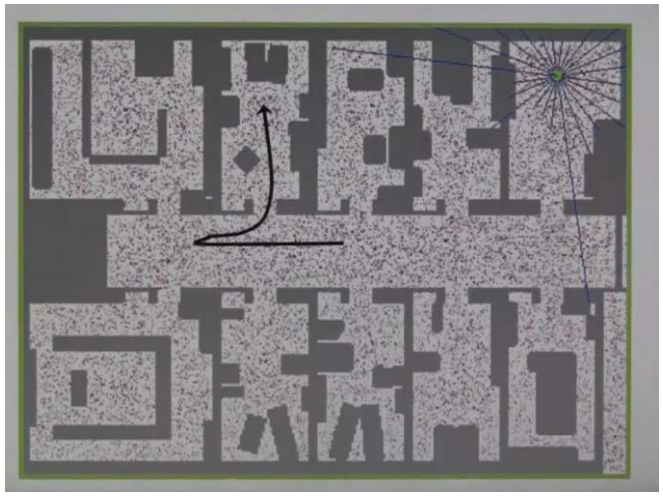
Navigation mobiler Systeme

Karten und Pfadplanung

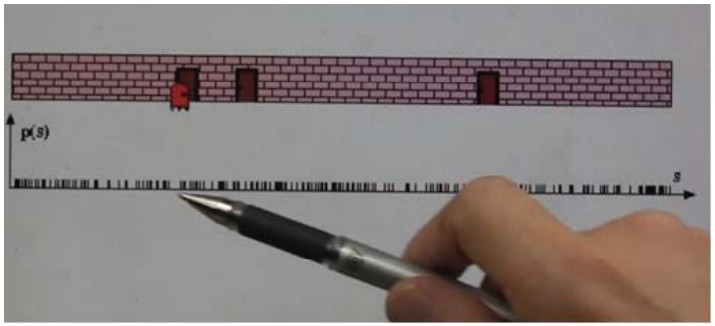
Dipl.-Inform. Ingo Boersch
In LV „Autonome Mobile Systeme“

Aus aktuellem Anlass: AI Class Oct-Dec 2011, Stanford, Sebastian Thrun + Peter Norvig, www.ai-class.com, Unit 11: HMMs and Filters -> Videos bei Youtube

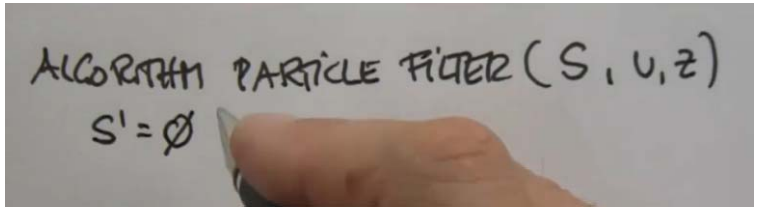
24. Particle Filters 3:46



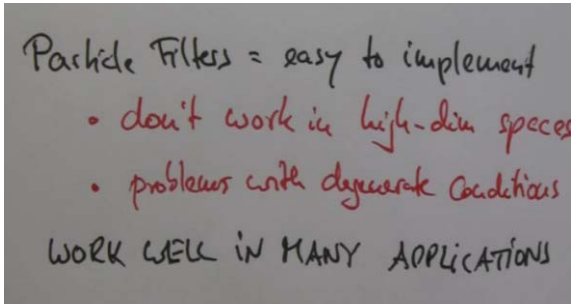
25. Localization and Particle Filters 4:27



26. Particle Filter Algorithm 2:57



27. Particle Filters Pros and Cons 1:38



Der Plan

ArLocalizationTask

Karten

- Arten von Karten
- Karten in ARIA, Kartenobjekte, Anfahren mit Actions

Lokale und globale Navigation

- Globale: Pfadplanung (Roadmap, Cell decomposition, Potenzialfeld, Value iteration)
- Lokale Navigation (die Fahrt): Dynamic Window Approach

Navigationsphänomene bei Tieren

>>Übung

Passive und aktive Lokalisierung

Kontrolliert die Lokalisierung die Roboterbewegung?

nein

Passive Lokalisierung (ARNL)

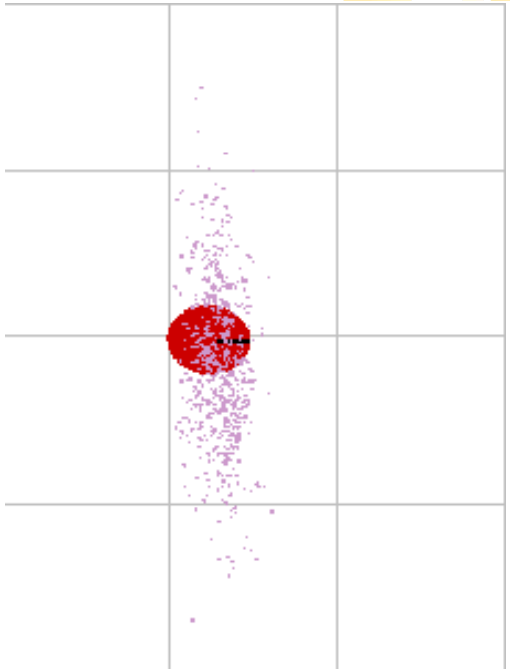
- Lokalisierungsmodul beobachtet nur die Bewegung
- Bewegung ist nicht auf beste Lokalisierung gerichtet (Wandern, Ziel anfahren)
- Bsp.: **ArLocalizationTask**

ja

Aktive Lokalisierung

- Lokalisierungsmodul plant und steuert (mit)
- Ziel: Lokalisierungfehler verringern
- Bsp: Küstennavigation

ArLocalizationTask



- Klasse aus ARNL
- Übernimmt die komplette passive Lokalisierung mittels Monte Carlo Localization, d.h. er kontrolliert die Partikelwolke
  Was ist ein Partikel?
- Wahlweise mit Sonar oder Laser
- Gibt den Laser Loclization Score aus –
  Was war das doch gleich?
- Schreibt die aktuelle Position selbständig in das Objekt der Klasse ArRobot –
  Wie bestimmt man aus der Partikelwolke die Position?
- Meldet sich, wenn es Probleme mit der Lokalisierung gibt –
  Woran erkennt man die Probleme?

ArLocalizationTask

- Startet separaten Thread
- benötigt ArRobot, ArSick, ArMap

Desktop→
Mobile Robots→
ARNL→
ARIA API Reference Documentation

Beispiel: arnlServer.cpp

// Konstruktor
*ArLocalizationTask (ArRobot *robot, ArRangeDevice *laser, char *mapName)*

// Lokalisierungsversuch an Home-Positionen (vorher in die Karte eintragen):
virtual bool localizeRobotAtHomeBlocking ()

// wenn nötig, Position selber setzen
void forceUpdatePose (ArPose forcePose, bool rayTrace=true)

// Laser Localization Score lesen
virtual double getLocalizationScore (void)

// Callback für ‚robot lost' registrieren
*void addFailedLocalizationCB (ArFunctor1< int > *functor)*

Übung:
Position setzen ohne Lokalisierung:
ArRobot.moveTo

Karten

Eine Karte ist ein **Modell** der Umwelt, welches die für die Navigation relevanten Aspekte der Umwelt im AMS abbildet.

=> ein Weltmodell, eine Wissensrepräsentation

Modellbegriff nach Herbert Stachowiak:

1. **Abbildung** eines Originals
2. **Verkürzung**: nicht alle Aspekte des Originals, nur relevante
3. **Pragmatismus**: Zweck (Für wen, wann, wozu)

Karten

Wozu werden Karten im Navigationsprozess genutzt?

1. **Positionierung (Lokalisation)**
   Korrespondenz der Beobachtung (Sonarbild, Landmarken, lokale Karte) mit der internen Karte
2. **Exploration (Mapping)**
   Erkunden einer unbekannten Umgebung, schrittweises Verfeinern der Karte

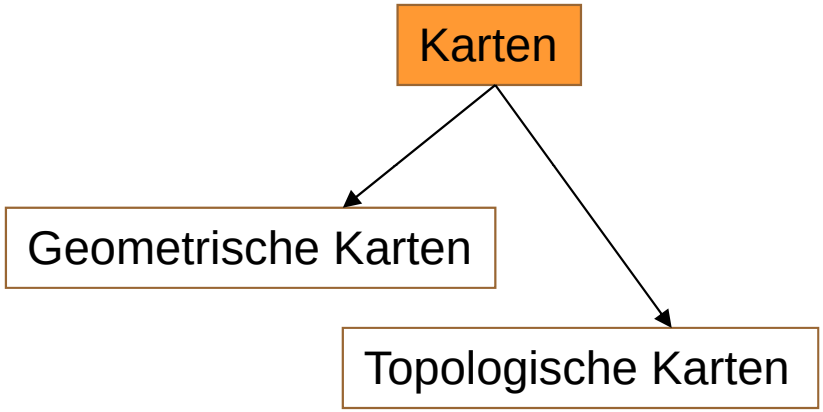
1.+2. gleichzeitig = **SLAM**
Simultaneous localization and mapping

3. **Pfadplanung**
   Finden eines begehbaren Weges

Arten von Karten

Was sind für die Navigation relevante Aspekte:

- Hindernisse und Freiräume
- Entfernungen
- Unsicherheit
- Gefährlichkeit
- Dynamik
- Windverhältnisse, Ladestationen …



2 wesentliche Arten von Karten

- geometrische Karten,
- topologische Karten

- in der Praxis Mischformen, z.B. in ARIA

Geometrische Karten

(engl.: location-based maps)

- Gittermodell mit euklidischen Koordinaten
- zusammengesetzt aus attributierten Kachelelementen
- analog dem Pixelbild in der Bildverarbeitung und damit intuitiv verständlich

Beispiel 1: **Belegungsgitter**
(engl. **occupancy grid**)

- Kachelwert zeigt Hindernis
- Diskreter Kachelwert:
  {Belegt, Frei, Unbekannt}

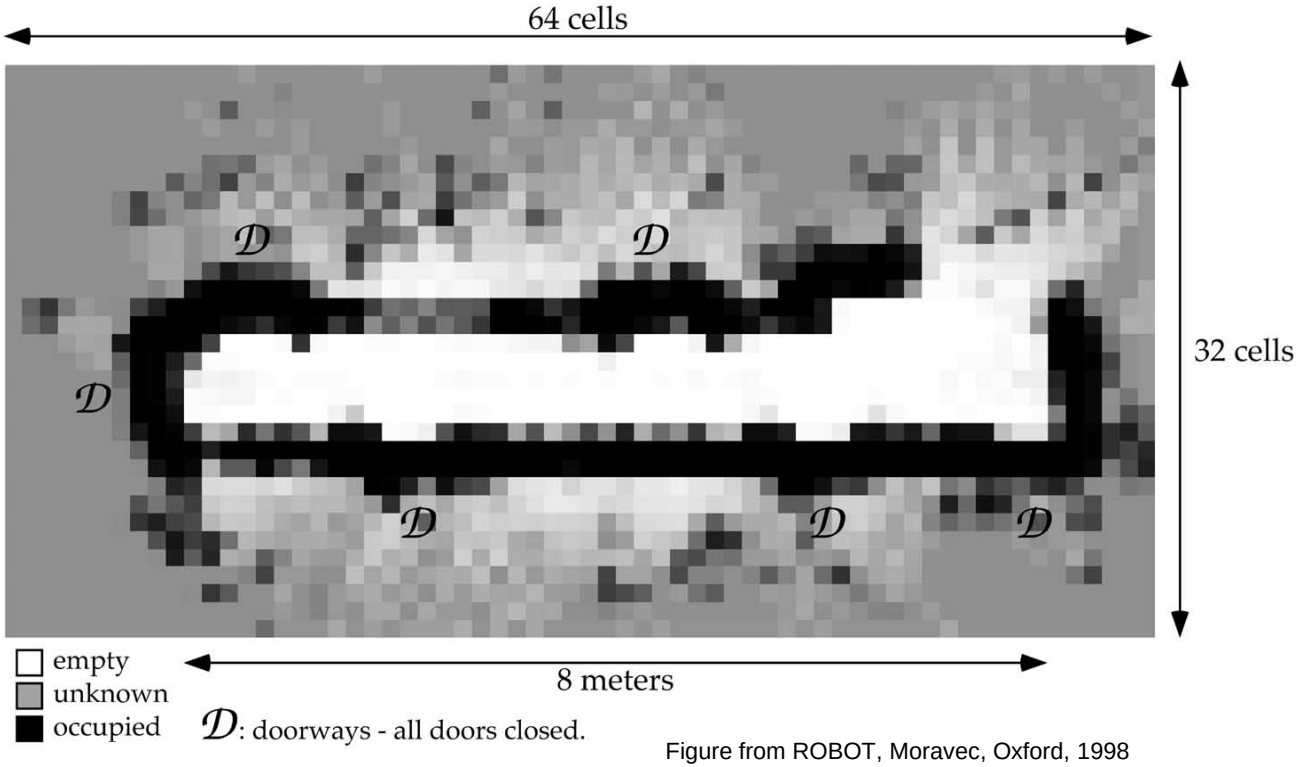
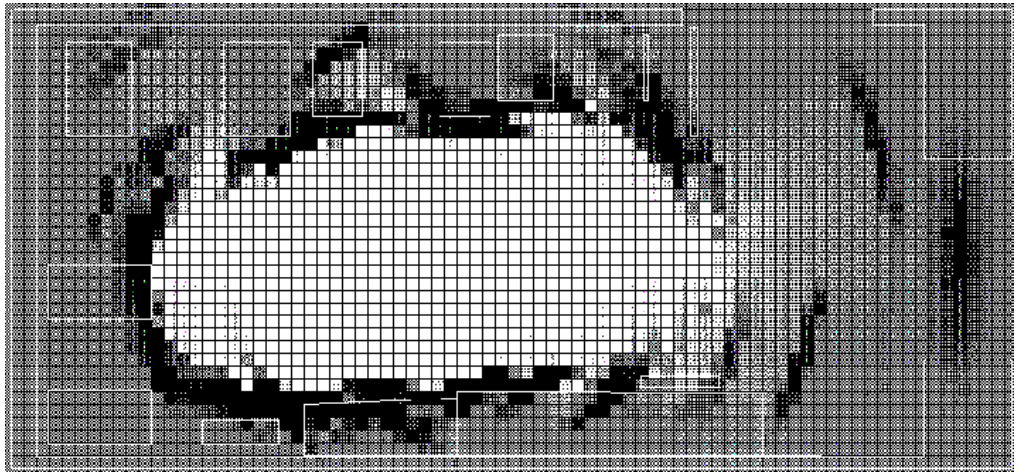


Figure from ROBOT, Moravec, Oxford, 1998

Geometrische Karten

- **Unsicherheit** abbilden: Kachelwerte aus dem Intervall [0,1]:
- Interpretationsvarianten:
  - z.B. **evidence grids** (Moravec, 1985, Navigation eines AMR mit Sonar)
    Gewissheit, dass sich in der Kachel ein Objekt befindet
  - **oder Wahrscheinlichkeit**, dass die gesamte Kachel frei ist
  - **oder Anteil** der besetzten Fläche in der Kachel ...



http://cmp.felk.cvut.cz/cmp/gallery/mobil.html

Geometrische Karten

Probleme geometrischer Karten:

- Kachelgröße
  - Kleine Kacheln: Speicherplatz und Rechenzeit, 3D!
  - Große Kacheln: keine Darstellung kleiner Objekte, kleiner Freiräume
  - Eine Lösung: Quadtree (flexible Auflösung), Octtree in 3D

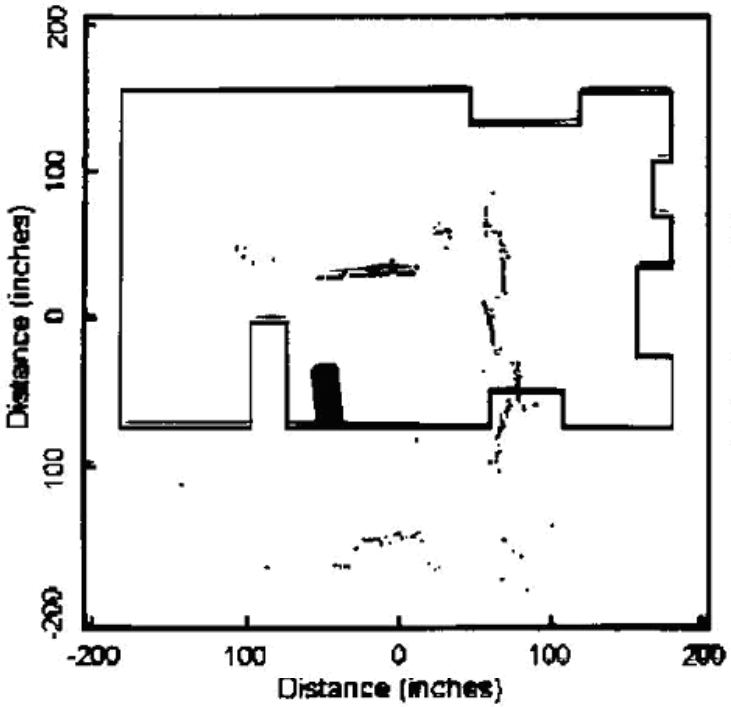
- Update von Objektpositionen (bspw. vom „Herrchen“) aufwändig

Vorteile:

- Repräsentation des freien Raumes (wird für Pfadplanung benötigt)

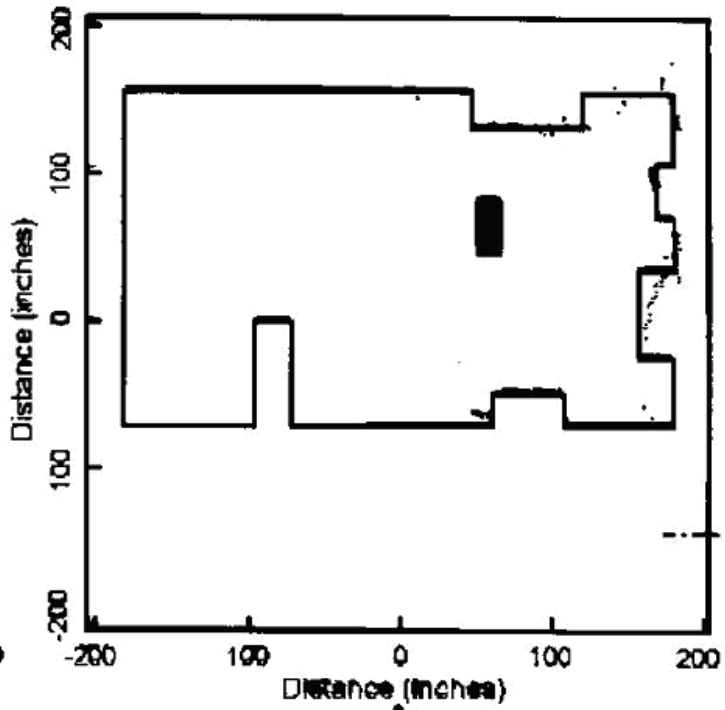
Geometrische Karten

- Lokalisation mittels Sonarscan und geometrischer Karte (Scan-Matching)



a.

Lokale Karte und geschätzte Position



b.

Gefundene Korrespondenz zwischen Scan und interner Karte

Geometrische Karte einer beleuchteten Decke
= Karte m

Nutzung für das Sensormodell
p(z|x,m) - Wie?

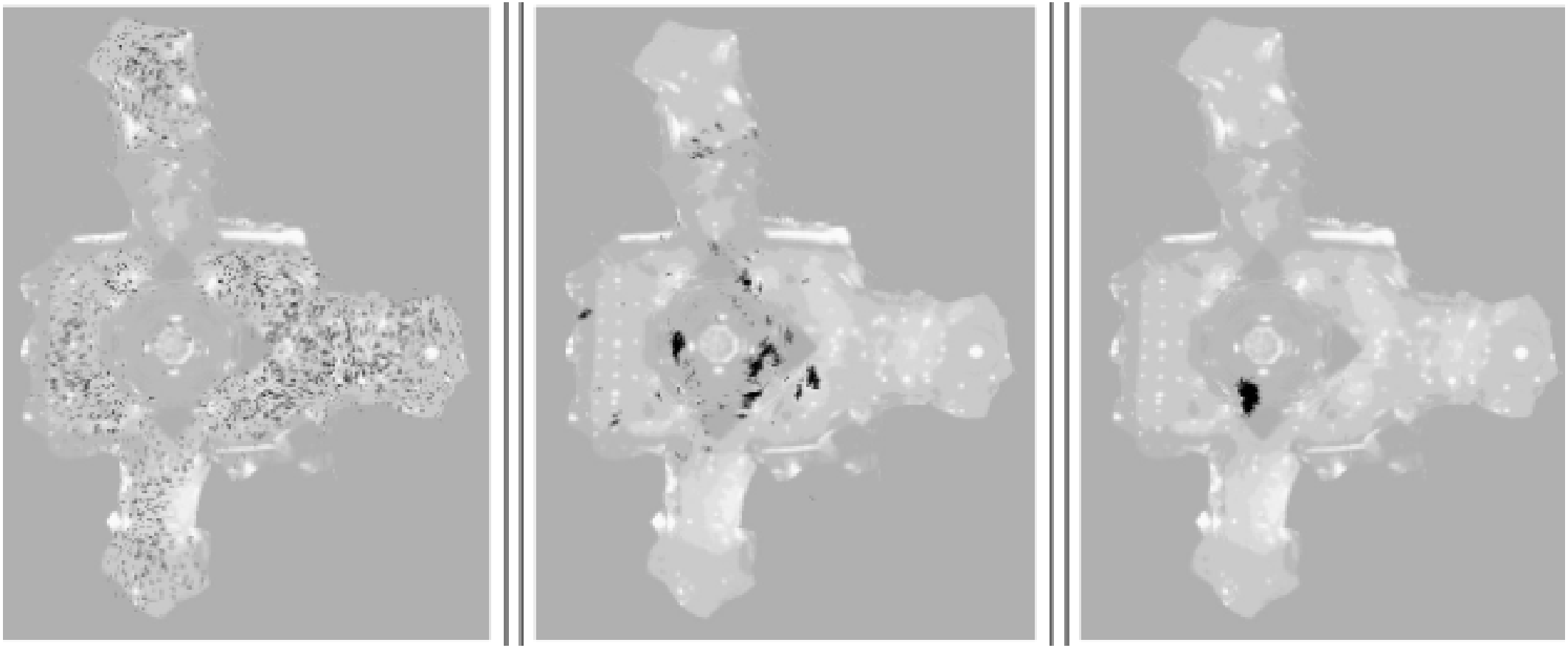
Aktuelle Beobachtung
z

An welcher Position x ist p(z|x,m) am größten?

Damit jedes Partikel wichten

[Frank Dellaert, in TBF05]

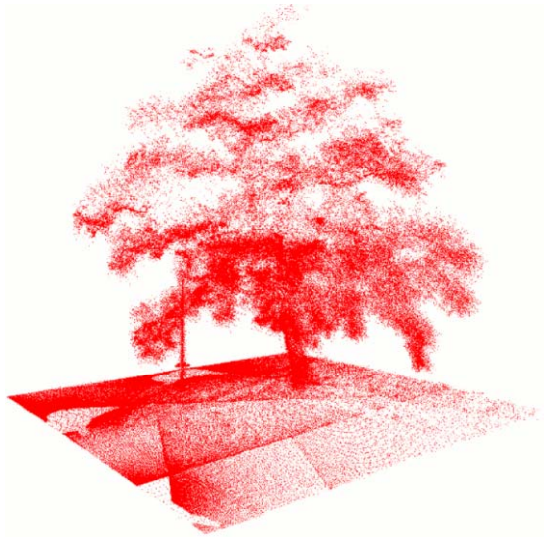
Monte Carlo Lokalisierung mit ceiling map



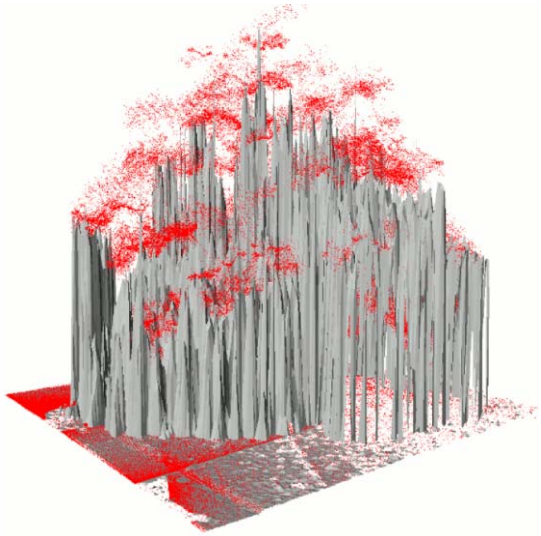
- Partikelwolke = Menge von Positionshypothesen

[ebenda]

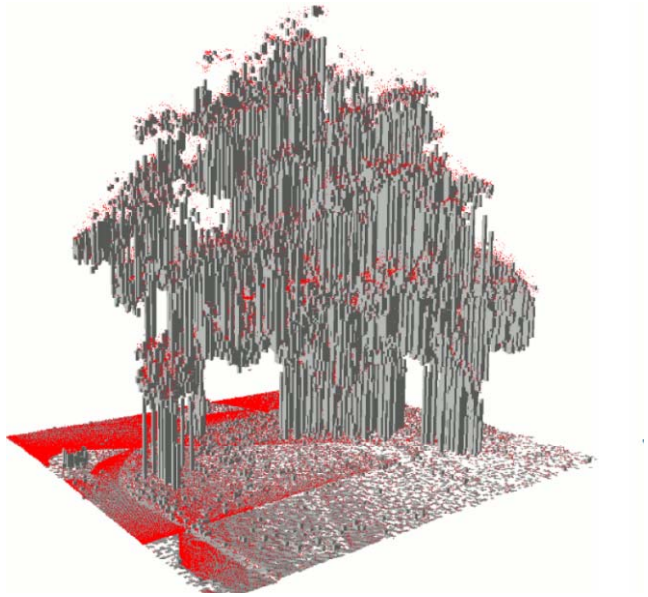
Geometrische Karten in 3D



- **Punktwolke**



- **Elevation Map**: eine Höhe pro 2D-Kachel

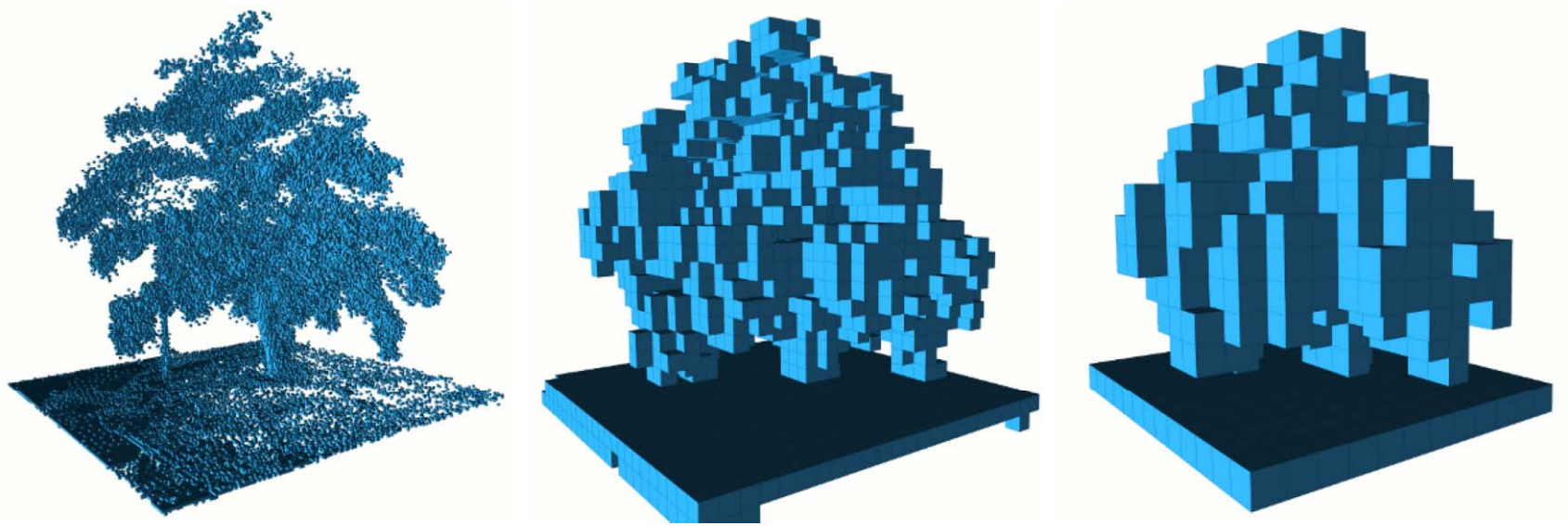


- **Multi-level surface map**
- 2.5D, wenige Werte pro 2D-Kachel

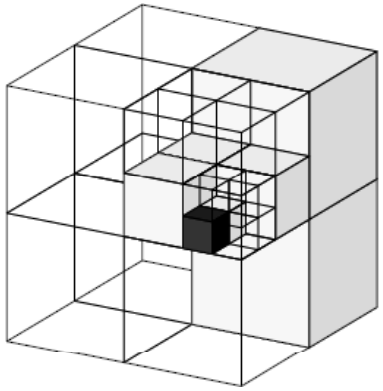
Hornung, A.; Wurm, K. M.; Bennewitz, M.; Stachniss, C. & Burgard, W.: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Auton. Robots, Kluwer Academic Publishers,* **2013**, *34*, 189-206

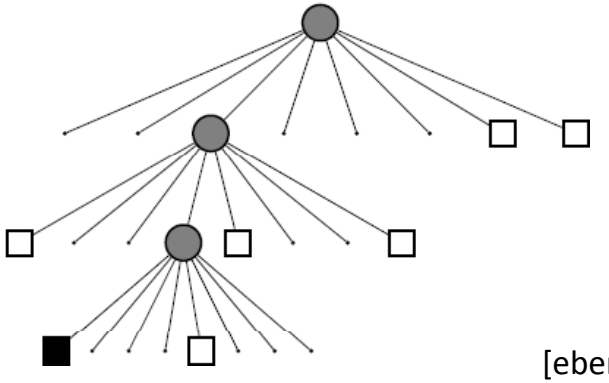
Geometrische Karten in 3D

- Volumetrisches Modell (occupancy grid in 3D)



- Beispiel OctoMap – ein **Octtree:**
- weiß = frei, schwarz = belegt
- links das volumetrische Modell, rechts der zugehörige Oct-Tree (immer 8 Kindknoten)





[ebenda]

Topologische Karten

Können Sie die geometrische Position der Mensa angeben?

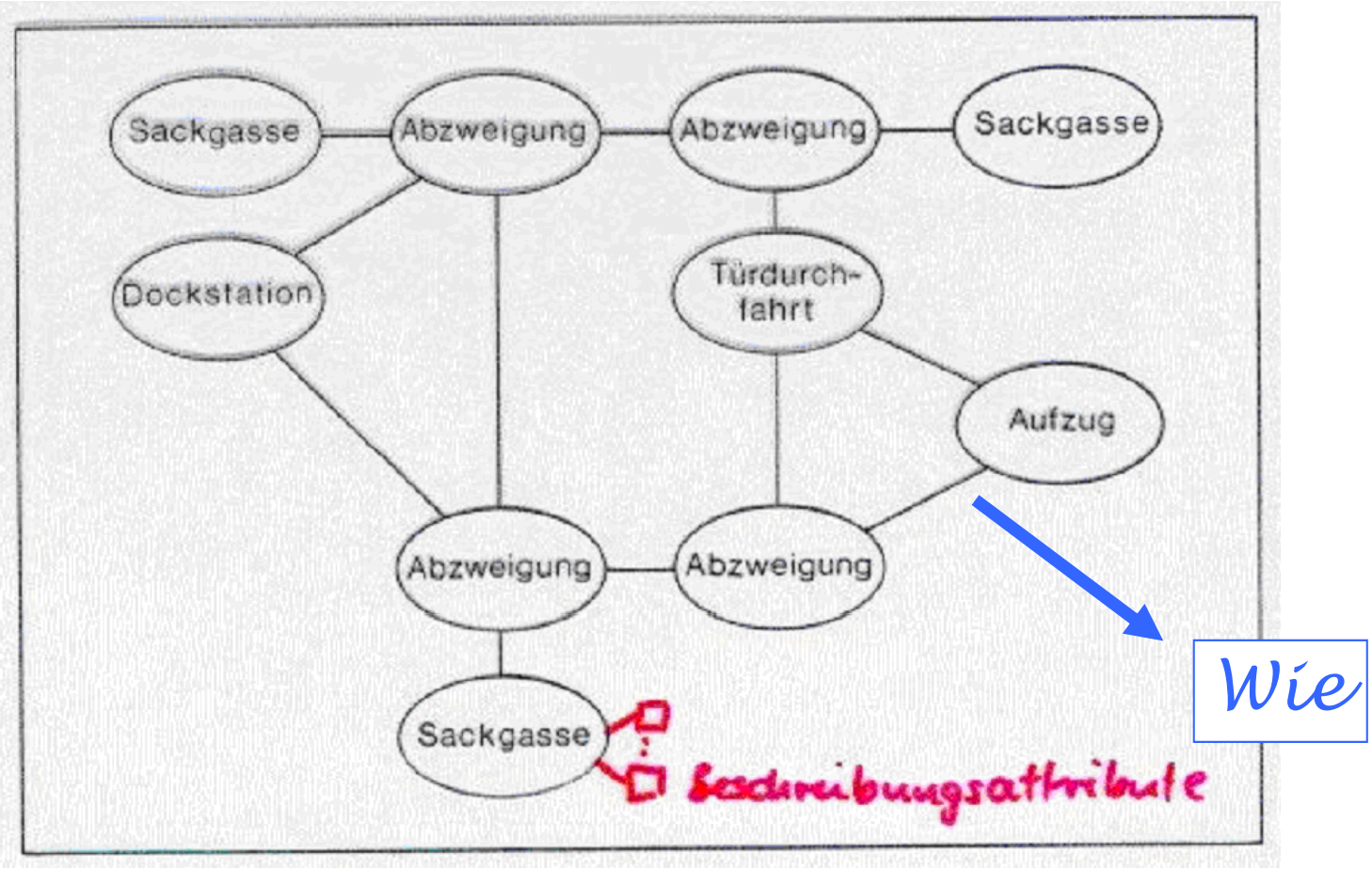
Wie kommen Sie dahin?

Topologische Karten sind Graphen

- **Knoten**: Orte mit entscheidungsbestimmenden Situationen (Beschreibung der Orte)
- **Kanten**: relative Positionen, Wege, Aktionen zwischen diesen Orten, Beziehungen
  => Bsp (Randow, 1998)

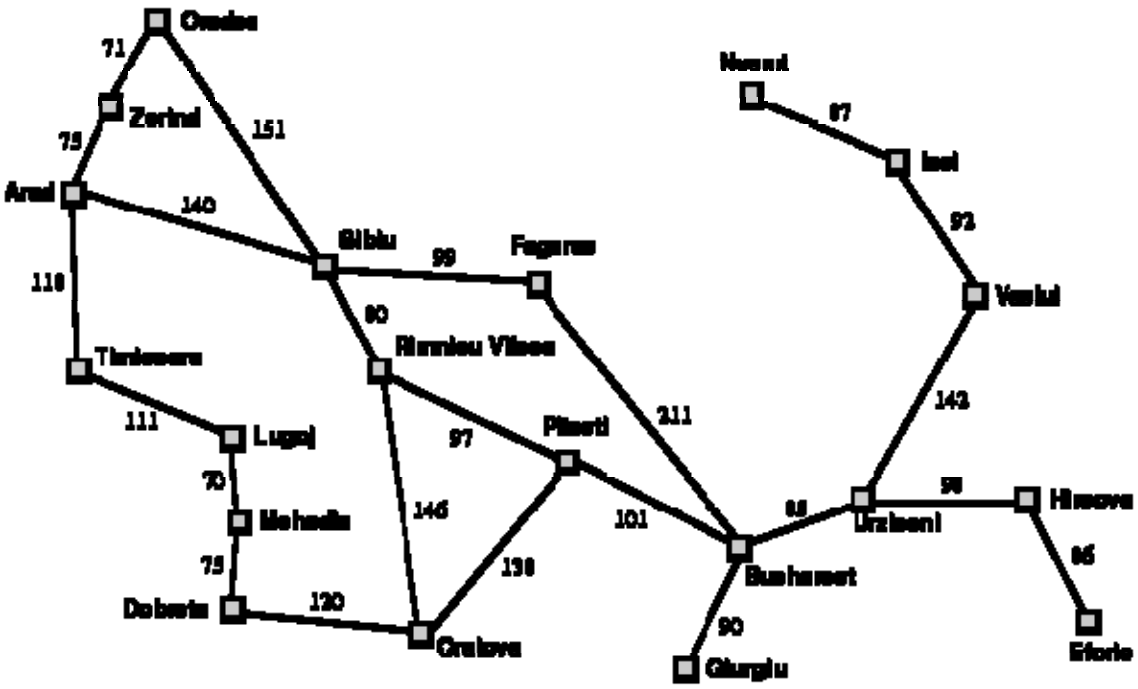
- Der menschlichen Vorgehensweise ähnlicher als geometrische Karten
- Höherer Abstraktionsgrad
- Speicherplatzsparend
- Anwendung in der *globalen* Navigation

Beispiel topologische Karte



- Beispiel topologische Karte aus „Roboter“

Eine topologische Karte: Straßenkarte von Rumänien



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- Suchen Sie mit A* einen optimalen Weg von Arad nach Bucharest!
- g(k) = Kosten des Weges vom Start bis zum Knoten k
- h(k) = Luftlinienentfernung nach Bukarest
- zulässige heuristische Funktion ?
- => ja, denn "Luftlinienheuristik„!

Topologische Karte + A*
=> Wegplanung

Gero von Randow

*„**Nichts geht über eine gute Karte**.*

*Gut ist sie, wenn sie das enthält, **was wir zur Navigation brauchen – nicht zu viel und nicht zu wenig**.*

Randow, Roboter, 1998

Karten in ARIA

- Geometrische Karte mit topologischen Elementen
- Verwendet als
  A) als **Karte** in ARIA (ARNL, SONARNL, Actions …) und
  B) als **Welt** im Simulator

- ASCII-File im MapFile-Format mit Endung .map
- Manuell bearbeiten oder mit *Mapper3*

cairns

- **Aufbau des Files:**
  1. Metadaten (Eigenschaften der Karte, **Kartenobjekte**, neue Klassen)
  2. Daten zum Belegungsgitter (occupancy map)

- **Daten =** Wahrnehmbare Objekte, die ‚echten' Objekte
  - **Punkte** (für Lokalisierung mit dem Laserscanner)
  - **Linien** (für Lokalisierung mit den Sonarsensoren)
  - Werden im Simulator zu: Hindernissen

MapFile-Format

1 2

ARMAP = ("2D-Map" NEWLINE) (**MetadataSection**) (**DataSection**)*

1

MetadataSection = MetadataLine*
MetadataLine = **MDKey** ":„ (WS MDVal)* NEWLINE
MDKey = StringID
MDVal = Integer / Float / StringID / KeyValPair

2

DataSection = ("DATA" NEWLINE PointLine*) / ("LINES" NEWLINE LineLine*)
PointLine = XPos WS YPos NEWLINE
LineLine = XPos WS YPos WS XPos WS YPos NEWLINE

- MDKey kann sein
  - Resolution, MinPos, MaxPos, NumPoints , LineMinPos, LineMaxPos, NumLines
  - Cairn: Kartenobjekte mit Bedeutung, bspw. Goal, Dock
  - MapInfo: Deklaration neuer Klassen für Kartenobjekte

* Wiederholung 0..n: im Original MapFileFormat.html in Augmented BNF, also vor dem zu wiederholenden Ausdruck

Kartenobjekte, Cairns


Cairn at Garvera, Surselva, Graubuenden, Switzerland.

- cairn (engl.) = Hügelgrab, Steinhaufen, hier Kartenobjekte
- Vordefinierte **Cairn-Typen**, also Klassen für Kartenobjekte:

Goal	Zielpunkt
GoalWithHeading	Orientierter Zielpunkt
RobotHome	Eine mögliche Startposition
Dock	Orientierter Punkt, an dem ein Docking-Manöver beginnen kann
ForbiddenLine und ForbiddenArea	Sollten nicht überfahren werden

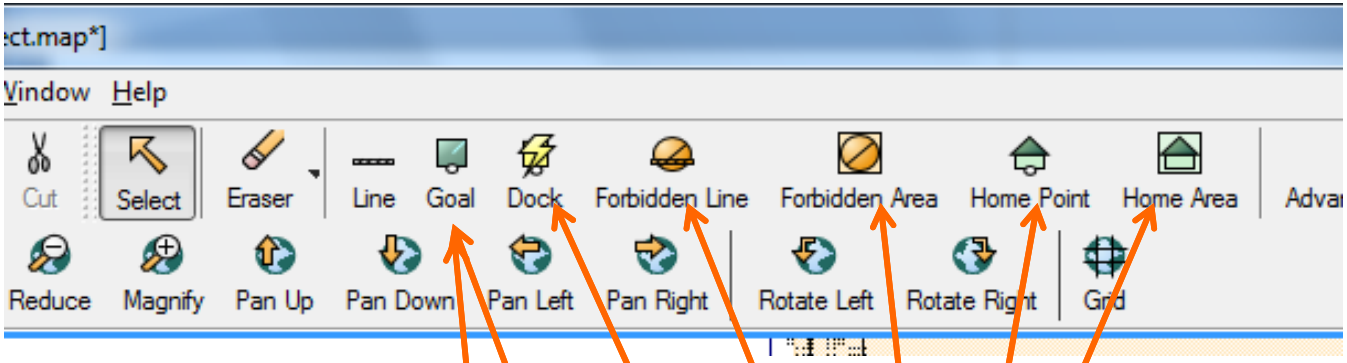
- Beispiel Kartenobjekt im MapFile definieren:

Cairn: Goal 9038 6296 0.000000 "" ICON "Tief im InfLab"

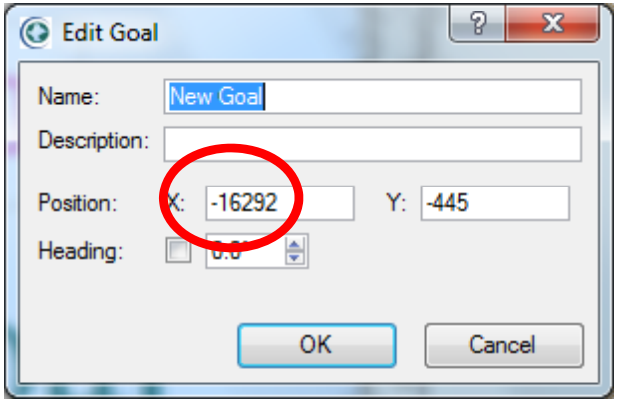
Cairn: RobotHome 9072 -1979 -179.662827 "" ICON "SweetHome„

Cairn = "Cairn:" WS CairnType WS XPos WS YPos WS Theta WS InternalName WS IconName WS Label [WS TypeSpecificData]

Kartenobjekte im Mapper



→Übung



Goal	Zielpunkt
GoalWithHeading	Orientierter Zielpunkt
RobotHome	Eine mögliche Startposition
Dock	Orientierter Punkt, an dem ein Docking-Manöver beginnen kann
ForbiddenLine und ForbiddenArea	Sollten nicht überfahren werden

Eigene Klassen für Kartenobjekte

- Definieren mit MapInfo → Verfügbar in Mapper3 + MobileEyes …
- Klassen setzen auf Typen auf:
  GoalType, DockType, LocationType, BoundaryType, SectorType

Beispiel1: eigene Klasse von Zielen - Steckdosen

MapInfo: GoalType Name=Steckdose "Label=Steckdose" "Desc=Hier gibts Strom"
Heading=Required Shape=VBars "Color0=0xff0000"

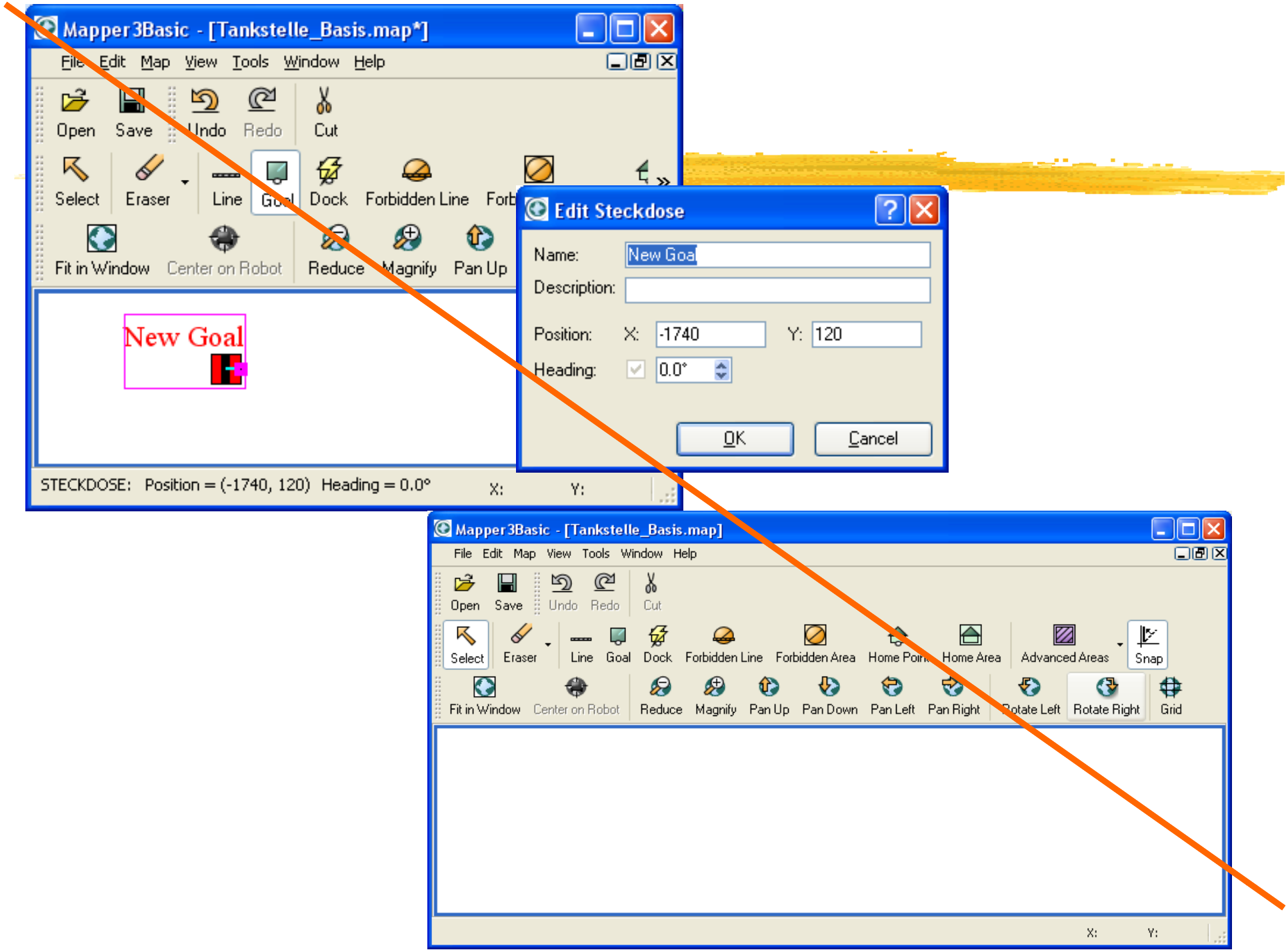
→ im Mapper3 bei Zieleingabe, Properties *)

Beispiel2: eigene Klasse von Sektoren – im Simulator bewegliche Box

MapInfo: SectorType Name=BoxObstacle Label=Sim.BoxObstacle "Desc=Herrchen vom Dogbot"
Shape=Plain "Color0=0x00FF00" "Color1=0x000000" Sim.Obstacle=yes

→ im Mapper3 neuer Toolbutton "Advanced Areas"

*) Beachten: bei Verwendung von GoalType, DockType -> StandardTypen Goal, GoalWithHeading etc. verschwinden


Mapper3Basic - [Tankstelle_Basis.map*]
File Edit Map View Tools Window Help
Open Save Undo Redo Cut
Select Eraser Line Goal Dock Forbidden Line
Fit in Window Center on Robot Reduce Magnify Pan Up
New Goal
STECKDOSE: Position = (-1740, 120) Heading = 0.0°
X: Y:
Edit Steckdose
Name: New Goal
Description:
Position: X: -1740 Y: 120
Heading: 0.0°
OK Cancel
Mapper3Basic - [Tankstelle_Basis.map]
File Edit Map View Tools Window Help
Open Save Undo Redo Cut
Select Eraser Line Goal Dock Forbidden Line Forbidden Area Home Area Advanced Areas Snap
Fit in Window Center on Robot Reduce Magnify Pan Up Pan Down Pan Left Pan Right Rotate Left Rotate Right Grid
X: Y:

ARIA-Bibliothek: Die Klasse ArMap

- Nicht reentrant, locken
- Übung

- ***ArMap()*** **Konstruktor**
  - *ArMap arMap(fileDir);* Kartenverzeichnis im Konstruktor
- ***readFile***, *writeFile* Laden einer Karte im MapFile-Format
- *getMapObjects()* modifizierbare Liste der Kartenobjekte
- ***findMapObject*** **benanntes Kartenobjekt suchen**
- *getLines(), getPoints(), getResolution()* Getter
- *setMapObjects(), setLines(), setPoints()* Setter

- Callbacks für Kartenänderung registrieren: *addMapChangedCB*
- Callbacks manuell: triggern *mapChanged()*
- **Map geladen?:** ***getFileName()***

- Kommandozeilenparamter: -map xxx.map oder im File arnl.p

Wie weit ist ein Kartenobjekt vom Roboter entfernt?

- ArMap

  *ArMapObject * findMapObject (const char *name, const char *type=NULL)*

  Gets a map object of given name and type if it exists.

  →Übung

- ArMapObject

  ArPose getPose (void) const Gets the pose of the object.

- ArRobot

  double findDistanceTo (const ArPose pose)

  Gets the distance to a point from the robot.

- Wie kann der Roboter zu einem Kartenobjekt fahren?

Anfahren von Objekten

Variante A) mit Actions: ***ArActionGoto***

This action goes to a given **ArPose** very naively.

This action naively drives straight towards a given **ArPose**. the action stops when it gets to be a certain distance (closeDist) from the goal pose. It travels at the given speed (mm/sec).

You can give it a new goal with **setGoal()**, cancel its movement with **cancelGoal()**, and see if it got there with **haveAchievedGoal()**. Once the goal is reached, this action stops requesting any action.

This doesn't avoid obstacles or anything, you could have an avoid routine at a higher priority to avoid on the way there... but for real and intelligent looking navigation you should use something like ARNL, or build on these actions.

→Übung

Examples:

gotoActionExample.cpp.

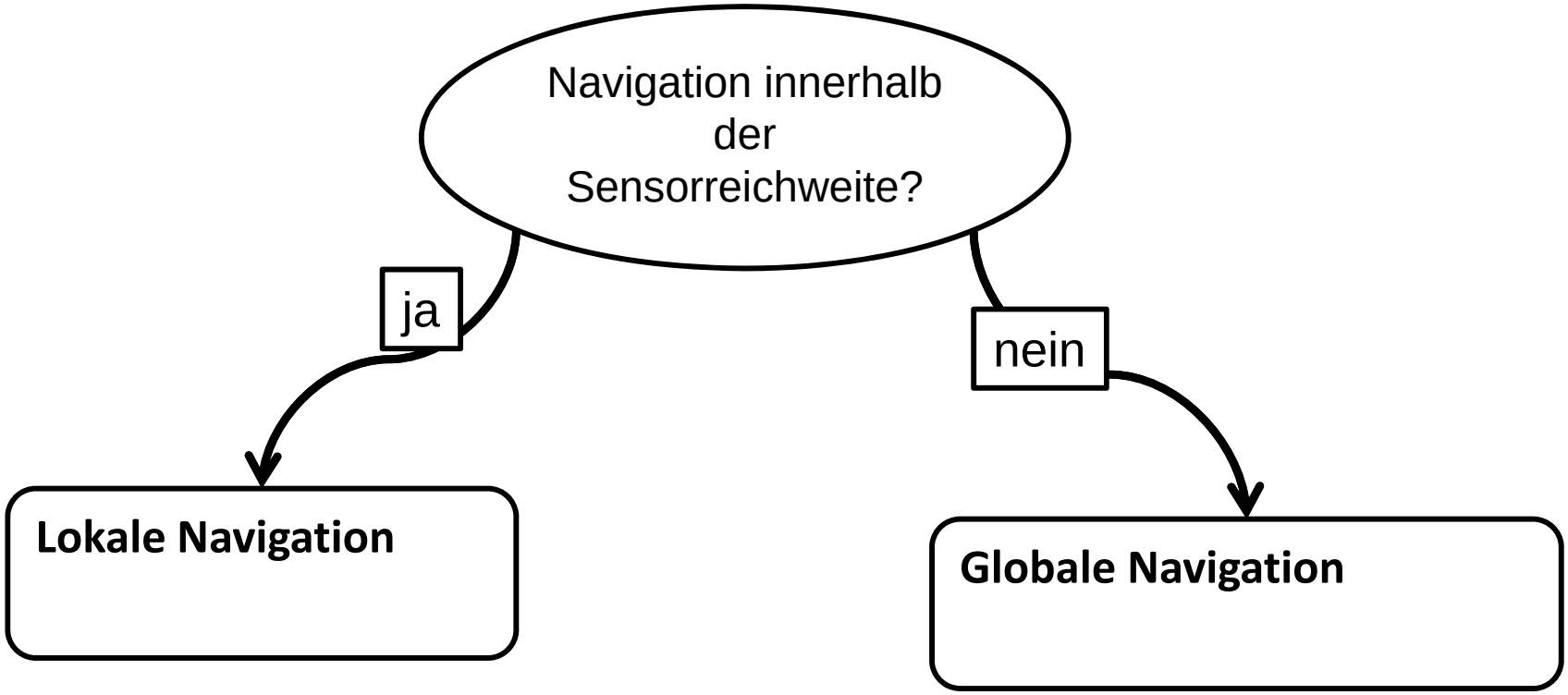
Variante B) mit Pfadplanung ***ArPathPlanningTask***

(elegante Variante, siehe arnlserver.cpp)

Wie komme ich dahin?

Pfadplanung

Lokale vs. globale Navigation



Lokale vs. globale Navigation

Lokale Navigation (Pilot)

- Navigation in der nahen Umgebung des Roboters (**Sensorreichweite**)
- z.B. Erreichen eines nahen Zielpunktes, Türdurchfahrt, Korridor-Folgen, Kollisionsvermeidung, Dogbot, Linienfolgen etc.
- Starke Sensorkopplung, Karte möglich, reaktive Ansätze

DWA heute in der LV?, siehe z.B. DA Ronny Menzel oder MA Helge Scheel

Globale Navigation (Navigator)

- Navigation im Großen, Grobplanung
- z.B. **Planung optimaler Wege**, Vermeidung von Umwegen & Sackgassen,
- Lose Sensorkopplung, Karte zwingend, deliberative Ansätze z.B. Suchalgorithmen

Warum diese Unterscheidung?

- Umwelt-Dynamik, wichtig für ...?
- Aufgabenänderung, wichtig für ...?
- Vorausplanung, wichtig für ...?
- Zeitbeschränkung, wichtig für ...?

Kombination notwendig => ergibt Steuerungsarchitektur mit Navigator, Pilot ...

- Aber diese Verzahnung ist nicht unproblematisch

Globale Navigation - Pfadplanung / Wegplanung

Suche eines (optimalen) Pfades
von der momentanen Position zum vorgegebenen Ziel.

Optimalitätskriterien:

- Kollisionsfreiheit
- Länge
- Fahrzeit
- Gesamtrotation
- Rechenzeit
- Sicherheit
- Küstennähe
- …

→ Mehrzieloptimierung

Verfahren zur Wegplanung

In geometrischen Karten

- **Roadmap**
  - Andere Namen: Wegkarten-, Straßenkarten-Verfahren
  - Arten: Visibility-Graph, Voronoi-Diagramme
- **Cell decomposition**
  - Andere Namen: Zellzerleguns-, Zellaufteilungsverfahren
  - Arten: Exakte, Approximierte
- **Potenzialfeld**
- **Value Iteration** (ARIA/ARNL)
- u.v.a

Diese Verfahren enden ebenfalls in einer Graphensuche

In topologischen Karten

- **Graphensuche**

Roadmap – Verfahren

Wegkarten-, Straßenkarten-Verfahren

Konstruktion eines Graphen aus befahrbaren Wegen im freien Raum
=> ergibt **Roadmap** (ungerichteter Graph)

damit

- Einschränkung der Komplexität
- Anwendung von Suchalgorithmen möglich, z.B. A*

Beispiele für Roadmap-Verfahren: Visibility Graph, Voronoi

Wie kann man derartige Graphen konstruieren?

Konstruktion der Roadmap

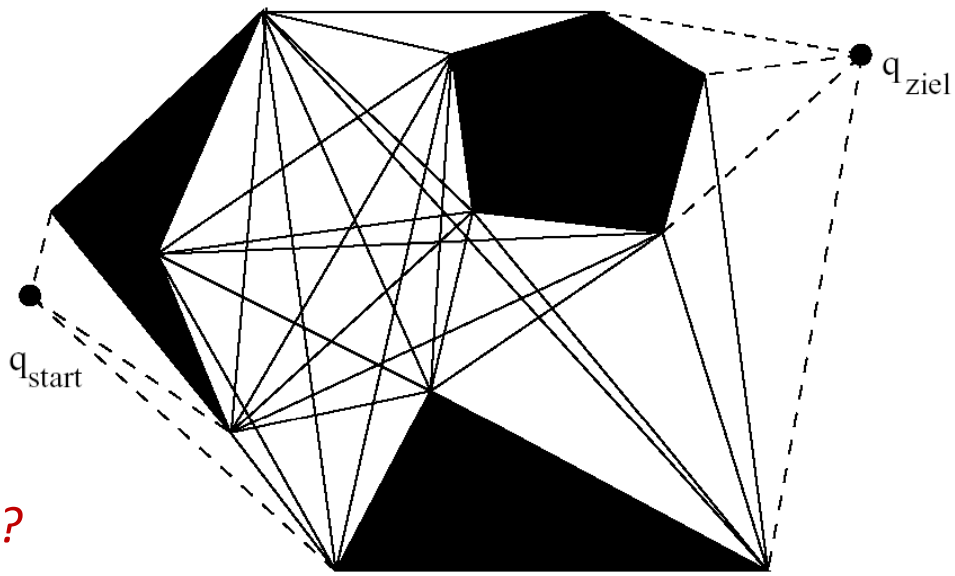
Visibility Graph

- N.J. Nilson 1969, Sichtbarkeitsgraph
- Polygone Objekte

Tangenten an Hindernissen

Idee:

- Paarweises Verbinden der Hindernisecken, des Startpunktes und Zielpunktes im freien Raum,
- zusätzlich die Objektkanten
- ergibt semifreie Pfade



- Anschließend Pfadsuche im Graph
- Kantenanzahl wächst exponentiell
- Vollständig
- *Wie finden wir den Weg in dem Graphen?*

A^*

Konstruktion der Roadmap mit Visibility Graph

Verbesserungen in Komplexität und Qualität des Weges

Komplexität

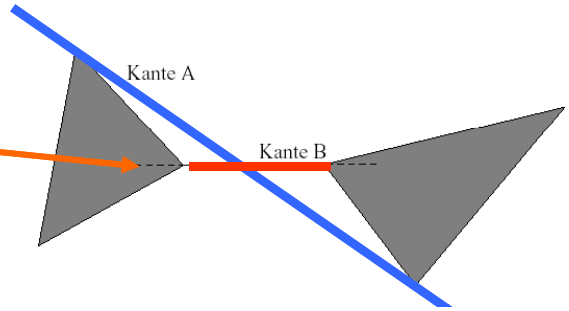
- Löschen ungünstiger Kanten (wie B)
- Konkave Ecken (wie V) weglassen

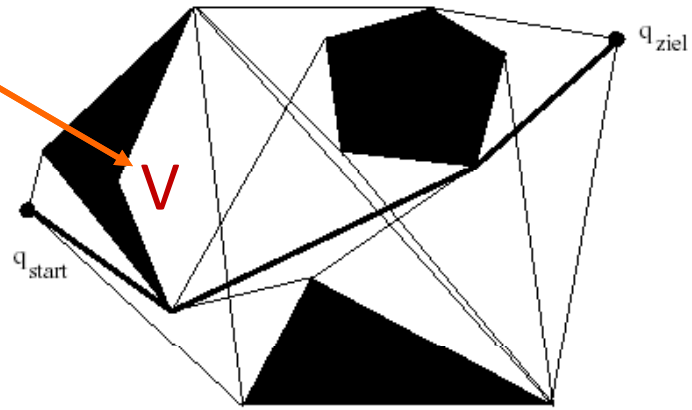
Qualität des Weges, Stochastische Umwelt

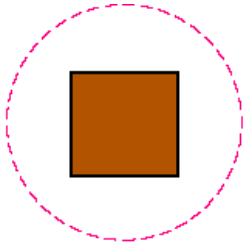
- Aufblähen der Hindernisse (obstacle growing)

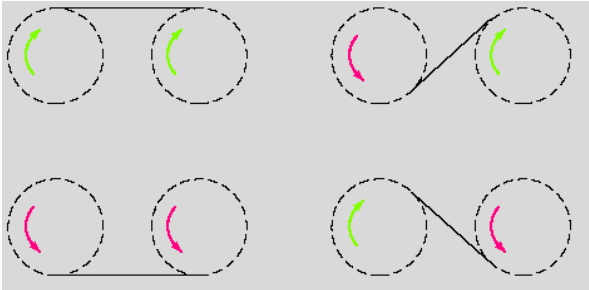
Komplexität und Qualität

- Moravec, kreisförmiges Aufblähen der Hindernisse um Ungewissheit (1979 AMR: Stanford Cart, 30m in 5 Stunden)





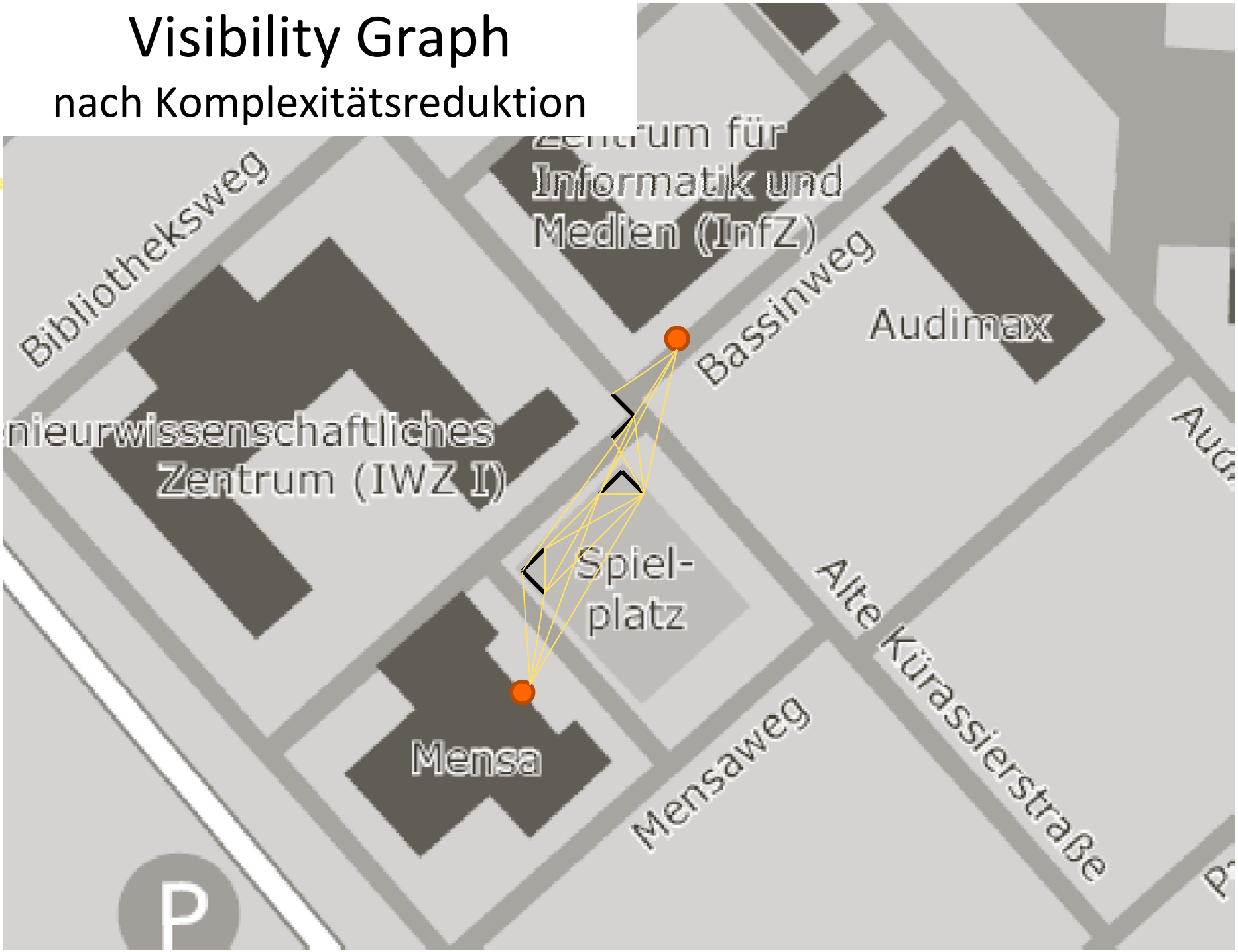




- dadurch unvollständig, eventuell Verlust des optimalen Weges

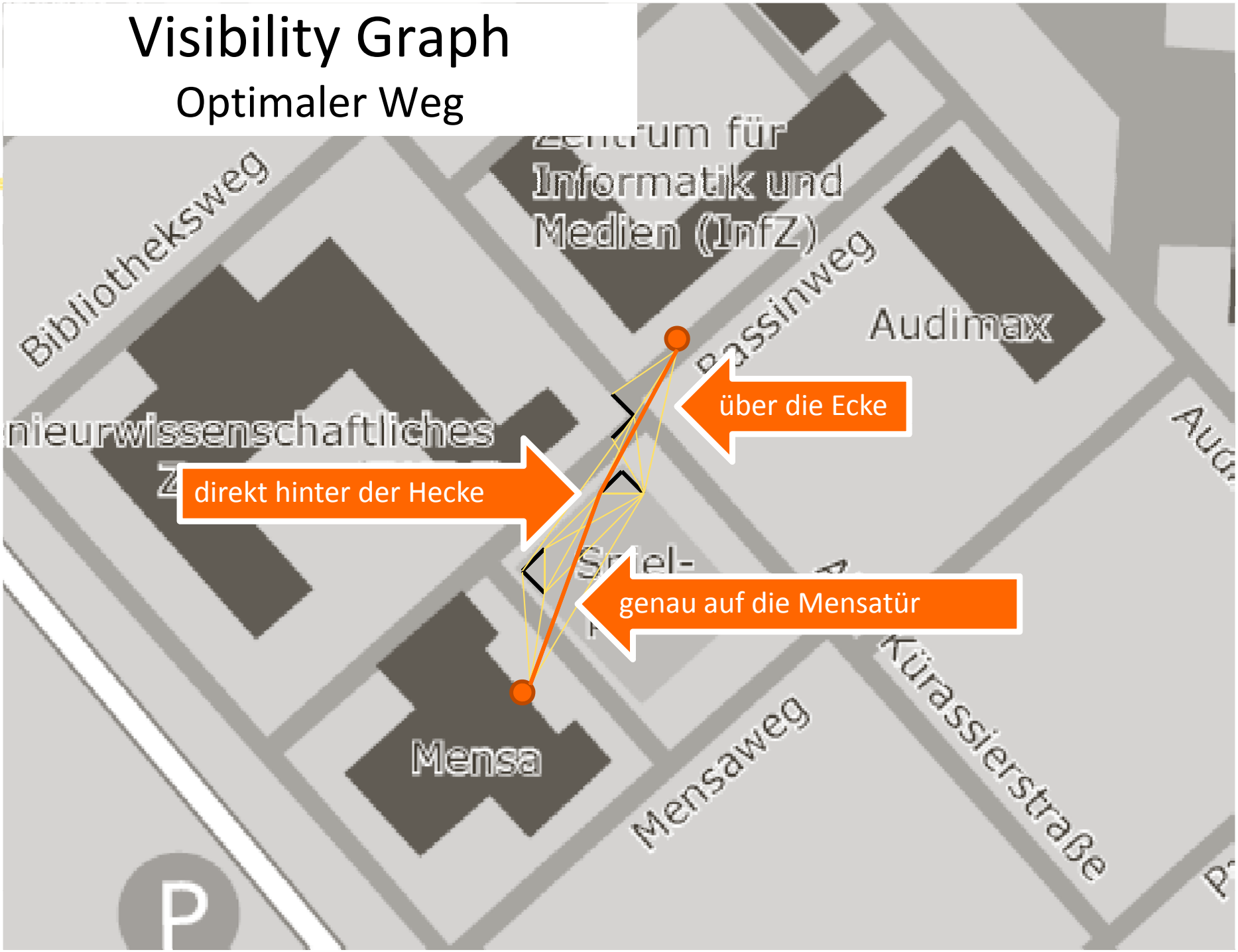
Visibility Graph

nach Komplexitätsreduktion



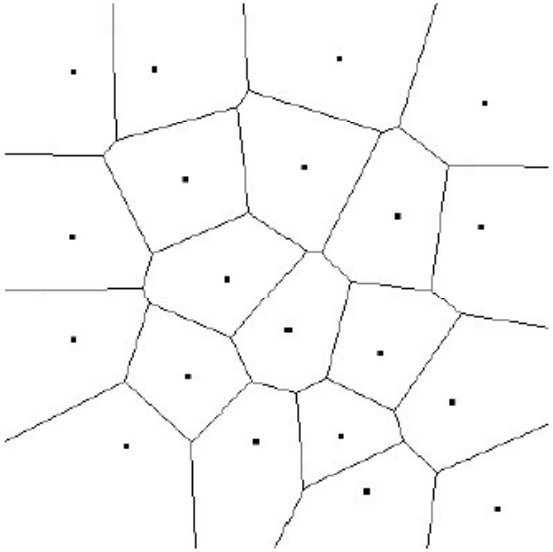
Visibility Graph

Optimaler Weg



Konstruktion der Roadmap
Voronoi-Diagramme

- M.G. Voronoi 1908
- Das Voronoi-Diagramm einer Punktmenge von n Punkten p1.. pn ist eine **Zerlegung der Grundmenge in n Regionen**. Jede Voronoi-Region VR(pi) für pi enthält genau die Punkte, die näher zu pi sind als zu irgend einem anderen Punkt pj.
- VR(pi) = { p | d(p,pi) < d(p, pj) für alle j ungleich i)



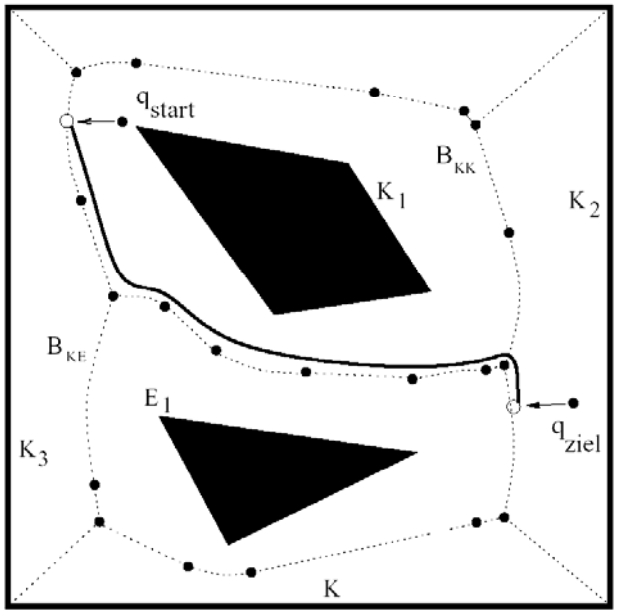
Konstruktion der Roadmap

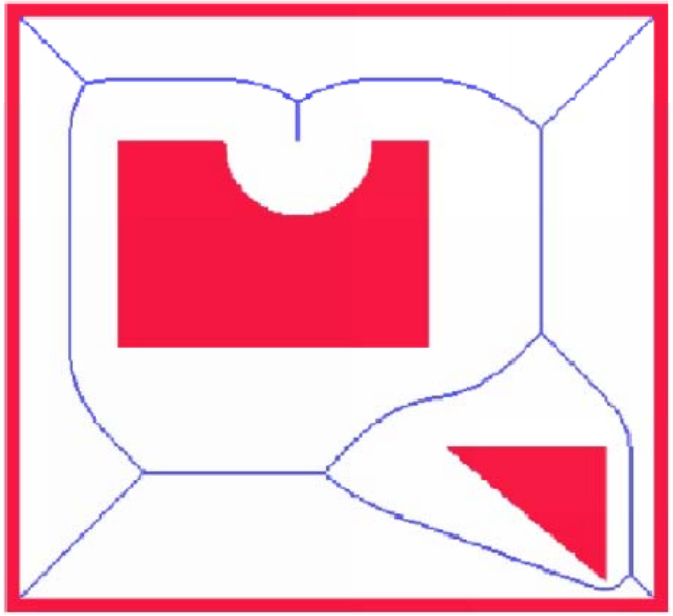
Retraction (Voronoi-Diagramme) II

- Anwendung auf Hindernisse
- Verbinden der Roadmap mit Start und Zielpunkt
- Dann wie üblich Suche im Graphen
- **Vorteil:** größtmöglicher Abstand zu Hindernissen
- vollständig

*schon wieder A**

u.U. lange Wege





Verfahren zur Wegplanung

In geometrischen Karten

- **Roadmap**
  - Andere Namen: Wegkarten-, Straßenkarten-Verfahren
  - Arten: Visibility-Graph, Voronoi-Diagramme

- **Cell decomposition**
  - Andere Namen: regionsorientierte, Zellzerleguns-, Zellaufteilungsverfahren
  - Arten: Exakte, Approximierte

- **Potenzialfeld**
- **Value Iteration** (ARIA/ARNL)
- u.v.a

In topologischen Karten

- **Graphensuche**

Cell decomposition

Vorgehensweise:

1. Zerlegung des Freiraumes in disjunkte konvexe Regionen (Zellen)
2. Interpretation der Regionen als Knoten, Regionenberührung als Kanten

Knoten, Kanten – ein Graph

*schon wieder A**

3. Graphsuche ergibt sogenannten Kanal – Liste von Regionen
4. Festlegung der Durchfahrtsroute im Kanal (Schwerpunkte der Zellen oder Mittelpunkte der Kanten)

Arten:

- exakte,
- approximierte

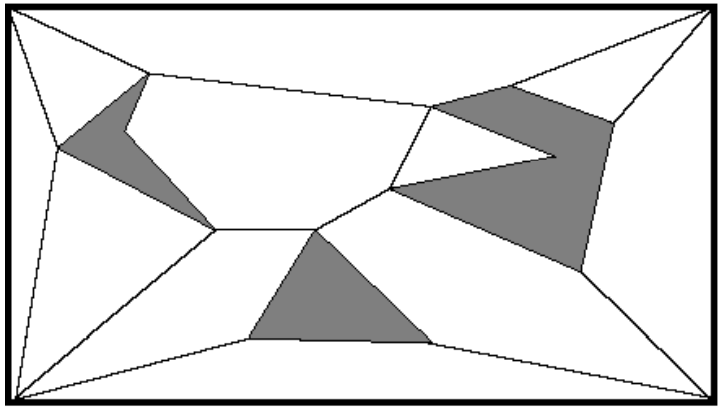
Wie wird der Freiraum zerlegt

Cell decomposition

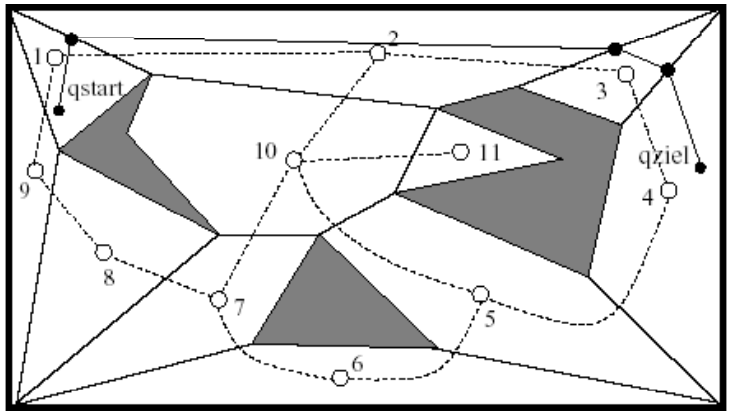
Exakte Zerlegung

- **Zerlegung ist vollständig**

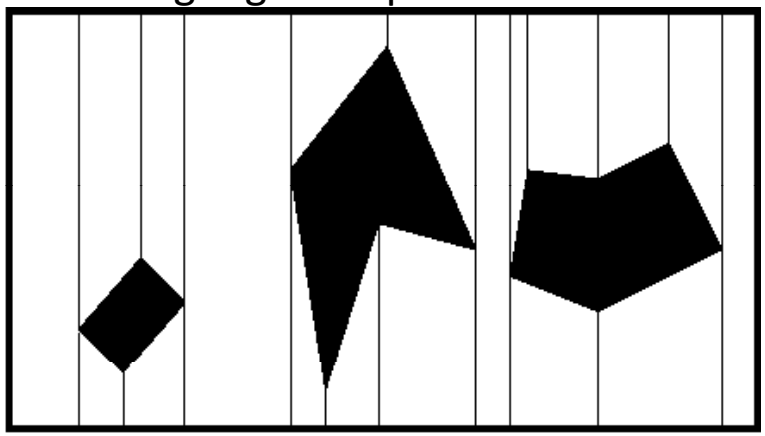
Zerlegung in Polygone

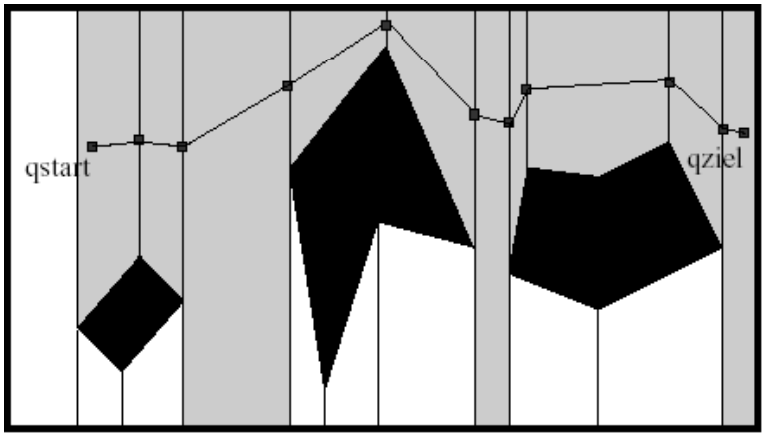


Kanal: Zellen 1,2,3,4



Zerlegung in Trapeze



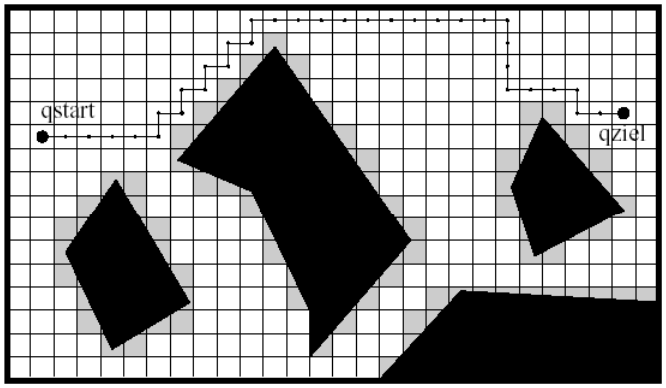


Cell decomposition

Approximierte Zerlegung

Einteilung des Freiraumes in Standardformen (Elementarzellen)

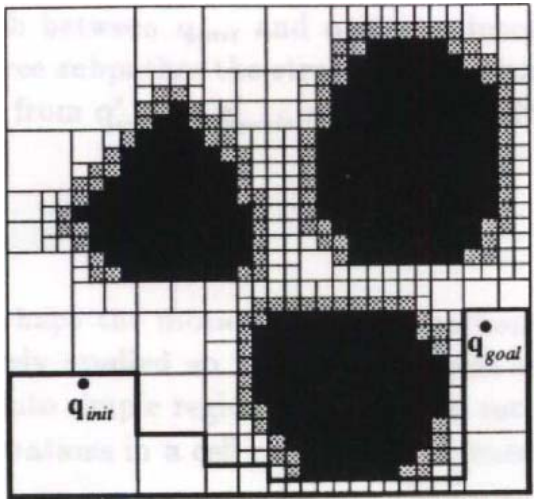
- Zerlegung ist unvollständig (Verschnitt)



z.B. Lozano-Perez und Brooks 1981: Quadrate

- Werte: Belegt, frei, teilweise belegt
- gleichmäßige Aufteilung
  => Rechenzeit ⇔ Vollständigkeit

- => besserer Ansatz:
  „divide and label“ (Quadtree)



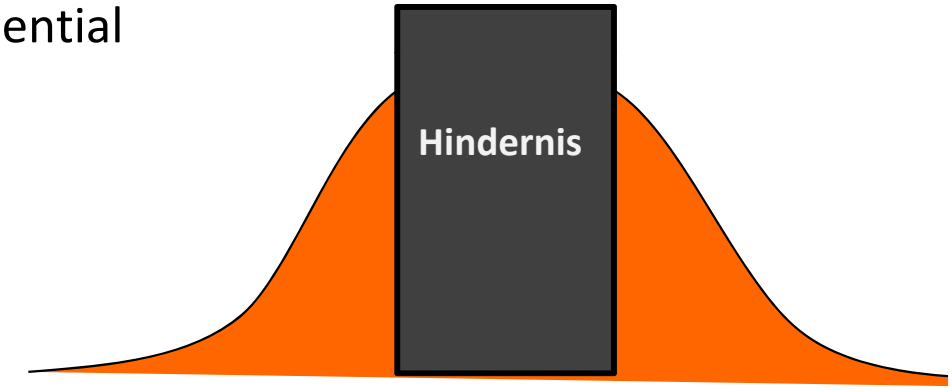
schon wieder A, geht's auch anders?*

Künstliche Potentialfelder

Pfadplanung in geometrischen Karten

- Ausgangspunkt: geometrische Karte mit Hindernissen und Zielpunkt

- Ziel erhält hohes anziehendes Potential (ähnlich Gravitation, Magnetismus)
- Hindernisse erhalten hohes abstoßendes Potential

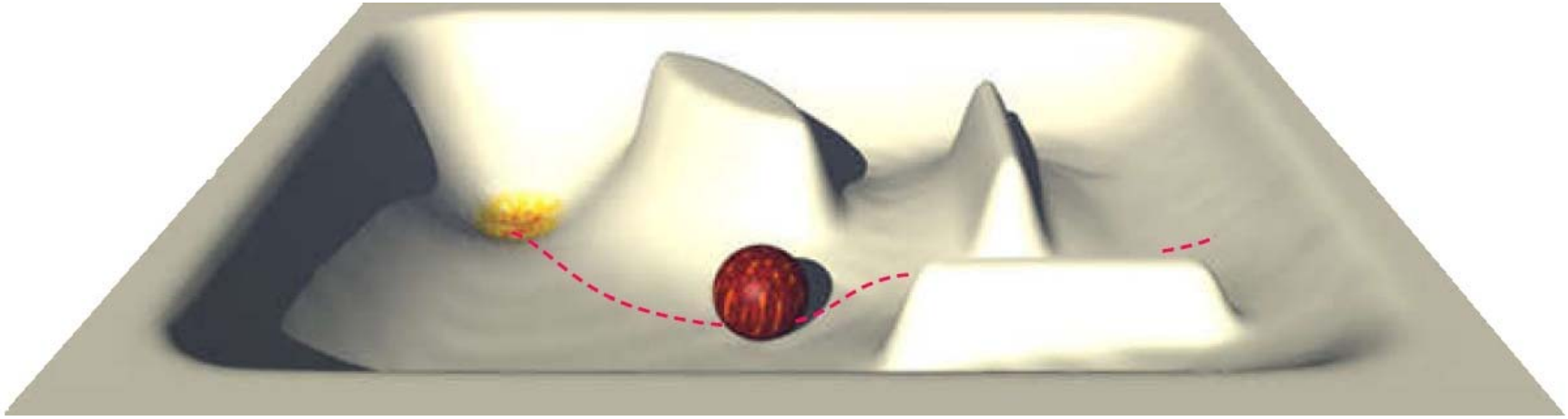


Vorgehen:

1. Erstellen des Potentialfeldes (Gebirge)
   - es gibt verschiedene Wege
2. Berechnen des Vektorfeldes (Anstiege, künstliches Kraftfeld)
3. Fahrt in Richtung der Vektoren

Beispiel - Ein Potentialfeld

- Robotertrajektorie ähnelt der Trajektorie einer Kugel, die zum tiefsten Punkt eines Gebirges rollt *



Grafik: Erich Rome

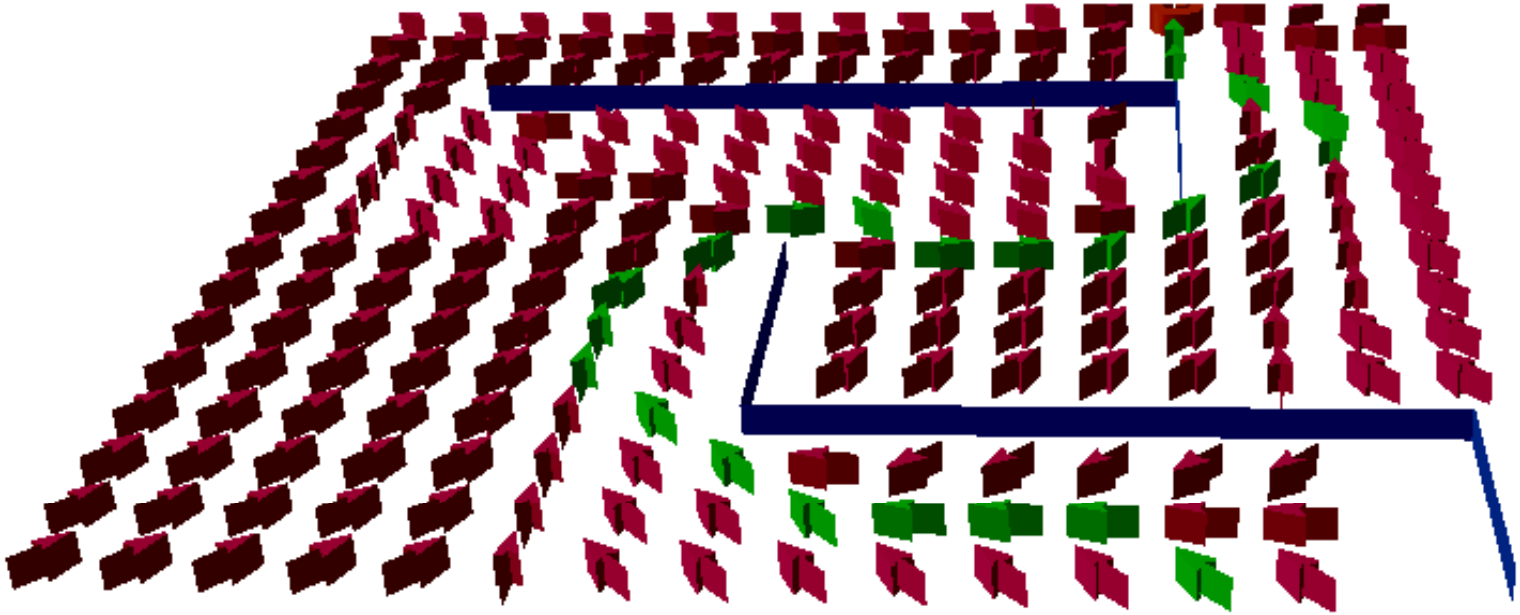
* ohne Trägheit

passt nicht zum Potentialfeld der vorigen Folie

Beispiel – Ein Vektorfeld

- Prinzip des steilsten Abstiegs ergibt das Vektorfeld (Kräfte)
- Folgt der Roboter den Vektoren, gelangt er oft auf einem kurzen Weg zum Ziel

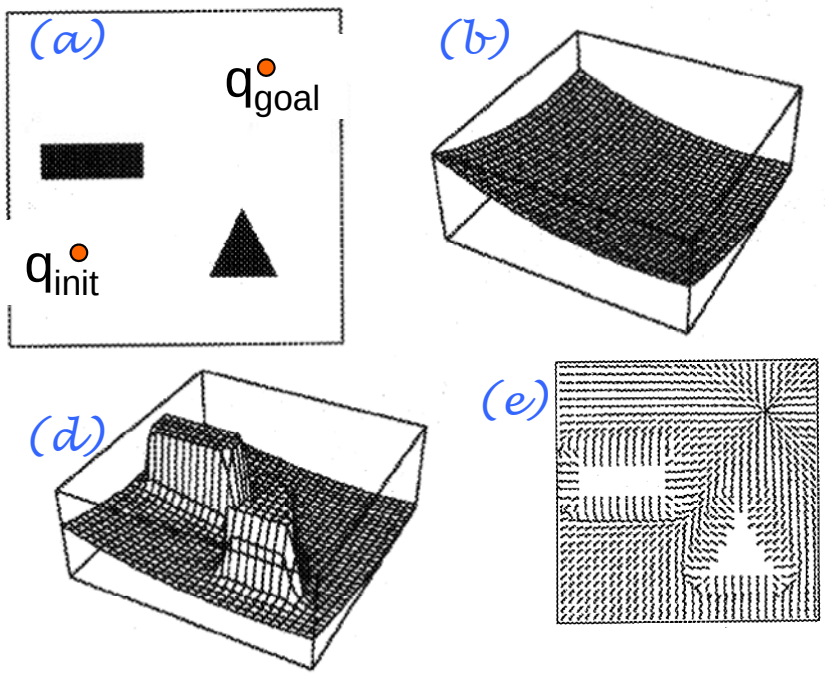
- oft, kurzer Weg vs. immer, kürzester Weg



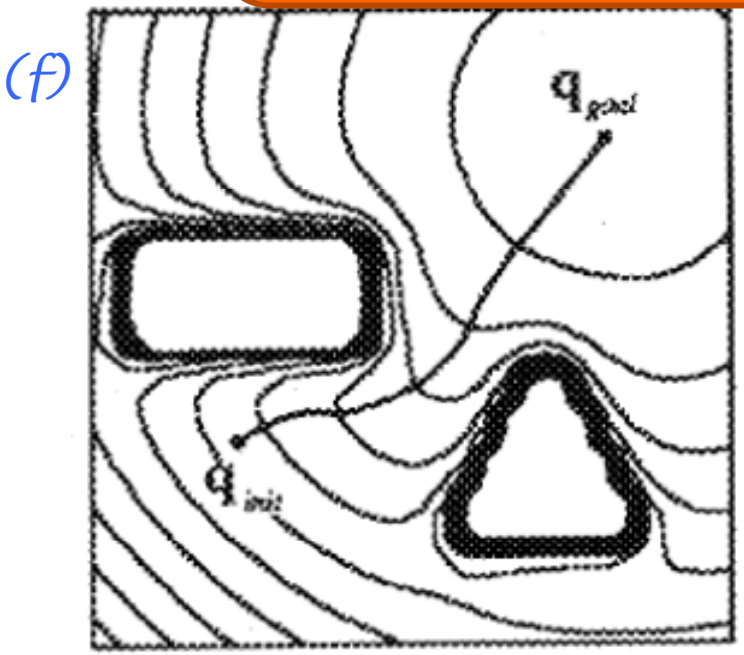
Grafik: Erich Rome

Potentialfeld – Komplettes Beispiel

q_{init} Startposition, q_{goal} Zielposition



Einfacher und gieriger Algorithmus nach Latombe, lokale Minima im Potential möglich, bspw. Bei U-förmigen Hindernissen



(a) Umwelt mit Start-/ Zielpunkt und 2 Hindernissen,
(b) anziehendes Potential des Zieles,
(c) abstoßendes Potential der Hindernisse,
(d) Summe der anziehenden und abstoßenden Potentiale,
(e) Kraftfeld symbolisiert durch Pfeile in der Ebene,
(f) resultierender Bewegungsablauf

Grafiken: J.-C. Latombe: Robot Motion Planning (Kluwer, 1991)

Algorithmus Value Iteration –

Welchen Wert (value) hat eine Position

- Erfolgreicher Algorithmus zum Finden von Policies bei Markov-Decision-Prozessen
- **Erzeugt Potentialfelder in ARIA/ARNL**
- Potential = Value = Optimale Kosten zum Ziel
- Konvergiert zu optimalen Wegen

1. Init: Ziel mit 0, alle anderen sehr hoch initialisieren:

$$V_{x,y} \leftarrow \begin{cases} 0, & wenn\ (x,y)\ Zielpunkt \\ \infty, & sonst \end{cases}$$

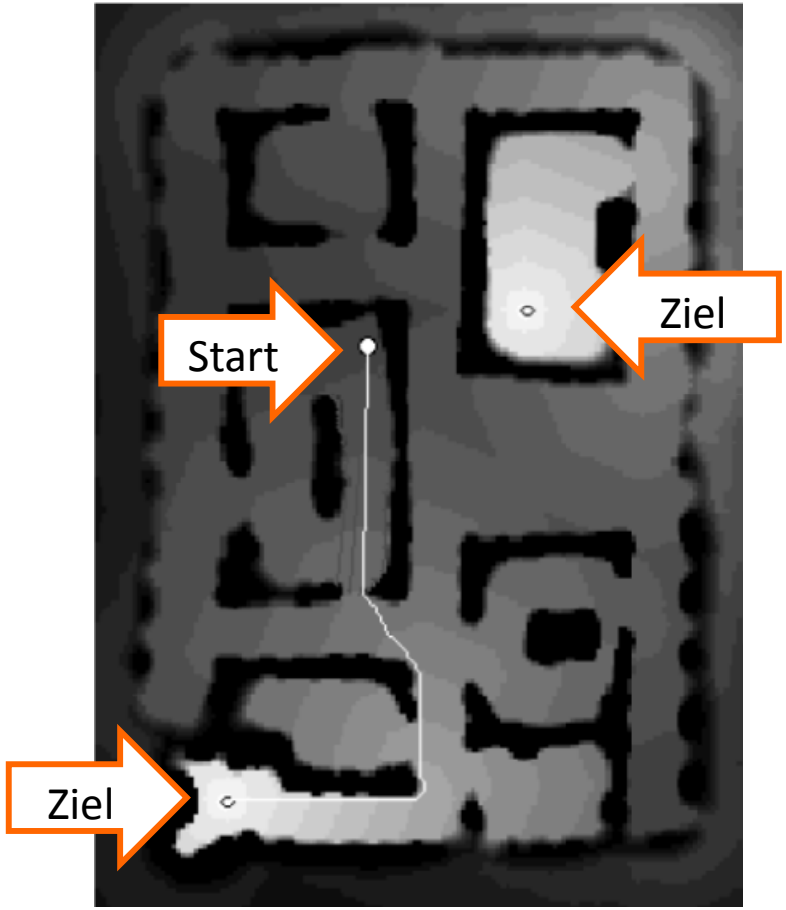
2. Iteration bis zur Konvergenz:
Für alle Kacheln des Freiraumes:

$$V_{x,y} \leftarrow \min_{\substack{k=-1,0,+1 \\ l=-1,0,1}} \left(V_{x+k,y+l} + \sqrt{l^2 + k^2} \right)$$

alle Nachbarn

Value des Nachbarn

Kosten zum Nachbarn



Thrun, S.; Bücken, A.; Burgard, W.; Fox, D.; Fröhlinghaus, T.; Henning, D.; Hofmann, T.; Krell, M. & Schmidt, T. Kortenkamp, D.; Bonasso, R. P. & Murphy, R. *(Eds.)* Map Learning and High-Speed Navigation in RHINO. *AI-based Mobile Robots: Case Studies of Successful Robot Systems, MIT Press,* **1998**

Beispiel Value Iteration

$$V(x) \uparrow \min_{x'} \left(V(x') + d(x,x') \right)$$

100	100	100	100	100	100
100	100	100	100	**0**	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	100	100	1	0	100
100	100	100	100	1	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	100	2	1	0	100
100	100	100	100	1	100
100	100	100	100	2	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	3	2	1	0	100
100	**3,4**	100	100	1	100
100	100	100	100	2	100
100	100	100	100	3	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	3	2	1	0	100
100	3,4	100	100	1	100
100	4,4	4,8	100	2	100
100	100	100	100	3	100
100	100	100	100	4	100
100	100	100	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	3	2	1	0	100
100	3,4	100	100	1	100
100	4,4	4,8	100	2	100
100	5,4	5,8	100	3	100
100	100	100	100	4	100
100	100	100	100	5	100
100	100	100	100	100	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	3	2	1	0	100
100	3,4	100	100	1	100
100	4,4	4,8	100	2	100
100	5,4	5,8	100	3	100
100	100	6,8	100	4	100
100	100	100	100	5	100
100	100	100	6,4	6	100
100	100	100	100	100	100

→

100	100	100	100	100	100
100	3	2	1	0	100
100	3,4	100	100	1	100
100	4,4	4,8	100	2	100
100	5,4	5,8	100	3	100
100	100	6,8	100	4	100
100	8,2	7,8	100	5	100
100	100	7,4	6,4	6	100
100	100	100	100	100	100

→

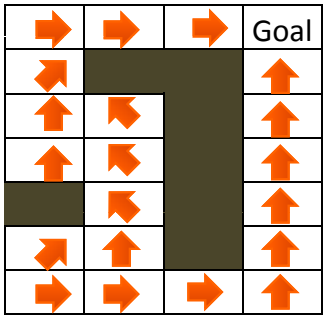
100	100	100	100	100	100
100	3	2	1	0	100
100	3,4	100	100	1	100
100	4,4	4,8	100	2	100
100	5,4	5,8	100	3	100
100	100	6,8	100	4	100
100	8,2	7,8	100	5	100
100	8,4	7,4	6,4	6	100
100	100	100	100	100	100

→

Fertiges Potentialfeld:

100	100	100	100	100	100
100	3	2	1	0	100
100	3,4	100	100	1	100
100	4,4	4,8	100	2	100
100	5,4	5,8	100	3	100
100	100	6,8	100	4	100
100	8,2	7,8	100	5	100
100	8,4	7,4	6,4	6	100
100	100	100	100	100	100

Policy ableiten: Gehe zum kleinsten Nachbarn



Optimale Pfade zum Ziel von jeder Position

Algorithmus von Latombe

- schnell (ein Durchlauf)
- Sackgassen möglich

Value Iteration

- aufwändig
- optimale Wege

- nicht nur Navigation, sondern allgemein Planung in Zustandsräumen (Beispiel gleich)
- nicht nur deterministische Automaten, sondern auch **optimale Planung bei unsicheren Aktionen** (Markov Decision Process) bei bekanntem Zustandsübergangs-modell P(s‘,|s,a)

Anderes Beispiel: Value iteration im Konfigurationsraum

- Roboter mit 2 Gelenken: 2-dim. Konfigurationsraum (Gelenkwinkel 1 und 2)

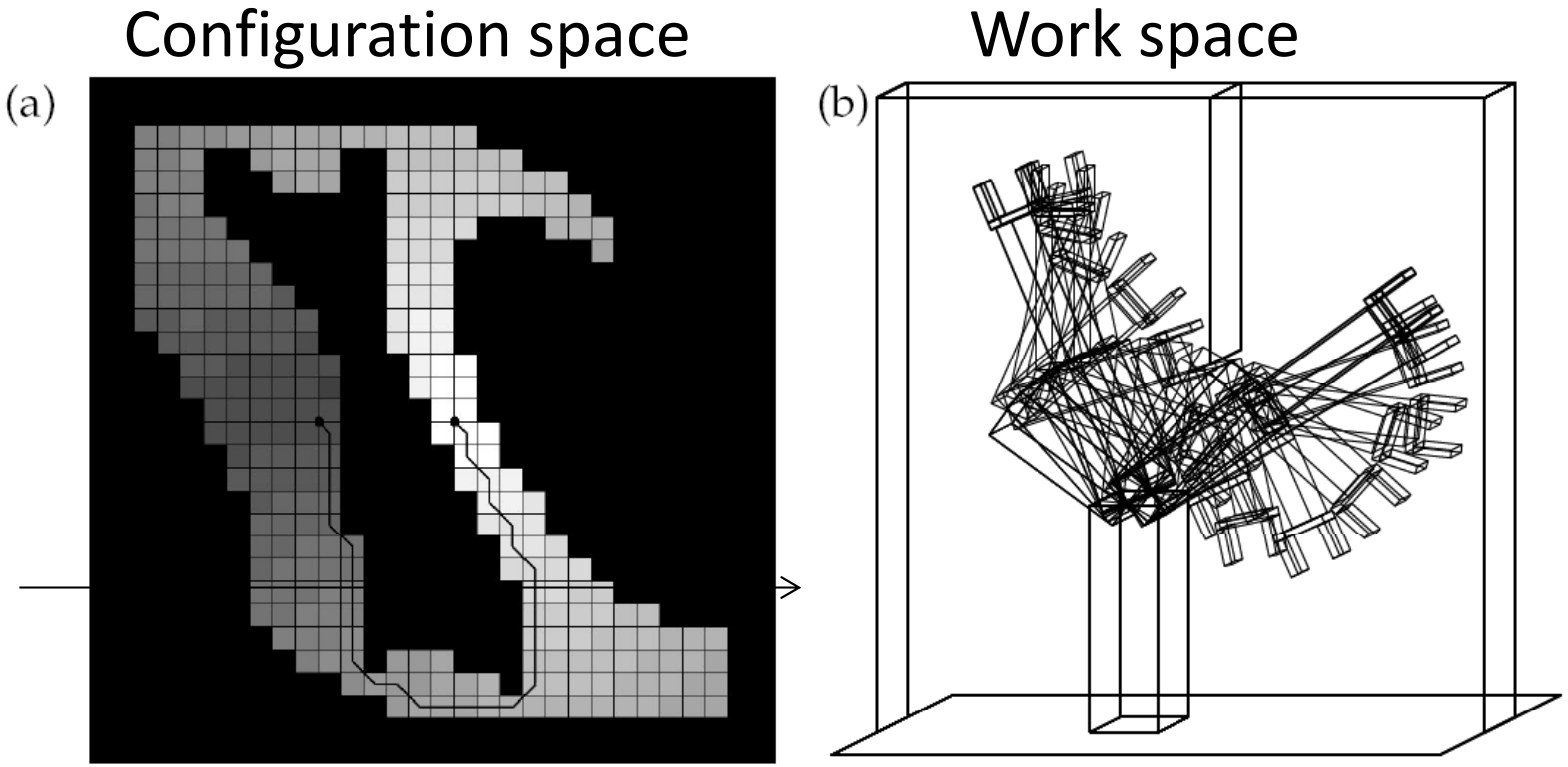


Figure 14.7 (a) Value iteration applied to a coarse discretization of the configuration space. (b) Path in workspace coordinates. The robot indeed avoids the vertical obstacle.

Planen in stochastischer Welt: MDP und allgemeine Iterationsformel der Value Iteration

Aus den konvergierten Values der Zustände lässt sich die Strategie ablesen, die den Erwartungswert der Belohnungssumme maximiert – also, welche Aktion in welchem Zustand zu wählen ist = ein Plan

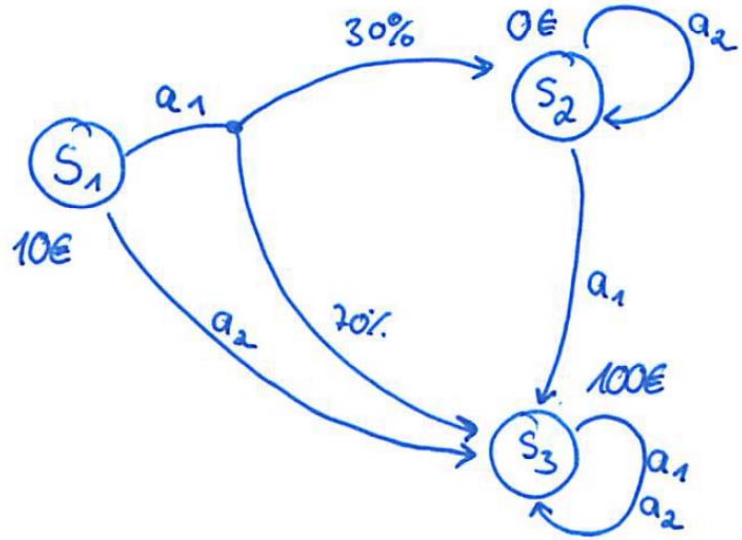
s = states

a = actions

gamma = discount factor

R = reward, Belohnung

V = value



$P(s_2 | s_1, a_1) = 0.3$

$P(s_2 | s_2, a_2) = 1$

$\vdots$

$R(s_3) = 100€$

$\vdots$

Iteration:

$$V(s) \leftarrow \max_a \left(\gamma \cdot \sum_{s'} p(s' | s, a) \cdot V(s) \right) + R(s)$$

Spezialisierung zur vorne gezeigten Bewegungsplanung mit steilstem Abstieg: Bewegungsmodell deterministisch, Reward = negative Kosten, Discountfaktor = 1 (Gesamtkosten zählen)

Potentialfelder – Fazit

Nachteile künstlicher Potentialfelder

- Erstellung aufwändig (bei Value iteration]
- Einfache Potentialfelder können **lokale Minima enthalten**

Vorteile künstlicher Potentialfelder, insbesondere Value iteration

- Mächtige Methode zur Planung bei Unsicherheit nicht nur in der Navigation, sondern allgemein bei MDP (Markov Decision Process)
- Einfach und intuitiv: „Dort will ich hin, dort nicht“
- Feld <u>wiederverwendbar</u> für alle Positionen / Zustände!
- Grundidee vieler Verfahren auch in der lokalen Navigation
  (z.B. Virtual Force Field, Elastic Band)

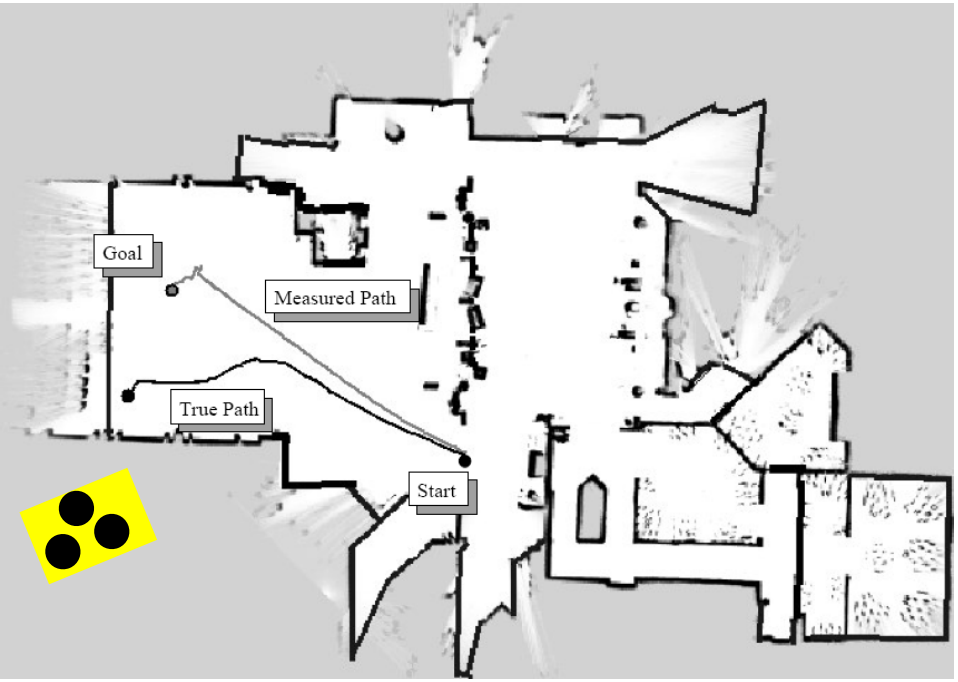
Erweiterung

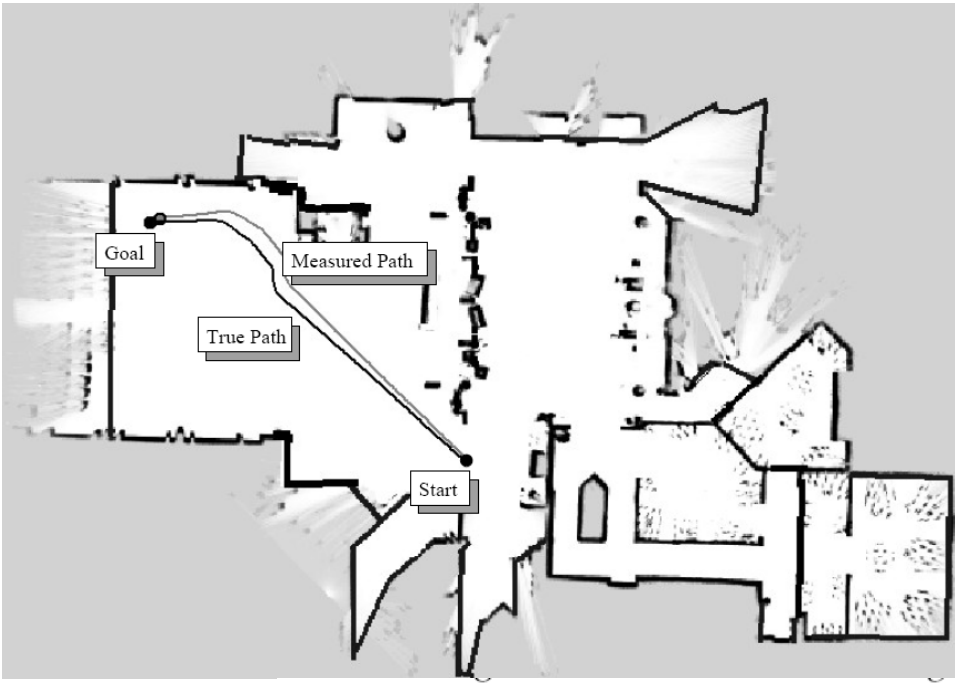
- Belohnung / Kosten nicht bekannt oder Bewegungsmodell nicht bekannt
  -> Reinforcement-Lernen

Küstennavigation (eng.: coastal navigation)

Ist kurz immer gut?

- Vermeiden von Gebieten, die die Lokalisierung gefährden
- Aktive Lokalisierung, d.h. Lokalisierung beeinflußt Pfadplanung und Fahrt



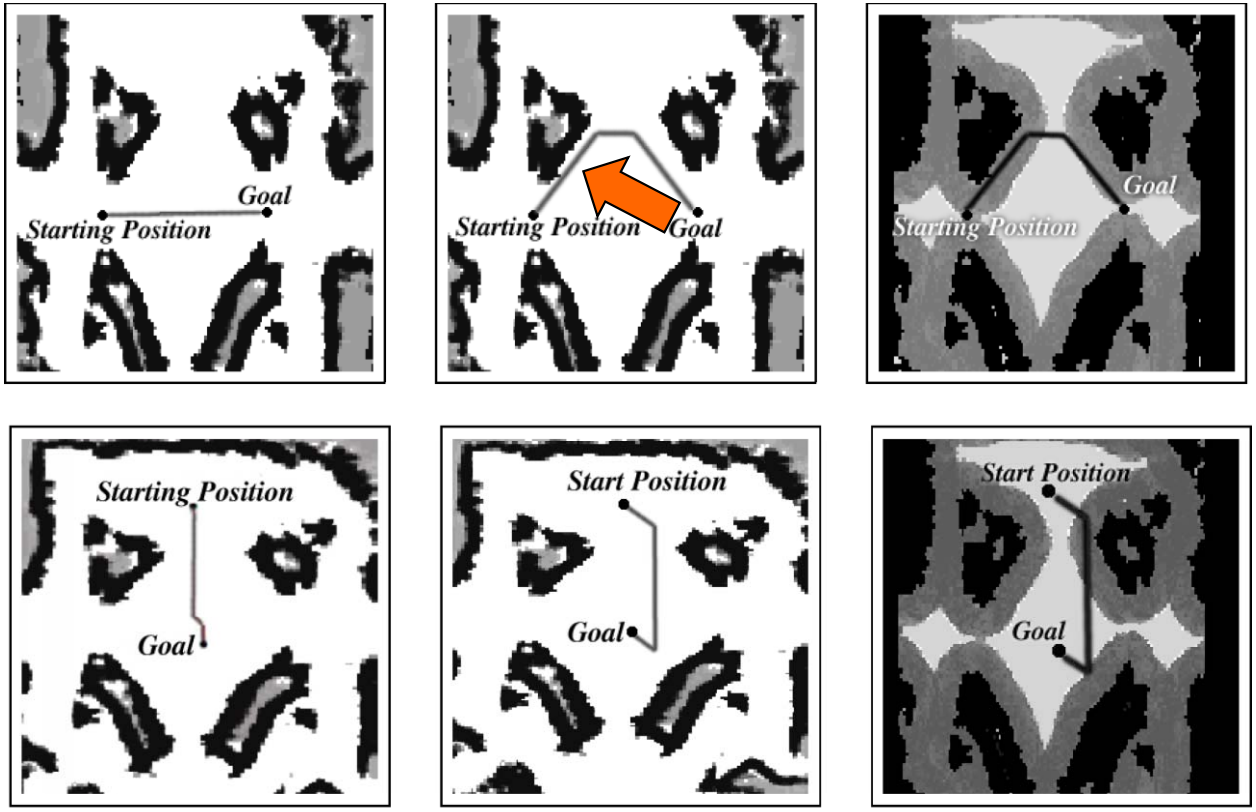


Images courtesy of Nicholas Roy, MIT.

Küstennavigation (eng.: coastal navigation)

Ist kurz immer gut?

- Vermeiden von Gebieten, die die Lokalisierung gefährden
- **Aktive** Lokalisierung, d.h. Lokalisierung beeinflusst Pfadplanung und Fahrt



(a) Conventional (b) Coastal (c) Sensor Map

- Coastal Navigation Planner:
  Pfade bevorzugt in der Nähe von Hindernissen

Abb. aus Roy, N. & Thrun, S. *Coastal Navigation with Mobile Robots* In Advances in Neural Processing Systems 12, 1999, 1043-1049

Probleme der globalen Navigation

Das war Pfadplanung

in topologischen Karten (Graphensuche) und

in geometrischen Karten

- Roadmap: Visibility-Graph, Voronoi-Diagramme
- Cell decomposition: Exakte, Approximierte
- Potenzialfeld: nach Latombe, Value iteration

Probleme der globalen Navigation

- Bewegliche Hindernisse
- Bewegliche Objekte
- Kinematik des Roboters
- **Unsicherheit** der Karte, der Beobachtung, der Bewegung

- Trotzdem, reicht globale Navigation nicht aus?

Auch bei JH

Planung und Ausführung (klassisch, top-down)

Planer

- der Stratege
- Setzt abstrakte Aufgaben (Bsp: Klausuren ins Dekanat bringen) in Navigationsaufgaben um.
- Start und Zielpunkte

Wohin soll ich gehen?
Task Planing

Navigator

- Pfadplaner, der Kapitän
- Konstruiert Pfad als Folge von Pfadsegmenten
- Übergibt die Pfadsegmente in Form von Kursvektoren (Liste von Zwischenzielen) an die Pilotkomponente.

Wie komme ich dahin?
Path Planning

Pilot

- Fahrer, Steuermann
- Setzt Kursvektoren in Steuerbefehle um und übergibt sie dem Lokomotionssystem.

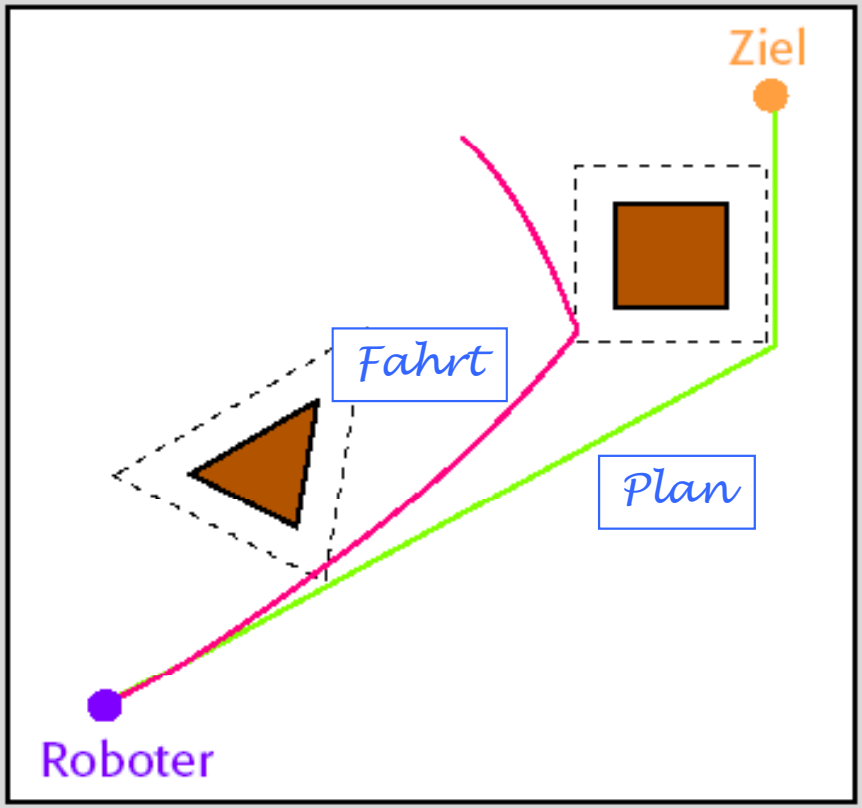
PSOS, ARCOS des Pioneer
Actions

- **Wozu die ganze Sensorik? Wer guckt aus dem Fenster? Keiner, alle oder nur der Pilot? Wozu „Wo bin ich?“, Wozu Selbstlokalisierung, wer weicht den Hindernissen aus?**

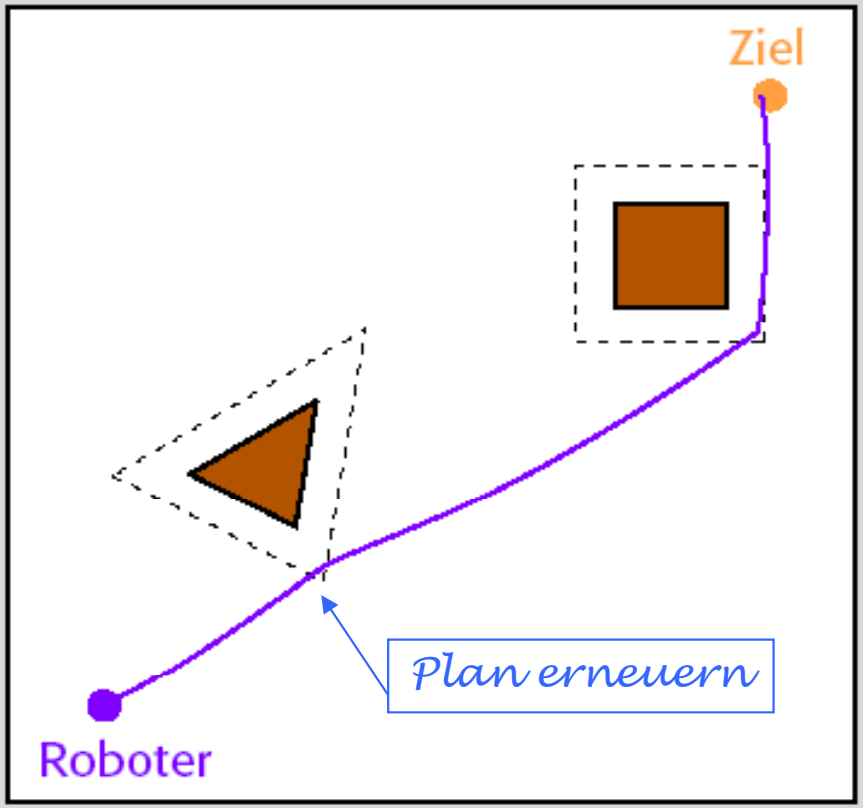
→

- **Ist es wirklich so einfach?!** **Fast**

Durchführung der Fahrt (1)



Abweichung vom Plan aufgrund von Unsicherheiten der Kinematik & Sensorik



Lokalisation und lokale Neuplanung

Verzahnung von globaler und lokaler Navigation

Wegplanung und Planausführung => Kontrollarchitekturen

- Orientierungslos?
  Die Karte an der <u>angenommenen Position</u> (Selbstlokalisation) muss im Pilot und/oder Navigator ständig mit der <u>aktuellen Wahrnehmung</u> verglichen werden (Kartenabgleich, Matching)

Was tun bei Abweichungen?

Probleme lösen ...

- ✓ Korrektur der Position in der internen Karte (Lokalisierung)
- ✓ Lokales Neuplanen, Hindernissen ausweichen, Pfad wieder erreichen (MOSRO, ARIA)
- ✓ Vertrauen auf Beschränkung der Abweichung (Stanley, DGPS-Fahrten)

oder eskalieren:

- Gesamtroute neu erstellen (globales Neuplanen)
- Wake-Up-Problem lösen
- Aufgabe abbrechen

jede Architektur enthält einen Punkt, in dem Wahrnehmung und Plan aufeinander treffen.

Die Fahrt

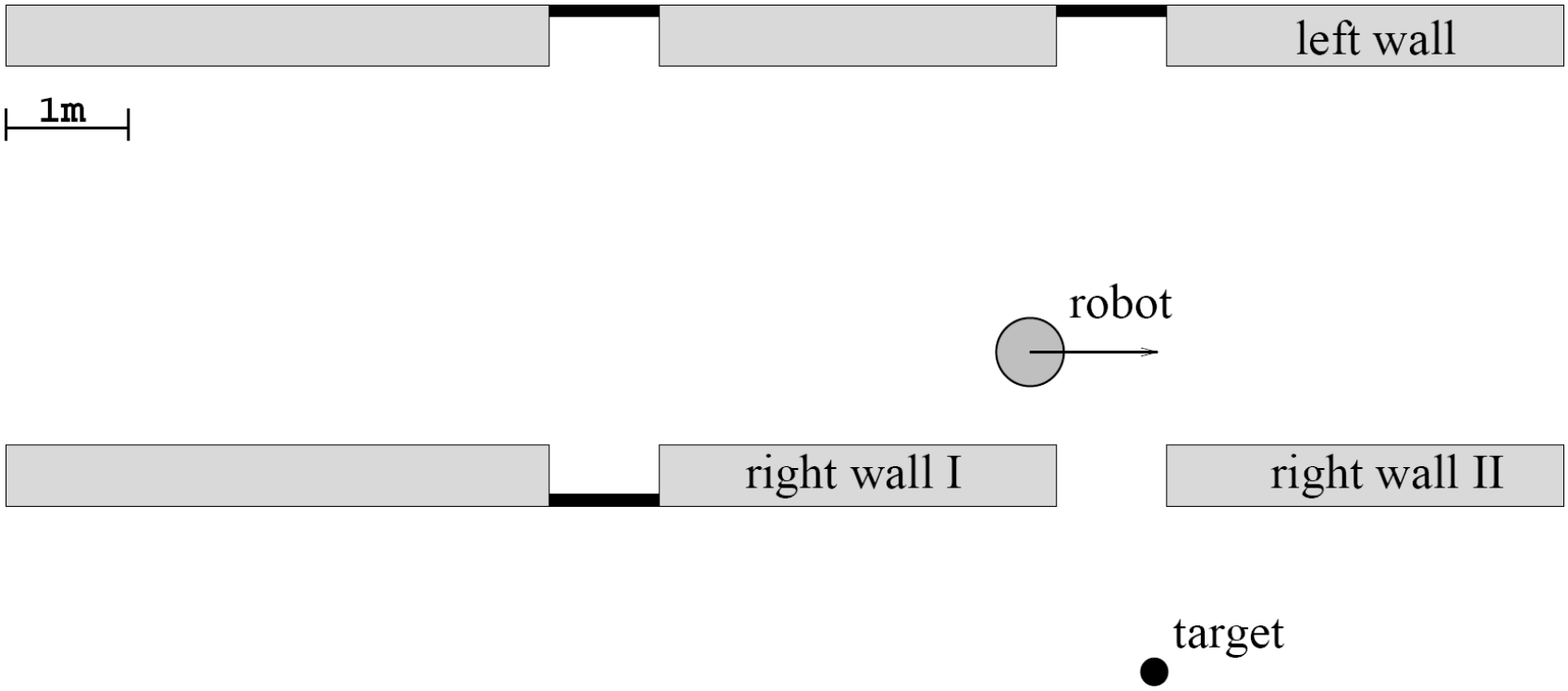
- Plan vorhanden, Lokalisierung bemüht sich, aber

Hindernisse tauchen auf

-> lokale Navigation: Ausweichen und zurück auf den Pfad

- in ARIA/ARNL: Dynamic Window Approach (DWA)

Dynamic Window Approach (DWA)



Fox, D.; Burgard, W. & Thrun, S. *The dynamic window approach to collision avoidance* Robotics & Automation Magazine, IEEE, 1997, 4, 23-33

Dynamic Window Approach (DWA)

- Steuerung mit Rotations- und Translationsgeschwindigkeit = eine Aktion
- → 2-dimensionaler **Aktionsraum** (velocity space)
- Welche Aktion (Geschwindigkeitspaar) ist zu wählen?
- jede Aktion erzeugt eine Bahnkurve
- Bahnkurven, die im nächsten Zeitschritt ein Hindernis treffen (dunkelgrau), sind verboten.

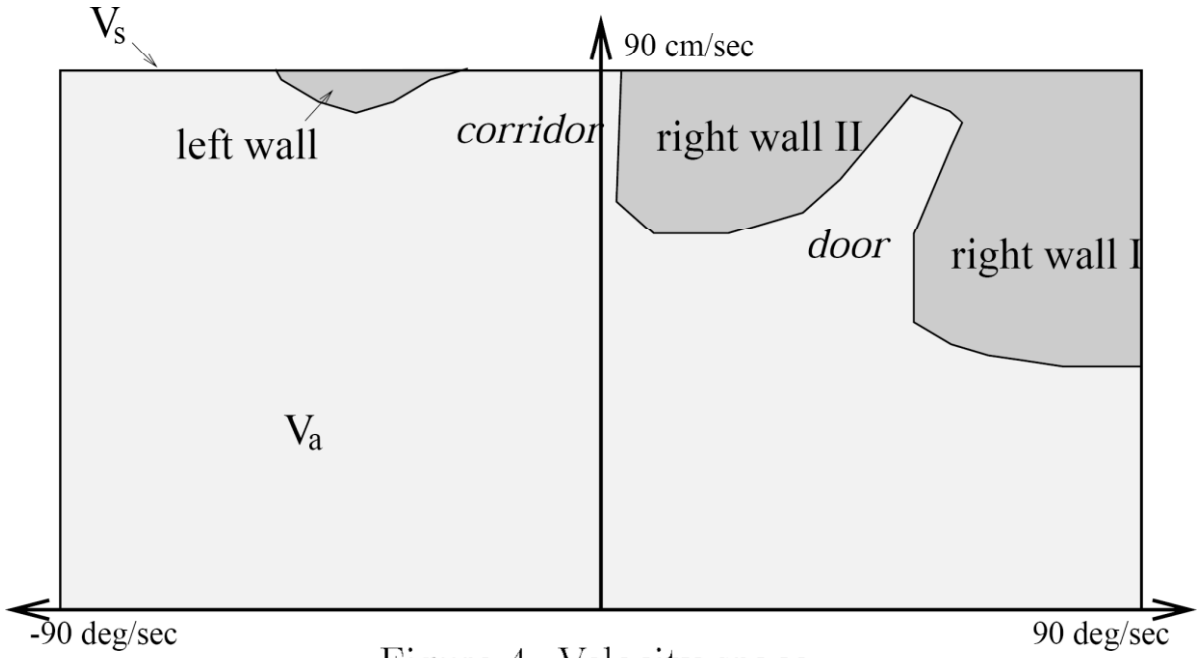
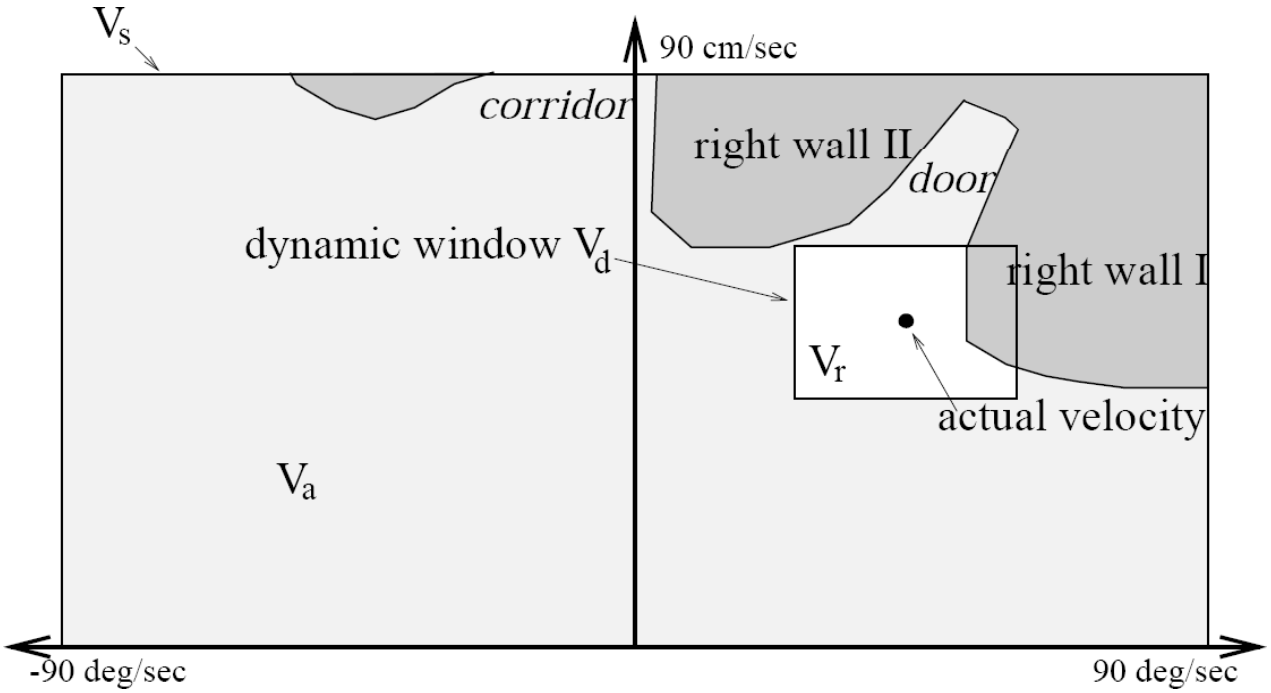


Figure 4. Velocity space

Dynamik-Fenster

- Welche Geschwindigkeiten sind ausgehend von der aktuellen Geschwindigkeit und den möglichen Beschleunigungen überhaupt zu erreichen? → Dynamik-Fenster



- Optimierung:
  Finde die beste Aktion (v, ω)
  aus dem weißen Sektor.

- Multiple Kriterien:
  Zielrichtung (heading),
  Hindernisabstand (dist),
  Geschwindigkeit (v)

$$\langle v^*, \omega^* \rangle = \underset{v,\omega}{\operatorname{argmax}} \left(-\alpha_1 \cdot |\mathrm{heading}(v,\omega)| + \alpha_2 \cdot \mathrm{dist}(v,\omega) + \alpha_3 \cdot v \right)$$

Dynamic Window Approach in ARIA/ARNL

- Auszug aus arnl.p, Paramter in MobileEyes:

HeadingWt 0.8	; range [0, 1], Heading weight for DWA. Unlike the ; heading objective computed from a destination pose ; on the path, as in the conventional DWA, we use a ; path matching function to match the arc made from ; the velocities with the desired computed path.
DistanceWt 0.1	; range [0, 1], Distance weight for DWA. Distance ; refers to the distance to collision if any if the ; robot continues on the path computed from the ; velocities.
VelocityWt 0.1	; range [0, 1], Velocity weight for DWA. Velocity ; refers to the linear velocity only.

Haben wir die Antwort auf „Wie komme ich zu meinem Ziel?“?

Pfadplanung in der Karte (globale Navigation) und Durchführung der Fahrt (lokale Navigation).

Unsicherheit der Sensordaten und der Bewegung können zu Abweichung der angenommenen Position von der realen Position führen.

Lösung durch probabilistische Lokalisierung oder ausreichende Genauigkeit der absoluten Positionierung oder komplizierter durch Eskalation zu hierarchisch höheren Modulen.

Es gibt kein allgemeindienliches Navigationsverfahren für AMS.

In ARIA/ARNL:

→Value Iteration
→Dynamic Window Approach

→Monte Carlo Localization

Orientierung von Menschen im Wald

- Aufgabe: Geradeausgehen
- Eingeschränkte Sicht nur auf die Füße
- Blaue Kurve: bewölkter Himmel
- Gelbe Kurve: sonnig
- Interpretation: Sonnenlicht hilft beim Halten der Richtung
- Uni. Stanford, 2011, AI-Class

Navigationsphänomene bei Tieren

- Odometrie beim Tausendfüßler
- Heimweg bei der Wüstenameise (Cataglyphis)
- Wasserläufer

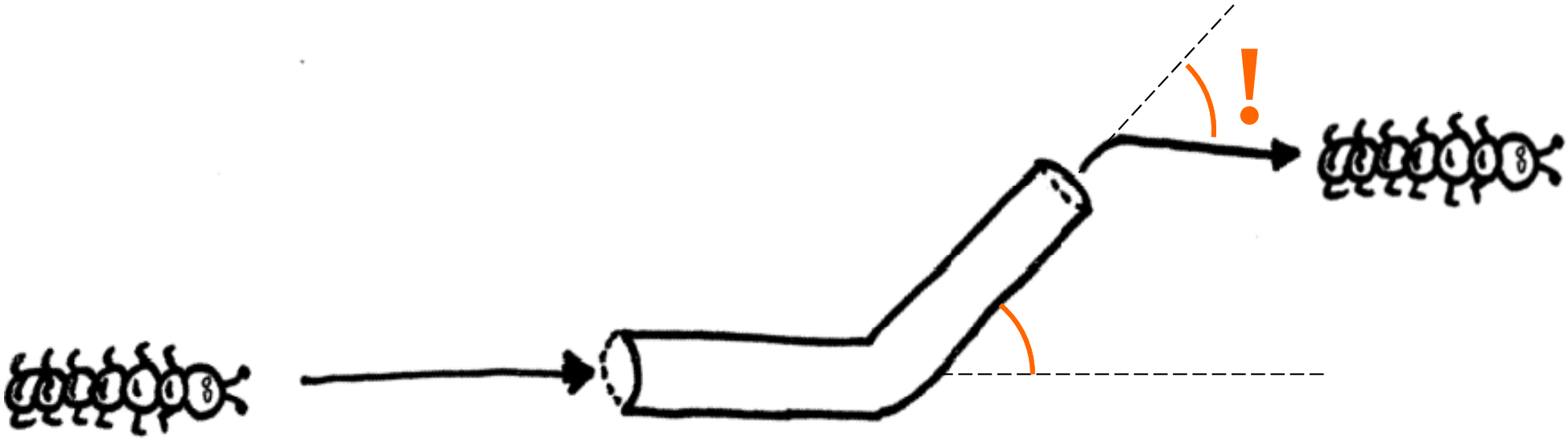
Odometrie beim Tausendfüßler

(Varju S. 142ff)

Tausendfüßler haben die Fähigkeit, auch ohne richtende Außenreize längere Strecken geradeaus zu laufen

Versuch1: Laufen durch einen engen Gang, der im Winkel *alpha* nach links geknickt ist

- **Beobachtung**: Kam der Tf. aus dem abgeknickten Teil heraus, so machte er im Mittel eine Rechtsdrehung um den gleichen Winkel *alpha*, so dass er seine ursprüngliche Richtung wiedererlangte
- **Deutung**: **Wahrnehmung der eigenen Bewegung (relative Positionierung)**



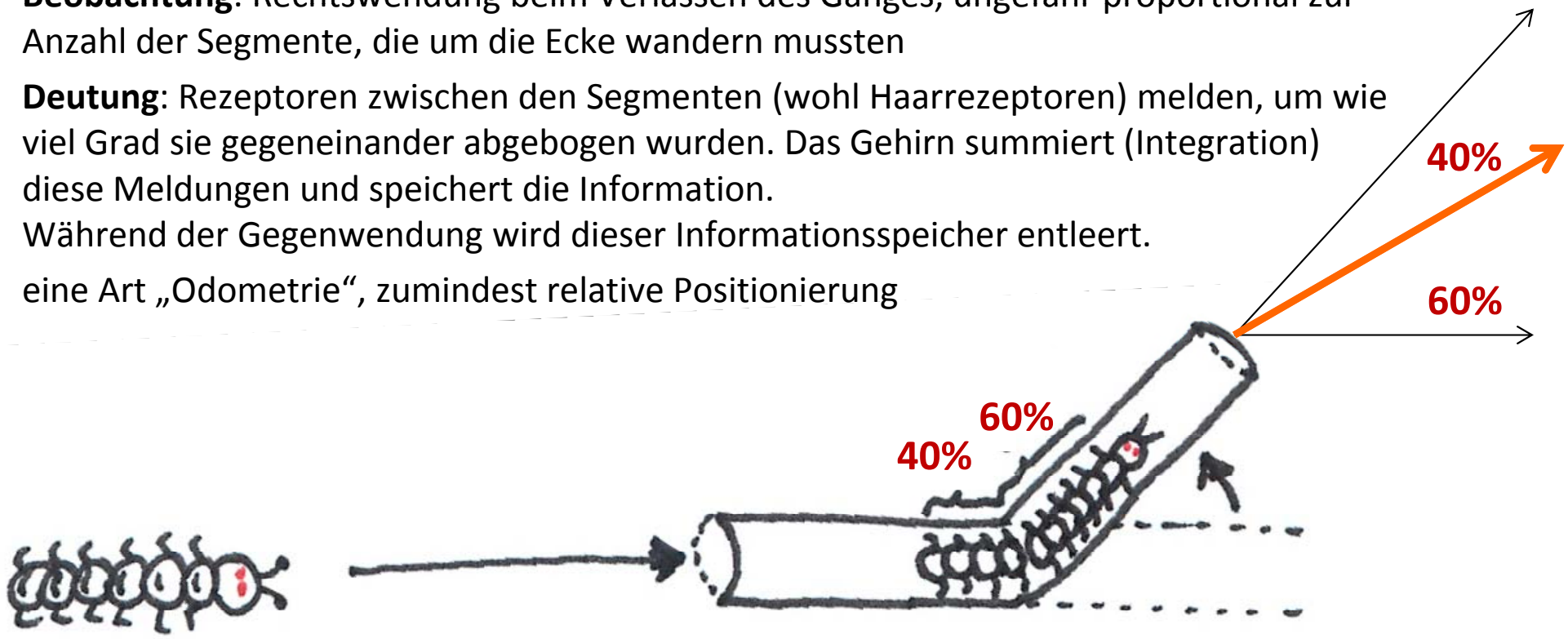
Odometrie beim Tausendfüßler

(Varju S. 142ff)

Tausendfüßler haben die Fähigkeit, auch ohne richtende Außenreize längere Strecken geradeaus zu laufen

<u>Versuch2: Knicken des Ganges nach links, während das Tier dort entlanglief: 1/3, 1/2, 2/3 der Körperlänge schon vorbei, hintere Segmente müssen um die Ecke.</u>

- **Beobachtung**: Rechtswendung beim Verlassen des Ganges, ungefähr proportional zur Anzahl der Segmente, die um die Ecke wandern mussten
- **Deutung**: Rezeptoren zwischen den Segmenten (wohl Haarrezeptoren) melden, um wie viel Grad sie gegeneinander abgebogen wurden. Das Gehirn summiert (Integration) diese Meldungen und speichert die Information.
  Während der Gegenwendung wird dieser Informationsspeicher entleert.
- eine Art „Odometrie“, zumindest relative Positionierung



Homing der Wüstenameise

(Rüdiger Wehner, Universität Zürich)

Homing = Heimfindevermögen

Wie findet eine Wüstenameise
Cataglyphis nach Wanderungen von
hunderten von Metern zum Nest zurück?

- Keine chemischen Stoffe (Hitze!)
- Wohngebiete arm an Landmarken
- Tiere sind Einzelgänger, legen keine Pfade an, markieren nicht mit Duftstoffen

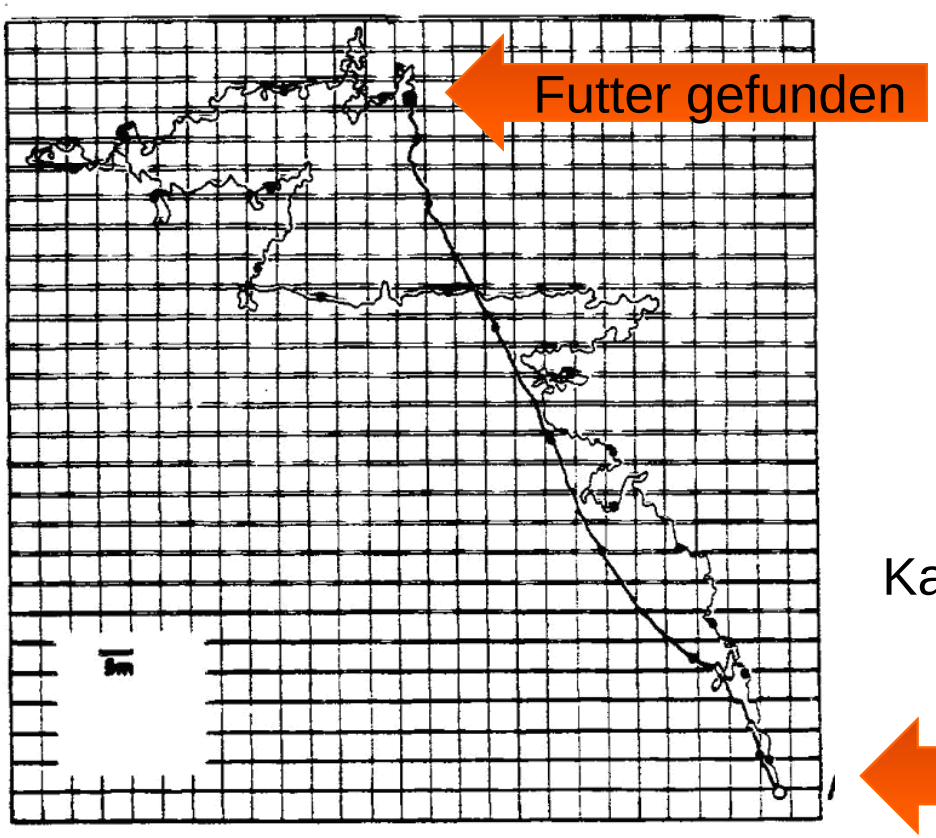
www.ifi.unizh.ch/groups/ ailab/projects/sahabot/

- **Ein Modell:**
  Pfad-Integration (Odometrie)
  + Kompass (Polarisationsmuster)
  + Sichtorientierung (ALV-Modell)

Gehirngröße

Odometrie bei der Wüstenameise (Cataglyphis) I (Varju S. 145ff)

- Versuch1: tunesische Wüste, Rückkehr zum Nest nach der Futtersuche
  Beobachtung: gerader Weg zum Nest, trotz stark gewundenem Hinweg.



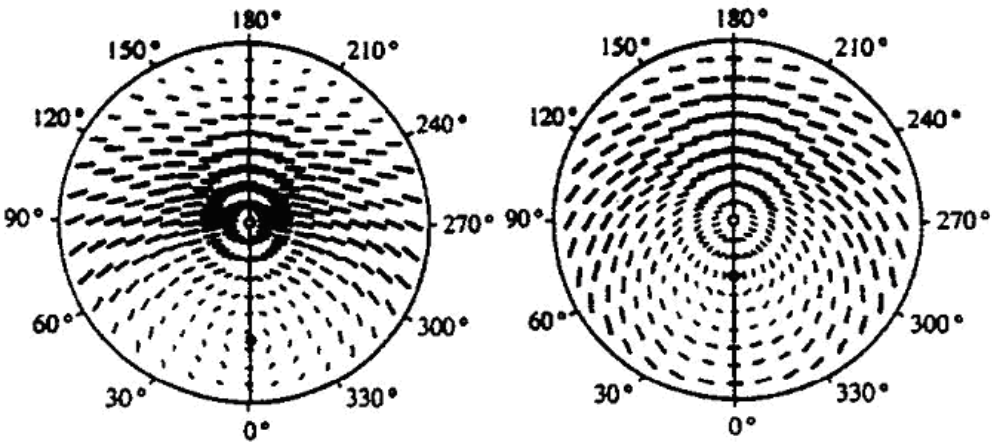
Kachelgröße 5m!

Odometrie bei der Wüstenameise (Cataglyphis) II

- Versuch2:
  - Versetzen der Ameise um 50m beim Rückmarsch
  - gerade Linie parallel zur Futter-Nest-Linie in 50m Abstand
  - **Deutung: Odometrie - mit Propriozeptoren (körpereigene Rezeptoren)?**

  - Absoluter Richtungssensor -> Hilfe bei der Odometrie
  - Ameisen das **Polarisationsmuster des Himmels** wahrnehmen können (ständig klarer Himmel in Tunesien)
  - Muster ist tageszeitabhängig: innere Uhr notwendig

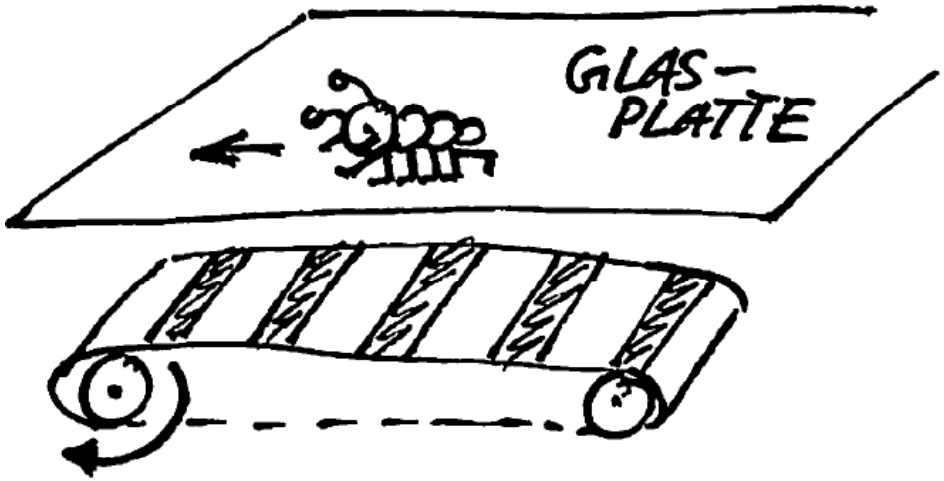
B



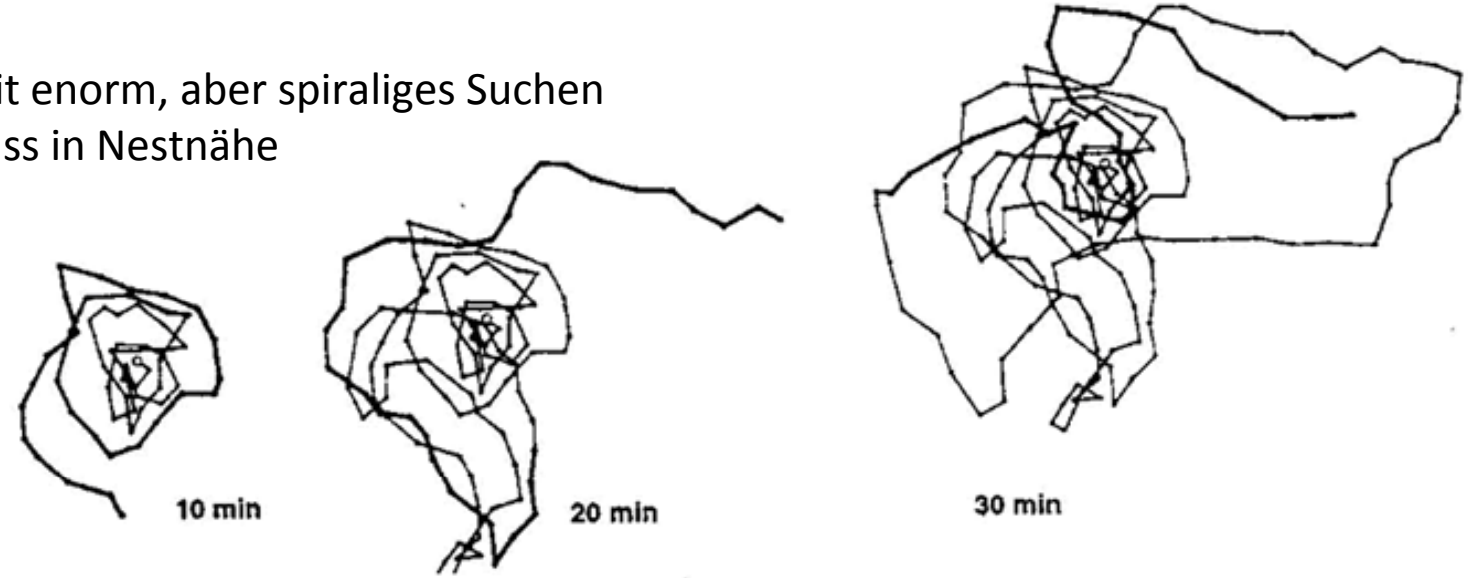
Odometrie bei der Wüstenameise (Cataglyphis) III

Frage: Wie nimmt die Ameise Entfernungen wahr?

- Schrittzahl (vermutet) + Geschwindigkeitssensor (Bodenbeobachtung, nachgewiesen)



- Genauigkeit enorm, aber spiraliges Suchen als Abschluss in Nestnähe



ALV-Modell
(D. Lambrinos, 1998, Uni Zürich)

ALV-Modell = average landmark vector model

Voraussetzungen:

- Interner Kompass
- Seltene auffällige Landmarken

Idee:

- Zu jedem Punkt des Raumes kann das sichtbare Bild der Landmarken als **Vektor** gespeichert werden
- Die Ameise merkt sich das Bild am Nest
- Sie kennt eine Methode (Sie auch bald), um aus einem beliebigen anderen Bild und dem Nestbild **die Richtung abzuleiten, in der das Nest liegt**

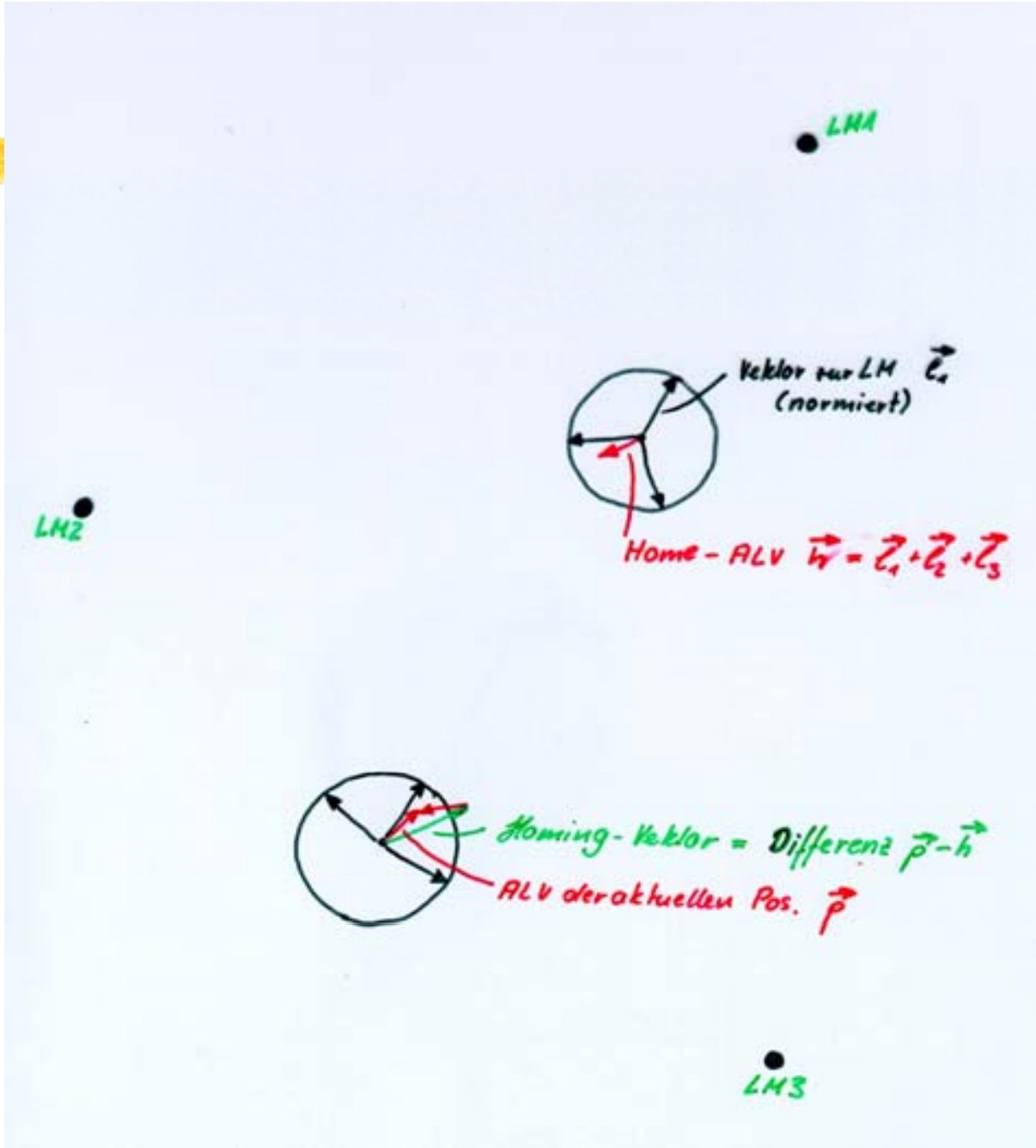
360° - Kamera und Sahabot

Figure 9: Detailed view of the 360° camera system of the Sahabot 2, consisting of a camera with vertical optical axis (within the case) and a conical mirror that is aligned with the axis of the camera and mounted within an acrylic glass tube. The axial alignment and the distance between mirror and camera can be adjusted. An infrared and a neutral density filter are mounted between camera and mirror.

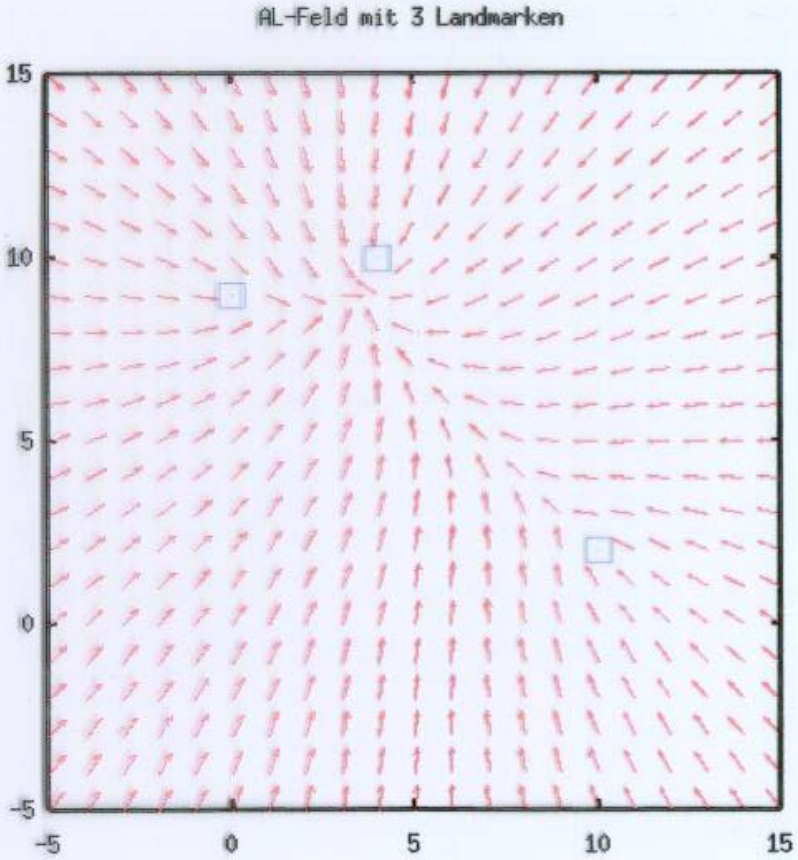
ALV Grundidee

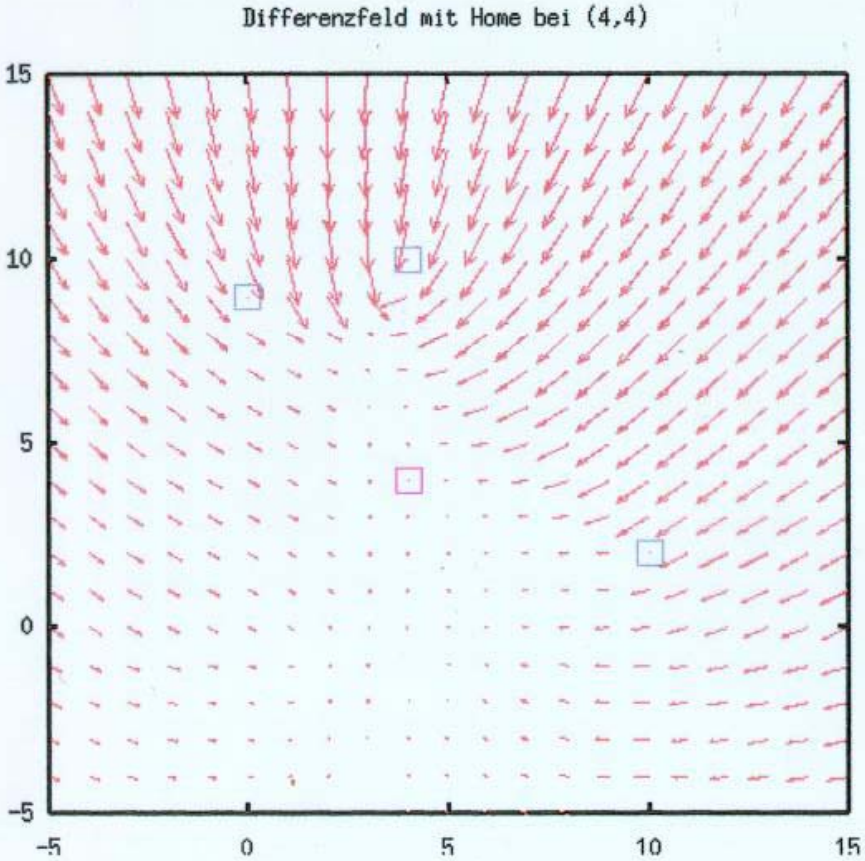
- Summe der Richtungsvektoren vom Insekt zu den Landmarken wird an verschiedenen Positionen gemerkt:
- ALV(Nest) = h
- ALV(Aktuelle Position) = p

- Die Differenz zwischen den ALVs p und h gibt eine ungefähre Zielrichtung von p nach h an
  – den Homing-Vektor



Landmark-Vektoren und Homing-Vektoren des ALV-Modells

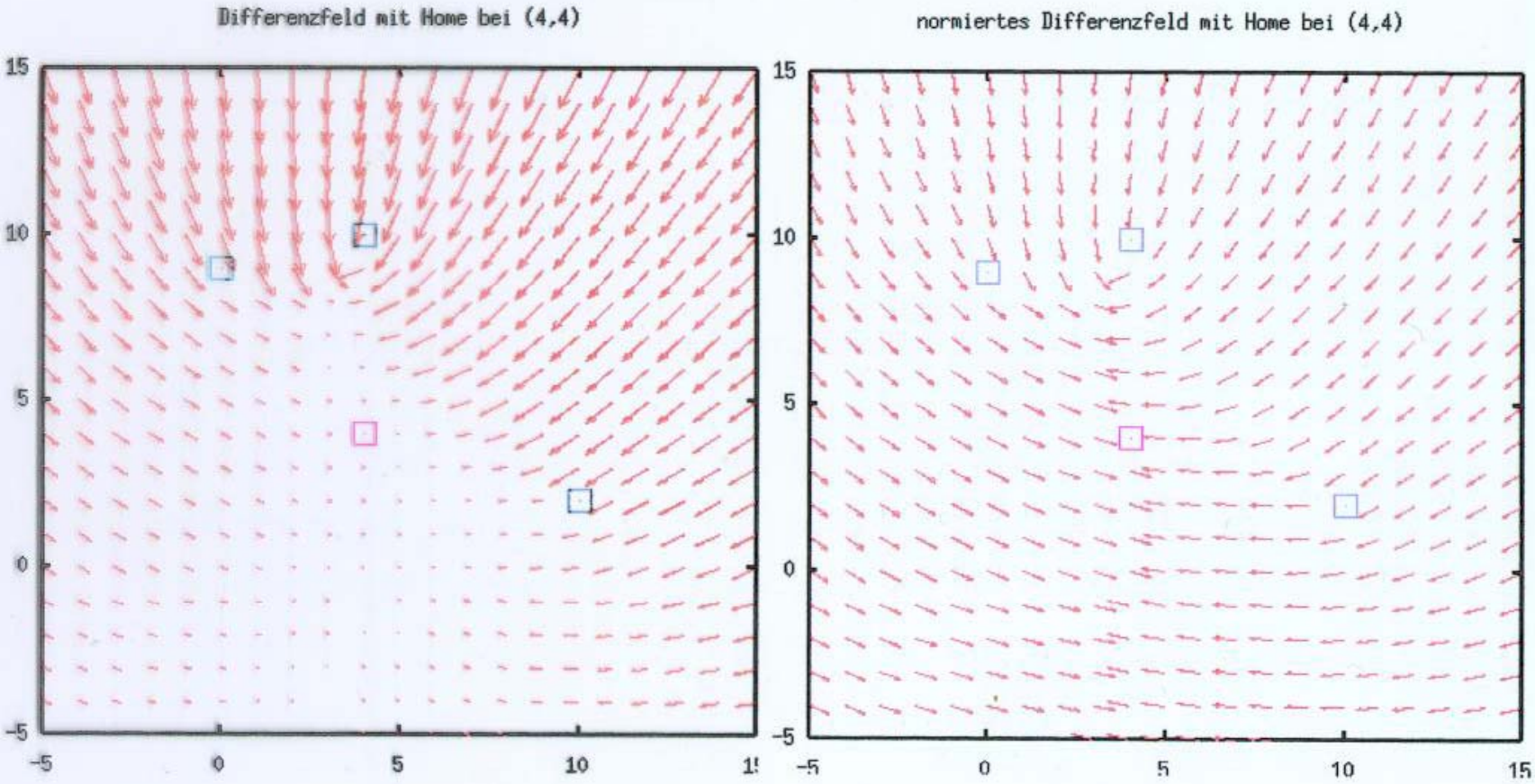




I. Boersch

3

Normierte Homing-Vektoren



I. Boersch

4

Wasserläufer

Wasserläufer

(Varju S.72ff)

- 90cm/s, Ruderschläge der Mittelbeine

Wie kommt der Wasserläufer zur Beute?

- Sehen? Lackschicht auf den Augen
  => immer noch Orten und Schnappen der Beute (aber geringfügig ungenauer)
- Wahrnehmen der Wellenbewegung des Wassers
  => Drehen in Beuterichtung (auf 10° genau)=> Beinstellung möglichst weit auseinander >Abb
- typische Wellenfronten (Wellenlänge) von Beutetieren: 2-12mm
  => hierfür Sensitivität am größten, (Höhenunterschiede 1-5 tausendstel mm wahrnehmbar)
- Vermutung: Wahrnehmung durch Sensoren für Gelenkverbiegung
- Unterscheidung von Beute, Geschlechtspartner, Rückenschwimmer (Feind), Kommunikation über Wasserwellen (Trommeln) zum Anlocken und Vertreiben von Artgenossen !

Wie hält der Wasserläufer auf fließenden Gewässern die Position? (Verdriftung)

- optisches System: Landmarkennavigation mit markanten Objekten am Ufer und Lichtquellen (Genauigkeit +/- 10 cm bei v=180m/h)

Wie werden neue Gewässer besiedelt?

- Aufbruch im Frühjahr, übers Land fliegen: Unterscheidung von Land und Wasser anhand Polarisation: unpolarisiertes Sonnenlicht wird durch Reflektion horizontal polarisiert (ebenfalls Plastikfolien, nasser Asphalt, Ölflächen), verbreitete Methode bei Insekten

Summary

Andere Phänomene

- Landmarkennavigation bei Bienen, Informations-austausch über Futterstellen (Schwänzeltanz)
- Zugvögel
- Zugverhalten von Aalen, Lachsen, Schildkröten, Kröten, Schmetterlingen, Heuschrecken, Libellen ..
- Verstecken und Wiederfinden von Futter (Meisen, ...)
- Brieftauben u.v.a.

Häufige Beobachtungen

- starke Anpassung an den Lebensraum (Nische) durch Evolution
- <u>Parallele</u> Nutzung aller wahrnehmbaren Umweltreize -> graceful degradation

Wolfgang Wiltschko (Uni Frankfurt, FB Biologie, Zoologisches Institut):

„ Das ist überhaupt manchmal zum Verzweifeln. Als ich mich auf Tauben verlegt habe, dachte ich, das ist nun endlich einmal ein einfaches Modell, um herauszufinden, wie ein Zugvogel in der letzten Phase seines Heimkehrfluges navigiert.
Und dann zeigte sich: Sie machen es heute so, morgen so; oder die einen so, die anderen aber ganz anders.“

Bilderquellen

[Everett, 1995] Everett, H. R., 1995, Sensors for Mobile Robots: Theory and Application, ISBN 1-56881-048-2, A K Peters, Ltd., Wellesley, MA.

[Borens 1996] Borenstein, J., Everett, H.R., Feng, L., 1996, " 'Where am I?' Sensors and Methods for Mobile Robot Positioning." Technical Report, The University of Michigan. This 230-page report is available in its entirety on the web and on CD-ROM.

[Konolige 1998] Kurt Konolige, 1998, Mobile Robots and Motion

[Saph61f 1998] Saphira Software Manual Version 6.1, 1998

[Deutsch 2004] Deutsch, Ch., 2004, Pfadplanung für einen autonomen mobilen Roboter, DA an TU Graz

[Rome 1996] Rome, E., 1996, Über Navigation autonomer, mobiler Roboter (AMR), KIFS 1996

[Loh 2000] Loh, M. , 2000, Proseminar: Bahnplanung in der Robotik, TU München

[Varju 2002] Varju, D., 2002, Mit den Ohren sehen und den Beinen hören, C.H.Beck

[Lambrinos 2000] Lambrinos, D., 2000, Saharan Ants and Robots, http://www.ifi.unizh.ch/groups/ailab/projects/sahabot/

u.a. im Text genannte