Intelligente Sehsysteme

12 Klassifikation von 3D-Punkten

3D-Laserabstandsmessungen, Punktwolken

3D-Deskriptoren, RGB-D-Sensoren

Volker Steinhage

Inhalt

- Zur Klassifikation von Punkten aus 3D-Punktwolken von 3D-Laserscans
  - Sensor & Punktwolken
  - Nachbarschaftssuche
  - Normalenvektoren & Deskriptoren
- RGB-D-Punktwolken
- Zur Klassifikation von Punkten aus RGB-D-Punktwolken über *Deep Learning*

Danksagung: zahlreiche Folien wurden von Jens Behley im Rahmen einer gemeinsamen Vorlesung am Institut für Informatik, Univ. Bonn, erstellt.

Zur Klassifikation von 3D Laserabstandsdaten

- Objekterkennung
  - Anwendungen: Robotik (Indoor/Outdoor), Fahrzeugassistenz, autonome Fahrzeuge
  - Klassen: Fußgänger, PKW, LKW, Busse Motorrad- und Fahrradfahrer
  - Auch: befahrbares/nicht befahrbares Terrain, Bordsteine, Fußgänger- und Fahrradwege

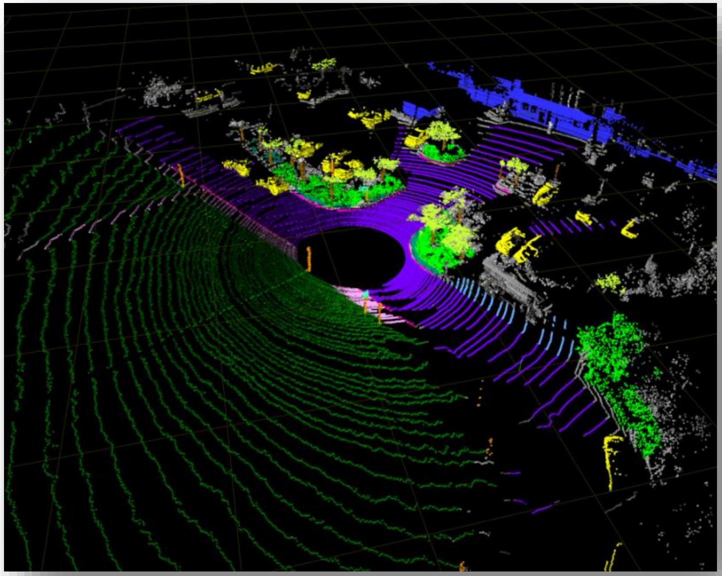
- Semantische Karten
  - Anwendungen: Autonome Exploration von Gefahrengebieten
  - Klassen: Gebäude, KFZ, Vegetation, etc.

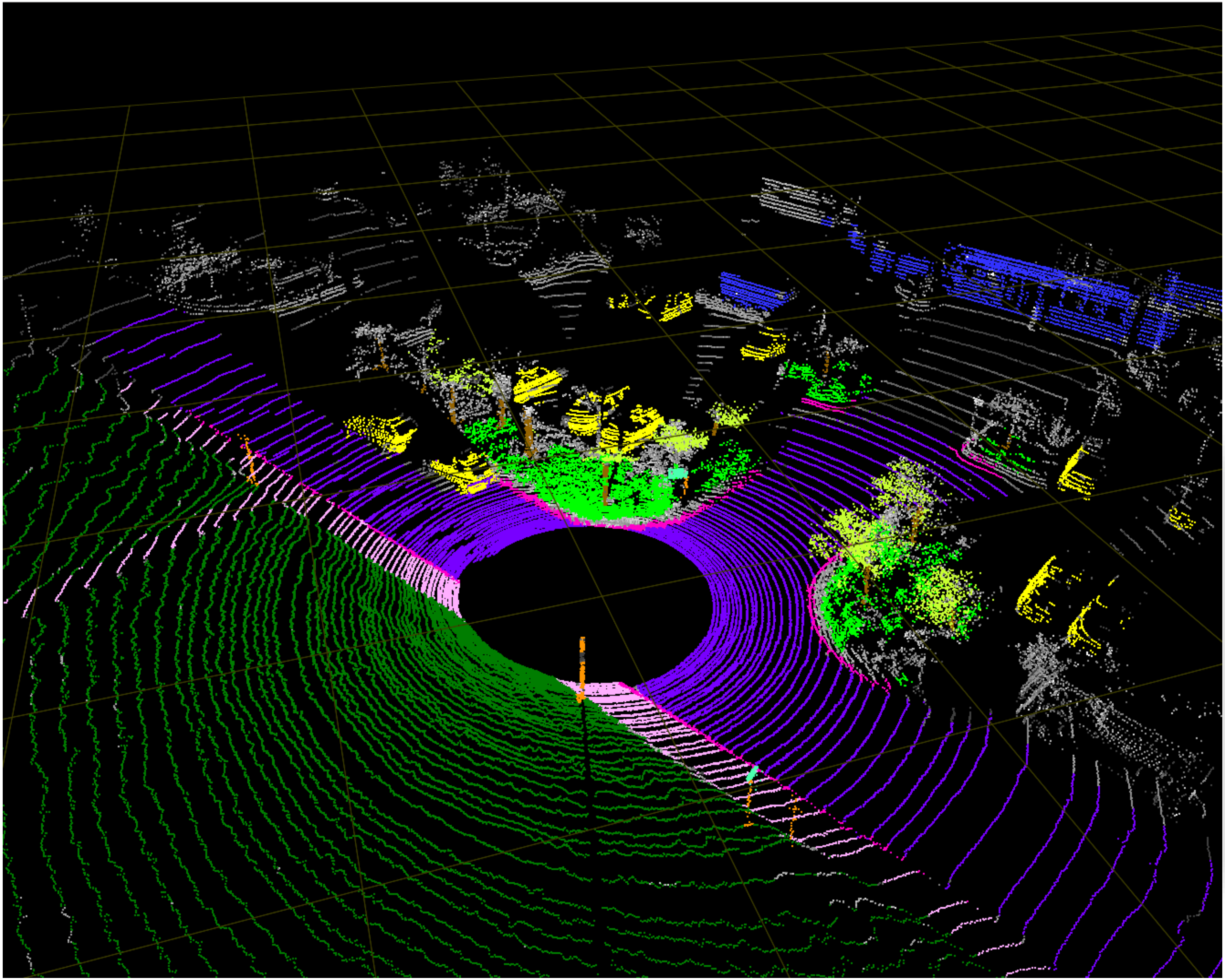
Fokus hier: Punktklassifikation
für Objekterkennung

Semantic Labeling in 3D Laser Scans

universitätbonn

Fraunhofer FKIE





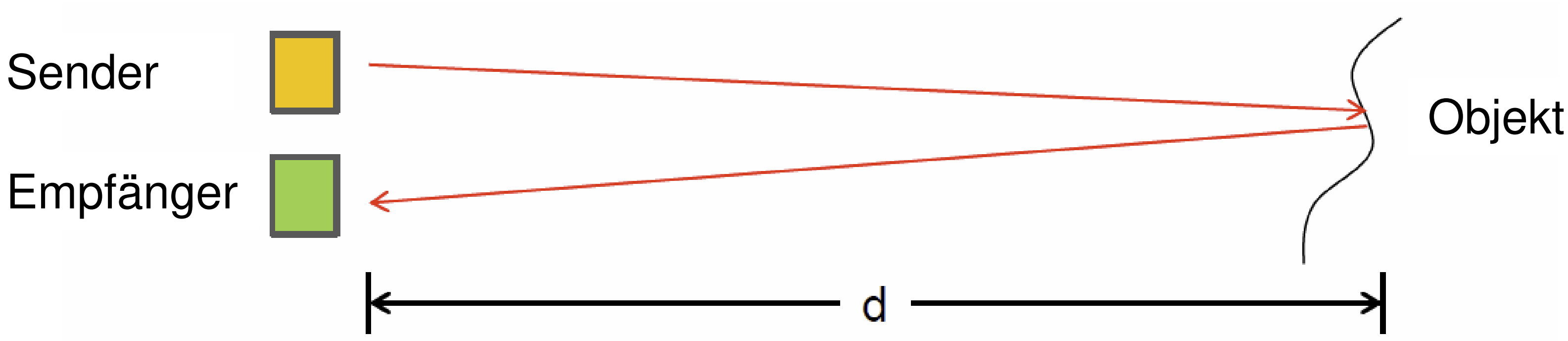
Ausrüstung

- QinetiQ Longcross mit inertialem Navigationssystem (INS)
- Velodnye HDL64-E (1.3 Million Datenpunkte, 5-15 Hz)

Slide credits for slides 4 - 22: Jens Behley, Institut für Informatik, Univ. Bonn

Abstandsmessung

- Abstandsmessung: Zeitmessung zwischen Sendung und Empfang von Laserstrahl bzw. dessen Reflektion *(time-of-flight)*



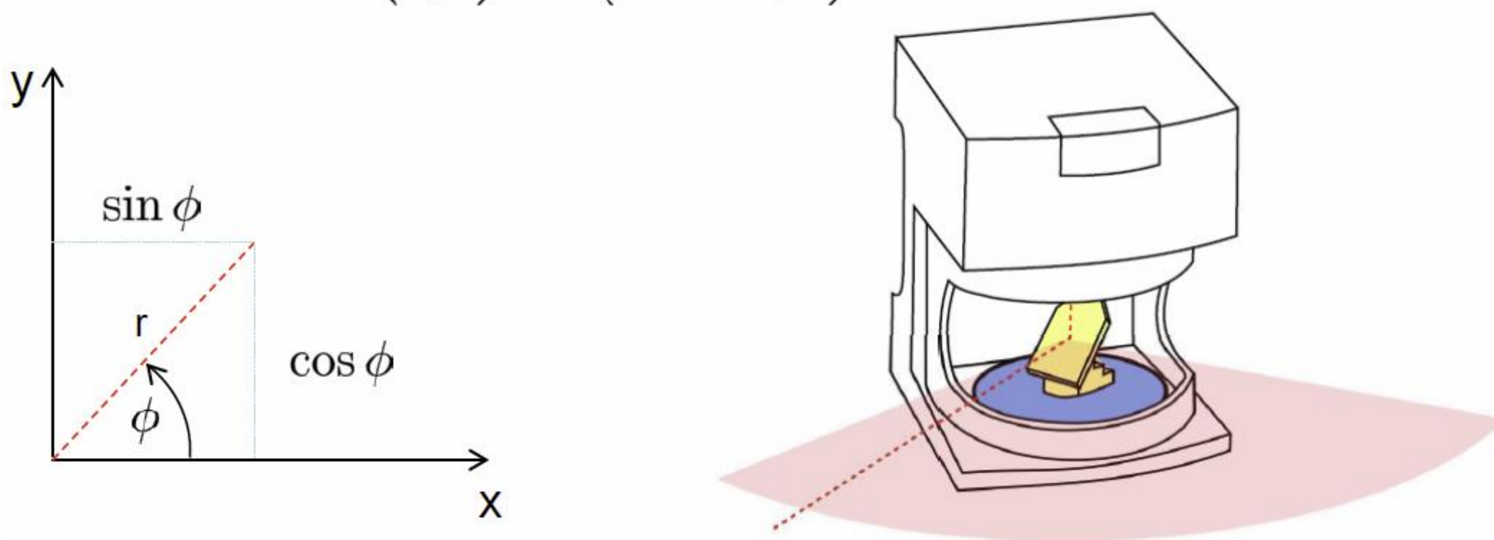
- Abstand d bei *time-of-flight* t_f und Lichtgeschwindigkeit c:

$$d \approx \tfrac{1}{2}\, t_f \cdot c$$

2D-Laserabstandssensoren

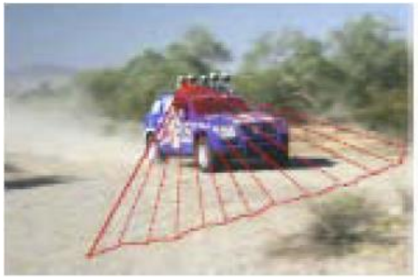
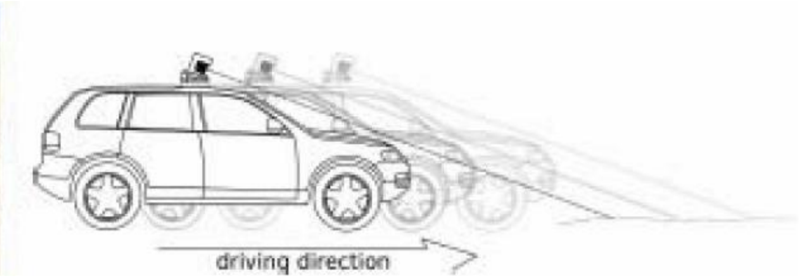
- Drehender Spiegel erzeugt Abtastebene
- Kartes. Koordinaten relativ zum Sensor aus Winkel ϕ und Abstand r ableitbar:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r \cdot \cos\phi \\ r \cdot \sin\phi \end{pmatrix}$$ (polar coordinates)



Von 2D- zu 3D-Laserabstandsmessung

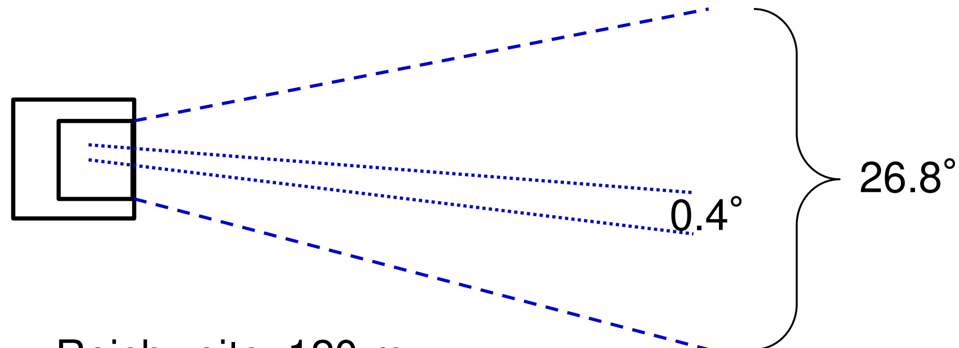
- Bewegung der Plattform selbst

- Neigung des Sensors in gewünschte Richtung
  - Plattform kann fix bleiben
  - i.A. langsam (wg. Stop/Motion)
- Rotation des Sensors in gewünschte Richtung
  - Plattform kann fix bleiben
  - Ständige Rotation, daher kein Stop/Motion
  - Bevorzugte Installation

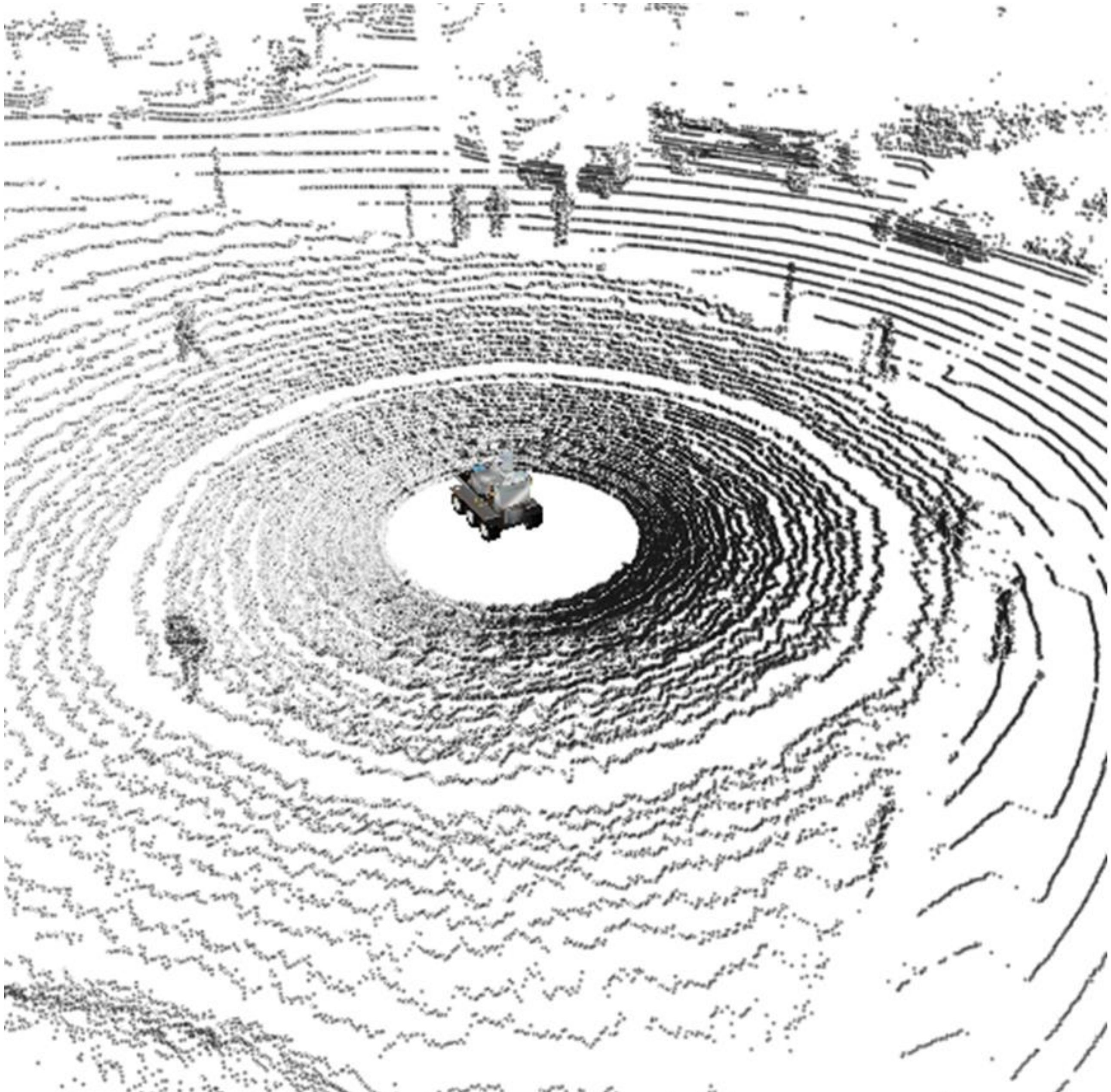
Velodyne HDL-64E (Daten)

- Ca. 1.3 Million Laserabstandsmessungen pro Sekunde
- 64 Laserstrahlen erzeugen eine Messebene
- Bis zu 15 Hz, d.h., Umdrehungen pro Sekunde

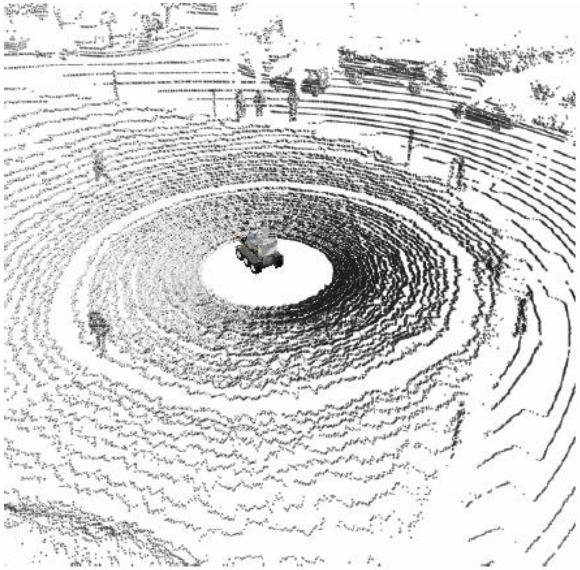


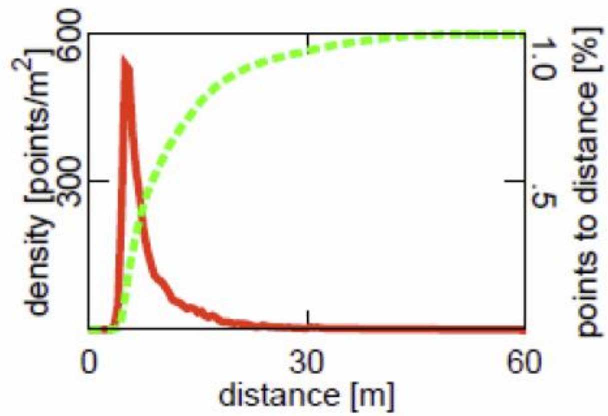
Reichweite: 120 m
Vertikale Auflösung: ~0.4°
Vertikales Sichtfeld: 26.8°
Horizontale Auflösung: ~0.09°
Horizontales Sichtfeld: 360°

Velodyne HDL-64E (Prinzip)



Punktwolken

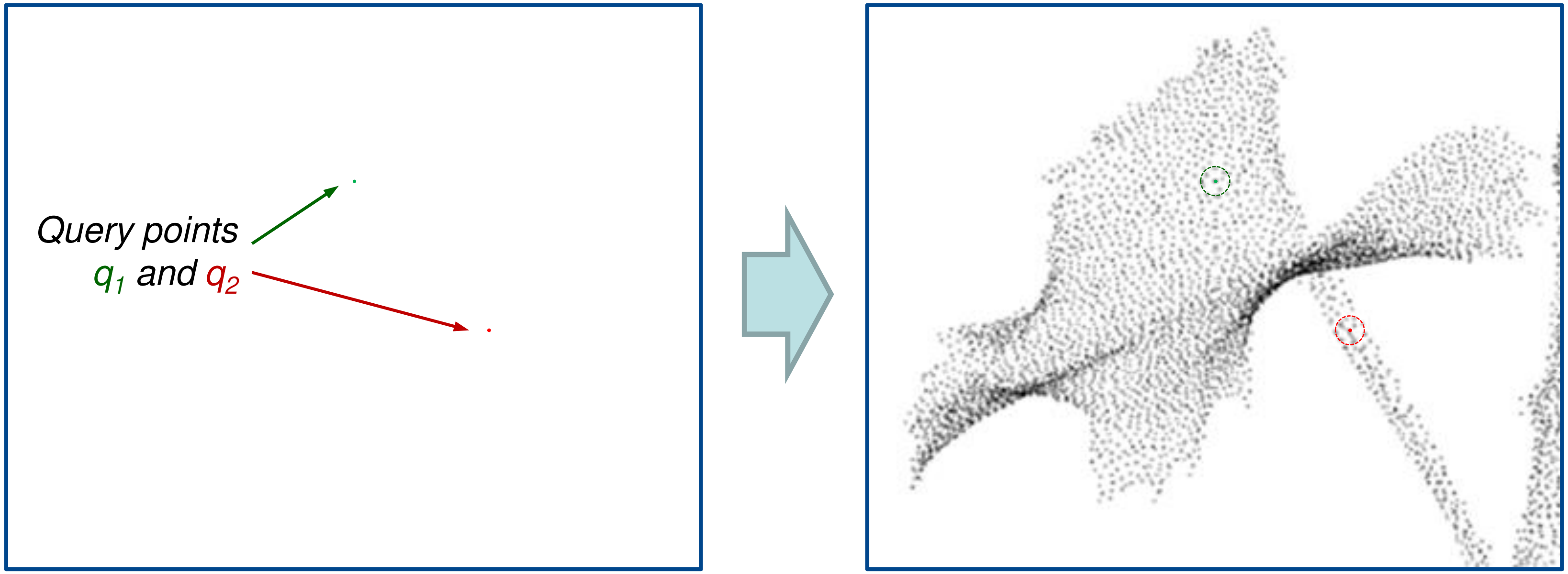




- Punktwolke = ungeordnete Menge von 3D-Punkten $\mathcal{P} = \{\mathbf{p} \in \mathbb{R}^3\}$
- Sehr hohe Punktezahlen → Effizienz
- Distanzabhängige Punktdichte → Robustheit

Nachbarschaftsinformation

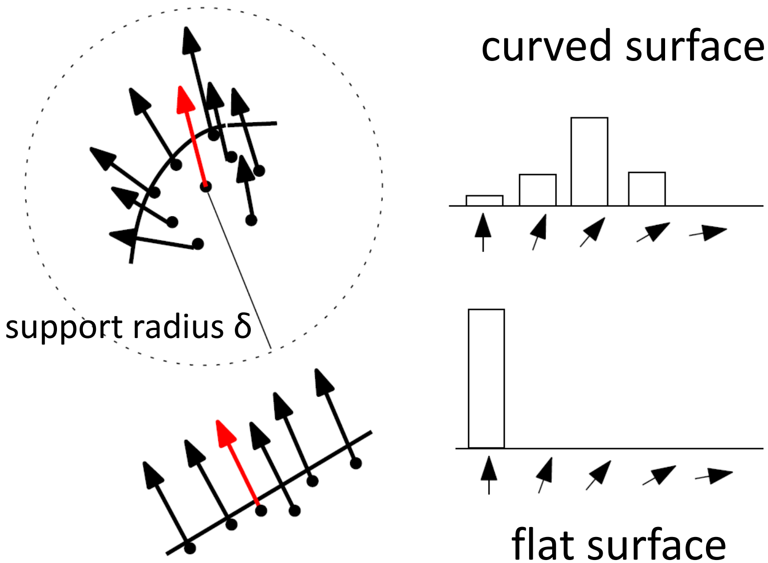
- Einzelne 3D-Punkte tragen keine Information



⇝ Lokale Nachbarschaft ist für Interpretation notwendig

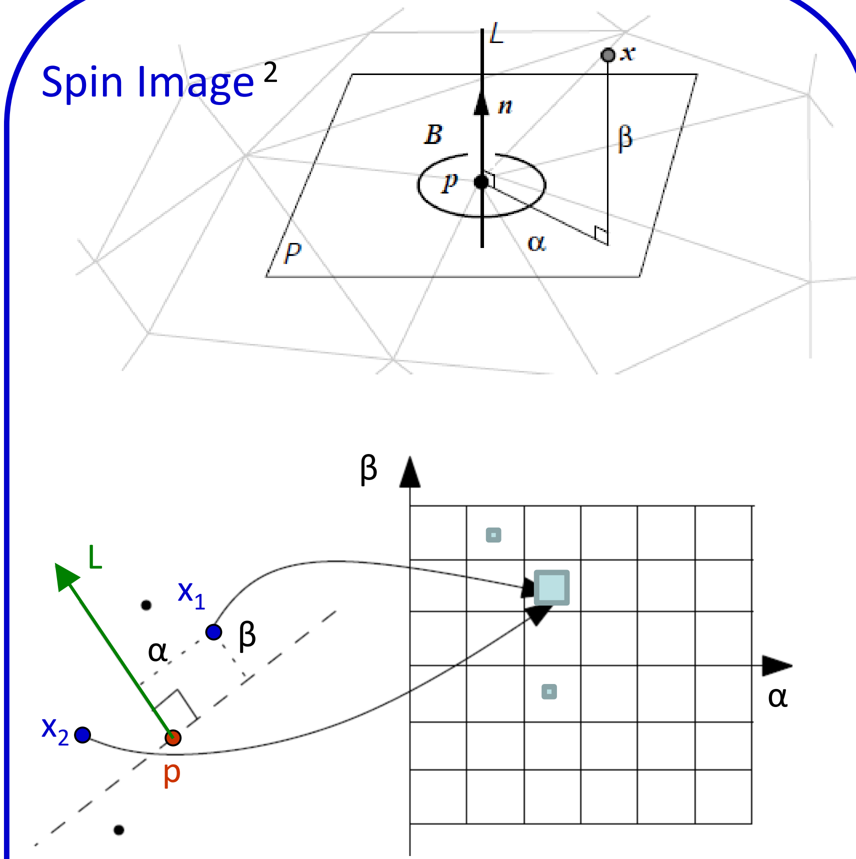
Histogramm-basierte Deskriptoren: Beispiele

Normal Orientation Histogram [1]



- normal orientation around query point
- spatial distribution of points lost

Spin Image [2]



- 2d histogram by line/plane distances
- rotational invariant about ref. axis

1.Triebel et al. *Robust 3D Scan Point Classification using Associative Markov Networks*, ICRA, pp. 2603—2608, 2006.
2.Johnson, Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. TPAMI, 21(5), pp. 433—449, 1999.

Nachbarschaftsinformation

- Aufgabe: Finden aller Nachbarpunkte zu einen Anfragepunkt (*query point*) **q** innerhalb einer Umgebung mit festem Radius *r :*

Fixed-radius neighbors: $\mathcal{N}(\mathbf{q}, r) = \{\mathbf{p} \in \mathcal{P} \mid ||\mathbf{p} - \mathbf{q}|| < r\}$

- Exhaustive Suche:
  - Iteration durch alle Punkte und Finden der Punkte in $\mathcal{N}(\mathbf{q}, r)$
  - Lineare Zeitkomplexität *O*(*N*) in der Zahl N der Punkte in $\mathcal{N}(\mathbf{q}, r)$

Nachbarschaftsinformation

Idee:

- Minimierung der Vergleiche: $\mathbf{p} \in \mathcal{N}(\mathbf{q}, r)$
- Partitionierung des 3D-Raums um schnell irrelevante Raumanteile für die Anfrage zu identifizieren
- Verschiedene Ansätze sind bekannt

  $\leadsto$ hier: *k*-d Bäume

  - Einfache Implementierung
  - Effizient für *3D fixed-radius nearest neighbor*

k-d-Baum

- Ein *k-dimensionaler Baum* oder *k-d-Baum* ist ein binärer Baum
- Jeder Blattknoten repräsentiert einen *k-dimensionalen Punkt*
- Jeder innere Knoten ist mit einer Dimension k_i der k Dimensionen und einer (k-1)-dim. Hyperebene, die orthogonal zur entspr. Dimensionsachse verläuft, assoziiert.
- Jede Hyperebene erzeugt zwei Halbräume
  - → Punkte, deren Wert (bzgl. Dimension k_i) kleiner oder gleich dem Positionswert der Hyperebene sind, werden dem einen Halbraum zugeordnet und im linken Teilbaum repräsentiert.
  - → Alle anderen Punkte werden dem anderen Halbraum zugeordnet und im rechten Teilbaum repräsentiert.

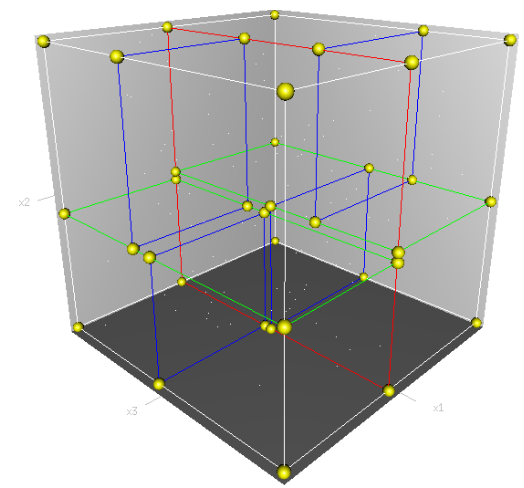


Abbildung: https://en.wikipedia.org/wiki/K-d_tree (29. Febr. 2016)

k-d-Baum: Konstruktion informell

Kanonische Methode:

- Aufbau über schrittweises und zyklisches Besuchen jeder Dimension
- Innere Knoten: Positionierung der neuen trennenden Hyperebene im Median aller Punkte des entsprech. Teilbaums bezogen auf die aktuelle Dimensionsachse
- Blattknoten sind solche mit nur einem Punkt
- Resultat ist ein balancierter *k*-d-Baum

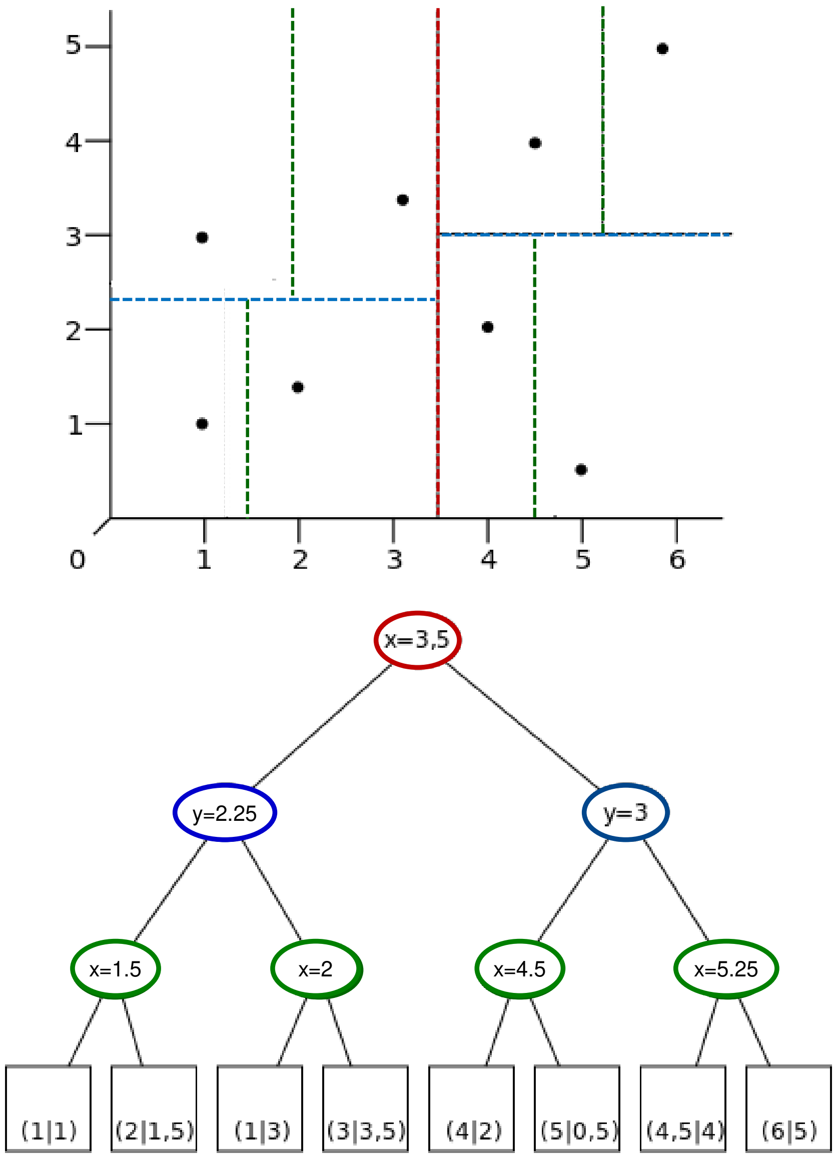


Abbildung nach: https://de.wikipedia.org/wiki/K-d-Baum (29 Febr. 2016)

k-d-Baum: Konstruktion als Pseudocode

Kanonische Methode:

```
function kdtree (list of points pointList, int depth)
{
if |pointList| = 1 then return leaf node with that point
 // Select axis based on depth so that axis cycles through all valid values
 var int axis := depth mod k;  // k = dimension of points
 // Sort point list and choose median as pivot element
 select median by axis from pointList;
 // Create node and construct subtrees
 var tree_node node;
 node.location := median;
 node.leftChild := kdtree(points in pointList before median, depth+1);
 node.rightChild := kdtree(points in pointList after median, depth+1);
 return node;
 }
```

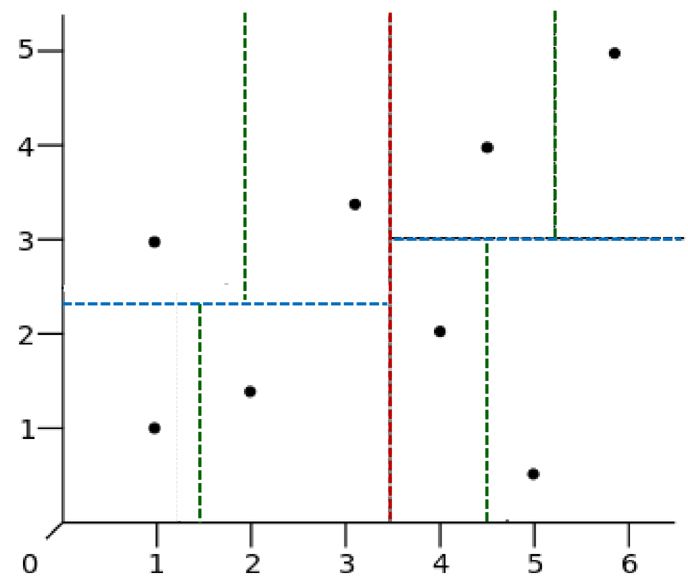
Berechnung des Median: bei gerader Anzahl von Punkten wird der Median als Mittelwert der beiden mittleren Werte der sortierten Daten gewählt (wie im Bspl.)

Abbildung basierend auf: https://en.wikipedia.org/wiki/K-d_tree (29. Febr. 2016)

k-d-Baum: Komplexität für Konstruktion

Kanonische Methode:

- Speicher: *O(n)*
- Tiefe: *O(log n)*
- Zeit: *O(n log n)*



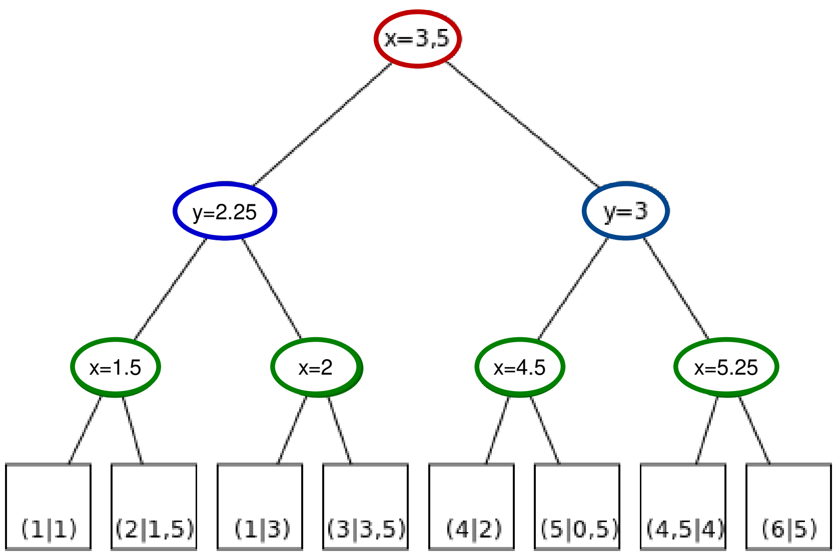


Abbildung nach: https://de.wikipedia.org/wiki/K-d-Baum (29 Febr. 2016)

k-d-Baum: Nächste-Nachbar-Suche (Algm.)

Ziel: *Pruning* von großen Teilbäumen bei Zeitkomplexität $O(\log n)$:

Geg.: Anfragepunkt (*query point*) $\boldsymbol{q}$

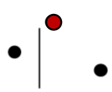
(1) Start mit Wurzelknoten $\boldsymbol{r}$ und großem Startwert d_M für minimale Distanz d

(2) Baumabstieg (in linken/rechten Teilbaum, wenn entspr. Dimensionswert von $\boldsymbol{q} \leq$ bzw. $>$ der Position der trennenden Hyperebene ist) bis zum Erreichen eines Blattknotens mit einem Punkt $\boldsymbol{c}$

(3) Wenn $d' = dist(\boldsymbol{q},\boldsymbol{c}) < d$ dann $d \leftarrow d'$ und Speichern von $\boldsymbol{c}$ als aktuellem nächsten Nachbarn $\boldsymbol{b}$ mit Distanz d

(4) Baumaufstieg mit Überprüfung der Vorgängerknoten von $\boldsymbol{b}$ ob die Distanz d_H der jeweiligen trennenden Hyperebene H zu **q** kleiner ist als d:

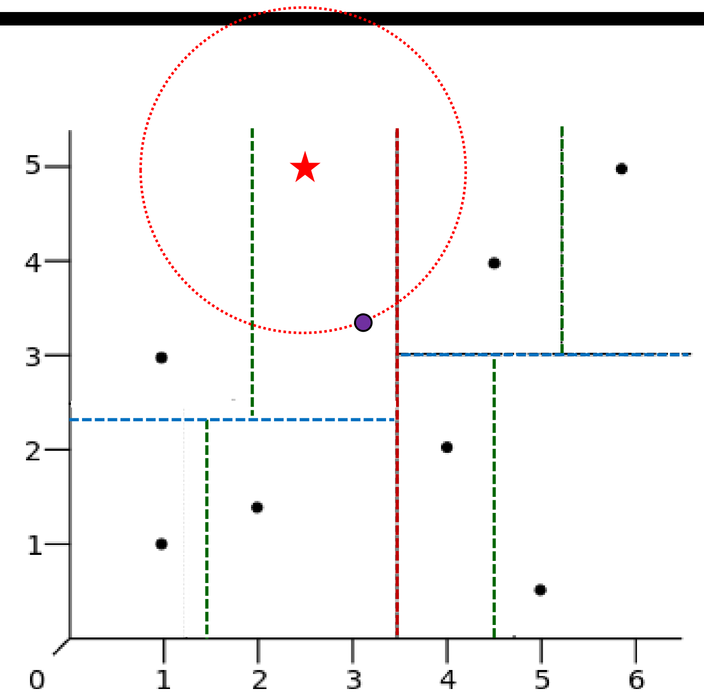
Wenn $d_H < d$ dann gehe zu (2) mit Wahl des anderen Teilbaums

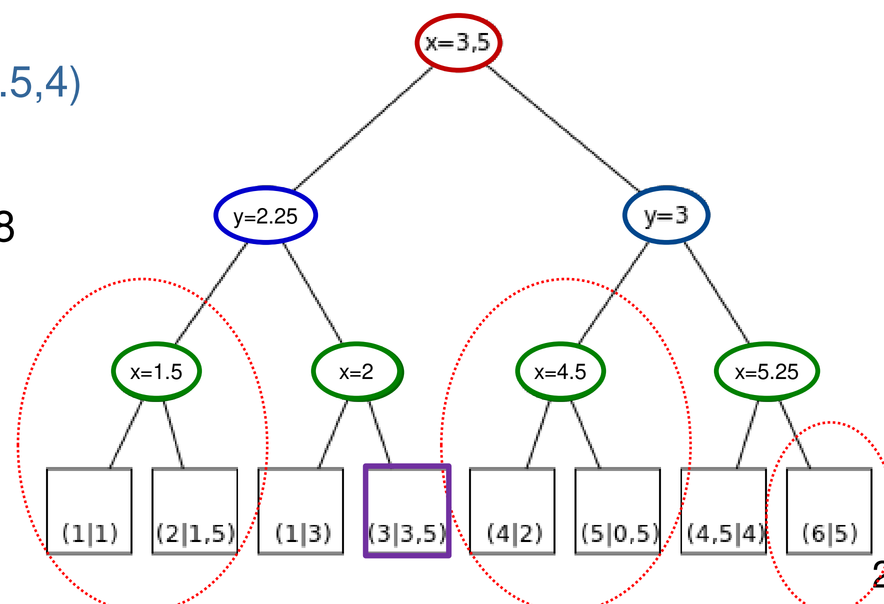


k-d-Baum: Nächste-Nachbar-Suche (Bsp.)

Query point **q** = (2.5,5), d=10

- Baumabstieg: 2.5 < 3.5, 5 > 2.25, 2.5 > 2
  → **c** = (3,3.5) → **b** = (3,3.5) mit *d* = 1.58
- Baumaufstieg: dist(**q**,(x = 2)) = 0.5 < 1.58
  → Baumabstieg: **c** = (1,3)
  dist(**q**,**c**) = 2.50 ↝ **b** bleibt (3,3.5)
- Baumaufstieg: dist(**q**,(y = 2.25)) = 2.75 > 1.58
  ↝ **b** bleibt (3,3.5)
- Baumaufstieg: dist(**q**,(x = 3.5)) = 1 < 1.58
  → Baumabstieg: 5 > 3, 2,5 < 5.25 → **c** = (4.5,4)
  dist(**q**,**c**) = 2.23 ↝ **b** bleibt (3,3.5)
- Baumaufstieg: dist(**q**,(x = 5.25)) = 2.75 > 1.58
  ↝ **b** bleibt (3,3.5)
- Baumaufstieg: dist(**q**,(y = 3)) = 2 > 1.58
  ↝ **b** bleibt (3,3.5)





k-d-Baum: Bereichssuche (Bsp.)

Deskriptoren basieren häufig auf *allen* Punkten innerhalb einer bestimmten Nachbarschaftsumgebung mit sog. *support radius r*.

⇝ rekursive *Range query* oder *Range search* startend mit v = Wurzelknoten und Hypersphäre S mit Radius *r* um Anfragepunkt **q**:

RSearch(v,r):

if v is a leaf **then** report the point stored in v if it lies in S
else if Region(v) is contained in S **then** report all points in the subtree of v
else if Region(left(v)) intersects S **then** RSearch(left(v),S)
If Region(right(v)) intersects S **then** RSearch(right(v),S)

- Mit *Region*(*v*) = Unterraum, der einem innerem Knoten v entspricht
- Zeitkomplexität: $O(k \cdot n^{(1 - 1/k)})$ bei k Dimensionen und n Punkten *

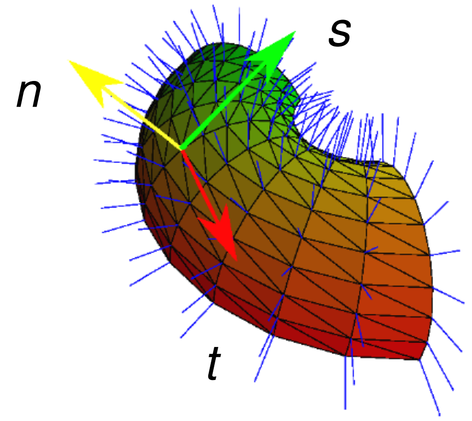
* Lee, D. T.; Wong, C. K. (1977). "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees". Acta Informatica **9**

Schätzung der Normalen

- Deskriptoren u.a. Punktwolkenoperationen nutzen Normalenvektoren

- Normalenvektoren sind orthogonal zur Oberfläche orientiert und als Vektorprodukt aus Tangentialvektoren **s** und **t** ableitbar:

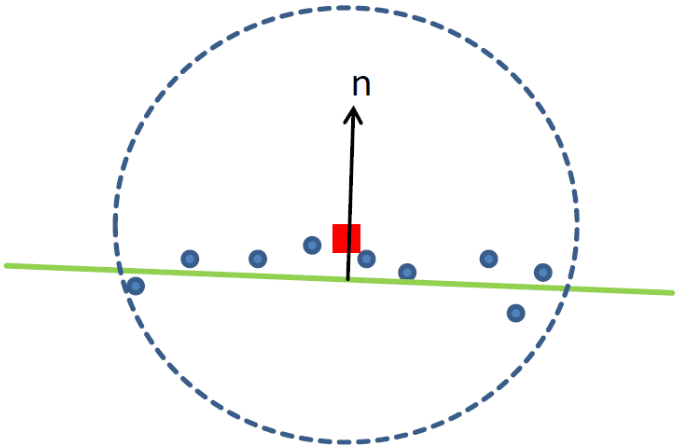
$$\mathbf{n} = \mathbf{s} \times \mathbf{t}$$



- Für Punktwolken sind Dreiecksnetze (triangular mesh) und Tangentialvektoren **s**, **t** nicht generell trivial ableitbar

- Eine Option ist dann die direkte Schätzung von Normalenvektoren aus den Punkten lokaler Nachbarschaften

Schätzung durch lokale Ebenenapproximation (1)



- Idee: Finde Ebene E, welche die Distanzen zu den Punkten in der Nachbarschaft von Anfragepunkt **q** minimiert
- Zur Schätzung des Normalenvektors **n** von Ebene E in **q** wird die Hessische Normalform genutzt:

$$\mathbf{n}^T \cdot \mathbf{p} - d = 0 \text{ mit } //\boldsymbol{n}//^2 = 1$$

- Zur Erinnerung: die (orientierte) Distanz d(**p**,E) eines Punkts **p** zur Ebene E:

$$d(\mathbf{p},E) = \mathbf{n}^T \cdot \mathbf{p} - d$$

Schätzung durch lokale Ebenenapproximation (2)

- Ziel: Finde Ebene E mit Normalenvektor $\mathbf{n}$, $||\mathbf{n}||^2 = 1$, und Skalar d, welche die Summe der Distanzen $d(\boldsymbol{p}_i, E)$, $\boldsymbol{p}_i \in \mathcal{N}(\mathbf{q}, r)$ für Anfragepunkt $\mathbf{q}$ minimiert:

$$\min_{\mathbf{n},d} 1/k \cdot \sum_i ||\mathbf{n}^T \cdot \mathbf{p}_i - d||^2 , \; i = 1,...,k$$

$$\text{mit } ||\mathbf{n}||^2 = 1 \text{ und } k = |\mathcal{N}(\mathbf{q}, r)|$$

- Lösung durch Hauptkomponentenanalyse auf Basis der Kovarianzmatrix über den nächsten Nachbarn des Anfragepunktes $\mathbf{q}$

Schätzung durch lokale Ebenenapproximation (3)

Genauer:

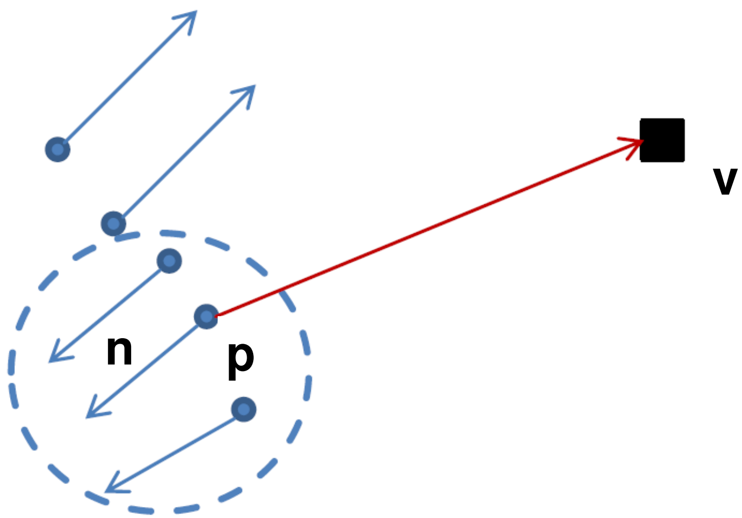
- Für jeden Anfragepunkt **q** Erzeugung der Kovarianzmatrix **C** über allen Nachbarpunkten $\boldsymbol{p}_i$, $\boldsymbol{p}_i \in \mathcal{N}(\mathbf{q},\mathrm{r})$:

$$\mathbf{C} = 1/k \cdot \sum_i (\mathbf{p}_i - \mathbf{p}_m)^T(\mathbf{p}_i - \mathbf{p}_m) = 1/k \cdot \sum_i \mathbf{p}_i \cdot \mathbf{p}_i^T - \mathbf{p}_m \cdot \mathbf{p}_m^T$$

$$\text{mit Mittelwert } \boldsymbol{p}_m = 1/k \cdot \sum_i \mathbf{p}_i \ \text{ für } k = |\mathcal{N}(\boldsymbol{q}, r)|$$

- **C** ist per Def. positiv semidefinit und symmetrisch
  - – Eigenwerte kodieren Varianzen in Richtung der Eigenvektoren
  - ⇝ Schätzung des Normalenvektors als Eigenvektor mit kleinstem Eigenwert minimisiert die quadratische Abweichung

Konsistente Normalenorientierung



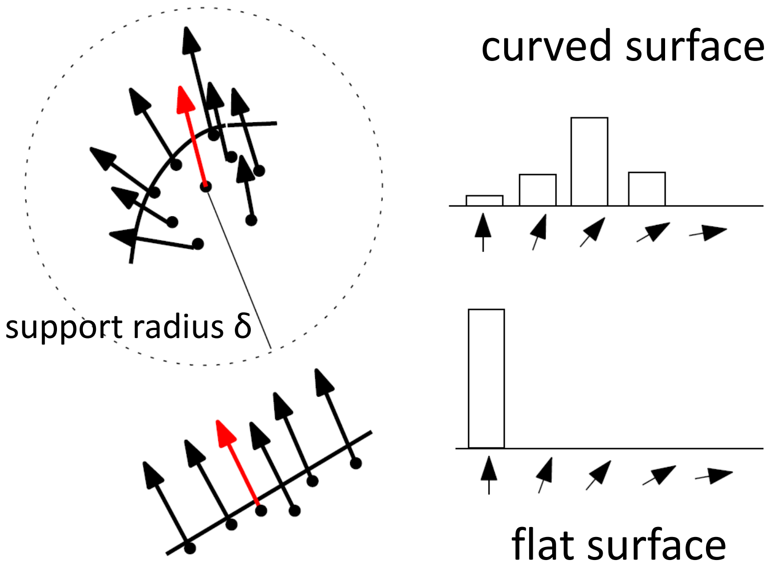
- **Problem**: Orientierung der Eigenvektoren ist ggf. nicht konsistent

- **Lösung** bei bekannten Beobachtungpunkt **v**:

  Umkehrung des Normalvektors $\mathbf{n}' = -\mathbf{n}$ von Punkt $\mathbf{p}$, wenn $\mathbf{n}^T \cdot (\mathbf{v} - \mathbf{p}) < 0$
  d.h.: Winkel zw. Vektor $\mathbf{v} - \mathbf{p}$ und Normalen $\mathbf{n}$ ist größer als 90°

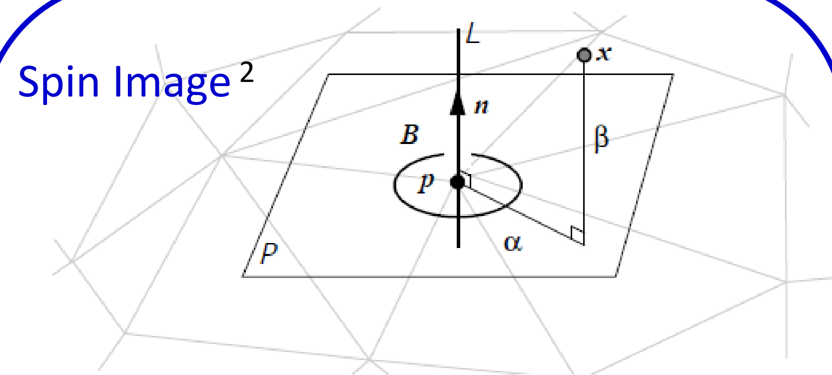
Histogramm-basierte Deskriptoren: Wiederholung

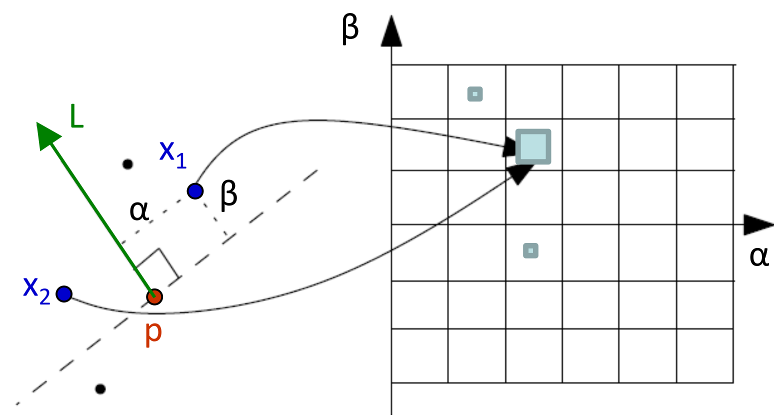
Normal Orientation Histogram [1]



- normal orientation around query point
- spatial distribution of points lost

Spin Image [2]





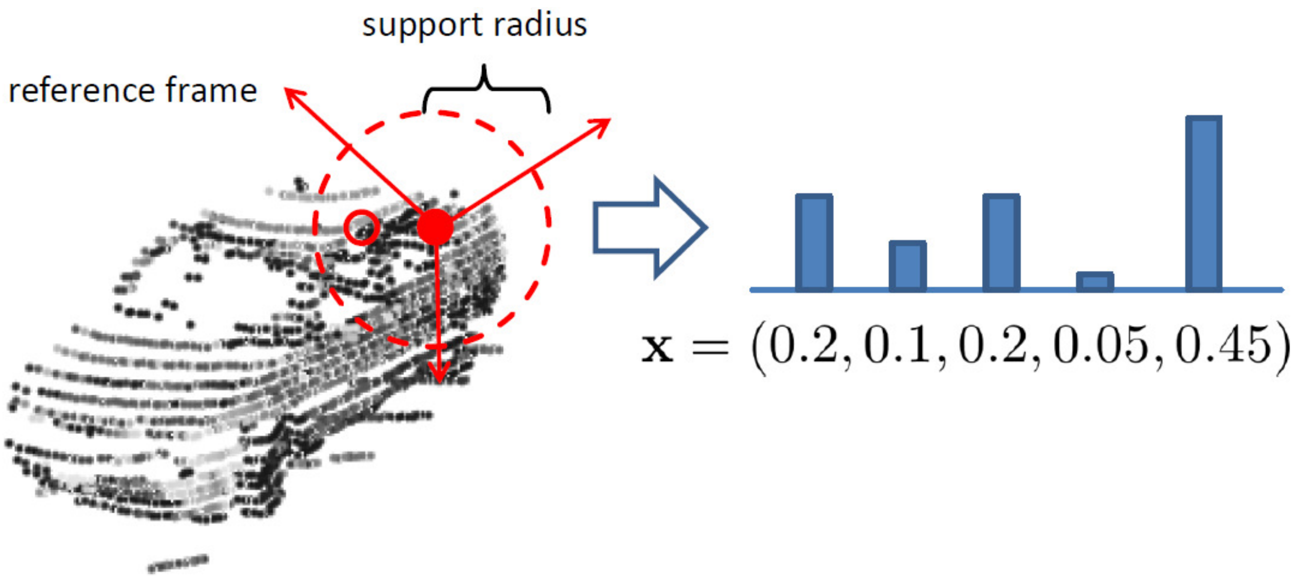
- 2d histogram by line/plane distances
- rotational invariant about ref. axis

1.Triebel et al. *Robust 3D Scan Point Classification using Associative Markov Networks*, ICRA, pp. 2603—2608, 2006.
2.Johnson, Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. TPAMI, 21(5), pp. 433—449, 1999.

Histogramm-basierte Deskriptoren: Anwendung

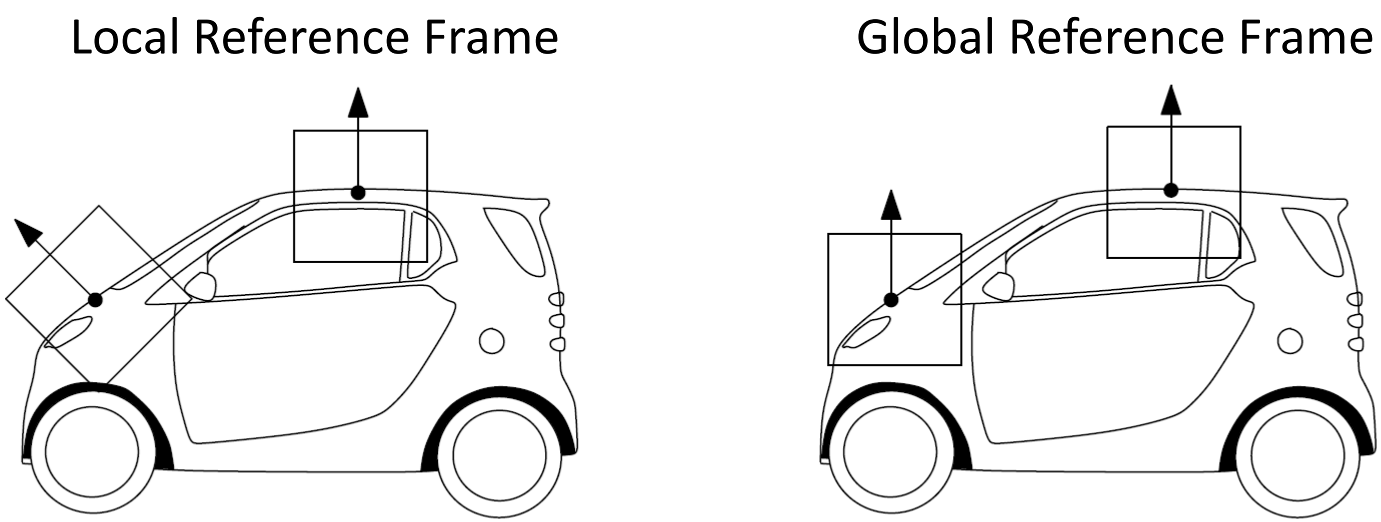
Hyperparameter:

1) Referenzachse (reference frame, reference axis)
2) Support radius, d.h., Radius der Nachbarschaftssphäre $\mathcal{N}(\mathbf{q},r)$
3) Anzahl der Histogramm-Bins



$$\mathbf{x} = (0.2, 0.1, 0.2, 0.05, 0.45)$$

Histogramm-basierte Deskriptoren: Referenzachse



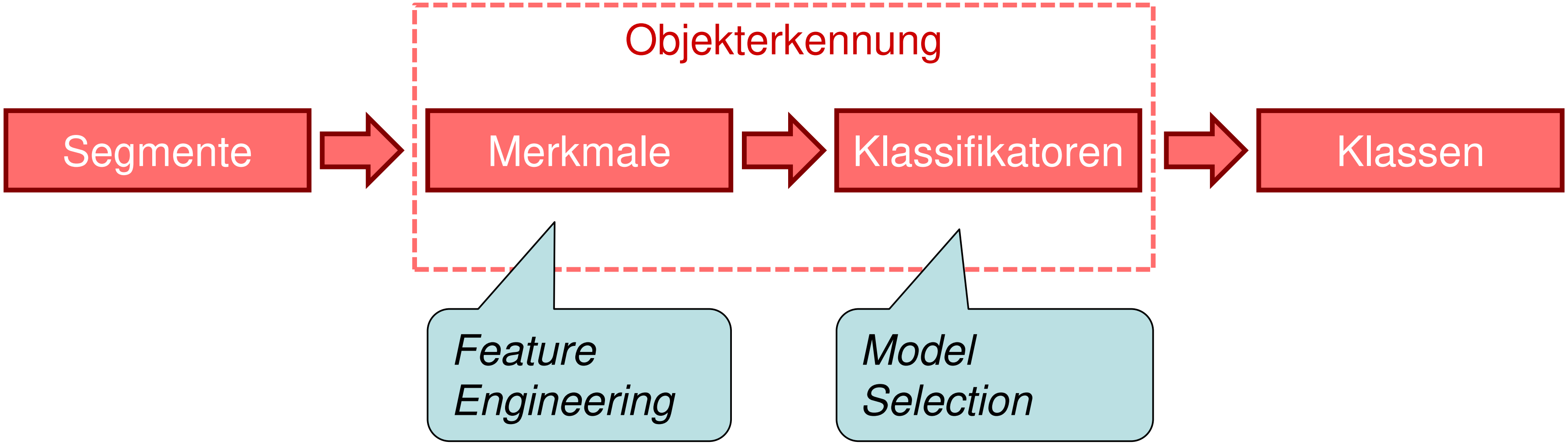
- Wahl ist abhängig von Applikation
- Generell:
  1. Normalenvektor von Anfragepunkt **q** (local reference frame)
  2. Festgelegte Referenzachse, z.B., Up-Vector eines Welt-/Szenenkoordinatensystems (global reference frame)

Entwurfsaspekte von Punkteklassifikation & Objekterkennung

Auch bei Punktwolken zeigen sich dieselben zwei Entwurfsaspekte

(1) Feature Engineering: Welche Merkmale/Deskiptoren sind geeignet?

(2) Model Selection: Welches Modell des maschinellen Lernens* ist geeignet?

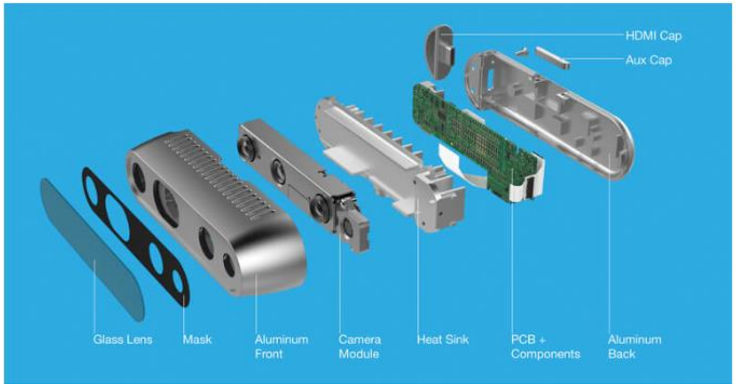


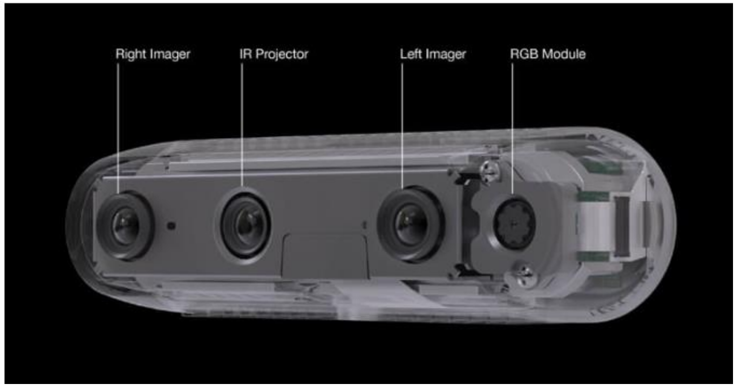
* Siehe Vorlesung „Grundlagen der KI“: Lernverfahren, Hyperparameter, etc. (SVM, LDA, Boosting, KNN, Clustering, …)

RGB-D-Bilder: RealSense

Verschiedene Sensoren erlauben auch die kombinierte Erfassung von Farb- und Tiefeninformation zu RGB-D-Bildern oder RGB-D-Punktwolken

- RealSense 435 Stereokamera





Bildquelle: https://www.intelrealsense.com/de/promo-depth-camera-d435/ (10.01.2020)

RGB-D-Punktwolken: Artec Space Spider

- Genauigkeit bis zu 50 Mikrometern
- Auflösung bis zu 100 Mikrometern

Bildquelle: https://www.artec3d.com/de/portable-3d-scanners/artec-spider

3D Pointclouds, left to right: Dornfelder, Calardis Blanc, Pinot Noir and Riesling grape bunch

Zusammenfassung (1)

- Zur Objekterkennung und Szenenrekonstruktion sind 3D-Laserabstandsdaten die Wahl als aktive Sensoren für Outdoor-Szenarien.
- Lasersensoren messen Distanzen durch Laufzeitmessung der emittierten Strahlen (*time of flight*, TOF))
- Lasersensoren erzeugen i.A. 3D-Punktwolken als ungeordnete Menge von 3D-Punkten. Damit wird die Suche nach nächsten Nachbarn von Anfrage-punkten aufwändig.
- Für die effiziente Suche nach nächsten Nachbarn werden Partitionierungen des 3D-Raums eingesetzt um schnell irrelevante Raumanteile für Anfragen zu identifizieren (*pruning*)
- *k*-dimensionale Bäume erlauben die Suche nach nächsten Nachbarn in sublinearer Zeitkomplexität
- Zur Charakterisierung von Punkten einer 3D-Punktwolke werden histogramm-basierte 3D-Deskiptoren verwendet

Zusammenfassung (2)

- Zur Berechnung von 3D-Deskiptoren und für andere Aufgaben sind häufig Normalenektoren von Punkten zu schätzen, indem eine lokale Ebenenapproximation durchgeführt wird
- 3D-Punktwolken sind auch durch optische Triangulationsverfahren ableitbar mit dem Vorteil auch Farbinformation für die Punkte vorzuhalten $\leadsto$ RGB-D-Punktwolken
- Zur Klassifikation von Punkten und der Objekterkennung in Punktwolken werden auch Ansätze von *Deep Learning* eingesetzt *

* Liu W, Sun J, Li W, Hu T, Wang P. Deep Learning on Point Clouds and Its Application: A Survey. Sensors (Basel). 2019;19(19):4188. Published 2019 Sep 26. doi:10.3390/s19194188