

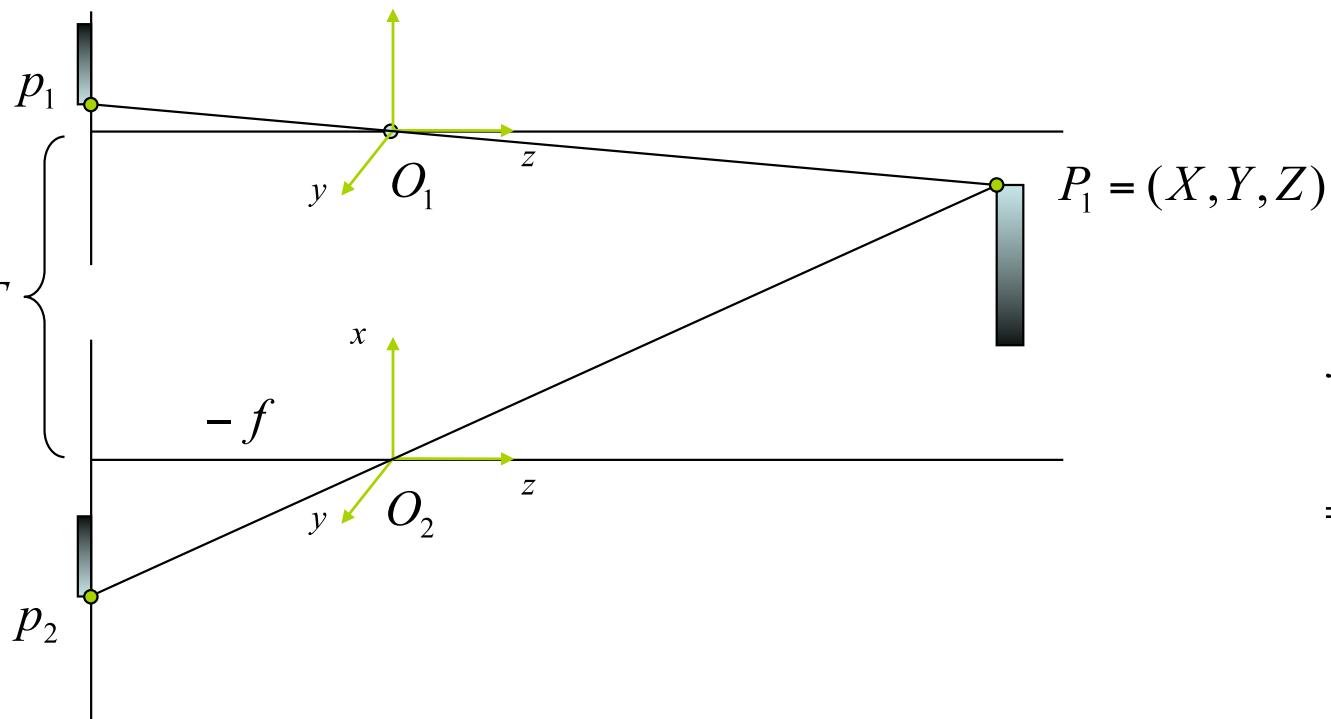
Robotik

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

SoSe 2021



► Zwei Kameras:



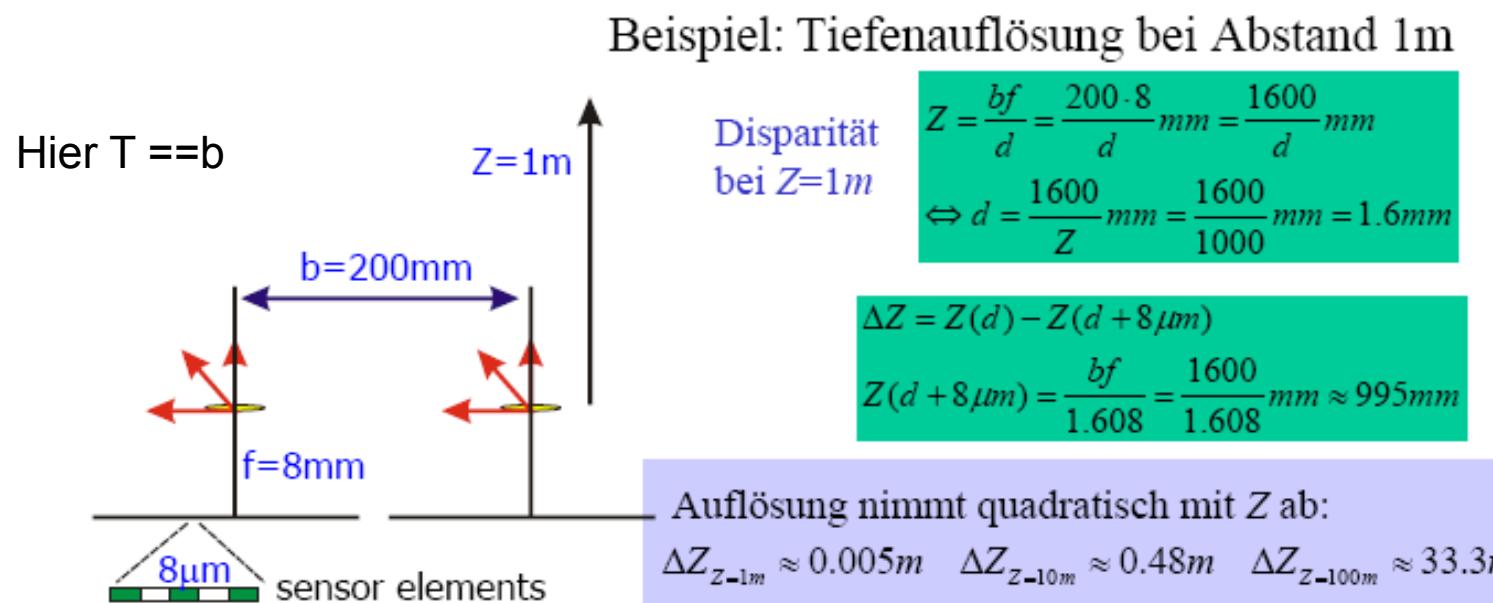
$$x_1 = -f \frac{X_1}{Z_1}, \quad x_2 = -f \frac{X_1 + T}{Z_1} = x_1 - f \frac{T}{Z_1}$$

$$\Rightarrow Z_1 = \frac{fT}{x_1 - x_2} = \frac{fT}{d}$$

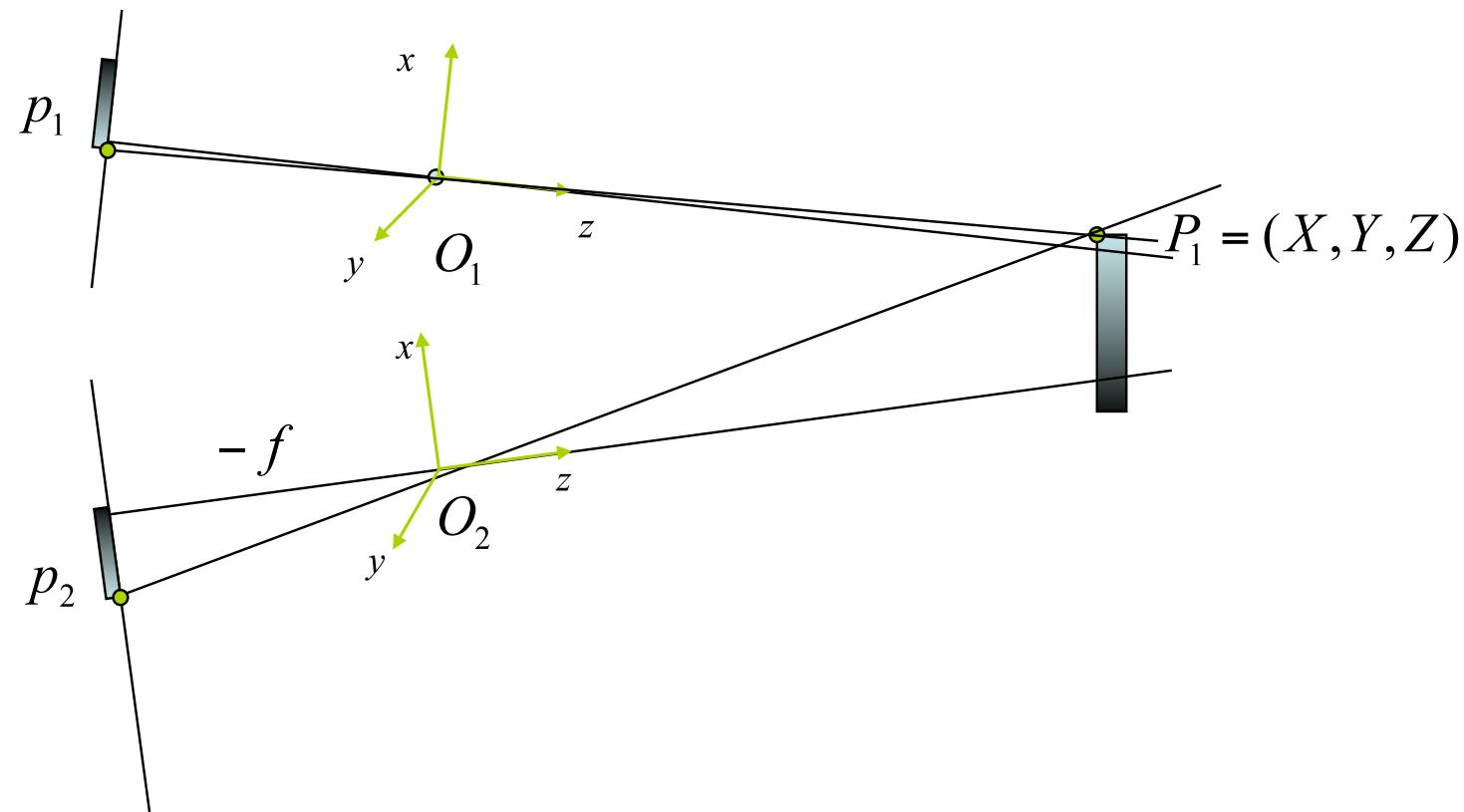
Die erreichbare **Tiefenauflösung** hängt im Wesentlichen von der **Baseline T** und der detektierbaren **Disparität $d = x_1 - x_2$** ab.

Wdh.: Stereo - Auflösung

$$Z = \frac{Tf}{d} \quad X = \frac{T}{2} \left(\frac{x_l + x_r}{d} \right) \quad Y = \frac{T}{2} \left(\frac{2y}{d} \right)$$

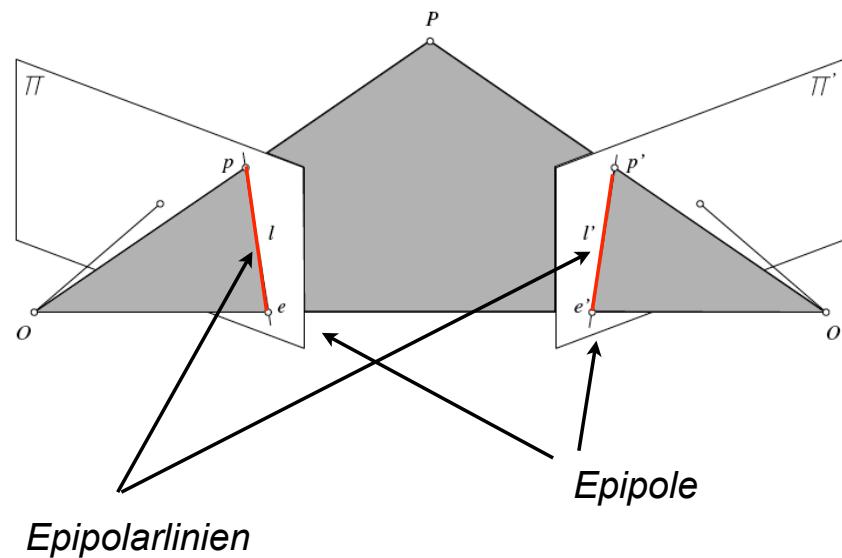


Wdh.: Stereo - Die Realität



Wdh.: Epipolargeometrie

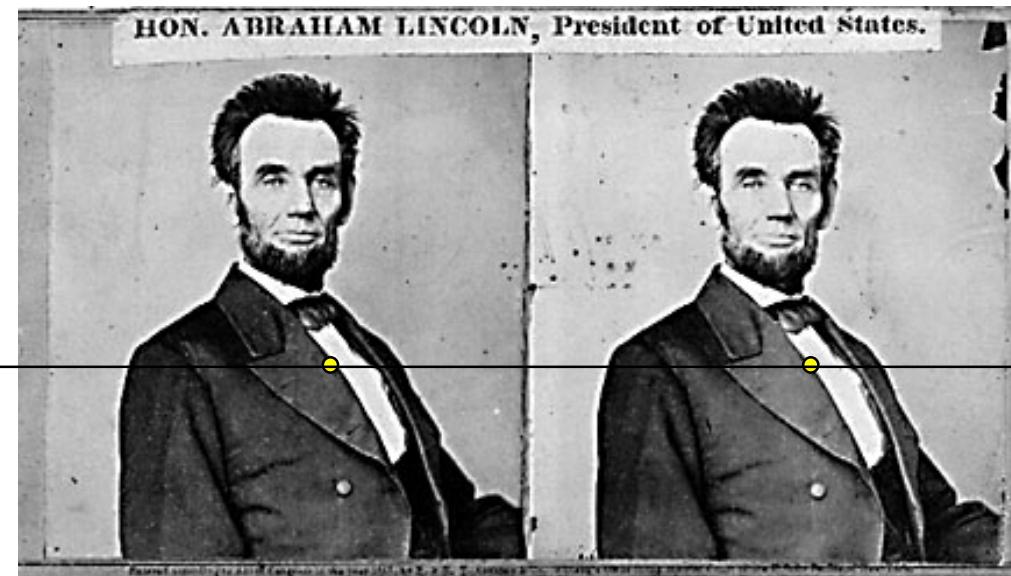
- Die Suche nach Korrespondenzen kann auf die Epipolarlinien beschränkt werden



Wdh.: Stereomatching

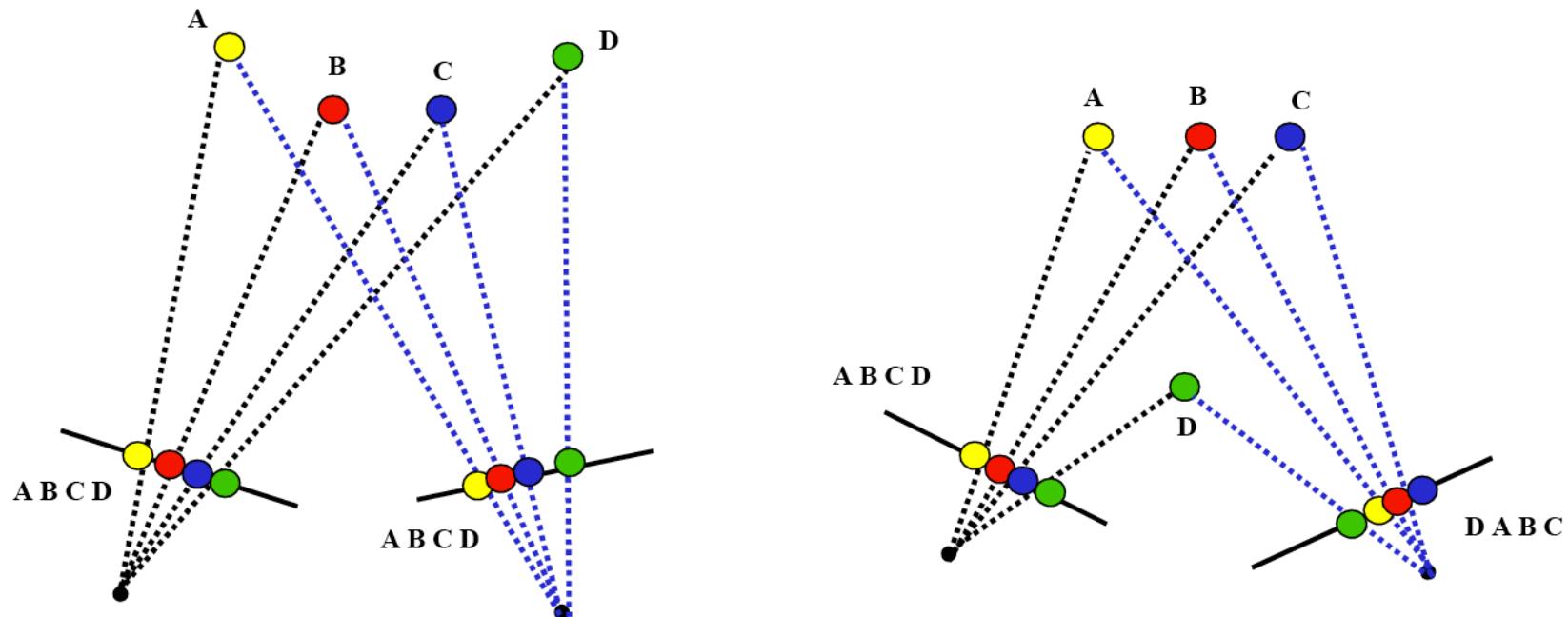
► Naiver Ansatz:

- Vergleiche alle Pixel der Epipolarlinie im linken Bild mit denen auf der entsprechenden Linien im Linken
- Wähle den Pixel mit minimalen Fehler
- ...geht so natürlich nicht, daher werden Regionen verglichen



Wdh.: Stereomatching

- ▶ Die Suche nach der besten Korrespondenz kann durch „Dynamic Programming“ realisiert werden
- ▶ Suche durch Randbedingungen, hier Punktanordnung, beschränken
- ▶ Verhindert Zyklen



Inhalt



Gliederung

4. Sensordatenverarbeitung
1. Entfernungsmessung
2. Robot Operating System
3. Sensorik
4. Bildverarbeitung
5. Sensordatenverarbeitung
6. Fortbewegung
7. Lokalisierung
8. Mapping
9. Navigation
10. Ausblick

Reduktionsfilter

- Solange für Sequenz von Messwerten a_i, \dots, a_j gilt: $\|a_i, a_j\| \leq \delta(i - j)$, ersetze a_i, \dots, a_j durch

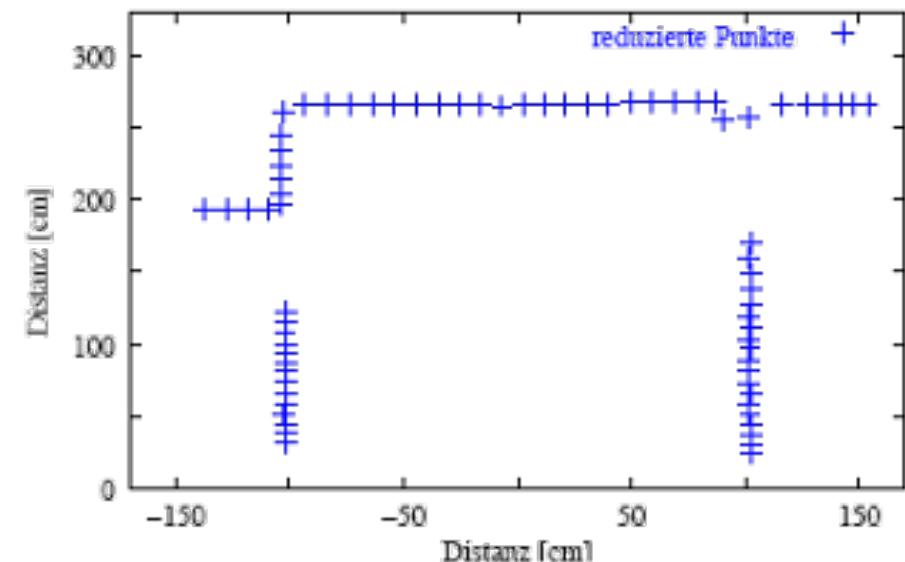
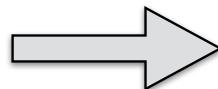
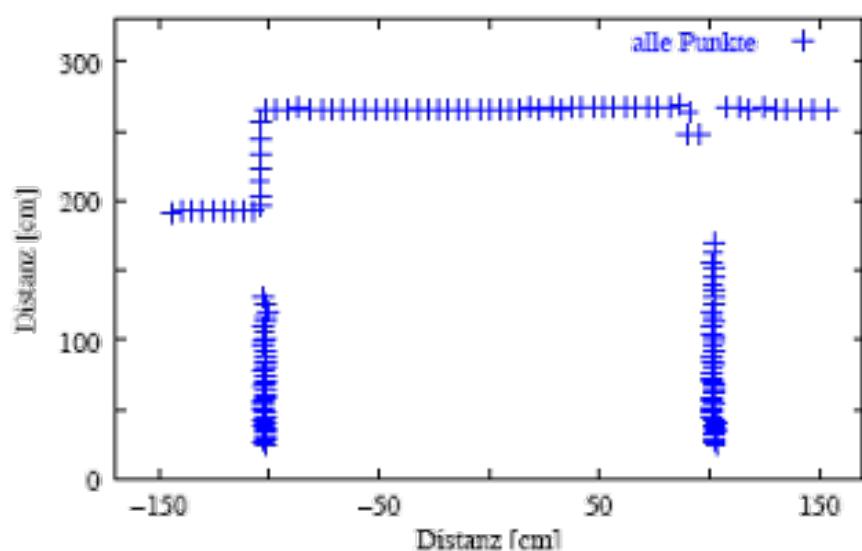
$$\frac{1}{j-i+1} \sum_{k=i}^j a_k$$

Ersetze eine Schar nah aneinander liegender Messwerte durch ihren Mittelwert

- Voraussetzungen:
 - Messwerte geordnet (wie beim Laserscanner)
 - Metrik $\|\cdot, \cdot\|$ auf Messwerten
 - δ -Schranke (hängt möglicherweise von der Punktzahl ab)

Eigenschaften des Reduktionsfilters

► Beispiel:



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

Medianfilter

- ▶ Ersetze jeden Messwert durch den Median seiner Umgebung aus k Werten:
- ▶ Ersetze für alle i den Wert a_i durch

$$\text{Median}\left(a_{i-\lfloor k/2 \rfloor}, \dots, a_i, \dots, a_{i+\lfloor k/2 \rfloor}\right)$$

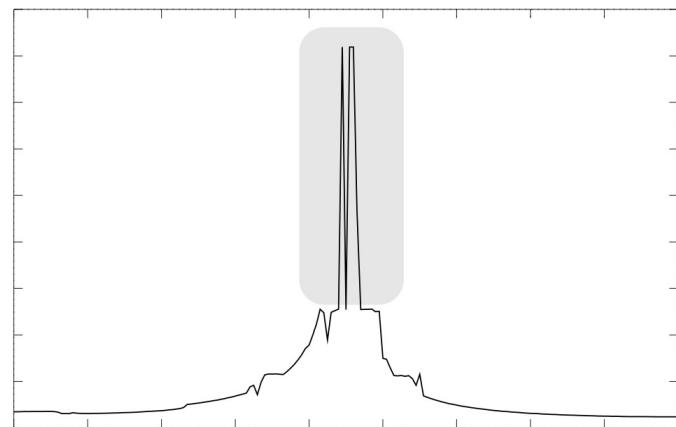
- ▶ Voraussetzungen:
 - ▶ N Messwerte
 - ▶ Metrik auf Messwerten (z.B. Euklidische Distanz)

Eigenschaften:

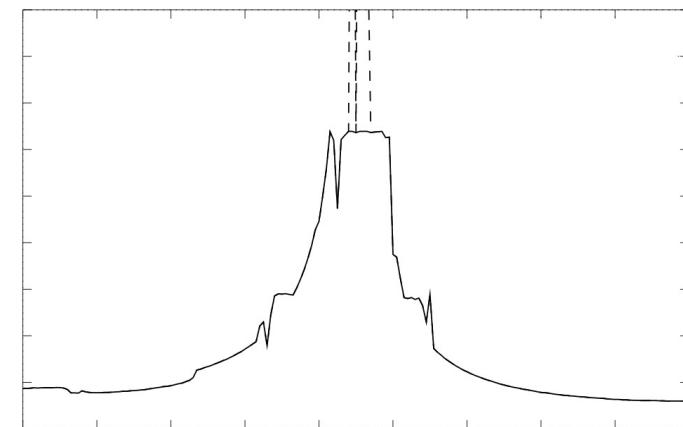
Für N Messwerte Zeitkomplexität $\mathcal{O}(N \cdot k \cdot \log k)$ wegen der Sortierung nach Distanz für die Medianberechnung.
Medianfilter verfälscht “echte” Sprünge in $< k$ Messwerten.

Beispiel Medianfilter

Originalscan

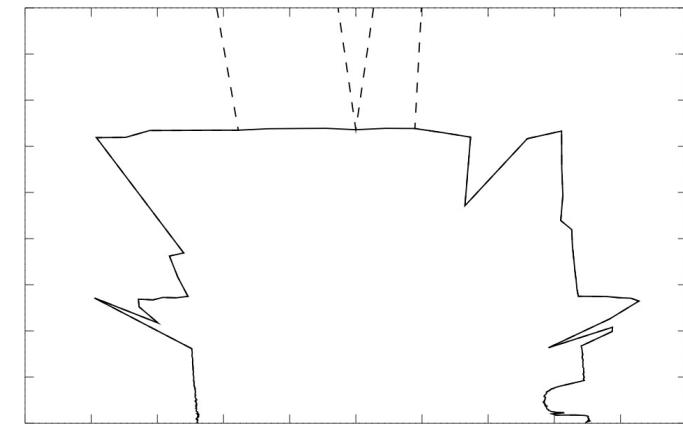
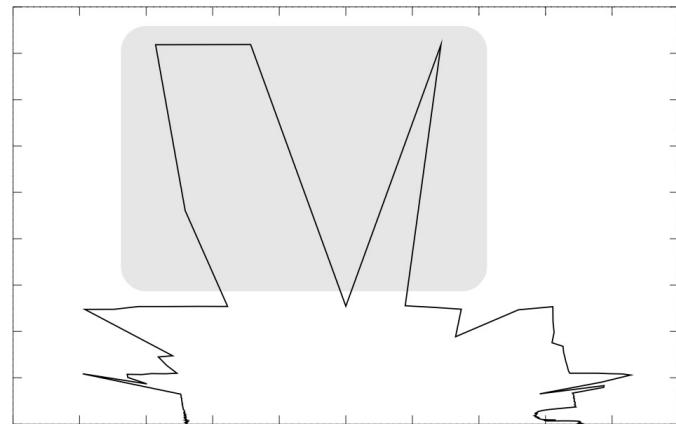


Scan nach Medianfilterung

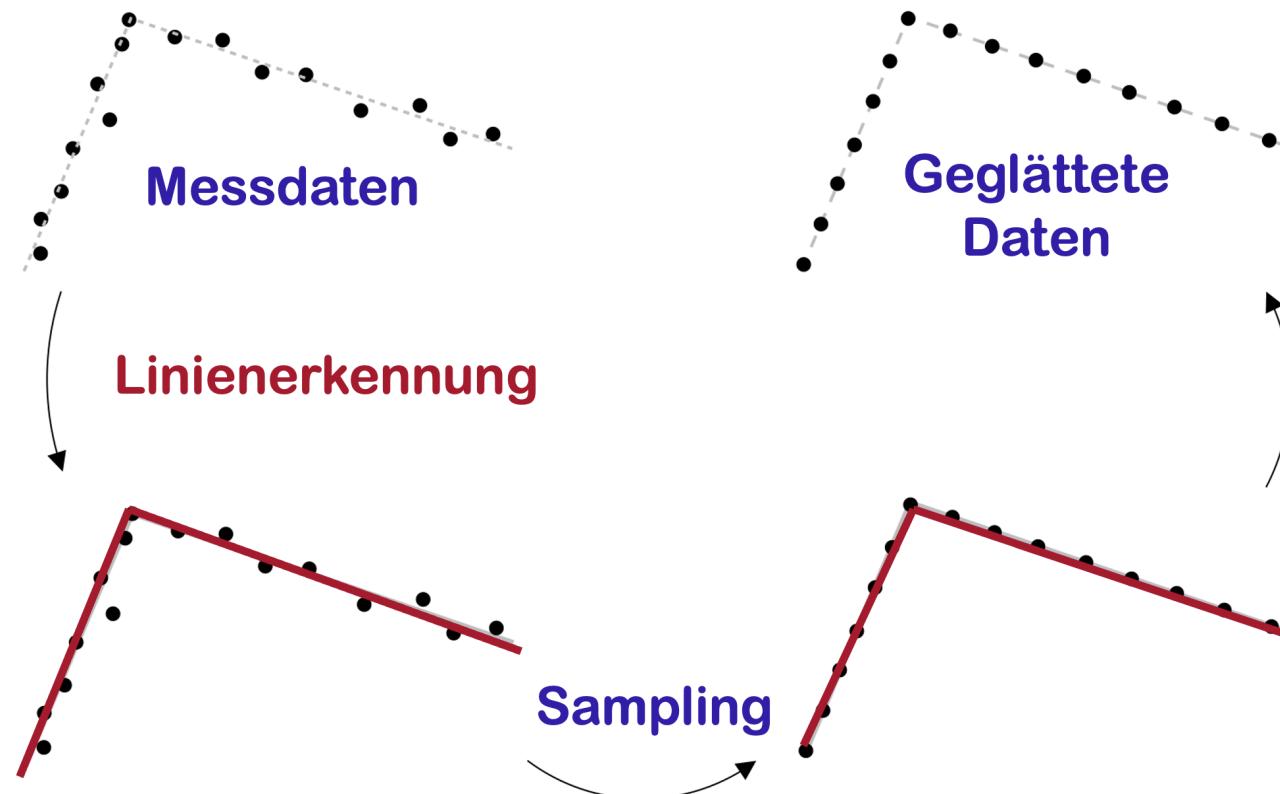


polar

karthesisch



Merkmale auf Entfernungsdaten: Linien



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

Online-Linienkonstruktion

- ▶ Idee für einen Online-Linienfinder (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 Red./Medianfilter 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 sind (z.B. $n = 3$)

Online-Linienfinder 1

Sei a_i, \dots, a_j die aktuell konstruierte Linie...

Mit neuem Streckenpunkt darf die Streckensumme nur wenig von der Luftlinie abweichen:

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

“Luftlinien-Argument” lokal am Ort der Linienerweiterung

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

Direkter Abstand $\|a_k, a_{k+1}\|$ und oder Abstand $\|a_{k-1}, a_{k+1}\|$ darf festes Maximum nicht übersteigen (“keine Linien durchs Nichts”)

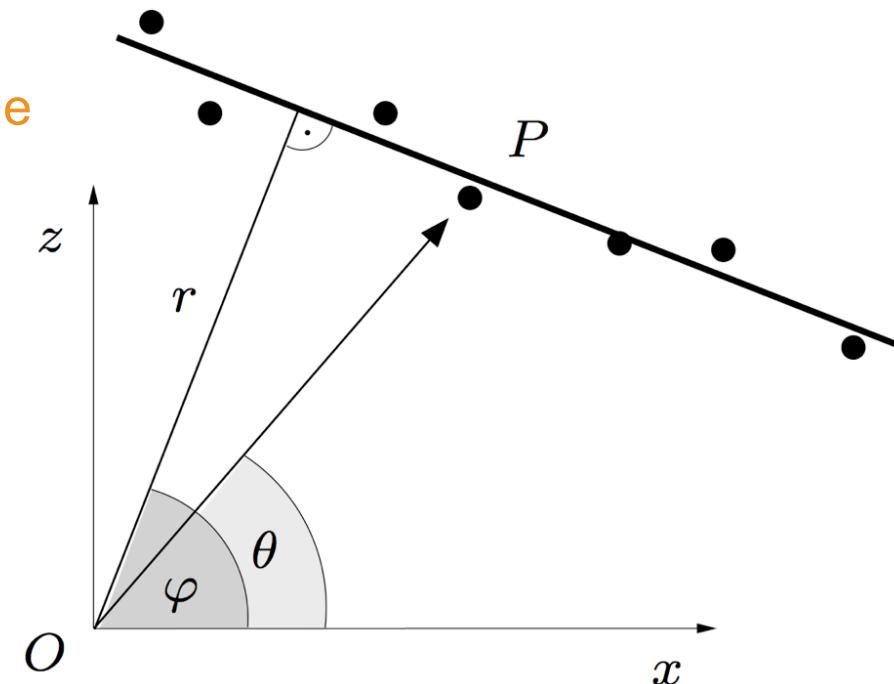
$\|\dots\|$: Euklidischer Abstand

$1 > \varepsilon, \varepsilon(k) > 0$

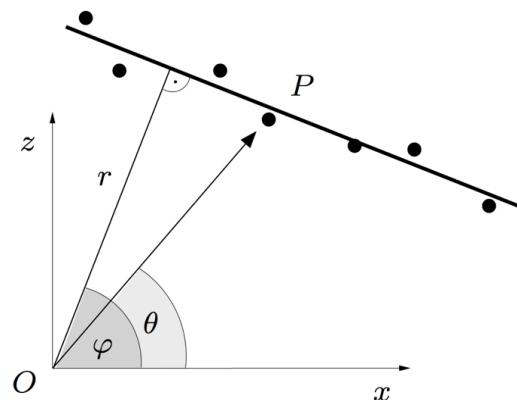
$\lim_{k \rightarrow \infty} \varepsilon(k) = 0$

Online-Linienfinder 2

- ▶ Tangentiallinien nach Lu/Milios
- ▶ Ausgangslage: ($k \in \mathbb{N}$, ungerade, klein, z.B. 7)
 - ▶ Für Punkt P unter Winkel θ suche Regressionsgerade durch je $\frac{(k - 1)}{2}$ Nachbarpunkte rechts und links
 - ▶ Normalen-Distanz auf die Linie ist r
 - ▶ $|\theta - \varphi|$ ist Winkeldifferenz zwischen Strecke \overline{OP} und der Normalen
- ▶ Heuristische Kriterien zur Akzeptanz der Regressionsgraden
 - ▶ $|\theta - \varphi|$ darf Maximalwert nicht überschreiten
 - ▶ andernfalls „flacher“ Scanwinkel oder Entfernungssprung in Daten
 - ▶ Summe der Distanzquadrate zwischen den k Punkten darf einen Maximalwert nicht überschreiten



Linienfindung nach Lu / Milius



Seien:

$$\bar{x} = \frac{1}{k} \sum_i x_i \quad S_{xx} = \sum_i (x_i - \bar{x})^2$$

$$\bar{z} = \frac{1}{k} \sum_i z_i \quad S_{zz} = \sum_i (z_i - \bar{z})^2$$

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

Lemma (Lu/Milius):

Die gesuchte Regressionsgerade ist definiert durch

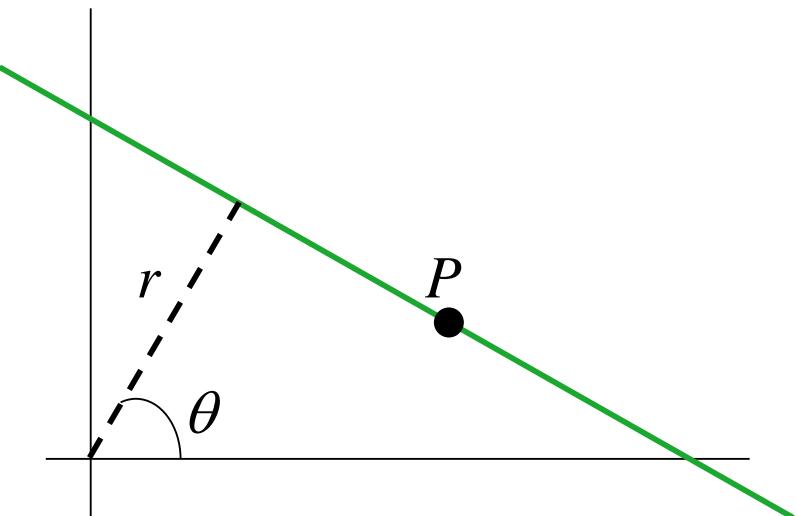
$$\varphi = \frac{1}{2} \arctan \frac{-2S_{xz}}{S_{zz} - S_{xx}} \quad \text{und} \quad r = \bar{x} \cos \varphi + \bar{z} \sin \varphi$$

Dabei gilt für den Approximationsfehler $E_{\text{fit}}(r, \varphi)$ d. Reg-Geraden:

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

Linienfindung 3 - Hough Transformation (1)

- ▶ Verfahren um parametrierbare geometrische Objekte zu finden
- ▶ Hier für Geraden im 2D Raum
- ▶ Mehr dazu in 3D-Sensordatenverarbeitung...

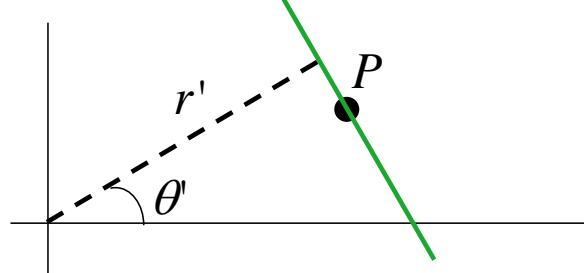
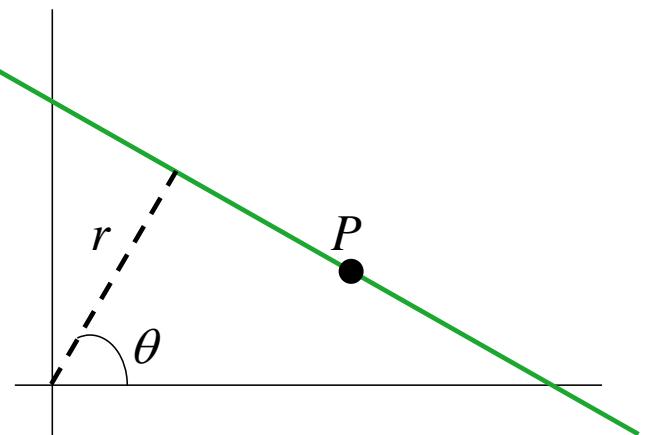


Geraden üblicherweise parametriert als $z = mx + b$
Was ist mit senkrechten Geraden?
Eigentlich $m \rightarrow \infty$
Darstellung im **Hough-Raum**: $r = x \cos \theta + z \sin \theta$
Wie bei Lu/Milos!

Hough-Transformation (2)

$$r = x \cos \theta + z \sin \theta$$

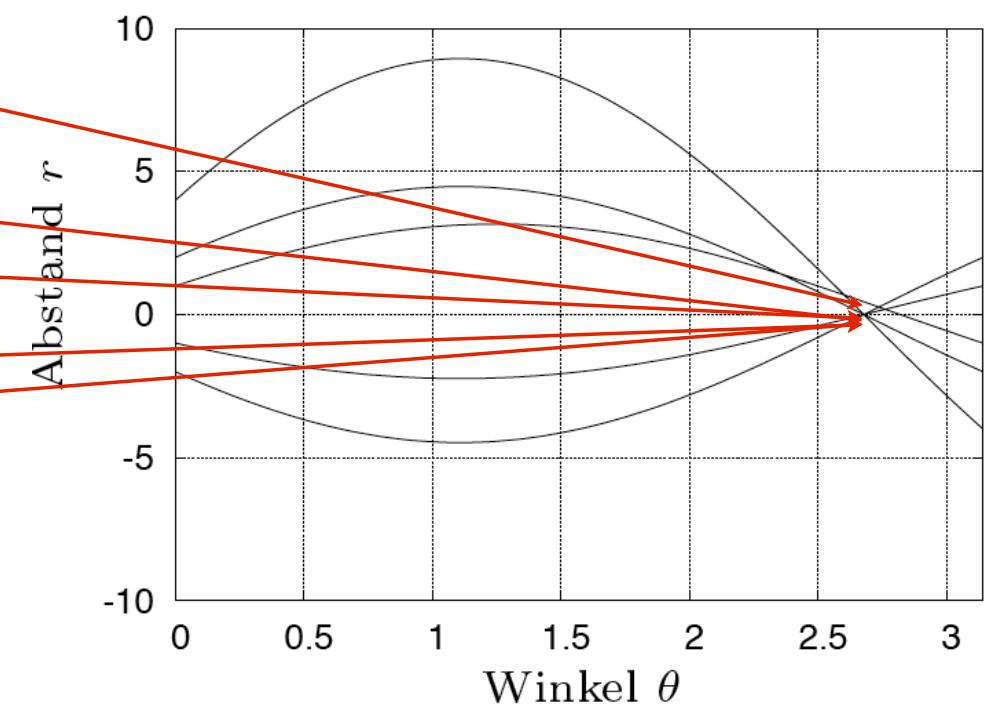
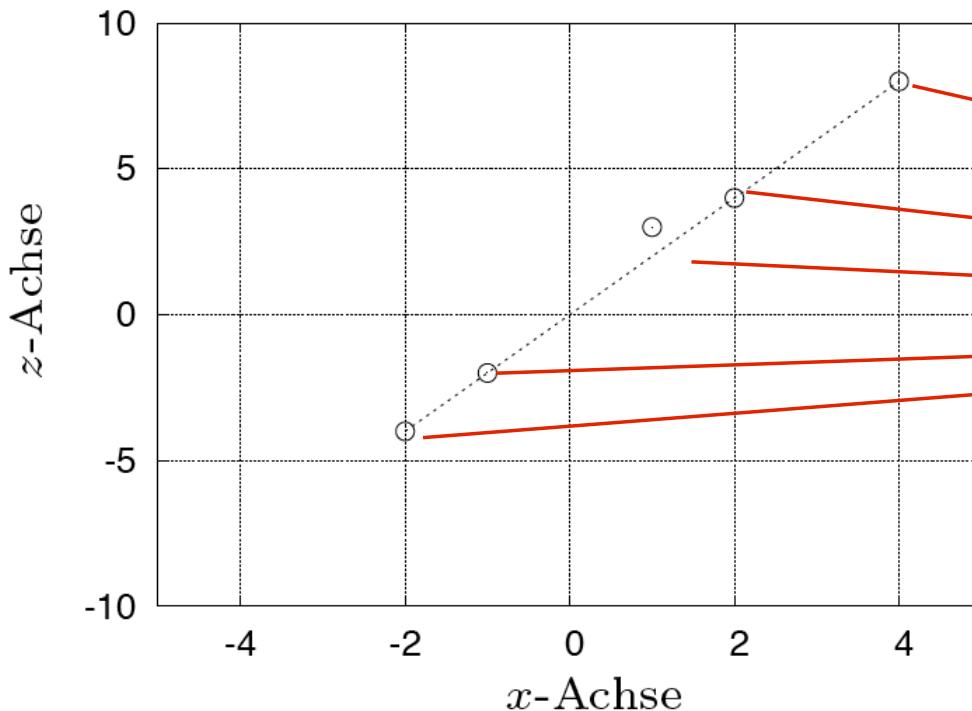
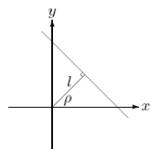
Ein Punkt $P = (x, z)$ im euklidischen Raum repräsentiert alle Geraden $G = (r, \theta)$ im Hough-Raum, die ihn durchlaufen. Eine Gerade $G = (r, \theta)$ im Hough-Raum repräsentiert alle Punkte $P = (x, z)$ im euklidischen Raum, die auf ihr liegen



Hough-Transformation (3)

► Beispiel:

$$l = x \cos \rho + y \sin \rho; \quad 0 \leq \rho < 2\pi, \quad l \geq 0$$



Hough-Transformation (4)

- Implementierung über ein „Akkumulator-Array“ $H[r][\theta]$

$$\theta = 0, \Delta\theta, 2\Delta\theta \dots$$

$$r = 0, \Delta r, 2\Delta r$$

Input: edge points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

Output: lines that go through these edge points

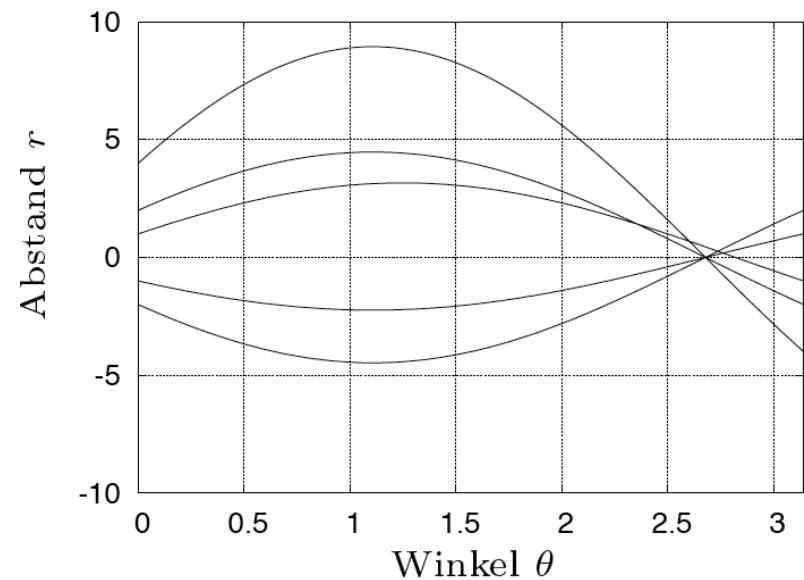
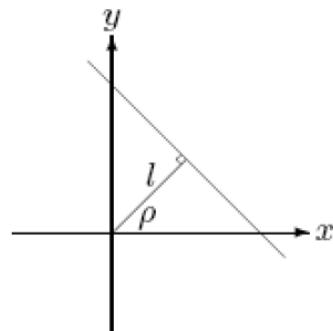
```
Set all elements of the accumulator H[p][l] to zero;
for (all pairs of edge points in k(r, c)) {
    calculate the corresponding p and l;
    H[p][l]++;
}
Detect peaks in H[p][l];
```

[Nüchter]

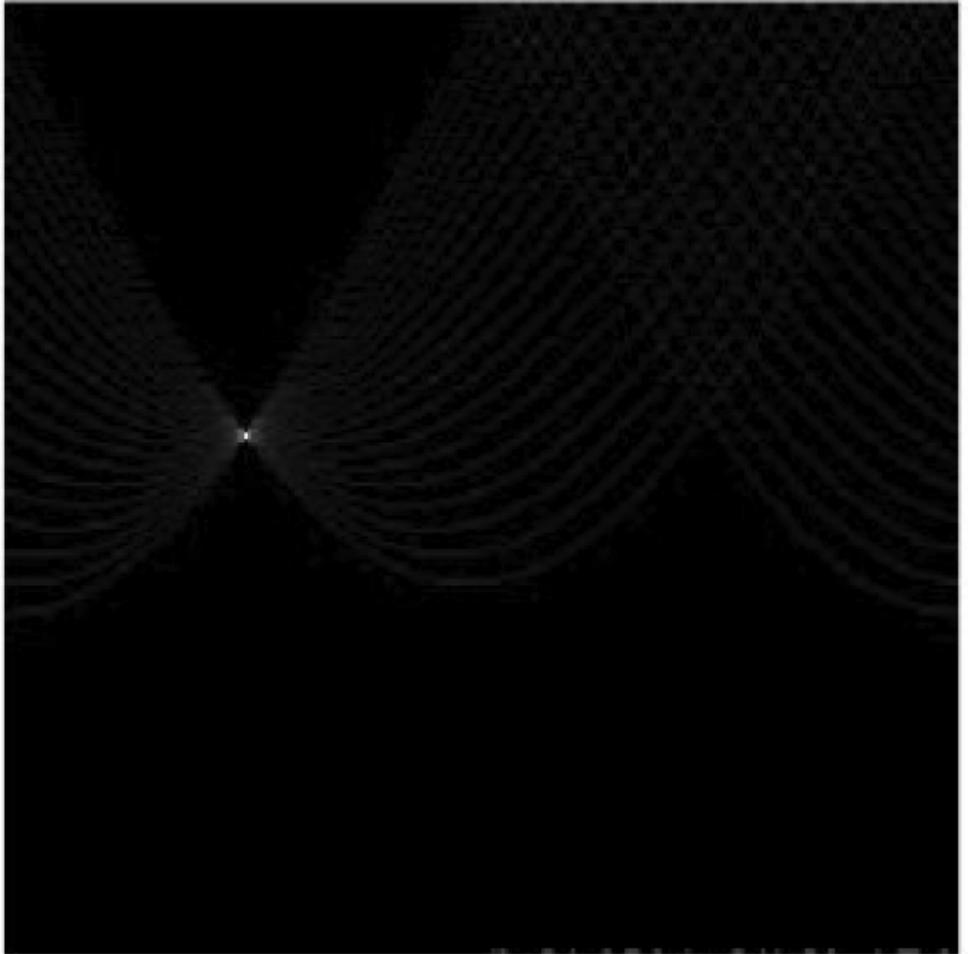
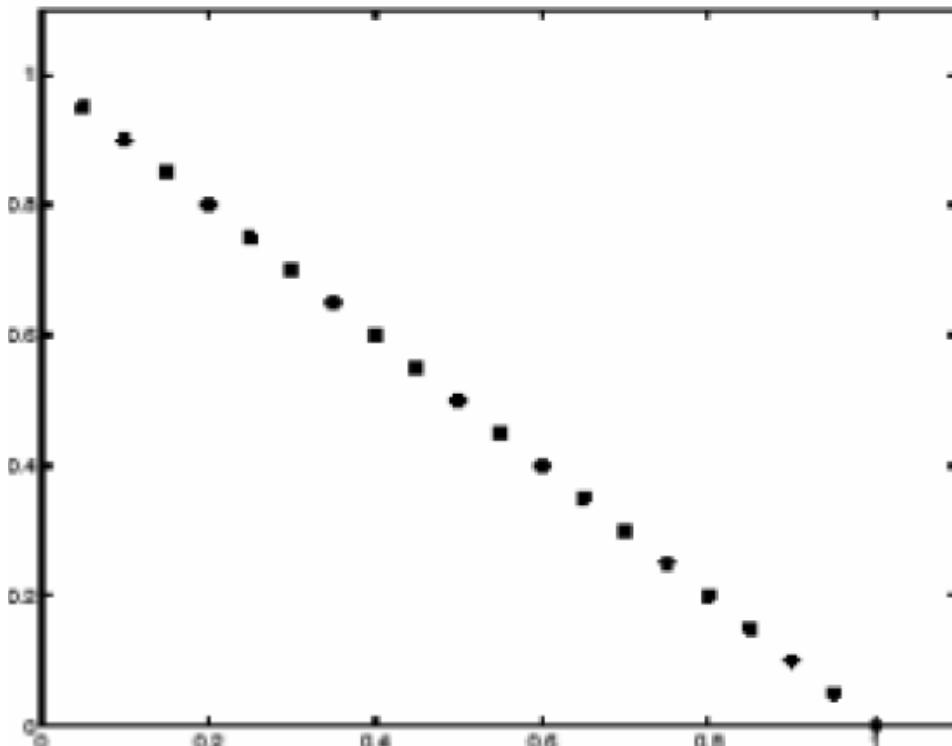
Hough-Transformation (5)

- ▶ Linien werden sich im Akkumulator-Array als Peaks darstellen
- ▶ Bei vielen Punktpaaren: $\mathcal{O}(n^2)$
- ▶ Akkumulator-Array mit wenigen Eingabepunkten problematisch
- ▶ Lösung: Betrachte alle Lösungen die die Randbedingung erfüllen
- ▶ Sinuise-Kurve im Hough-Raum

$$l = x \cos \rho + y \sin \rho; \quad 0 \leq \rho < 2\pi, \quad l \geq 0$$



Hough-Transformation (6)



Hough-Transformation (7)

- Input: Edge points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Output: Lines, that go through these edge points

1. Transform all edge points (x_i, y_i) according to

$l = x_i \cos \rho + y_i \sin \rho$ from the xy-space to the pl-space.

```
for ( $\rho = 0; \rho < 2\pi; \rho += d\rho$ )
    for ( $l = 0; l < l_{max}; l += dl$ )
```

$H[\rho][l] = 0;$

```
for ( $i = 1; i \leq n; i++$ )
```

```
    for ( $\rho = 0; \rho < 2\pi; \rho += d\rho$ ) {
```

$l = x_i \cos \rho + y_i \sin \rho;$

$H[\rho][l] ++;$ /* Regard the discretisation of l */

/* Increment according to edge width $s(x_i, y_i)$ poss. */

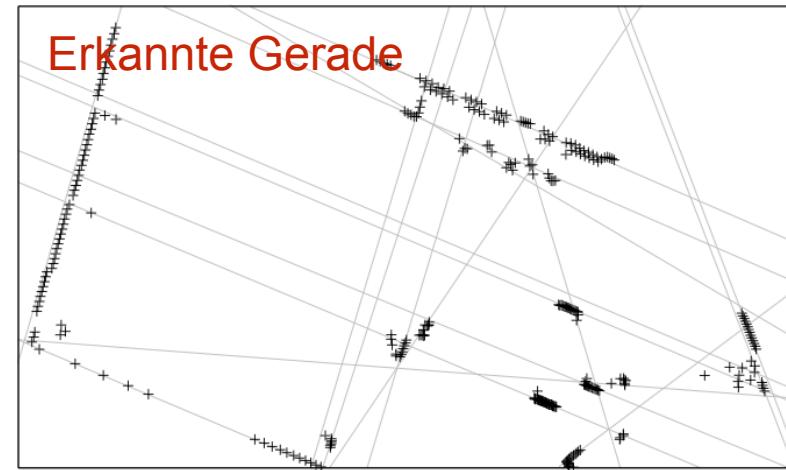
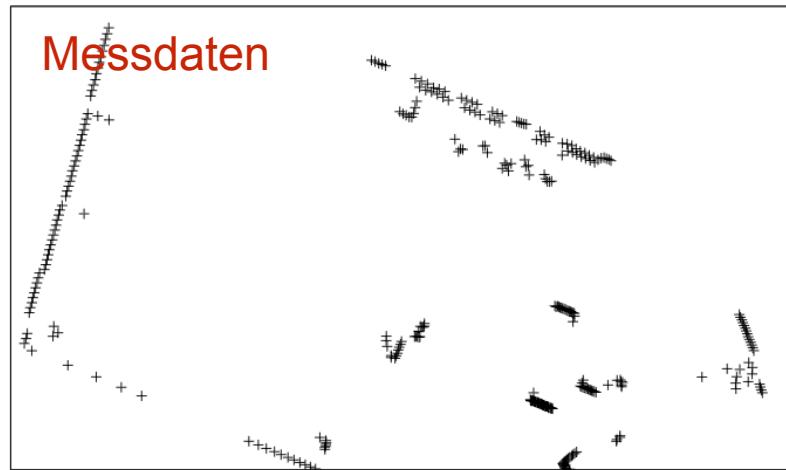
}

2. Search for cluster points in pl-space, i.e. in $H[\rho][l]$.

3. All cluster points (ρ_0, l_0) define a line

$l_0 = x \cos \rho_0 + y \sin \rho_0$ in xy-space.

Hough-Transformation (7)

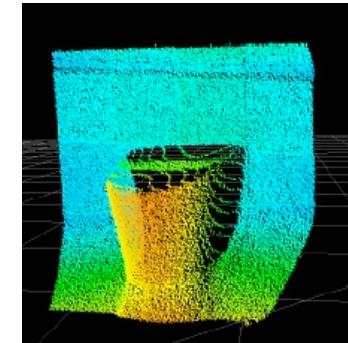
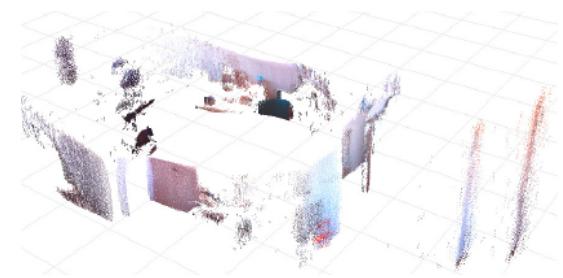
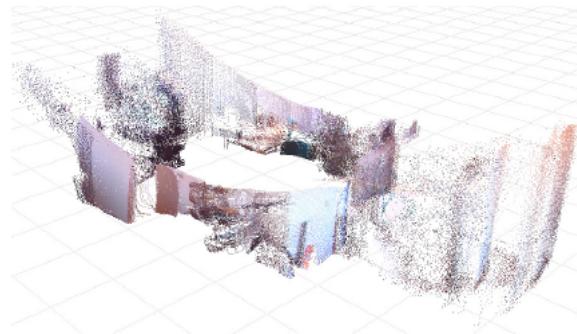


Ausblick - 3D-Punktwolken

- ▶ Tiefenfilter für Kantensprünge
 - ▶ Statistical Outlier Removal
 - ▶ MLS-Filterung
 - ▶ Ebenendetektion
 - ▶ 3D-Hough
 - ▶ ...

$$s_i = \sum_{j \in N_i} s_j e^{-\frac{1}{\sigma_d^2} (d_i - d_j)^2}$$

aktueller Pixel → s_i
 Nachbarpixel → s_j
 Varianz → σ_d^2
 Amplitude des Sprungs → $(d_i - d_j)^2$

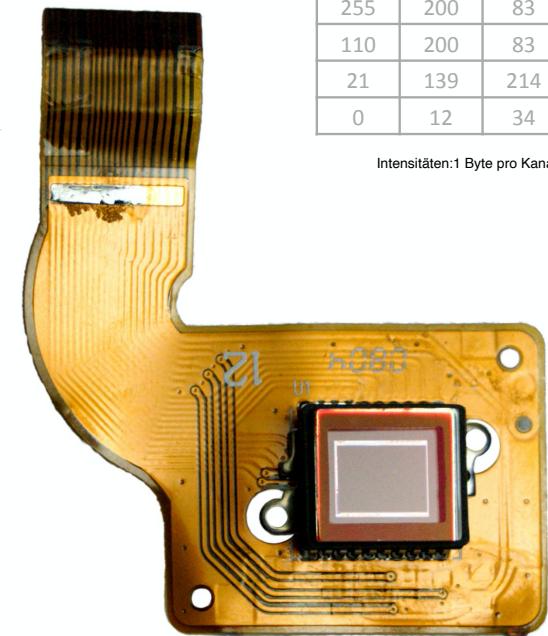
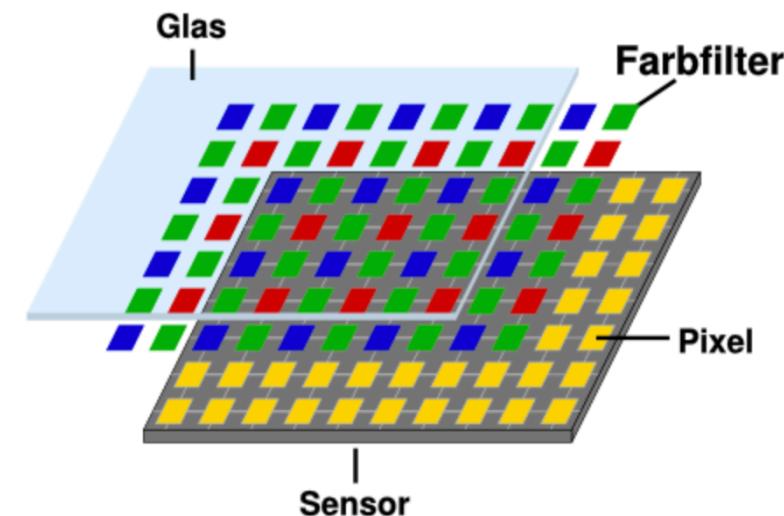
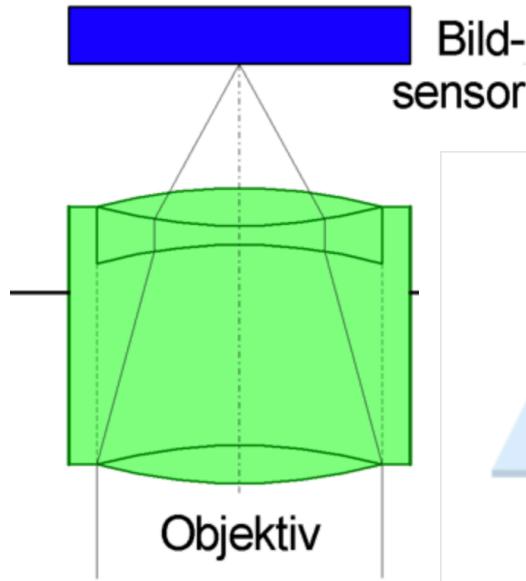




Gliederung

4. Sensordatenverarbeitung
- 2.1. Entfernungsdaten
- 3.2. Bildverarbeitung
4. Sensordatenverarbeitung
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick

Erinnerung:



255	200	83	83
255	200	83	83
110	200	83	21
21	139	214	0
0	12	34	28

Intensitäten: 1 Byte pro Kanal

Quelle: <https://www.elmar-baumann.de/fotografie/techtutorial/sensoren-02.htm>.

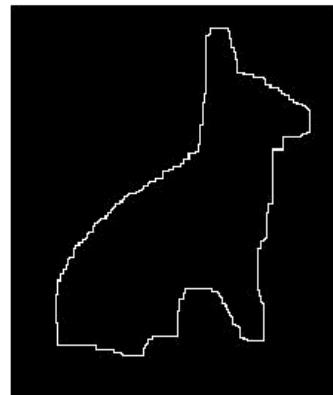
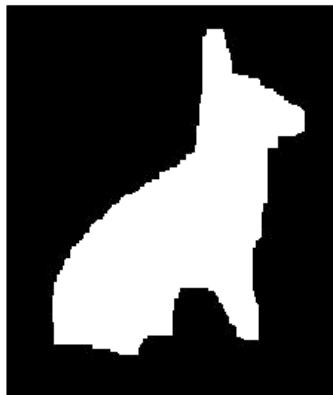
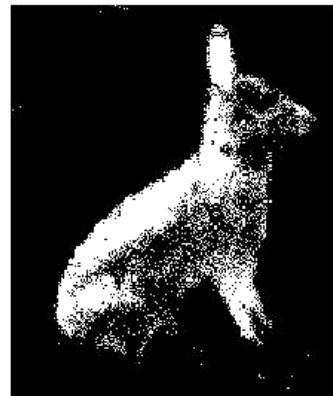
Von C-M - Eigenes Werk, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=2150801>

Segmentierung (1)



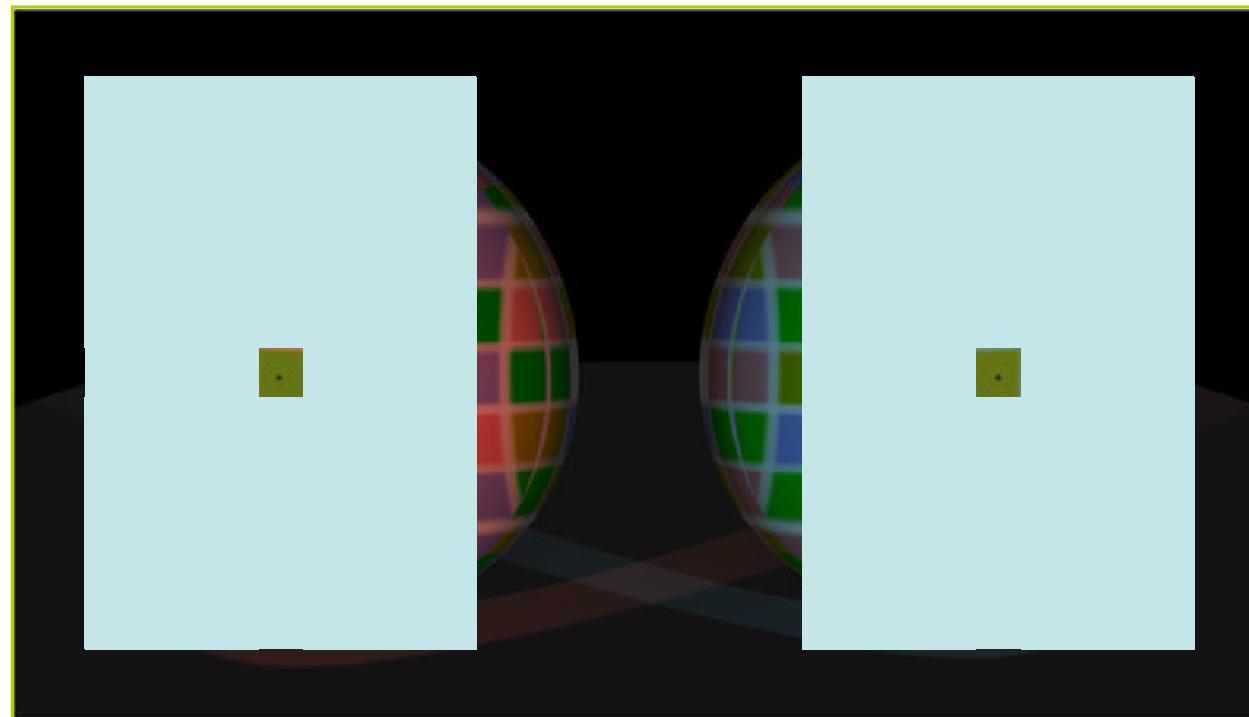
- ▶ Zerteilung eines Bildes in Segmente ➡ Trenne interessantes (Objekte) vom Hintergrund
- ▶ Übersegmentierung (zu viele Segmente) vs. Untersegmentierung (zu wenig Segmente)
- ▶ Bottom Up - Segmentierung aufgrund lokaler Affinität
- ▶ Top Down - Segmentierung aufgrund von Objektzugehörigkeit
- ▶ Flächen vs. Keypoints

Segmentierung (2)



Bildmerkmale (1)

- ▶ Was ist ein Bildmerkmal ?



- ▶ Informationen über bedeutungsvolle Teile eines Bildes, die es ermöglichen
- ▶ Häufig werden Features für Objekterkennung und Bild-Bild-Matching benutzt