LIDAR-based Perception for Offroad Navigation

Michael Himmelsbach* Felix von Hundelshausen Hans-Joachim Wünsche

abstract: We describe a LIDAR-based autonomous navigation approach applicable to both urban and non-urban environments. At the core of the method is a set of "tentacles" that represent precalculated trajectories defined in the ego-centered coordinate system of the vehicle. Similar to an insect's antennae, they fan out with different curvatures discretizing the basic driving options of the vehicle. We detail how the approach can be used for exploration of unknown environments and how it can be extended to combined GPS path following and obstacle avoidance allowing save road following in case of GPS offsets.

1 Introduction

We present a simple method for autonomous robot navigation in unknown environments. We describe and demonstrate it using our vehicle MuCAR-3, a VW Touareg equipped with a Velodyne 64 beam 360 degrees LIDAR.

While quite complex approaches to mobile robot navigation exist, our research was driven by the search for simplicity: *What is the simplest approach that lets a robot safely drive in an unknown environment?* Our basic intention underlying the approach is to let our robot move within an unknown environment similarly to how a beetle would crawl around and uses its antennae to avoid obstacles. Indeed, our basic approach consists of using a set of virtual antennae that we call "tentacles" probing an ego-centered occupancy grid for drivability.

2 Occupancy Grid

We use a two dimensional occupancy grid, where each cell stores a value expressing the degree of how occupied that cell is by an obstacle. We assume that the LIDAR data of the last frame has been transformed into a list of points in a global cartesian 3D coordinate system taking into account the vehicle's motion (exploiting an INS system on board). Each cell's value is then computed to be the maximum absolute difference in z-coordinates of all points falling into the respective grid cell. We refer to this value as the cell's *occupancy value*.

For accumulating data of successive LIDAR turns, we use a grid wrapping technique. While to the outside the vehicle always appears to be aligned with the grid's coordinates, the vehicle is allowed to freely move internally, keeping track of the vehicle's pose in grid coordinates. All positions outside the grid boundaries get wrapped back by simple 2D modulo operations. With wrapping, different coordinates are mapped to the same grid

*Universität der Bundeswehr, Technik Autonomer Systeme, e-mail: michael.himmelsbach@unibw.de.

cell. Thus, we define a ROI around the vehicle and assure unique coordinates within this ROI. This is done by clearing all cell's information outside the ROI. Clearing cells is done efficiently with methods from computational geometry: the last and the current ROI are expressed as polygons in grid coordinates, and a clipping operation yields a polygonal representation of the area from old ROI that is not part of the new ROI. The resulting polygons are then converted to tristrips and the triangle scan conversion algorithm yields the grid coordinates of cells to be cleared.

3 Tentacle selection and execution

We use 16 speed depending sets of tentacles, each of which contains 81 different tentacles corresponding to a specific velocity. The speed sets cover a range of velocities from 0 to 10 m/s. All tentacles are represented in the local coordinate system of the vehicle. They start at the center of the vehicle's rear axis and take the shape of circular arcs. Each arc represents a trajectory corresponding to a specific steering angle.

Every tentacle is assigned a *support area*, defined as the set of grid cells within a radius d_s to any point on the tentacle. Similarly, a *classification area* is defined, including all cells within a distance $d_c < d_s$. Both d_s and d_c increase with sets for higher speeds. For selecting a tentacle, first all drivable tentacles from current speed set are determined. For every drivable tentacle, we calculate 3 properties – clearance, flatness and trajectory-leading – described by numerical values. These values are later combined into a single decision value, and selecting a tentacle is simply a matter of minimizing the decision value.

Drivability

Both a tentacle's drivabilty as well as the distance to the first obstacle can be determined at the same time. For this purpose, a histogram aligned along each tentacle is filled by projecting each cell in the classification area to a histogram bin and incrementing its counter if the occupancy of the cell indicates an obstacle. When all cells are done, we slide a window covering 5 bins over the histogram. Once the sum of bins within this windows exceeds 2, the first obstacle has been detected at length to the first window bin.

Even if an obstacle can be found, the tentacle might still be drivable for some time. Thus, we calculate the crash distance the vehicle needs to stop using a constant convenient deceleration plus a safety distance. A tentacle is finally considered undrivable only if the first obstacle is within this crash distance.

Clearance

The *clearance value* is the first of three decision-affecting values computed for each drivable tentacle. To make those values comparable they are all normalized to the range $[0, ...1]$. For the *clearance value*, this means passing the distance to the first obstacle to a shifted sigmoid-like function.

Flatness

While all drivable tentacles can be executed without causing damage to the vehicle, paths leading over smooth terrain are to be preferred. Also, free space should be exploited

as much as possible. This is expressed in terms of the *flatness value*, computed as the weighted average of all occupancy values of cells in the support area of a tentacle. Normalization of the flatness value is done in analogy to that of the clearance value.

Trajectory

While the latter two values aim at obstacle avoidance in an unknown environment, the *trajectory value* pushes the vehicle towards following a given trajectory, e.g. defined by GPS waypoints. The trajectory value is computed by considering the distance and tangent orientations of two corresponding points, one sampled from the tentacle, the other from the trajectory. Each tentacles trajectory value is normalized by considering the min. and max. values over all tentacles.

Combining Clearance, Flatness and Trajectory-leading

For each drivable tentacle, the values $v_{\text{clearance}}$, v_{flatness} and $v_{\text{trajectory}}$ are linearly combined to a single decision value. By assigning weights to the values the behavior of our approach can be tuned, and the tentacle minimizing $v_{combined}$ gets selected. If no drivable tentacle exists, the tentacle with the farthest obstacle is selected, and the vehicle is commanded to stop before reaching it. Both a simple similarity method as well as a more elaborated Kalman Filtering approach can be used for hysteresis.

Tentacle execution

While the curvature of the selected tentacle might just be commanded as steering angle to the vehicle, we decided to execute tentacles by what we call fragment execution: the tentacle is transformed to some global coordinates (e.g., integrated from IMU) and passed as the target trajectory to a path controller. The advantage of this method is that trajectory execution can be controlled at a higher frequency allowing to compensate local disturbances using INS information.

4 Results and conclusion

Our approach was tested and demonstrated in numerous scenarios experimenting with different parameter settings and execution modes. At the C-Elrob 2007 we run the vehicle in exploration mode. That is the vehicle decided to drive always towards the flattest areas without regarding any given trajectory. Tentacle execution was done by temporal filtering of the selected tentacles and fragment execution as described above. Even this simple approach worked surprisingly well, and at the C-Elrob 2007 MuCAR-3 drove a combined urban and non-urban course in record time.

The main advantage of our approach is its simplicity and the fact that it is well-approved in many different scenarios including offroad terrain, mountain areas with serpentines and conventional residential roads. However, it is still a data-driven, reactive approach and no recognition of objects takes place. Hence, we see our approach at the lowest level in a hierarchical approach providing a protective shield in case recognition and tracking fails.

LIDAR-basierte Perzeption in Offroad Szenarien

Michael Himmelsbach
Dr. Felix von Hundelshausen
Prof. Dr.-Ing. H.-J. “Joe” Wünsche

Universität der Bundeswehr München

Department of Aerospace Engineering
Autonomous Systems Technology
Prof. Dr.-Ing. Hans-Joachim Wünsche

technik autonomer systeme

Chronologie des *tentacles* Ansatzes

März '07: Komplexer Ansatz

- Erkennen von Bordsteinen
- Gruppieren von Bordsteinen zu Straßenhypothesen
- Aufbau und tracking eines Straßenskeletts

August '07: C-ElRob 07

- Erster Einsatz des "*tentacles*" Algorithmus

Juni '07: Einfacher Ansatz

- Erkenntnis, dass der komplexe Ansatz nicht bis zur C-ElRob 07 zu realisieren ist
- Motivation: ***Was ist der einfachste Ansatz für autonome Navigation in unbekannter Umgebung?***
- Entwicklung eines ersten Algorithmus mit Spitznamen "*tentacles*"

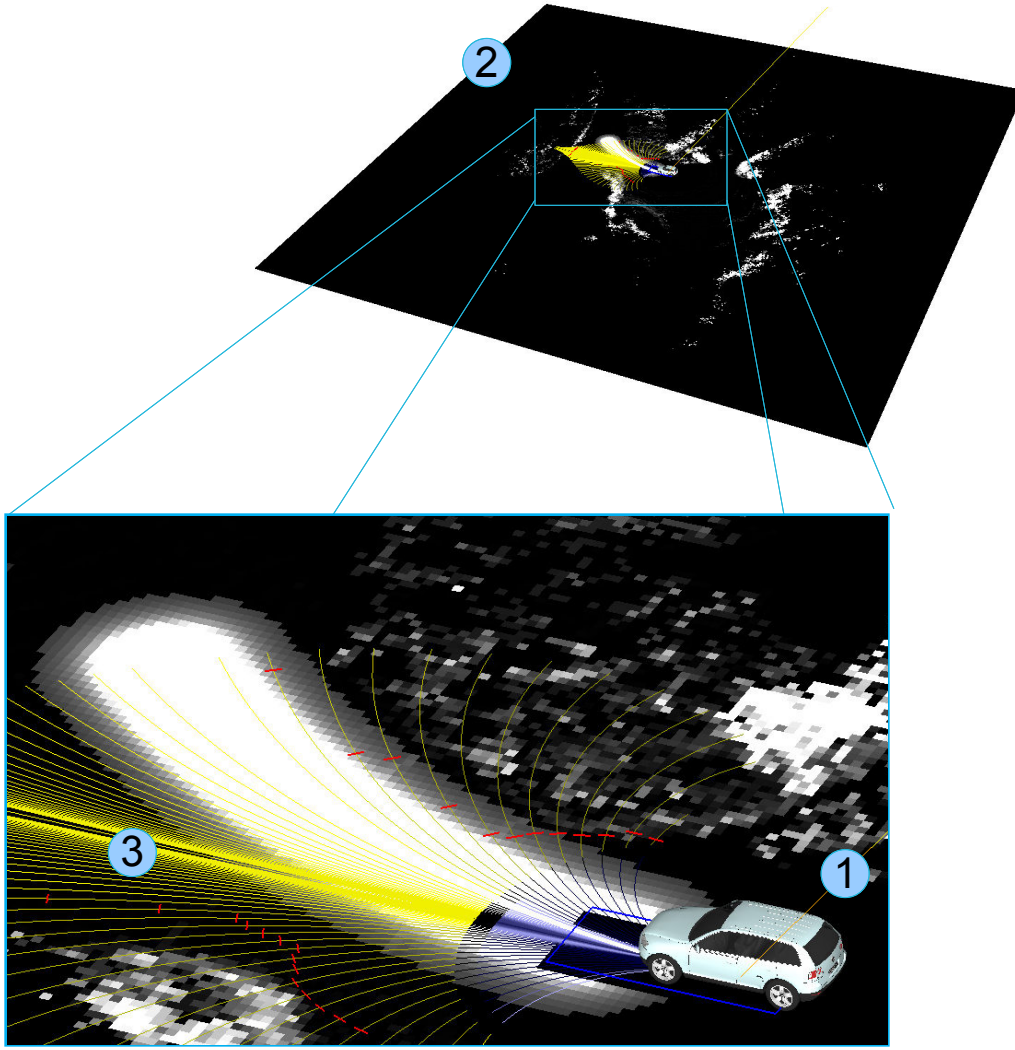
November '07: DARPA Urban Challenge

- Verbesserung des "*tentacles*" Algorithmus und Integration in das System von Team *AnnieWay*, Karlsruhe

Übersicht – der *tentacles* Basis-Algorithmus

1. Roboter MuCAR-3
2. Ego-zentriertes occupancy grid
3. tentacles

   eine fixe Anzahl vorberechneter lokaler Trajektorien



Basis-Algorithmus

- 1. Auswahl des besten *tentacle* (LIDAR framerate, 10Hz)
- 2. Hysteresis, zeitliches Filtern
- *3. tentacle execution*: Ausführen des aktuellen *tentacle.*

Occupancy Grid – Belegtheit von Zellen (1/4)

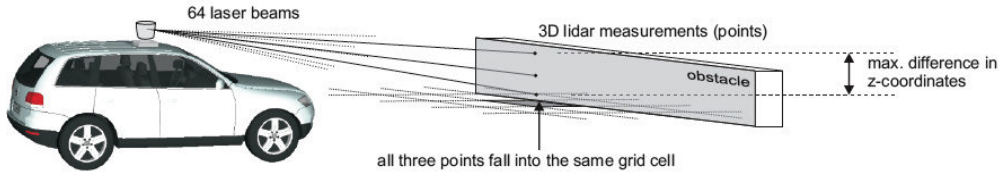
technik autonomer systeme

Inertial korrigierte Punktwolke

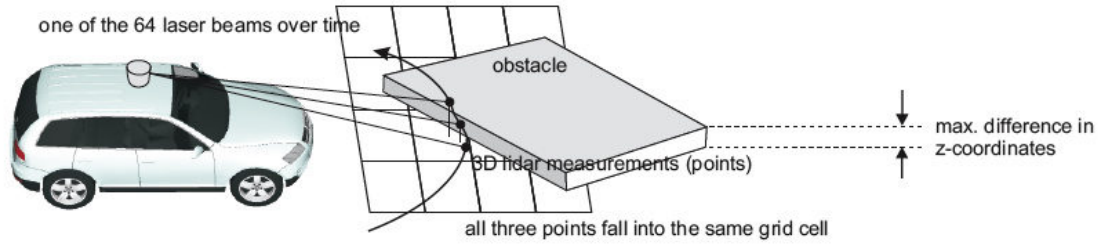
- Transformation jeder Messung in globales Koordinatensystem unter Berücksichtigung der Fahrzeugbewegung
- Rücktransformation aller Punkte in Fahrzeugkoordinaten am Ende einer LIDAR-Umdrehung

Aktualisierung des grids mit neuer Punktwolke

- Abbildung jedes 3D Punktes auf eine Zelle c im grid
- Aufzeichnen der minimalen und maximalen gemessenen Höhe über jeder Zelle
- Höhendifferenz über einer Zelle bestimmt den Grad ihrer Belegtheit c_{oc} (*occupancy value*)



Hinderniserkennung durch 3 Laserstrahlen mit versch. vertikaler Ausrichtung

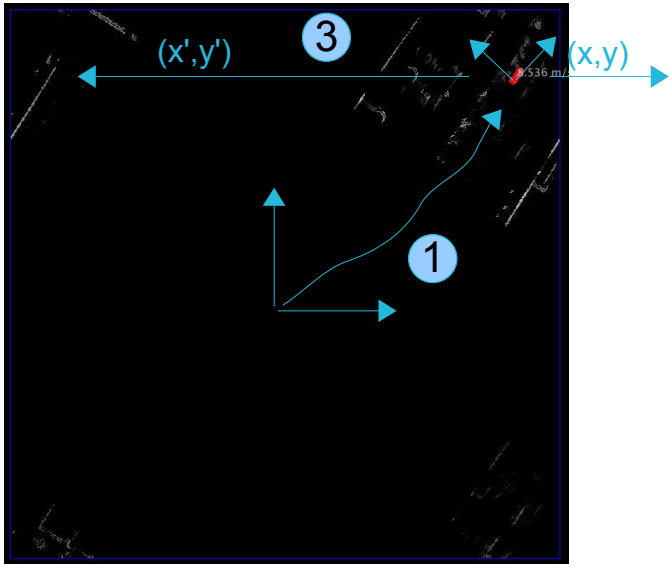


Hinderniserkennung durch einen Laserstrahl bei versch. Rotationswinkeln

Occupancy Grid – Datenakkumulation (2/4)

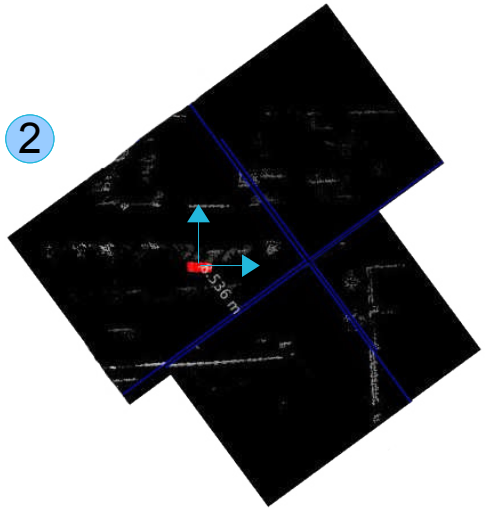
Erdfestes grid mit Wrapping

- 2 Sichtweisen
  - Intern: aktuelle Fahrzeugpose in Gridkoordinaten (1)
  - **Extern: Ego-zentriertes grid**! (2)
- „*wrapping*" von Koordinaten außerhalb der Grenzen des grid zurück in das grid
  - Einfache modulo Operation



Datenakkumulation

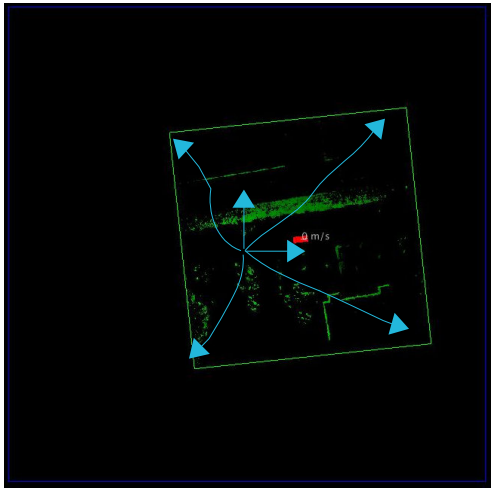
- Akkumulation mehrerer Punktewolken in das grid
- Fahrzeugbewegung in Gridkoordinaten durch **INS**/IMU/ICP
- Problem: *wrapping*
  - Abbildung unterschiedlicher Koordinaten auf die selbe Gitterzelle (3)



Occupancy Grid – Datenakkumulation (3/4)

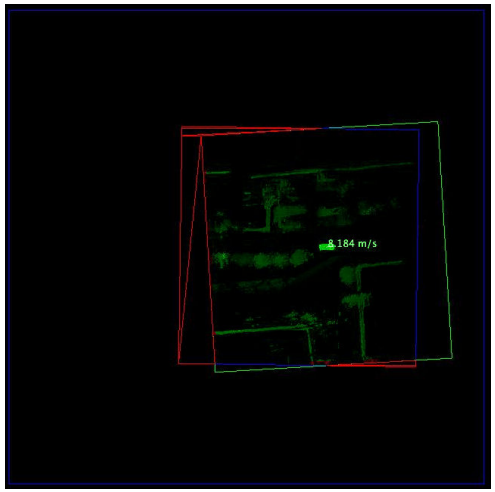
Lösung: Eindeutige Abbildung innnerhalb eines fahrzeugzentrierten ROI

- Quadrat in Gridkoordinaten, kein *wrapping* der Eckpunkte
- Aktualisierung mit Fahrzeugbewegung
- Zurücksetzen der Daten aller Zellen ausserhalb des ROI



Effizientes Zurücksetzen von Zellen

- ROI repräsentiert durch Polygon in diskretisierten Gridkoordinaten (“Bildkoordinaten” über Zellen)
- Bestimmen der Differenz von neuem und altem ROI durch *polygon clipping*
- Zerlegung der Differenz in Dreiecke
- Anwenden des *triangle scan conversion* Algorithmus' liefert Koordinaten aller Zellen innerhalb eines Dreiecks
- *wrapping* der Gridkoordinaten und Zurücksetzen der entsprechenden Zellen



Behandlung dynamischer Objekte

- Seperater Algorithmus zum Erkennen und Verfolgen dynamischer Objekte
- Vorhersage zukünftiger Positionen und Setzen der entsprechenden Zellen als belegt

Basis-Algorithmus Schritt 1 – Auswahl eines *tentacle*

Gewünschtes Navigationsverhalten

- Vermeide Hindernisse
- Bevorzuge ebenen Grund
- Folge einem vorgegebenen GPS-Pfad

Eigenschaften von *tentacles* repräsentiert durch realwertige Skalare

- Befahrbarkeit $v_{drivable} \in \{0, 1\}$
- Ebenheit $v_{flatness} \in [0; 1]$
- Distanz zum nächsten Hindernis $v_{clearance} \in [0; 1]$
- Übereinstimmung mit dem GPS-Pfad $v_{trajectory} \in [0; 1]$

Bestimmen des “besten”, zielführenden *tentacles*

- Klassifikation aller *tentacles* in befahrbar und nicht-befahrbar
- Berechnung der Eigenschaften Ebenheit, Distanz zum nächsten Hindernis und Übereinstimmung mit dem GPS-Pfad für alle befahrbaren *tentacles*
- “bestes” *tentacle* ist befahrbar und hat minimale gewichtete Linearkombination aller Eigenschaften:

$$v_{combined} = a_{flatness} v_{flatness} + a_{clearance} v_{clearance} + a_{trajectory} v_{trajectory}$$

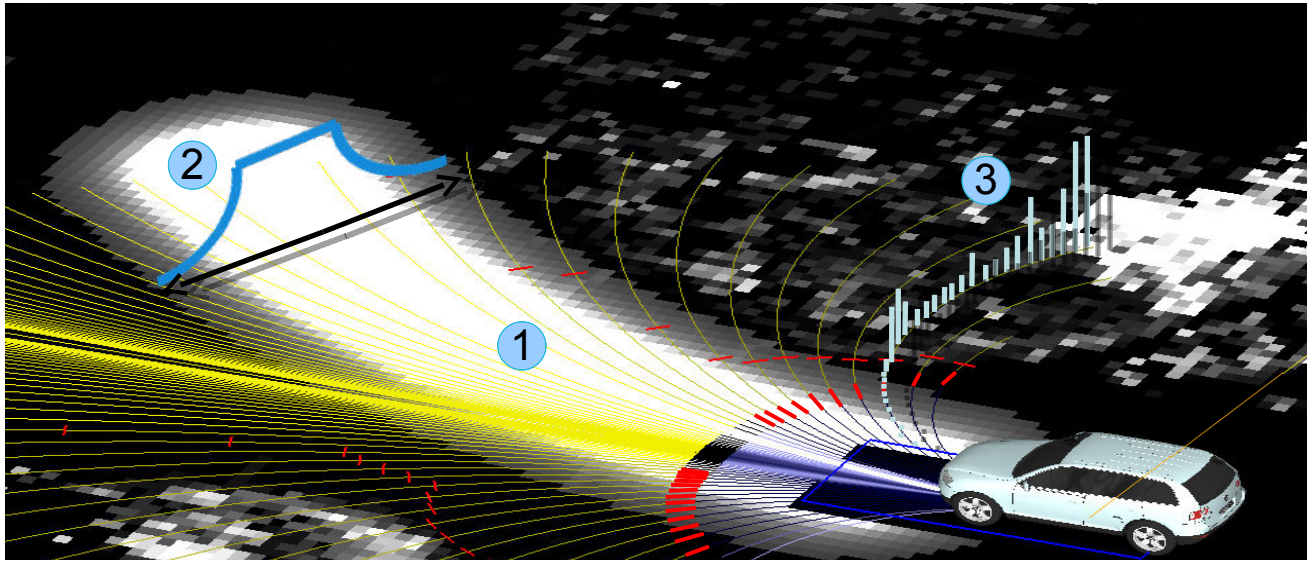
$$a_{flatness}, a_{clearance}, a_{trajectory} \in \mathbb{R}$$

tentacle – support & classification area

Jedes *tentacle* besitzt eine support area S (1)

Information für jede Zelle $c \in S$ innerhalb der support area

- Gewicht gemäß einer lateralen Profilfunktion c_w (2)
- Index zu einem Histogram, das entlang des *tentacle* angeordnet ist c_{hist_index} (3)
- Information, ob Zelle zur Bestimmung der Befahrbarkeit des *tentacle* beiträgt $c_{classification}$
- **offline vorberechnet**



Ebenheit ist gewichtetes Mittel der occupancy values aller Zellen in der support area:

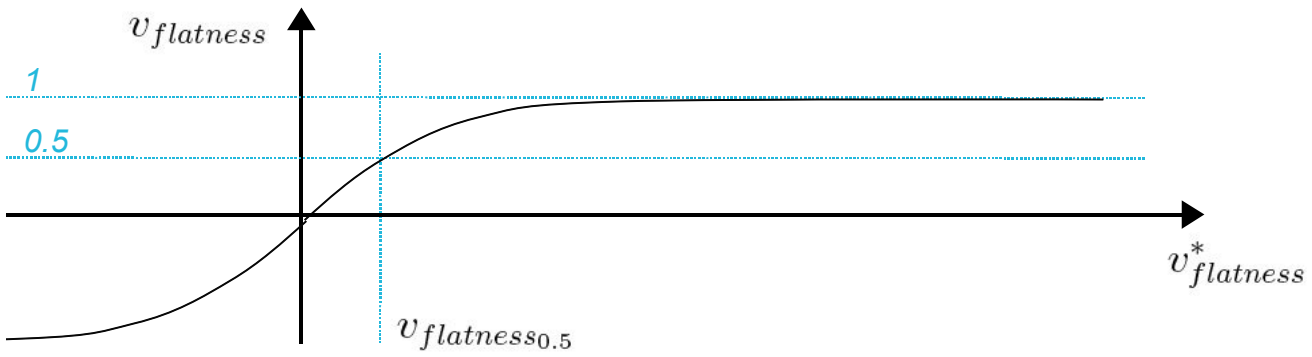
$$v^*_{flatness} = \frac{\sum_{c \in S} c_w c_{oc}}{\sum_{c \in S} c_w}$$

Problem: Keine obere Grenze für $v^*_{flatness}$

- Normalisieren von $v^*_{flatness}$ mit einer skalierten, verschobenen Sigmoid-Funktion:

$$v_{flatness} = \frac{2}{1 + e^{-c_{flatness} v^*_{flatness}}} - 1$$

- Bestimmen von $c_{flatness}$, so dass $v_{flatness} = 0.5$ für $v^*_{flatness} = v_{flatness_{0.5}}$



tentacle – Distanz zum nächsten Hindernis

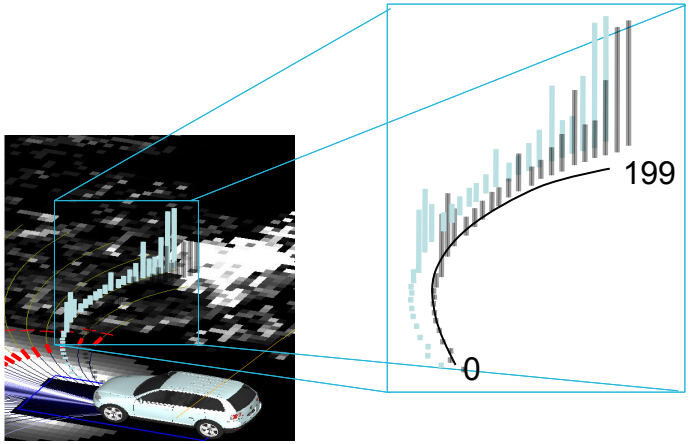
Jedes tentacle hat ein längs ausgerichtetes Histogram

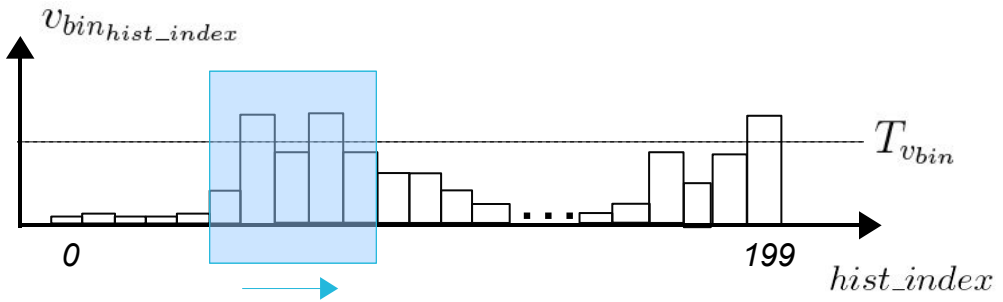
- Jedes bin des Histograms aggregiert die gewichtete Summe aller Zellen, deren senkrechte Projektion auf das tentacle in das bin fallen:

$$v_{bin_i} = \left(\sum_{\{c \| c \in S \wedge c_{hist\_index} = i\}} c_w c_{oc} \right) / \left(\sum_{\{c \| c \in S \wedge c_{hist\_index} = i\}} c_w \right)$$

Hinderniserkennung

- Ein Fenster der Größe 5 gleitet über das Histogram
- Ein Hindernis gilt als erkannt, wenn 2 der bins im Fenster eine Schwelle $T_{v_{bin}}$ überschreiten
- Die Distanz zum ersten bin im Fenster, in dem ein Hindernis erkannt wurde, ist die Distanz zum nächsten Hindernis v_{dist}
- $v_{clearance}$ durch Normalisieren von v_{dist} analog zu $v_{flatness}$





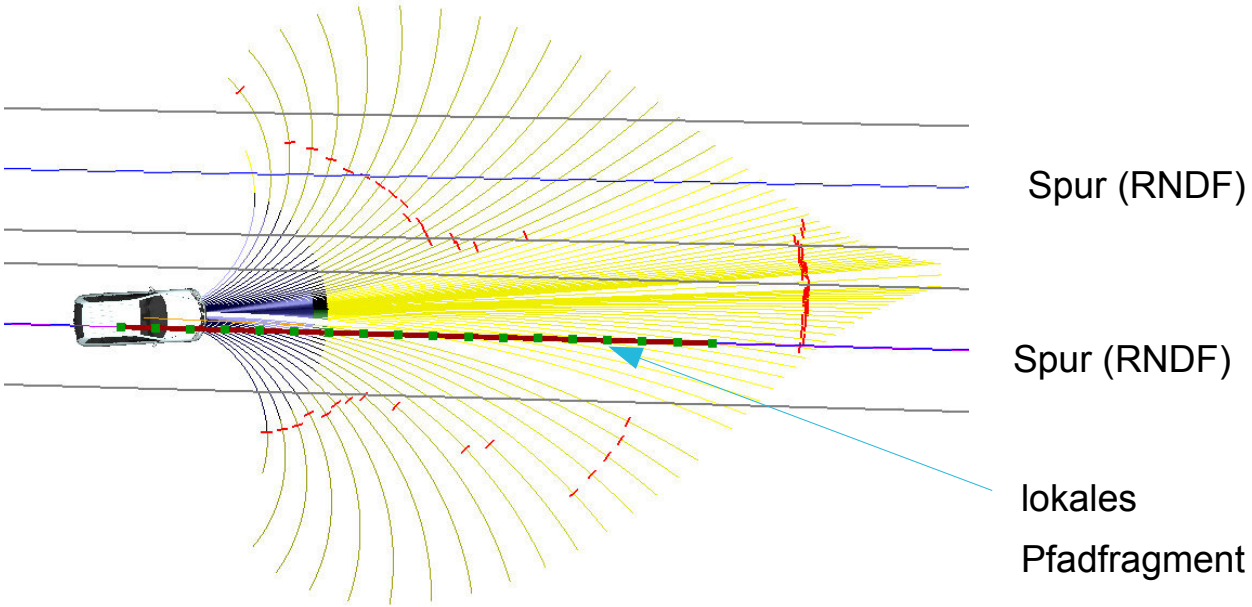
tentacle – Übereinstimmung mit dem GPS-Pfad (1/3)

Gegeben einen globalen Pfad

- Spärliche GPS Punkte (C-ElRob 2007)
- RNDF (*road network definition file*, DARPA Urban Challenge 2007)

Sampling eines lokalen Pfadfragments

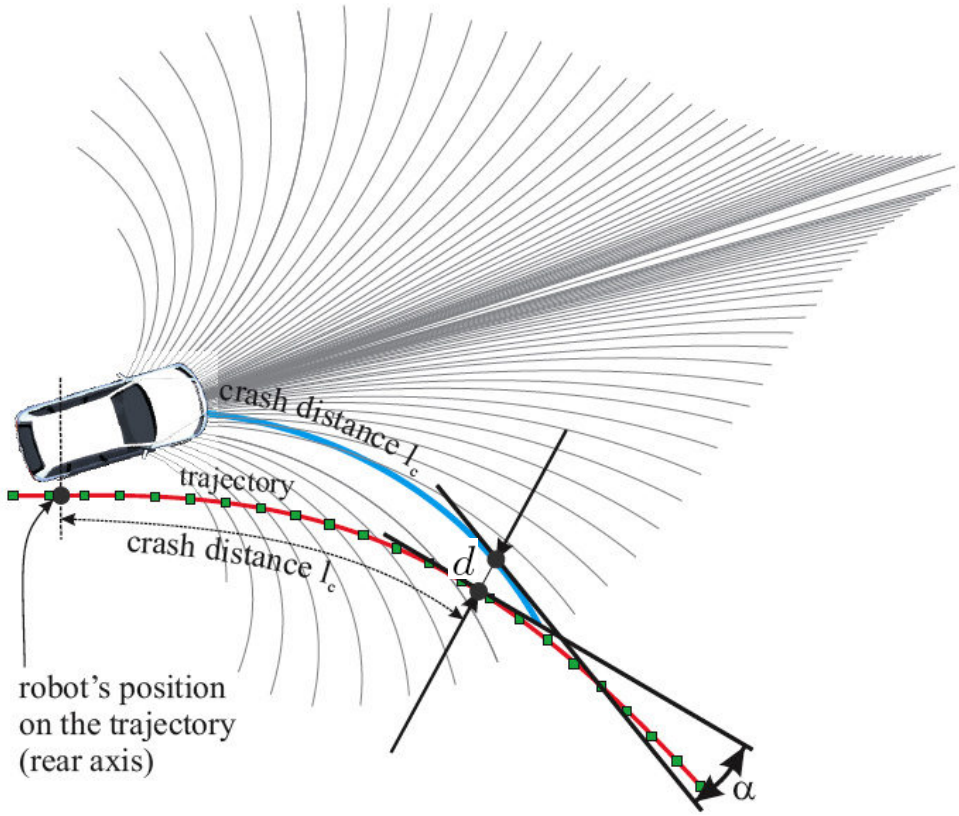
- Senkrechte Projektion des Fahrzeugs auf den globalen Pfad
- Annähern des lokalen Pfades durch wenige Punkte
- Transformation der Punkte nach Fahrzeugkoordinaten



tentacle – Übereinstimmung mit dem GPS-Pfad (2/3)

Berechnung von $v^*_{trajectory}$

- zwei korrespondierende Punkte auf *tentacle* und lokalem Pfadfragment
- Berücksichtigung von Distanz d zw. beiden Punkten und Orientierungsunterschied α der Tangenten in den Punkten



- $v^*_{trajectory} = d + c_{trajectory} \|\alpha\|$
- $c_{trajectory} \stackrel{def}{=} 4\,[m/rad]$
- l_c - Länge des Bremsweges bei aktueller Geschwindigkeit

tentacle – Übereinstimmung mit dem GPS-Pfad (3/3)

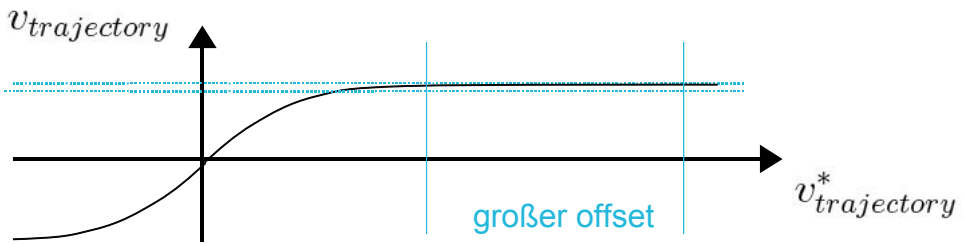
technik autonomer systeme

Problem

- Wenn das Fahrzeug einen großen offset zum GPS-Pfad hat, wird $v^*_{trajectory}$ sehr groß
- Der Einfluß der anderen Eigenschaften $v_{flatness}$ und $v_{clearance}$ ist dann sehr gering

Normalisieren von $v^*_{trajectory}$

- Sigmoid-Funktion nicht geeignet, da bei großem offset allen *tentacle* ein ähnlicher Wert zugewiesen würde und $v_{trajectory}$ dann keinen Effekt auf $v_{combined}$ hätte



- Bestimmen der minimalen und maximalen $v^*_{trajectory}$ Werte aller *tentacles*, $v^*_{trajectory_{min}}$, $v^*_{trajectory_{max}}$
- Berechnen von $v_{trajectory}$ für jedes *tentacle*

$$v_{trajectory} = \begin{cases} \frac{v^*_{trajectory} - v^*_{trajectory_{min}}}{v^*_{trajectory_{max}} - v^*_{trajectory_{min}}} & , v^*_{trajectory_{min}} \neq v^*_{trajectory_{max}} \\ 0 & , \text{otherwise} \end{cases}$$

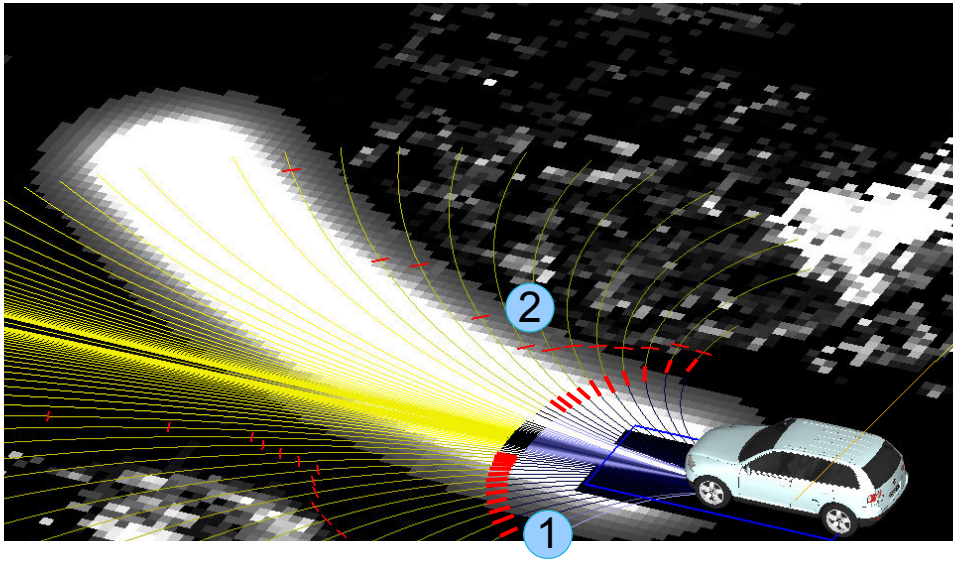
tentacle – Befahrbarkeit

Bestimmen, ob ein *tentacle* prinzipiell befahrbar ist

- Distanz zum ersten Hindernis v_{dist} bereits berechnet
- Klassifikation eines *tentacle* als nicht befahrbar, wenn das Fahrzeug innerhalb der Distanz zum ersten Hindernis nicht zu stehen kommen kann

$$v_{drivable} = \begin{cases} 1 & , v_{dist} < l_c \\ 0 & , \text{otherwise} \end{cases}$$ (1)

- Berechnung der *crash distance* l_c unter der Annahme einer komfortablen konstanten Verlangsamung des Fahrzeugs und einer Sicherheitsdistanz

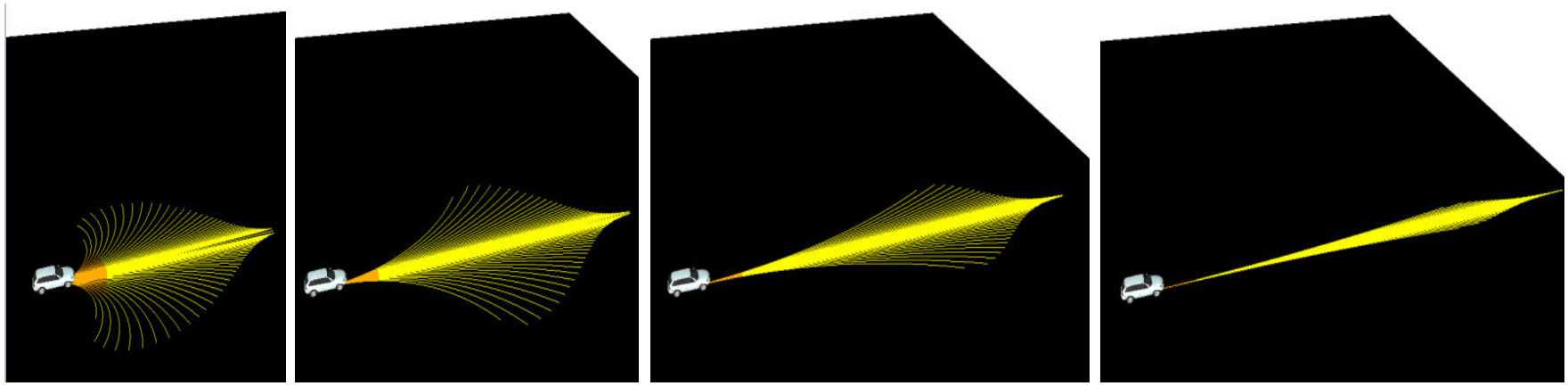


- Die roten Balken zeigen die Distanz entlang des tentacle, wo das Fahrzeug die Sicherheitsbedingungen verletzt (2)
- Berücksichtigung der Dimension und Orientierung des Fahrzeugs
- Eine Ecke des Fahrzeugs könnte ein Hindernis treffen, auch wenn der Schwerpunkt es nicht tut

tentacle speed sets

Geschwindigkeitsabhängige Vorauswahl einer Untermenge von *tentacles*

- 12 sets mit jeweils 81 *tentacles* für Geschwindigkeiten von 0 bis 10 m/s
- Mehr sets für niedrige Geschwindigkeiten
- Stärkere Krümmungen der *tentacles* bei niedrigen Geschwindigkeiten
- Maximale Krümmung der *tentacle* innerhalb eines sets beschränkt durch maximale erlaubte Zentripetalkraft
- Lateral geringere Ausdehnung der support areas der *tentalces* bei niedrigen Geschwindigkeiten erlauben das Durchfahren schmaler Passagen



Geschwindigkeit

Basis-Algorithmus Schritt 1 – Zusammenfassung

technik autonomer systeme

Basis-Algorithmus

- **1. Auswahl des besten *tentacle***
  - Auswahl eines *speed sets*
  - Berechnen der Eigenschaften aller *tentacles* im *speed set*

    $v_{flatness} \in [0; 1]$

    $v_{clearance} \in [0; 1]$

    $v_{trajectory} \in [0; 1]$
  - Klassifikation aller *tentacle* in befahrbar/nicht-befahrbar $v_{drivable} \in \{0, 1\}$
  - Berechnen von $v_{combined}$ für jedes befahrbare *tentacle*

    $v_{combined} = a_{flatness} v_{flatness} + a_{clearance} v_{clearance} + a_{trajectory} v_{trajectory}$

    $a_{flatness}, a_{clearance}, a_{trajectory} \in \mathbb{R}^0$
  - Bestimmen des besten tentacles als jenes mit minimalen $v_{combined}$
- 2. Hysteresis, zeitliches Filtern
- *3. tentacle execution*: Ausführen des ausgewählten *tentacle*

Basis-Algorithmus Schritt 2 – Hysteresis

technik autonomer systeme

Einfache Methode

- Bestimmen der Menge von *tentacles*, die „fast so gut“ sind wie das beste *tentacle*
- Schwellwert auf Differenzen

$$|v_{combined} - v_{combined_{min}}| \leq T_{v_{combined}}$$

- Wähle aus dieser Menge das *tentacle*, das dem im letzten Zyklus ausgewählten *tentacle* geometrisch am ähnlichsten ist
- Einfaches Ähnlichkeitsmaß über der Krümmung der *tentacles*

Zeitliches Filtern

- Klothoidenmodell für die Strasse
- Einspurmodell für Fahrzeugdynamik
- zusätzliche *tentacles* mit Ablage und Gierwinkel zur Strasse
- Ausgewähltes *tentacle* kann als Messung einer Krümmung, einer Ablage und eines Gierwinkels interpretiert werden
- Anwenden eines Kalman-Filter Ansatzes produziert modellbasierte, gefilterte Schätzung des auszuführenden *tentacles*

Basis-Algorithmus Schritt 3 – *tentacle execution*

Auswahl eines *tentacle* mit 10Hz

- Nur ein kleiner Teil zu Beginn der *tentacles* wird tatsächlich gefahren
- Abhängig von der Geschwindigkeit des Fahrzeugs

Experimente mit 3 verschiedenen Methoden zur Ausführung von tentacles

- **Direkte Ausführung**:

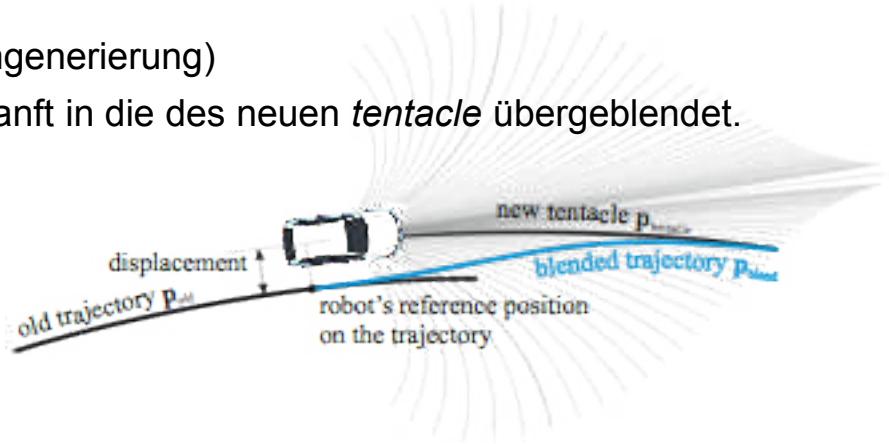
  Jedes tentacle besitzt eine Krümmung, die direkt als Lenkwinkel an das Fahrzeug kommandiert wird. Regelkreis arbeitet mit 10Hz.

- **Trajektoriengenerierung**

  Eine mit der Geometrie des *tentacle* korrespondierende Trajektorie wird in den IMU-integrierten Raum transformiert. Auf diese Trajektorie wird dann von einem Regler ausgeregelt. Regelkreis arbeitet mit 25Hz.

- **trajectory blending** (benutzt Trajektoriengenerierung)

  die Trajektorie des letzten *tentacle* wird sanft in die des neuen *tentacle* übergeblendet. Regelkreis arbeitet mit 25Hz.



C-ElRob 2007 – Offroad Navigation (1/2)

Militärisches Testgelände, Monte Ceneri, Schweiz

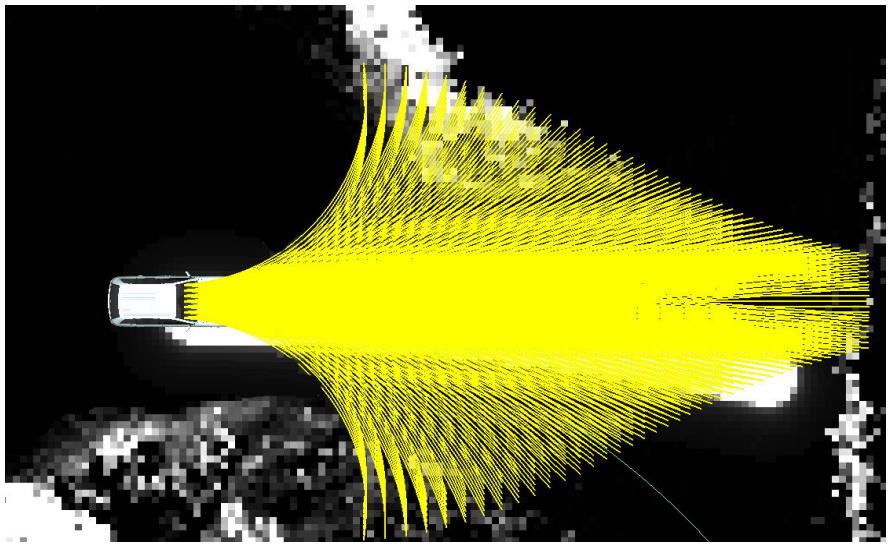
- Leistungsschau für autonome Fahrzeuge, kein Wettbewerb
- kombinierter urbaner/nicht-urbaner Kurs
- wenige GPS-Punkte gegeben

Explorationsmodus

- $a_{flatness} = 1, a_{clearance} = a_{trajectory} = 0$
- Bevorzugung ebenen Untergrunds
- GPS-Punkte ignoriert
- manuelle Intervention an Kreuzungen

Tentacles

- *tentacles* mit Ablage
- Hysteresis durch zeitliches Filtern
- Ausführen des gefilterten *tentacles* durch Trajektoriengenerierung



Abfahren des Kurses in Rekordzeit

- 31 min (nächster Teilnehmer 85min)

Zusammenfassung

Einfacher, zuverlässiger Ansatz für autonome Navigation

- Funktioniert ohne (C-ElRob 07) und mit GPS-Information (DARPA Urban Challenge 07)
- Funktioniert in urbaner und nicht-urbaner Umgebung, in Wohngebieten und Gebirgsgegenden mit Serpentinen
- Soll Wahrnehmung von Fahrspuren oder anderen Objekten nicht ersetzen, sondern als *fallback* dienen
- Nützlich als unterste Ebene eines hierarchischen Systems oder wenn keine zur Navigation geeignete Objekte wie Strassen oder Spuren erkannt werden

Geplante Erweiterungen

- Lernen der tentacles von einem menschlichen Fahrer
- *tentacles* für Rückwärtsfahren
- Auswahl situationsabhängiger tentacle sets