42TIN2450 | Research project

Keuzetrack: AI & Robotics

Self-driving car

Omschrijving

Tijdens dit project wordt er in team een prototype voor zelfrijdende wagentjes geïmplementeerd. De wagentjes moeten autonoom een pad kunnen bepalen om te navigeren van punt A naar punt B, op een gekend routenetwerk. Meer details hierover volgen verder in dit document.
In eerste instantie wordt er puur vanuit een simulatie omgeving gewerkt. Mogelijk wordt dit in een later stadium ook gekoppeld aan een fysieke testopstelling. Er moet in ieder geval steeds rekening gehouden worden met het feit dat de wagentjes in beide omgevingen (virtueel en fysiek) moeten functioneren.
De wagentjes dienen aangestuurd te worden met ROS.

Omgeving

Simulatie

De simulatie omgeving die we aanraden, maakt gebruik van WeBots, een robot simulatie framework dat nog maar recent gratis en *open source* geworden is. We hebben deze omgeving reeds voorzien voor jullie, maar jullie mogen zeker nieuwe testomgevingen creëren om specifieke scenario's te kunnen testen. De koppeling tussen ROS en de WeBots simulatie omgeving wordt ook gedemonstreerd in een eenvoudig voorbeeld dat we ter beschikking zullen stellen.

Het wagentje dat in de demo gebruikt wordt, is voorzien van een aantal motoren voor de aansturing en een frontcamera waarvan de beelden opgevraagd kunnen worden. Uiteraard mogen jullie ook hier experimenteren met andere types, configuraties, …

Routenetwerk

Zoals eerder gezegd, is het netwerk van berijdbare routes van tevoren vastgelegd en gekend voor de software. Het netwerk kan voorgesteld worden als een *graaf*. Grafen zullen ook aan bod komen in de lessen van het vak *AI & Robotics*.
Vanuit een bepaalde node kan de robot telkens enkel naar aangrenzende nodes bewegen. Specifieke informatie over het routenetwerk en de manier waarop deze gegevens opgeslagen zijn, is terug te vinden in de documentatie.

Autonome navigatie

Path planning

Een pad van node A naar node B wordt berekend door te bepalen langs welke nodes het wagentje moet rijden om in node B te geraken. Het wagentje moet standaard altijd de kortste route kiezen en moet de nodige berekeningen zo snel mogelijk kunnen uitvoeren, zodat er geen *delay* merkbaar is tussen het geven van een commando en het moment dat het wagentje vertrekt.

Path Execution

Nadat er een pad bepaald is van node A naar node B, moet het wagentje dit traject afleggen. In de meeste gevallen zal voor zo'n pad dus een reeks van nodes berekend worden die één voor één bezocht moeten worden, om uiteindelijk het eindpunt te bereiken. Zoals eerder gezegd zijn paden tussen twee verbonden nodes altijd recht. In eerste instantie raden we aan om het wagentje strak van node naar node te laten rijden. Nadien kunnen er eventueel optimalisaties toegevoegd worden om het pad van het wagentje 'realistischer' te maken. (bv. afgeronde hoeken)

Obstacle Avoidance

Op de berijdbare weg kunnen zich ook hindernissen bevinden. Bepaalde nodes kunnen daarom afgesloten worden om aan te duiden dat er niet langs gereden kan worden. (*roadblocks*) In dat geval moet het wagentje (in *real time*) een alternatieve route berekenen om toch in het eindpunt te geraken. Het wagentje krijgt een bericht dat een node is afgesloten.

Object Recognition

Tenslotte moeten de wagentjes ook de *camera feed* gebruiken om een set van eenvoudige verkeersborden te herkennen. Hiervoor raden we het gebruik van de OpenCV library aan. Deze library biedt heel wat functies aan voor beeldverwerking en volstaat voor het *vision* gedeelte van dit project. Er moeten hier dus nog geen geavanceerde AI/Computer Vision technieken aangewend worden.

De verkeersborden die minstens herkend moeten worden:

- een verkeersbord dat aangeeft dat een bepaalde straat (= *edge* van de graaf) afgezet is. (m.a.w. het wagentje mag er niet door rijden)
- een verkeerslicht dat rood is: het wagentje moet stoppen tot er een groen verkeerslicht getoond wordt
- een verkeerslicht dat groen is: het wagentje mag weer vertrekken

Extra's

Extra's kunnen geïmplementeerd worden om je als team te onderscheiden. We geven hieronder enkele voorbeelden/ideeën, maar in overleg met de lectoren kunnen er zeker ook

andere voorstellen als extra toegevoegd worden. Onderzoek zelf op welke manieren de genoemde voorbeelden uitgevoerd zouden kunnen worden.

- realistischere beweging van het wagentje
- fysieke robot (bv. *TurtleBot*) aansturen in echte omgeving i.p.v. enkel simulatie
- variatie in bewegingspatronen voor paden met gelijke lengte
- bewegende of onverwachte hindernissen ontwijken
- camera gebruiken om commando's aan het wagentje te geven
- meerdere niveaus in de omgeving ondersteunen (brug, tunnel, …)
- meerdere te bezoeken locaties aan het wagentje geven, waarna het wagentje de meest efficiënte route berekent om alle locaties te bezoeken (*Travelling Salesman Problem)*
- simulatie runnen op workstations in *AI & Robotics Lab*, ROS node lokaal (ROS over netwerk)
- meer realistische modellen gebruiken in de simulatie: https://cyberbotics.com/doc/automobile/introduction
- verschillende toegelaten snelheden op verschillende wegen, ook keuze voor 'snelste' route

Referenties

GitHub repo: https://github.com/PXLAIRobotics/researchproject1819

ROS: http://www.ros.org/
WeBots robot simulator: https://cyberbotics.com/
OpenCV: https://opencv.org/
Allerlei WeBots tutorials: https://cyberbotics.com/doc/guide/
TODO tutorial WeBots + ROS