i,j) I(i+u, j+v)$$

abgekürzt: $\mathbf{R} = \mathbf{H} * \mathbf{I}$

$*$ **Faltungsoperator**