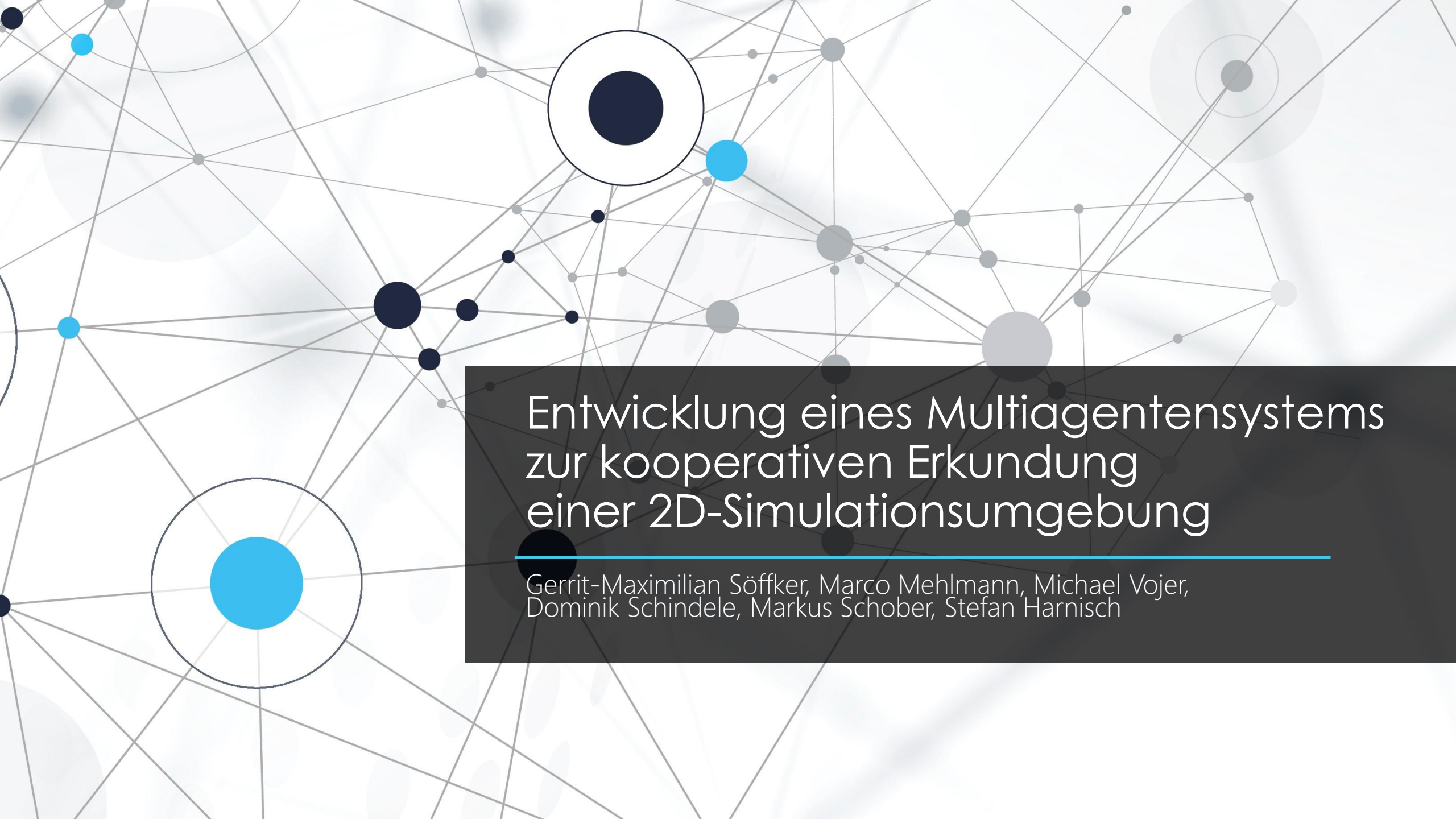
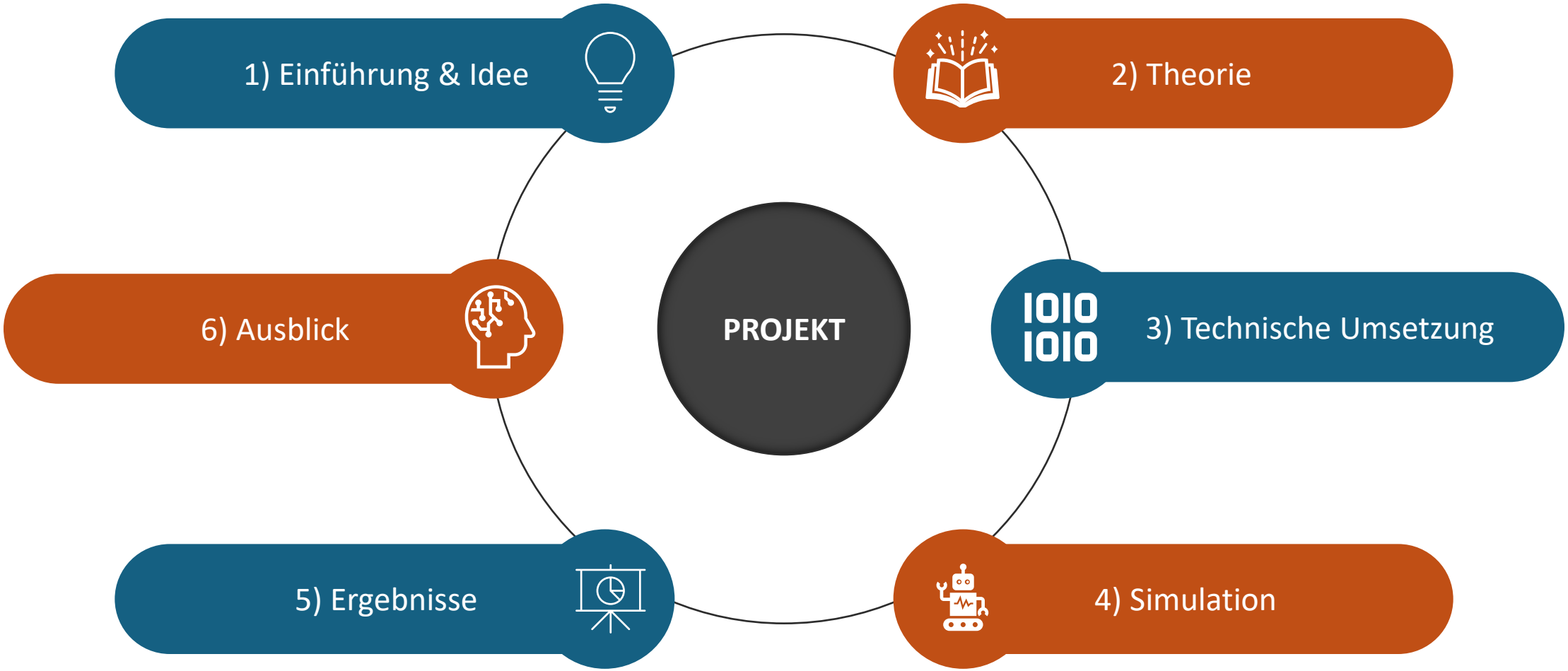


Entwicklung eines Multiagentensystems zur kooperativen Erkundung einer 2D-Simulationsumgebung

Gerrit-Maximilian Söffker, Marco Mehlmann, Michael Vojer, Dominik Schindele, Markus Schober, Stefan Harnisch

Projektaufbau



Einführung ins Thema

Automatische Kartierung unbekannter Gebiete – eine Herausforderung für die Robotik

- **Relevante Anwendungsgebiete:**
  - **Weltraumerkundung:** Kartierung fremder Planeten
  - **Katastrophenhilfe:** Erkundung von Katastrophengebieten zur Lageerfassung
  - **Dynamische Umgebungen:** Navigation in sich ständig verändernden Szenarien

Multiagentensysteme (MAS) bieten vielversprechende Lösungen

Autonome Agenten arbeiten:

- Selbstständig
- Kommunizieren ohne zentrale Steuerung
- Verbessern Effizienz
- Robustheit
- Zeitersparnis

Schlüsselparameter sind:

- Wegfindungs-Algorithmen
- Anzahl der Agenten
- Sensortechnik

Erste Laborergebnisse zeigen signifikante Fortschritte und einen Ausblick auf zukünftige Entwicklungen

Einführung ins Thema

Was ist ein MAS? – Definition

- Ein Multiagentensystem ist ein System aus mehreren selbstständig handelnden Einheiten (Agenten) welche mittels Kommunikation ein gegebenes individual- oder kollektiv Problem in einer geteilten Umgebung lösen.

- **Vorteile**
  - Gesteigerte Robustheit bei Ausfall einzelner Agenten
  - Einfache Skalierbarkeit durch flexibles Hinzufügen von Agenten

Ziel der Projektarbeit: Entwicklung eines Multiagentensystems zur kooperativen Erkundung einer 2D-Simulationsumgebung

Explorations-Algorithmen im Überblick

- Effizienz der Kartierung abhängig vom Explorations-Algorithmus
- Zwei grundlegend verschiedene Implementierungen:

Einfacher nicht-informierter Ansatz

Informierter, zielgerichteter Ansatz

Explorations-Algorithmen im Überblick

Vorgehen
Agent bewegt sich zufällig bis:
- Hindernis
- Zeitlimit

Ändert dann zufällig seine Richtung

Intelligenz
Agent berücksichtigt nicht, ob:
- Bereich bereits bekannt
- schon befahren wurde

Auswahl-Kriterium
- Basis-Referenz zum Messen von Effizienzgewinn

Vorgehen
Umgebung in drei Bereiche unterteilt:
- Bekanntes Gebiet
- Unbekanntes Gebiet
- Frontiers

Gezielte Bewegung zu Frontiers zur effizienten Erkundung

Intelligenz
- Strategische Zielauswahl
- Kommunikation
- Routenfindung

Auswahl-Kriterium
- Etablierter und fundamentaler Ansatz
- Weniger komplex als neuere Verfahren

FBE – Der Erkundungszyklus

Karten-Update: Agent nimmt Umgebung wahr und aktualisiert lokale Karte

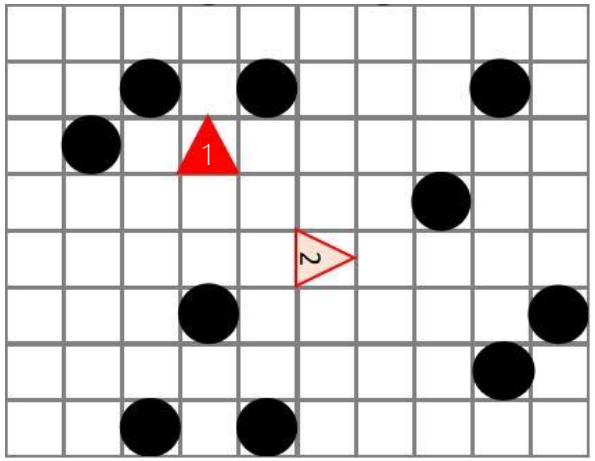
Grenzen-Update: Neue Frontiers werden identifiziert

Grenzen-Auswahl: Frontiers werden bewertet, "bestes" Ziel wird ausgewählt

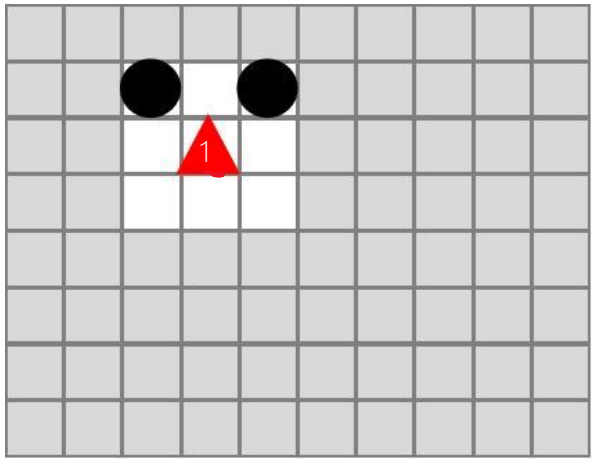
Pfadfindung: Agent plant Route zum Ziel (A*-Algorithmus)

Bewegung: Agent bewegt sich in Richtung des Ziels

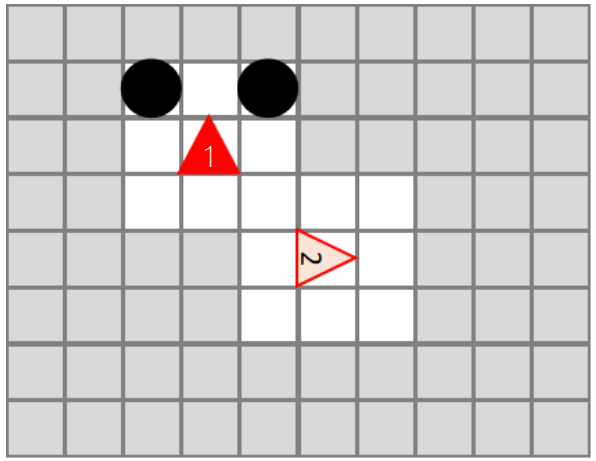
Globales Grid



Lokales Gedächtnis



Geteiltes Gedächtnis



FBE – Die Grenzauswahl

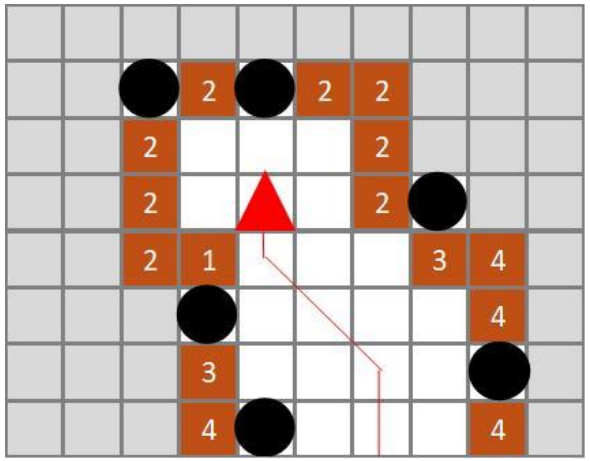
Grenzauswahl entsprechend Kostenfunktion:

$$C(f_i) = \lambda_{Distanz} * m_{Distanz}(f_i) - \lambda_{Größe} * m_{Größe}(f_i) + \lambda_{Orientierung} * m_{Orientierung}(f_i)$$

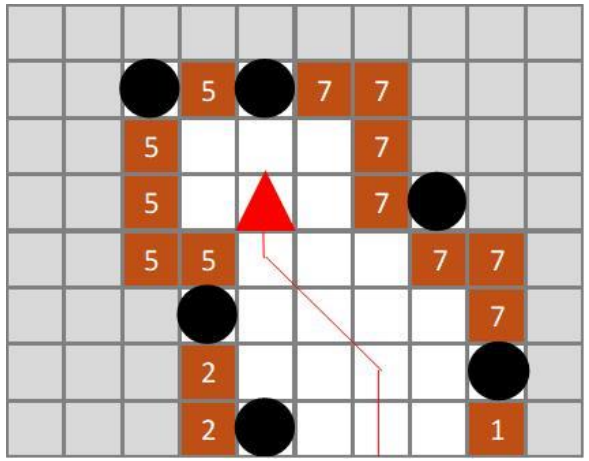
Bewertungskriterien:

- **Distanz:** Abstand Agent zu Frontier
- **Größe (Länge):** Anzahl zusammenhängender Frontier-Zellen
- **Orientierung:** Ausrichtung des Agenten zum Frontier

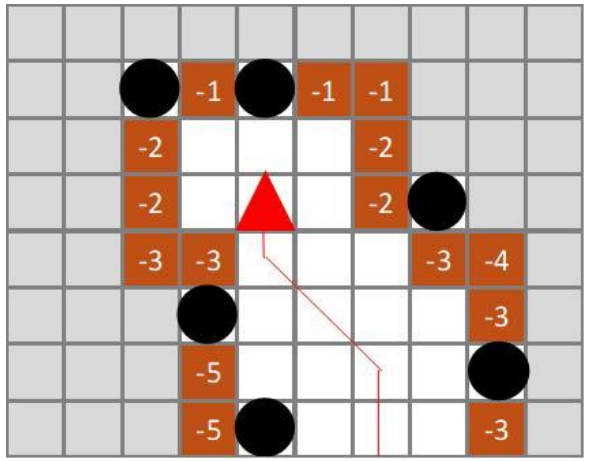
Distanz



Größe



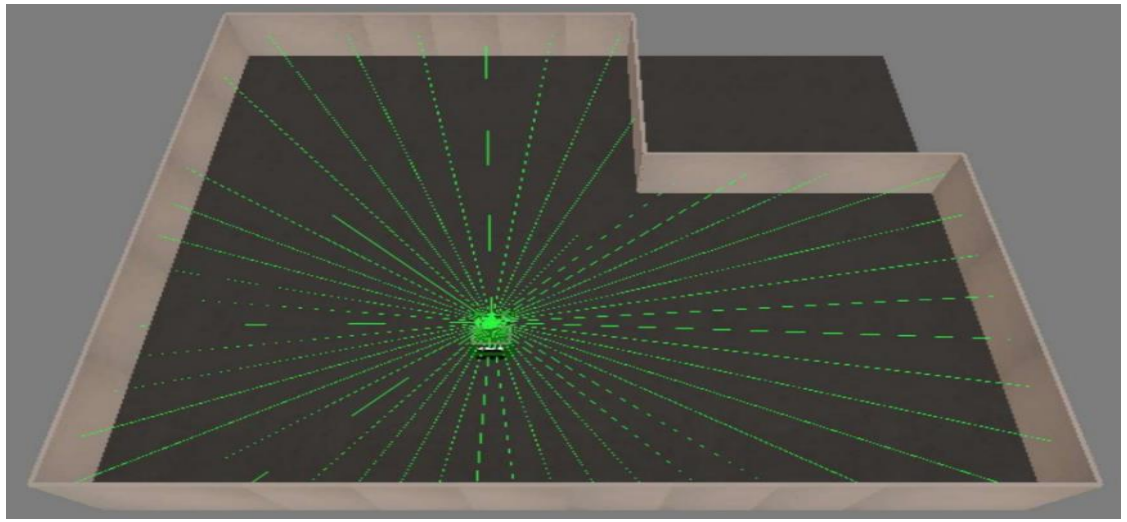
Orientierung

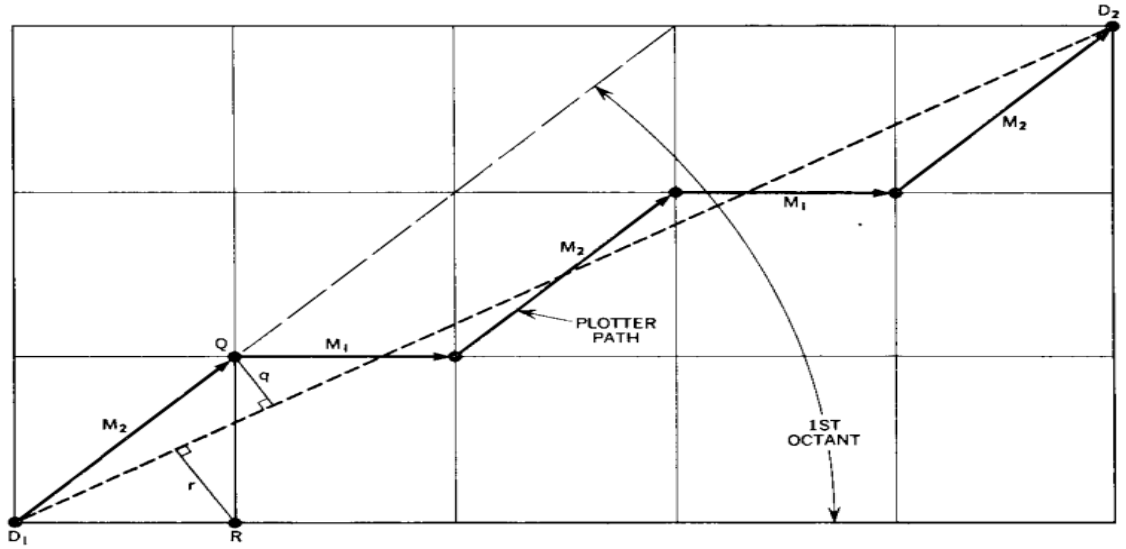


Umgebungswahrnehmung: Raycasting, Bresenham

Raycasting / LiDAR:
Simuliert Aussenden von "Strahlen", um Distanz zu Hindernissen zu messen

Bresenham-Algorithmus:
Approximierte Darstellung der "Strahlen" des Raycastings auf einem 2D-Grid





Pfadfindung: A*

Finden des optimalen Wegs zur ausgewählten Frontier

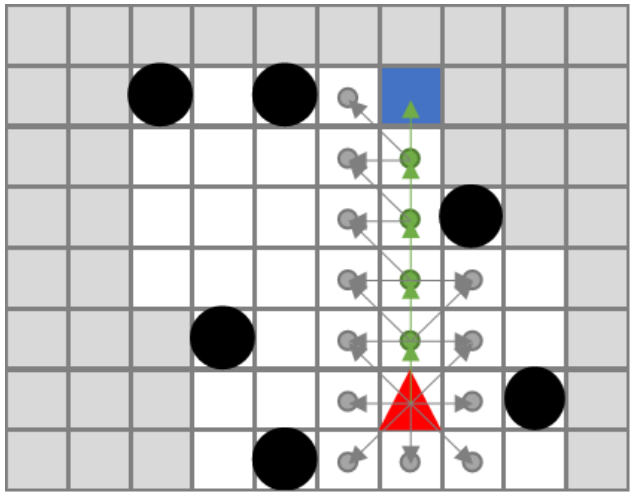
Informierte Suche:

- Erweiterung des Dijkstra-Algorithmus
- Nutzt zusätzlich eine Heuristik (Schätzfunktion)

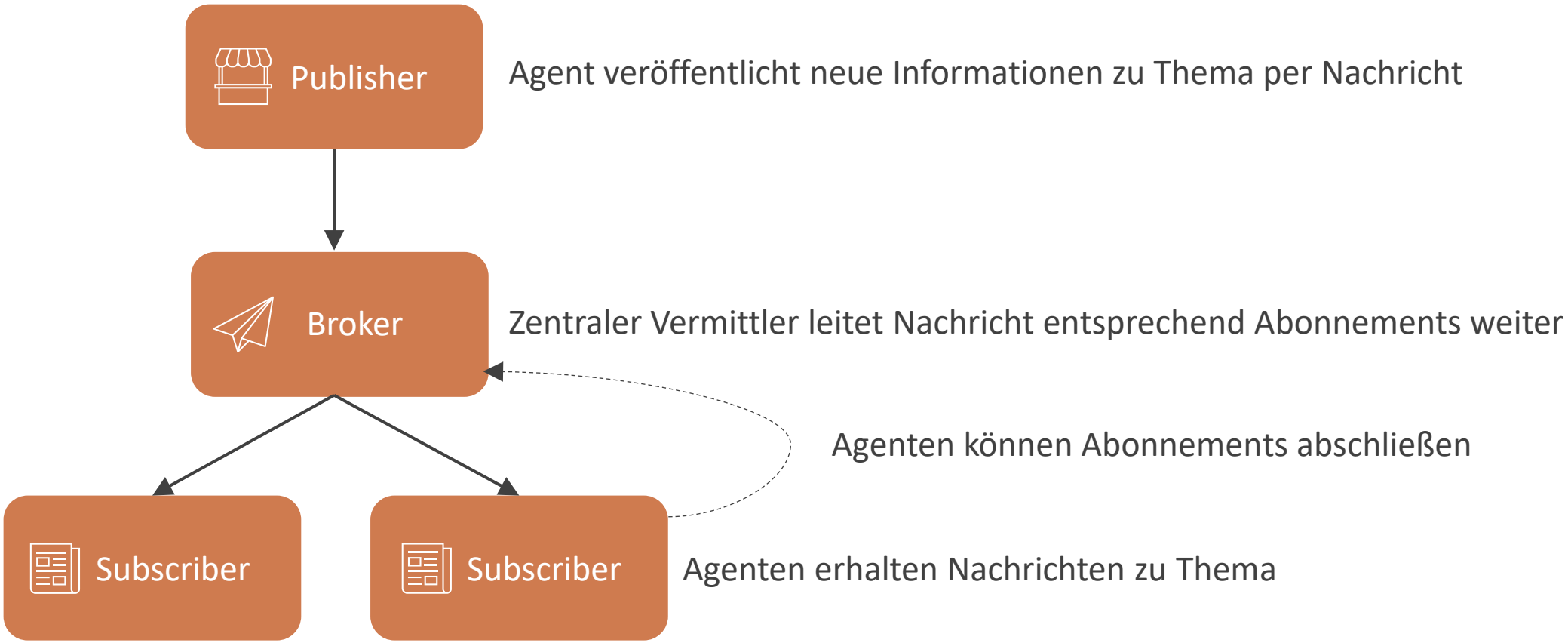
Kostenfunktion

$$f(x) = g(x) + h(x)$$

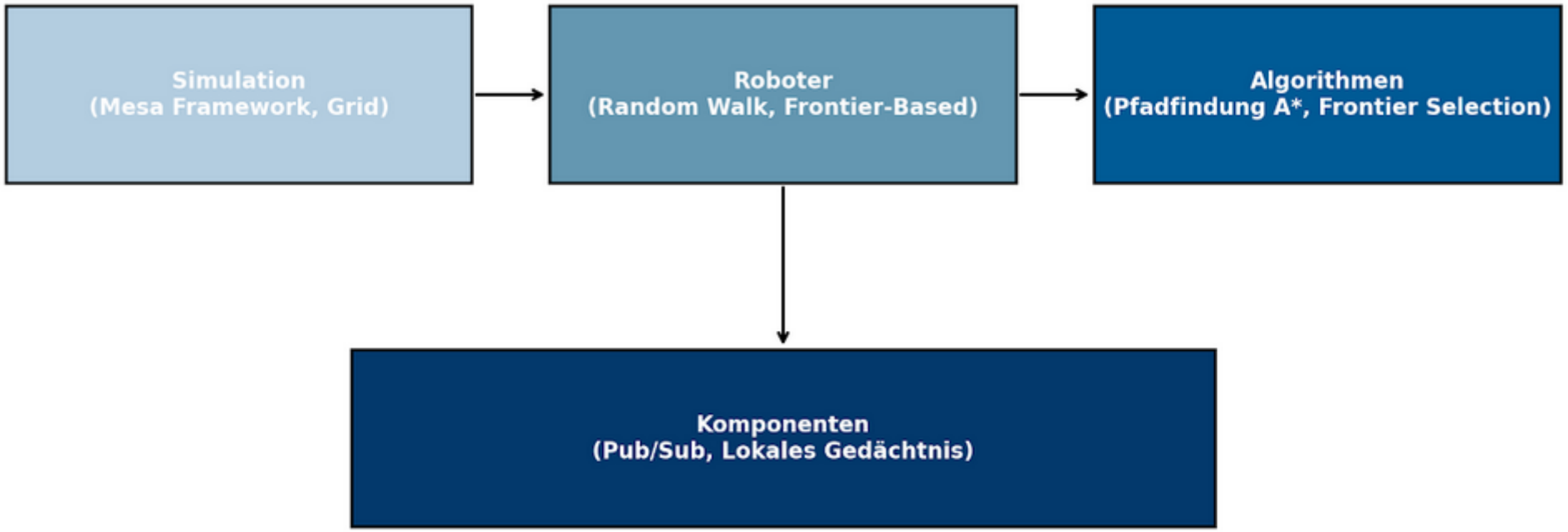
$g(x)$: tatsächliche Kosten vom Start zum x
$h(x)$: geschätzten Kosten von x zum Ziel (Heuristik), hier Euklidische Distanz



Kommunikation: Publish-Subscribe



Architektur & Tech-Stack



- Factories: einfache Erweiterung neuer Algorithmen
- Lose Kopplung: Publish/Subscribe Kommunikation
- Trennung von Logik & Simulation
- Lokales & geteiltes Gedächtnis für Effizienz

Rahmenbedingungen

Veränderliche Parameter

- Anzahl Roboter
- Kantenlänge Grid in Feldern
- Roboter Algorithmen
- Sichtweite in Feldern
- Sichtfeld in Grad
- Seed
- Gewichtungsfaktor der Distanz des Roboters zu Grenze
- Gewichtungsfaktor der Länge der Grenze

Randbedingungen

- 2D-Moore-Grid
- Collision Boundary
- Multigrid-Systems
- Euklidische Distanz
- Andere Agenten gelten als Hindernisse

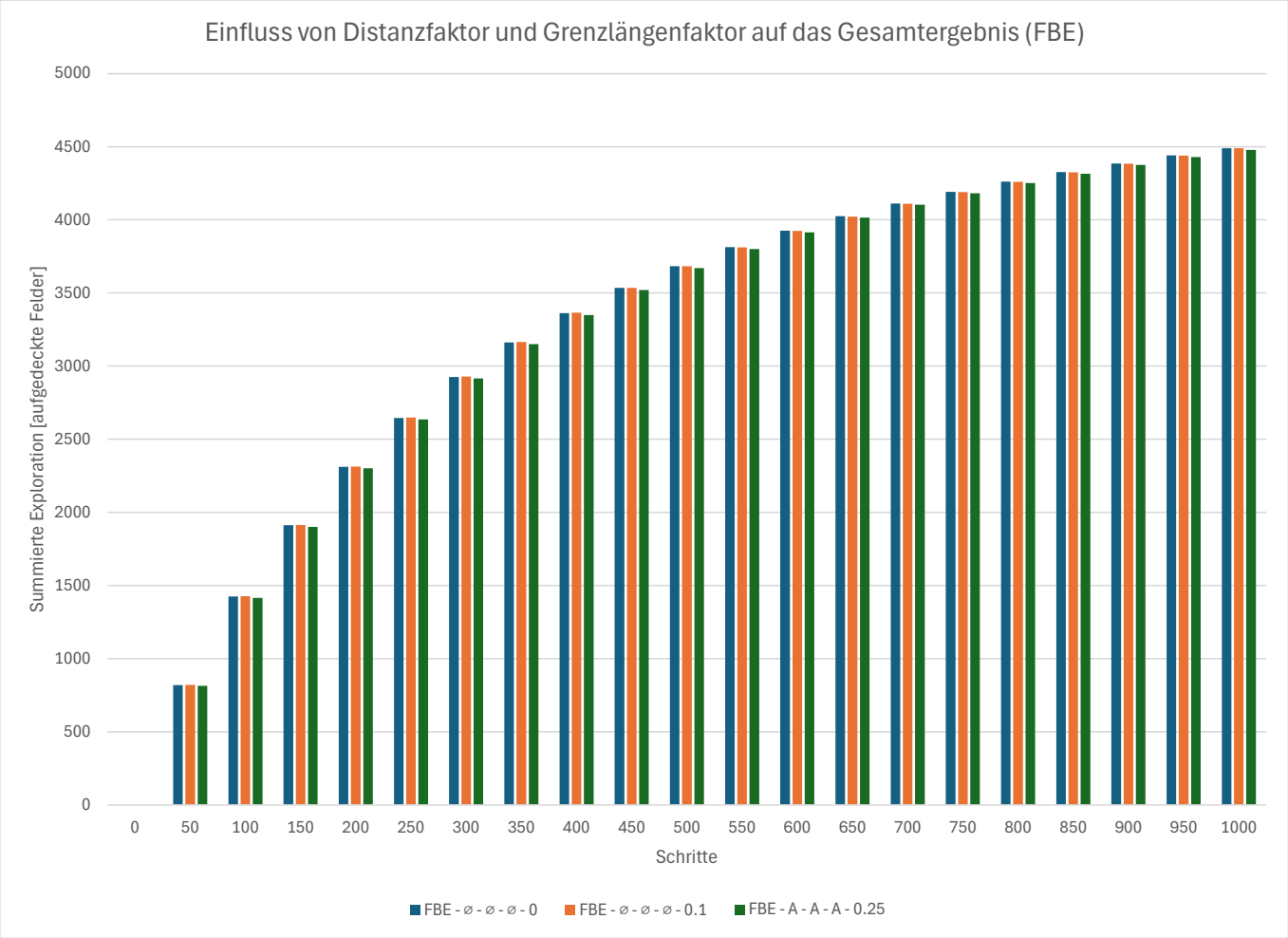
Namenskonvention

Robot Type - Grid Größe - Anzahl der Roboter – Sichtweite des Roboters (Radius) – Sichtfeld in Grad (Angle) – Gewichtungsfaktor Frontiergröße

- Bsp.: FBR – 100 – 9 – 1 – 180 – 0,1
- Gewichtungsfaktor für die Distanz zum Frontier wird nicht gesondert erwähnt

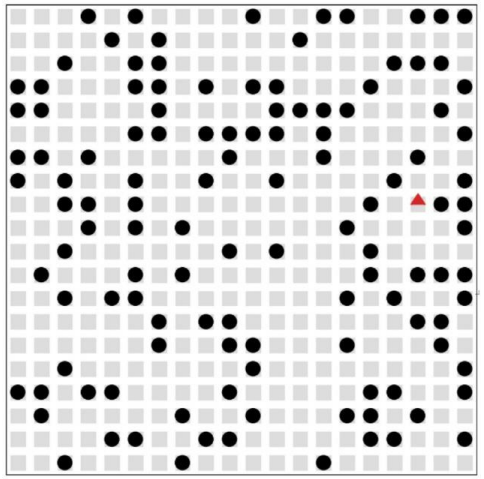
Simulation

Ergebnis



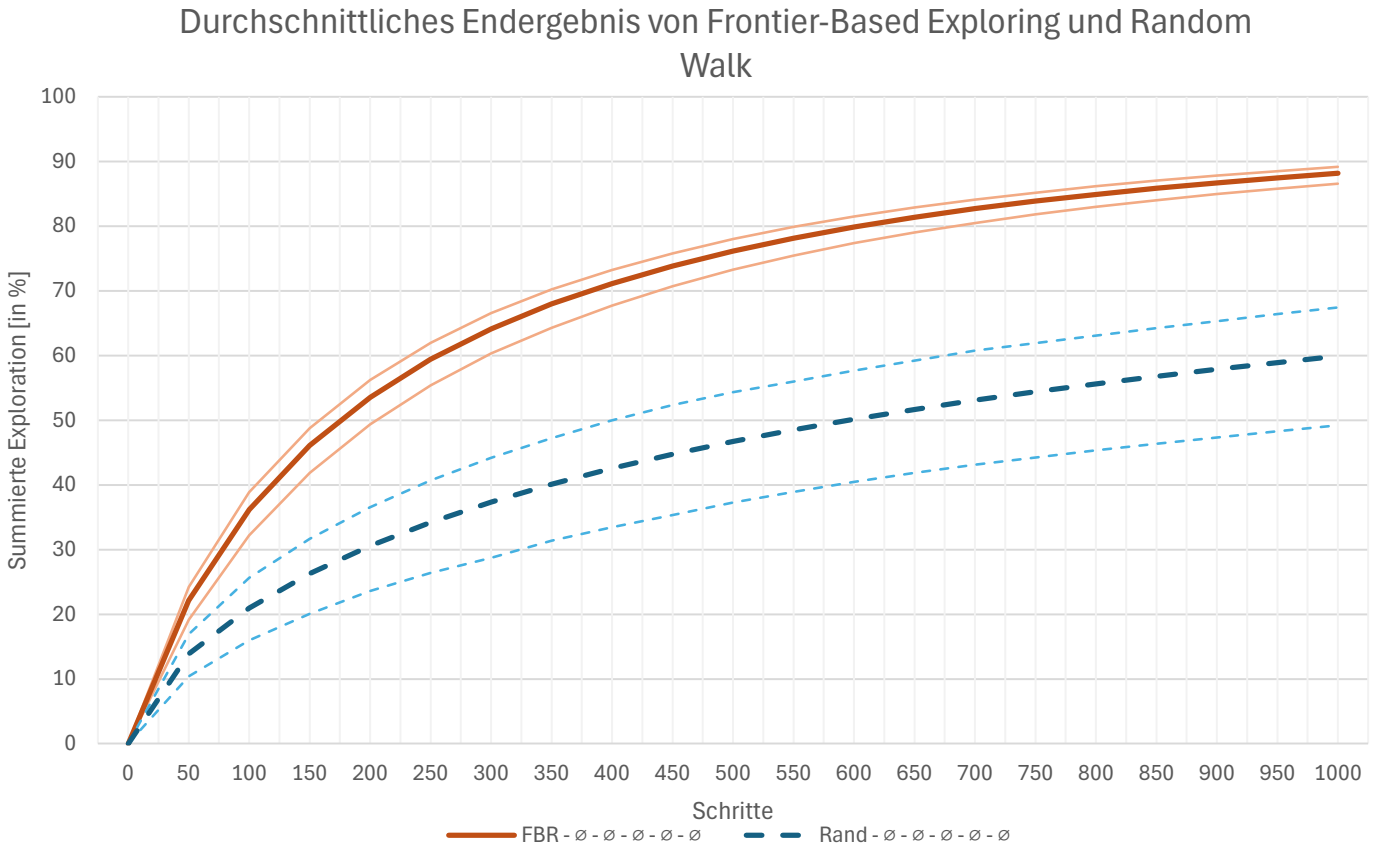
- Unterschiedliche Parameter für Frontier-Based
  - Länge Grenze
  - Distanz Grenze

- Keinen signifikanten Unterschied



- Wenige zusammenhängende Grenzen
  - Algorithmus Grenzen mit der Länge 1

Ergebnis

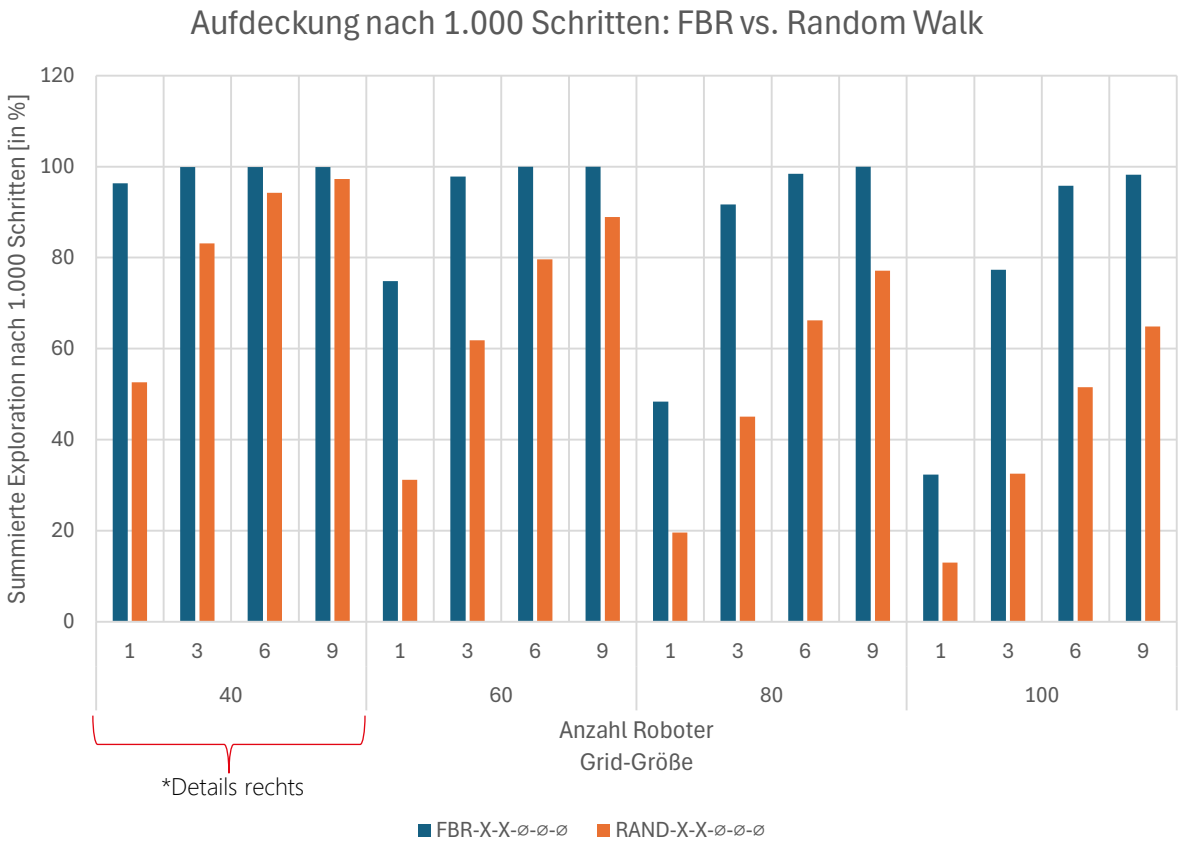


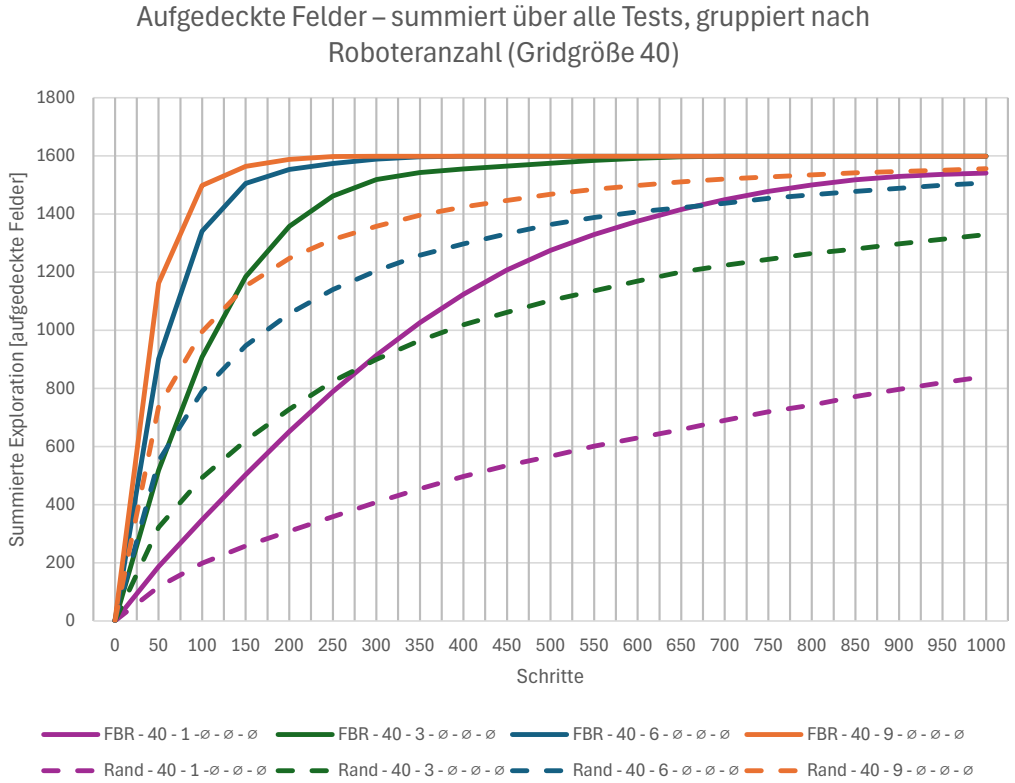
- Durchschnittliche summierte aufgedeckte Felder in Prozent
- Gesamtheit aller Tests aufgeteilt
- Frontier-Based Roboter vs. Random Walk Roboter

- Verbesserung von 19,1 %
- Streuung -18,2 %

	Min [%]	Durchschnitt [%]	Max [%]
FBR - ø - ø - ø - ø - ø	86,5	88,2	89,2
Streuung FBR	- 1,6	2,6	1,0
Rand - ø - ø - ø - ø - ø	49,2	59,9	67,4
Streuung Random	- 10,7	18,2	7,5

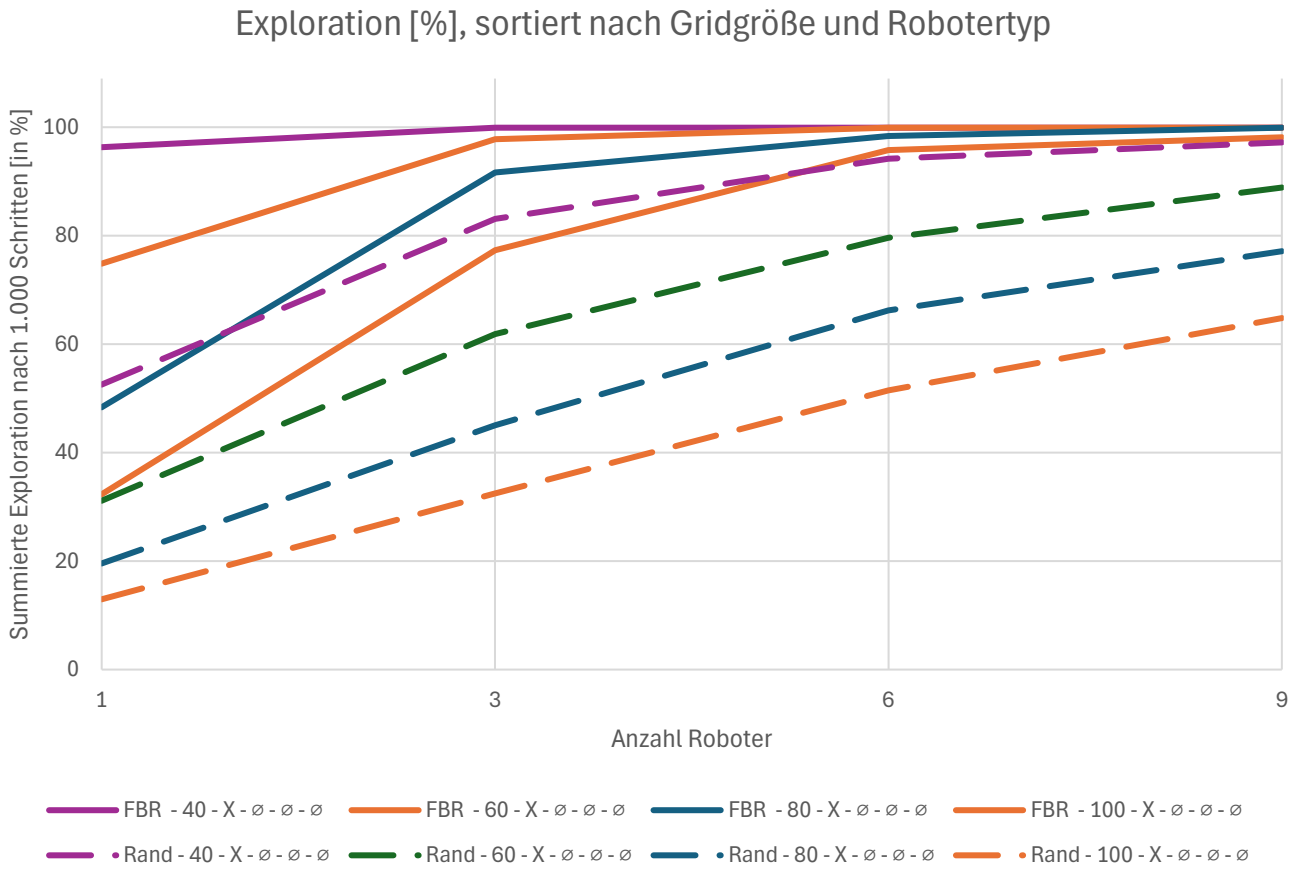
Ergebnis





- Differenz nur 2,66 % nach 1.000 Schritten
  - FBR nach 150 Schritten - 1.564 aufgedeckte Felder
  - Random nach 1.000 Schritten - 1.555 aufgedeckte Felder
- Verbesserung von 19,1 %
- Zeitersparnis von 567 %

Ergebnis



- Durchschnittliche summierte aufgedeckte Felder in Prozent
- Nach 1.000 Schritten
- Frontier-Based Roboter vs. Random Walk Roboter

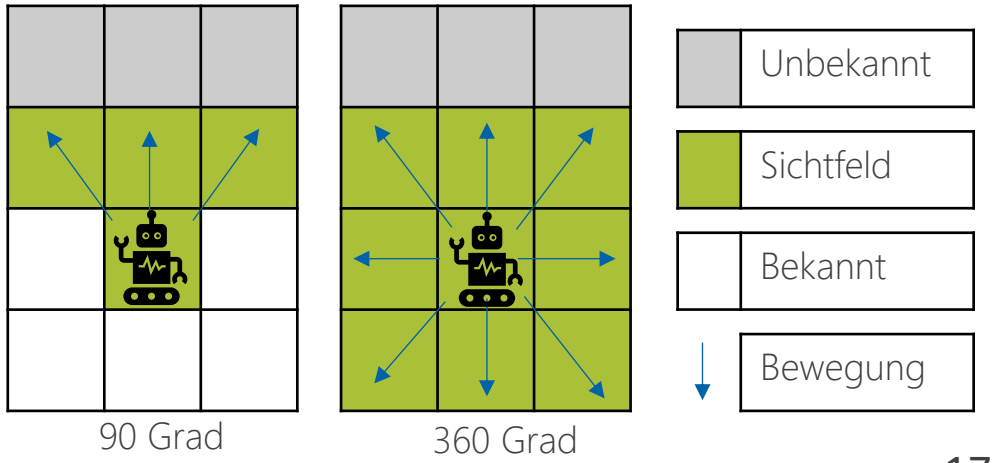
- Random Walk nahezu Lineare Steigerung
  - 10 % je zusätzlichem Roboter
- FBR mehr als 6 Roboter keine Verbesserung
  - Bei großen Grids 5 % je zusätzlichem Roboter

Ergebnis

- Neuaufdeckungen je Schritt
- Verschiedene Sichtwinkel

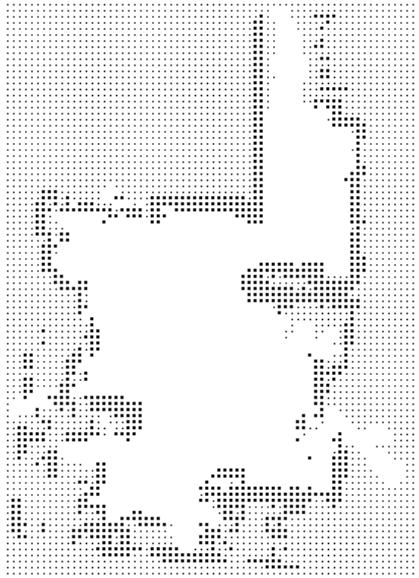
- Beginn mehr Felder unbekannt als bekannt
  - Dauer der Simulation kehrt sich dieses Prinzip um

- Random Walk: Abfall bei einem 360 ° Sichtfelds
  - Bewegungsraum wird vergrößert
  - Bewegung nur im Sichtfeld

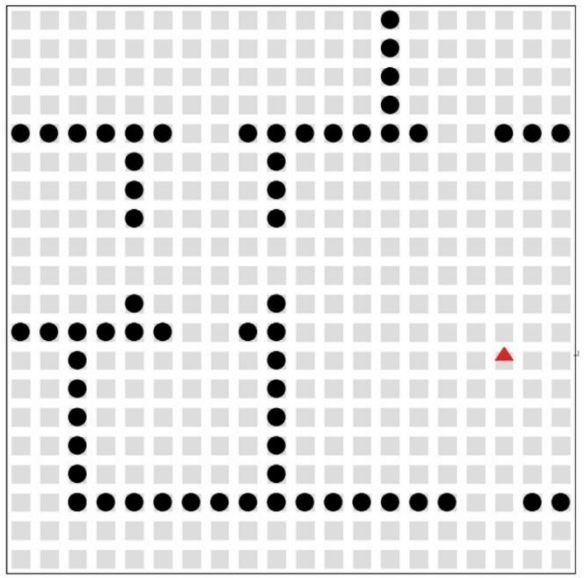


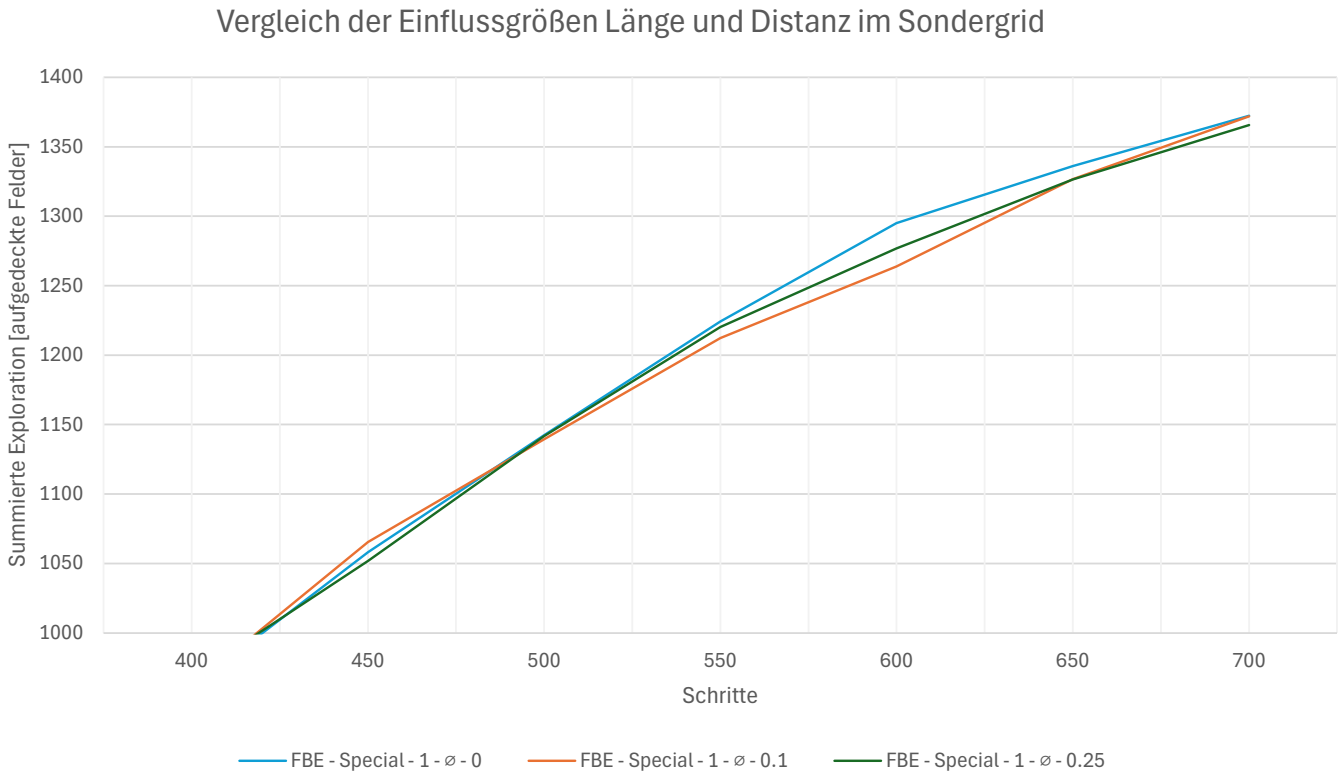
Ergebnis

Literatur



Special





- Manuell erstellte Karte (Special)
- Nachahmung der Vorlage
- „Wohnungs“-Idee

- Kein Unterschied
  - Max. 2 Felder
  - Im Laufe der Simulation legalisiert
- Widerspricht Literatur
  - Weitere Untersuchungen nötig

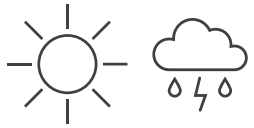
Herausforderungen

Viele aktuelle Multiagentensysteme basieren auf dem Ansatz einer Kontrollinstanz
Umstellung nur über längeren Zeitraum

Forschungsergebnisse nur schwer auf reale Szenarien übertragbar

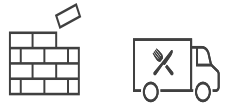
Beispiele:

Unterschiedliche Untergründe

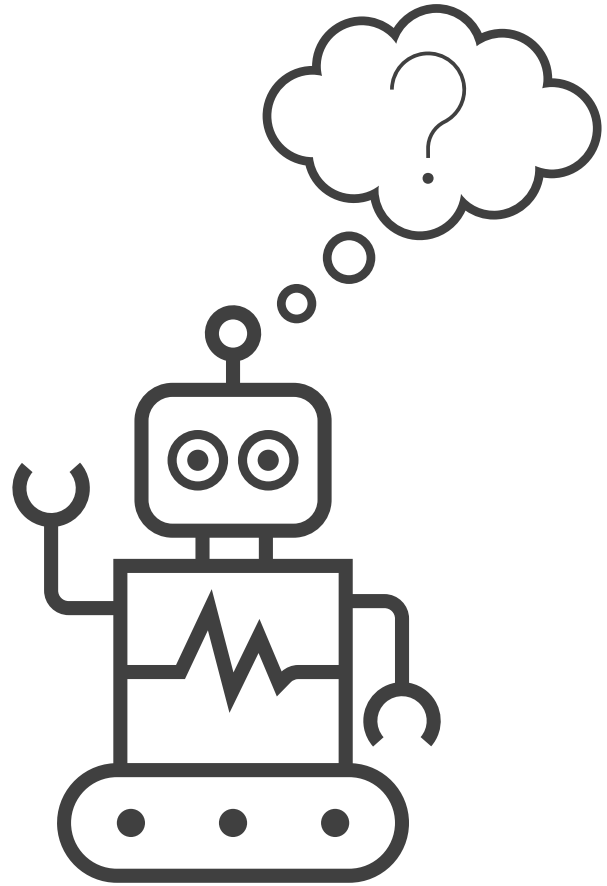


Variierende Wetterbedingungen

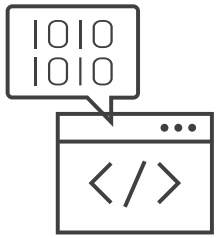
Kommunikationsqualität



Bewegliche Hindernisse

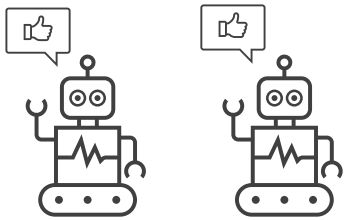


Aktuelle Forschung



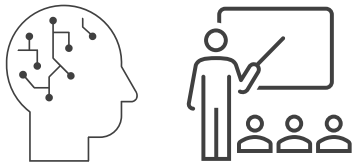
Integration und Entwicklung von MAS

- Erweiterung agentenorientierter Programmiersprachen
- Debugging- und Verifikationstechniken
  - Model Checking



Effizientere Kommunikation und Kooperation

- Kommunikationsstrategien
- Zusammenarbeit von Agenten



Multi-Agent Reinforcement Learning

- Deep Reinforcement Learning
  - Modellierung komplexer Aufgaben
  - Steuerung von Roboterschwärmen

- Reinforcement Learning
  - Verbesserung von Lern- und Entscheidungsfähigkeiten

Projektabschluss

Idee

MAS

Random vs. Frontierbased

TECHNISCHE UMSETZTUNG

Python

Mesa Framework

SIMULATION

Anzahl Roboter
Grid-Größe
Roboter
Algorithmen
Sichtweite
Sichtfeld
Seed
Faktor Distanz
Faktor Länge

ERGEBNISSE

Erhöhte Effizienz

Geringere Streuung

Zeitersparnis

AUSSICHT

Kooperations-strategien

Realistischere Sensorik & Kommunikation

Deep Reinforcement Learning

Vielen Dank für Ihre Aufmerksamkeit

Wir freuen uns auf Ihre Fragen und den gemeinsamen Austausch