Task 3 – Genetische Algorithmen

Alex, Yaroslav, Manuel

EvoTest

30.11.2016

Letzter Stand

- Funktionierende Umwelt
  - Logik
  - Grafische Ausgabe
- Größtenteils funktionierender Parkassistent:
  1. Berechne den Zielpunkt → Funktionierte
  2. Berechne die Fahrspur dorthin → Funktionierte sofern das Fahrzeug nicht zu dicht an der Parklücke stand
  3. Berechne aus der Fahrspur die Steuersignale → Funktionierte approximativ
  4. Berechne aus den Steuersignalen die Trajektorie (inklusive Distanzberechnung und Kollisionserkennung) → Funktionierte

Bugfixing

1. Fahrspurberechnung
   - Problem: Geometrische Berechnung suchte iterativ einen bestimmten Punkt auf einer Halbgeraden, die diesen Punkt nicht enthielt
   - Lösung: Parkvorgang auf diesem Weg nicht möglich $\rightarrow$ Parke nicht direkt ein sondern suche zuerst einen Punkt, von dem aus das einparken geht
2. Steuersignalberechnung
   - Problem: Iterative Berechnung $\rightarrow$ Statt Kreissehne wird eine Tangente gefahren $\rightarrow$ Je größer das Berechnungsintervall, desto größer die Abweichung von der Fahrspur (s. Abbildung 1)
   - Lösung: Kreissehne implementiert

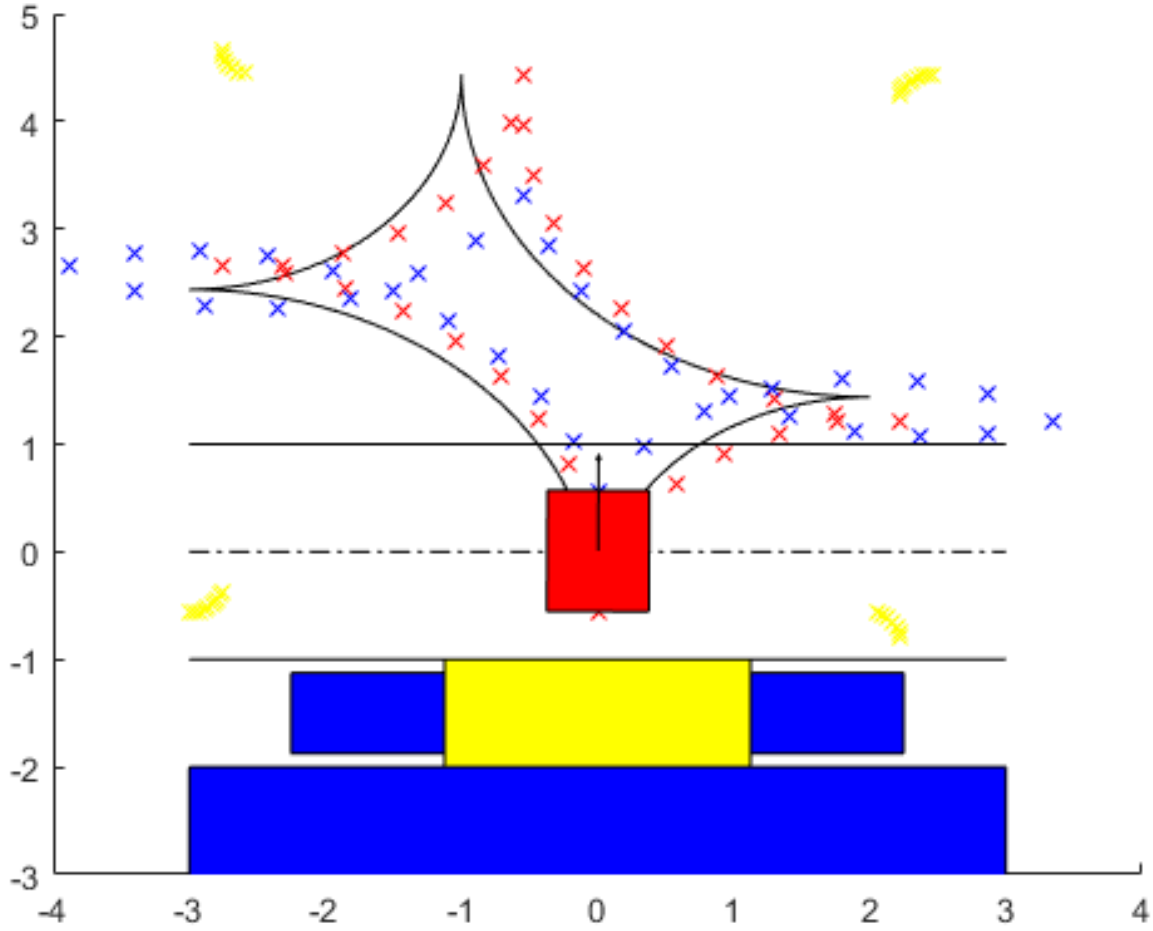

Abbildung: Fahrspur (schwarz) und tatsächliche Positionen des Fahrzeugs (rot: Hinterachse, blau: Vorderachse)

Genetische Algorithmen: Implementierung GA

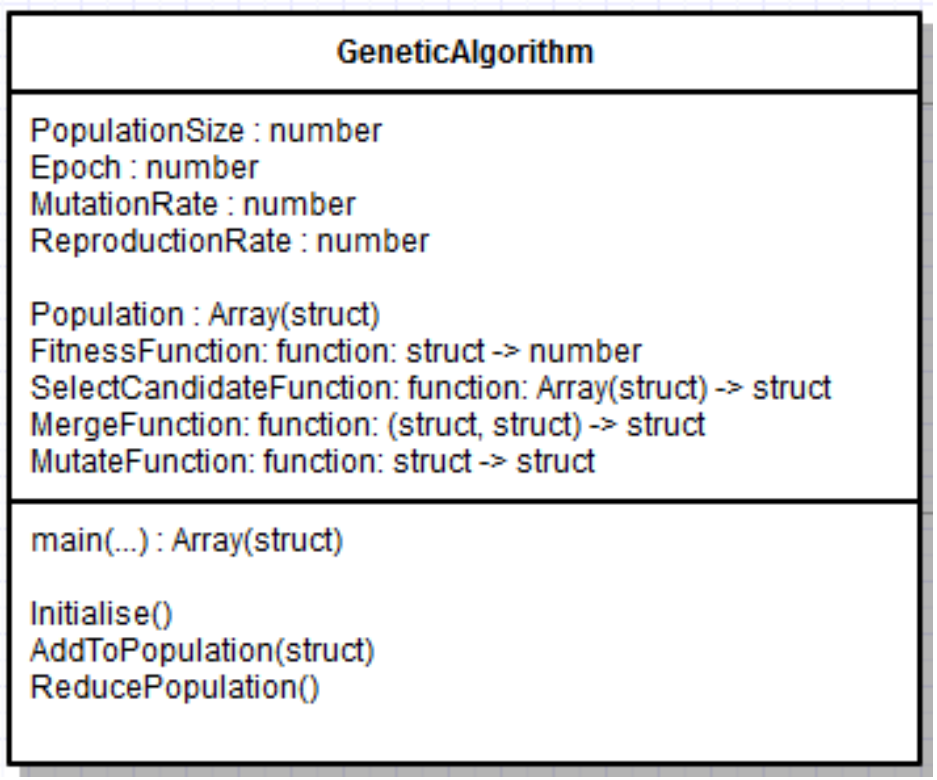

Abbildung: Klassenkarte Genetischer Algorithmus

- PopulationSize: Größe der Population
- Epoch: Akutelle Epoche
- MutationRate: Mutations Rate
- ReproductionRate: Prozentsatz der Populationsgröße, der zu generieren ist pro Epoche
- FitnessFunction: Function-Handle der Fitnessfunktion
- SelectCandidateFunction: Function-Handle der Selektionsfunktion
- MergeFunction: Function-Handle der Crossover-Funktion
- MutateFunction: Function-Handle der Mutationsfunktion

Genetische Algorithmen: Implementierung Chromosom

Chromosome enthalten alle Werte, mit denen ein Testszenario eindeutig erzeugt werden kann.

- Position $_{\text{Fahrzeug}}$: $(-7.5, -1) \leq (X, Y) \leq (7.5, 4)$ *
- Orientierung $_{\text{Fahrzeug}}$: $0 \leq \text{Angle} \leq 2 \cdot \pi$
- Größe $_{\text{Parkplatz}}$: $(2.25, 1) \leq (L, D) \leq (5, 2)$ *

Durch diese Definition ermöglichen wir links und rechts Einpark-Szenarios sowie andere für vorwärts und rückwärts Einparken. Die Werte werden alle als 8-Bit Integer gespeichert und in die für sie vorgesehene Spanne interpoliert.

*: elementwise - $\leq$

Implementierung: GA-Funktionen

- Selektion:
  1. Standard: $p(\text{chr}) = \frac{\text{Fitness}_{\text{chr}}}{\sum_{c \in \text{Population}} \text{Fitness}_c}$
  2. Alternativ: $p(\text{chr}) = \frac{1}{\text{PopulationSize}}$
- Crossover: Wähle zufällig Werte der “Eltern“
- Mutation: Jedes Bit wird mit Wahrscheinlichkeit entsprechend der MutationRate geflippt

- Fitness. Kombination durch multiplikation von einzelnen Fitness-Werten:
  1. Simpel:

$$\text{Fitness} = \frac{1}{1 + \text{minDistance}}$$

  2. Mindestabstand $dist_{desired}$:

$$\text{Fitness} = \frac{dist_{desired}}{\text{minDistance}}$$

  3. Minimaler Parkplatz (L, D):

$$\text{Fitness} = \frac{1}{L + D}$$

  4. Minimaler Abstand zu Simulationsbeginn $dist_{start}$:

$$\text{Fitness} = \frac{1}{1 + dist_{start}}$$

Probleme

- Derzeit werden bei PopulationSize = 10 und ReproductionRate = 50% pro Sekunde nur ungefähr 5 Epochen durchgeführt
- Durch den Greedy-Ansatz, der die Population nicht ersetzt, sondern die besten n Chromosome behält, konvergiert der Algorithmus relativ schnell gegen relativ gleiche n Ergebnisse