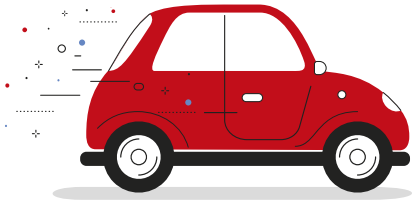
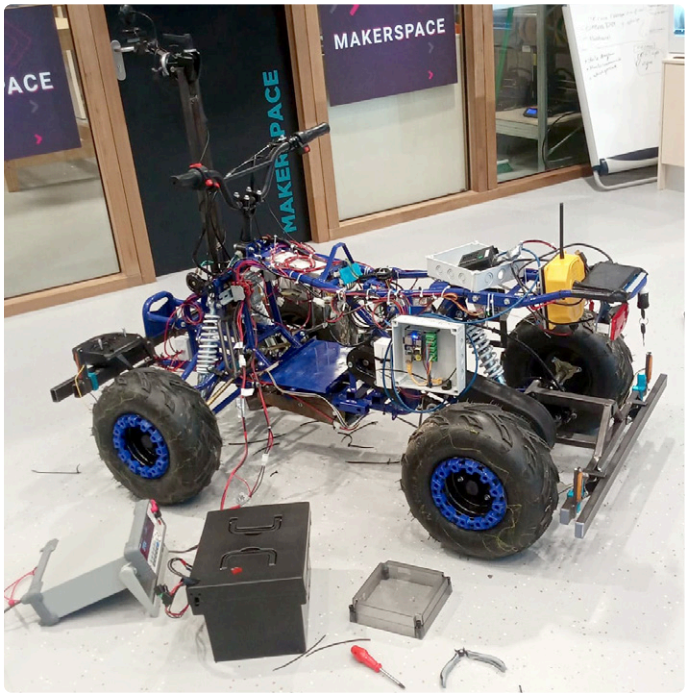
Autonoom rijden op basis van AI



De Self Driving Challenge 2024 van de RDW

Door Edwin van den Oetelaar, Sieuwe Elferink en Teade Punter (Nederland)

Zelfrijdende auto's beloven veiligere wegen, meer onafhankelijkheid voor mensen met een handicap en efficiënter, milieuvriendelijker verkeer. Ontdek hoe een team van een Nederlandse universiteit deze mogelijkheden in de praktijk heeft gebracht en met hun autonome voertuig de eerste plaats behaalde in de Self Driving Challenge van de RDW.

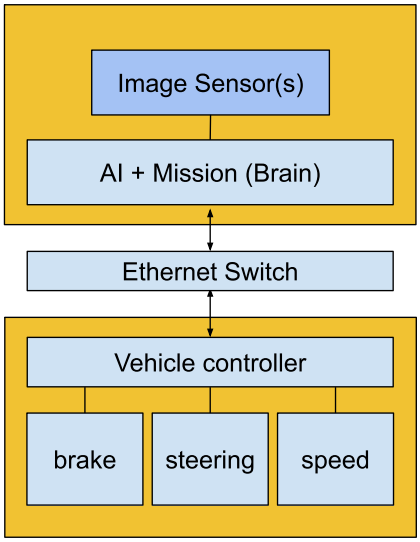


Fontys Quad (open categorie).

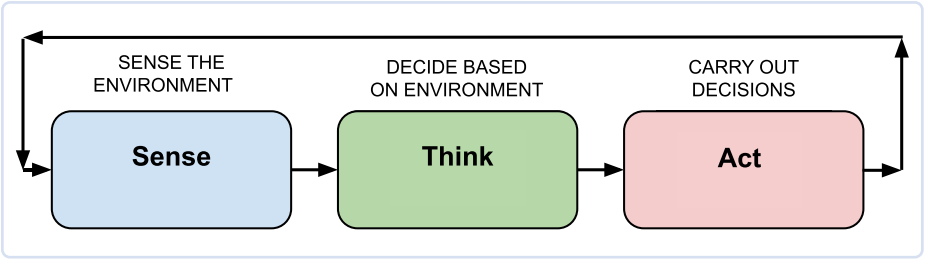
Elk jaar organiseert de Nederlandse wegautoriteit RDW de Self Driving Challenge (SDC). De SDC is een competitie voor Nederlandse universiteiten en hun studenten om een zelfrijdende oplossing te bouwen voor een auto-achtig voertuig. Het is bedoeld om kennis op te doen over complexe technologie voor autonome besluitvorming in een auto. Elk jaar wordt een nieuw doel gesteld voor de challenge en krijgen deelnemende studententeams de opdracht om software voor hun voertuig te ontwikkelen om dit doel zo goed mogelijk te behalen. Het voertuig moet over een aangewezen parcours navigeren, verkeersborden volgen, stoppen bij verkeerslichten en parkeren in gemarkeerde vakken.

De challenge heeft twee categorieën: ***closed***, waarin deelnemers een door RDW geleverde elektrische kart met een standaard Intel i5 computer gebruiken, en ***open***, waarin teams hun eigen oplossing mochten bedenken. **Figuur 1** laat voorbeelden van voertuigen in elke categorie zien. Team Fontys won de SDC ***open*** categorie in 2024. De open categorie stelde ons in staat ons eigen voertuig te ontwikkelen met custom hardware en software. Op deze manier wilden we graag de mogelijkheden van machinelearning-gebaseerde softwareontwikkeling door onze studenten verkennen. De ***open*** categorie maakte het mogelijk

Figuur 1. Voertuigtypes in de challenge van 2024: Renault Twizy (open categorie), RDW kart (closed categorie).



Figuur 2. Gelaagde voertuigarchitectuur (soft en hard realtime).



Figuur 3. Onze simpele "Sense, Think, Act"-oplossing.

om te focussen op machinelearning-gebaseerde oplossingen in plaats van klassieke algoritmen, wat meer vrijheid gaf om te experimenteren. Omdat we hoge processor eisen verwachtten, kozen we voor het NVIDIA Orin platform met CUDA-versnelling [2] op Linux in plaats van de standaard i5-setup. We hebben een eigen quad ontworpen om onze eigen elektronica en mechanische systemen beter te verkennen. Dit artikel doet verslag van ons ontwerp en onze ervaringen bij het ontwikkelen van onze auto.

Besturingen

We hebben ons systeem en ontwikkelgroep opgesplitst in twee delen: *Vehicle Controller* (snelheid, sturen, odometrie, low latency 1000 Hz regellus) en het *Brain* (mission control, planning en sensorverwerking, onafhankelijk in een tragere besturingskring). Beide waren verbonden via betrouwbaar bekabeld 100 Mbit Ethernet, wat onze versie van een ruggengraat is. Een blokschema van het systeem wordt getoond in **Figuur 2**.

Onze SDC-voertuigbesturing kende vier grote uitdagingen: sturen, snelheidscontrole, remmen en communicatie. De *Vehicle Controller* was gebaseerd op een ESP32, verbonden via bekabeld 100 Mbit Ethernet. Voor signaalintegriteit maakten we printen met optische-isolatie. Ook implementeerden we een PID-regelsysteem binnen de motorbesturing en stuurbesturing, en integreerden we een protocolstack die fail-safe mechanismen ondersteunt zodat het altijd betrouwbaar werkt. Sturen was de grootste technische uitdaging, waardoor we verschillende oplossingen hebben getest. Eerdere pogingen omvatten roterende servo's, lineaire actuatoren en zelfs een stuurbekrachtigingssysteem van een Suzuki Alto, maar deze waren te zwak of te langzaam. Uiteindelijk ontwikkelden we een hoogkoppel-motorsysteem met een geïntegreerde versnellingsbak, waarbij een stevig metalen tandwiel direct op de stuurkolom is gemonteerd. Dit bood het benodigde vermogen en de respons om het voertuig goed te kunnen besturen.

Om nauwkeurig sturen te garanderen, integreerden we een MAQ473 Hall-gebaseerde absolute hoeksensor [3] en voegden we mechanische eindstops toe met eindschakelaars die de motorvoeding onderbreken om overmatige-rotatie en schade te voorkomen. Snelheidscontrole werd geregeld met een galvanisch gescheiden PWM-naar 0-5 V-uitgangscircuit dat we zelf hebben ontworpen en gebouwd. Voor communicatie combineerden we bekabeld 100 Mbit Ethernet (met scheidingstransformatoren) met het open-source IP/Zenoh-protocol [4] voor autonome besturing en voegden we een 6-kanaals PPM-gebaseerde RC-invoer toe voor handmatige besturing.

We integreerden een gecertificeerd RF-noodstopsysteem, ontworpen om direct de motorvoeding op hardwareniveau te onderbreken. Dit was een belangrijke veiligheidsmaatregel voor directe uitschakeling bij onverwacht gedrag. Autonoom rijden vereist strikte veiligheidsmaatregelen, en de organisatoren van de challenge stelden een noodstopsysteem verplicht als eis voor deelname.

Voor waarneming kozen we voor een enkele ZED-X stereo dieptecamera [5] in plaats van LiDAR of radar. Onze aanpak is AI-gebaseerd en huidige deep-learningmodellen zijn makkelijker toe te passen op cameradata, wat dit de meest praktische keuze maakt. LiDAR voegt complexiteit en kosten toe, terwijl onze setup eenvoudig en goedkoop blijft—kernprincipes in ons ontwerp.

Mechanische en elektrische aanpassingen

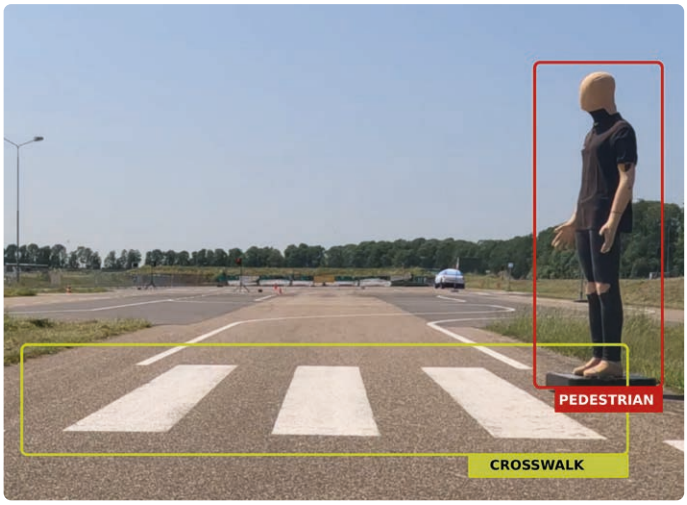
We hebben het voertuig verstevigd met gelaste metalen bumpers aan de voor- en achterkant, voorzien van schakelaars voor botsingsdetectie. De stroom kwam van het originele 48 V DC lead-acid accupakket, waaruit we alle benodigde spanningen genereerden met geïsoleerde DC/DC-converters (24 V voor sturen, 12 V voor de Orin en 5 V voor microcontrollers). Een elektrisch bediende hydraulische rem werd ontworpen en toegevoegd, maar raakte beschadigd tijdens het testen en was daarom niet operationeel tijdens de challenge. We kozen voor elektrisch remmen; we sloten de motorwikkelingen kort van de BLDC-motor om direct te stoppen.

De hardware, elektronica en software zijn parallel ontwikkeld, wat zorgde voor afstemming van de API's om integratie mogelijk te maken. Teamleden waren afhankelijk van elkaar om hun onderdelen af te ronden, met veranderingen en updates door het hele systeem tot aan de laatste dagen voor de challenge.

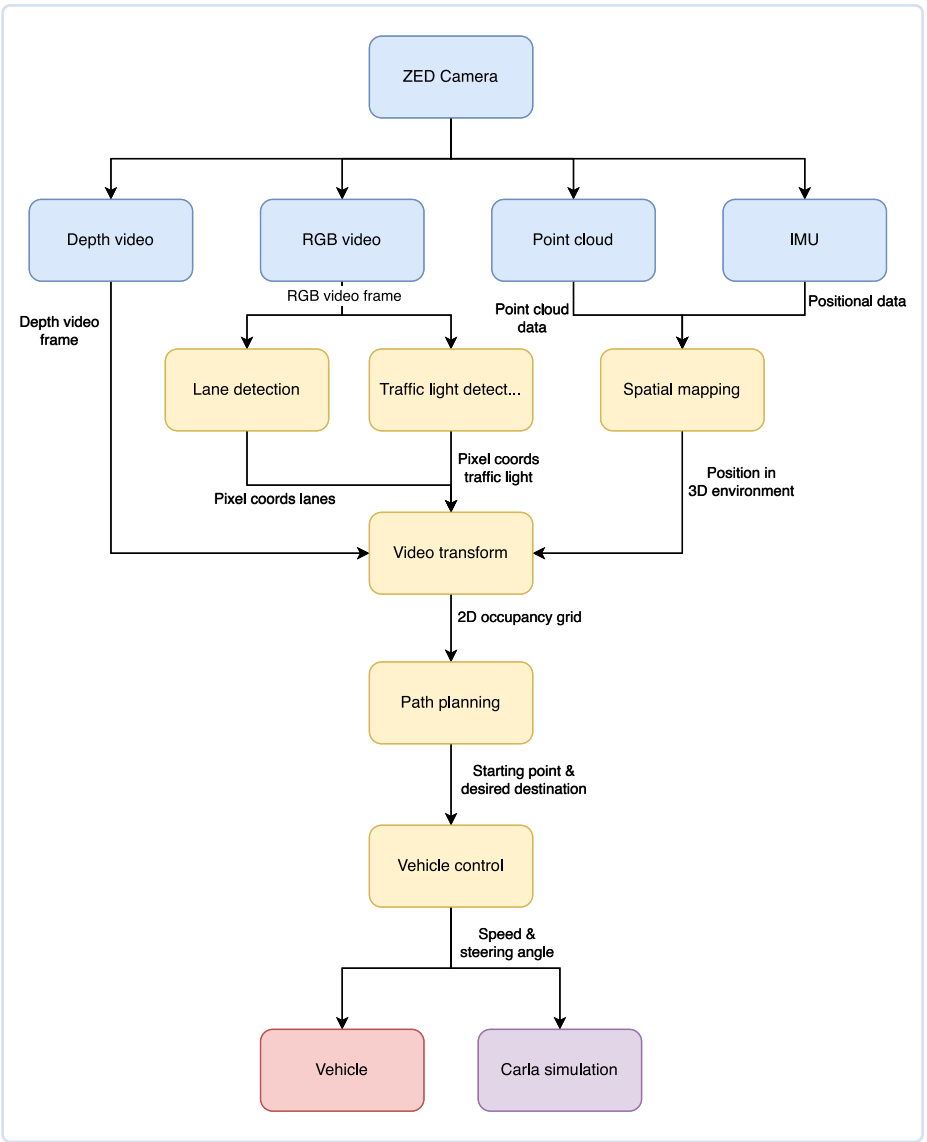
Navigatiesysteem

Het navigatiesysteem is verantwoordelijk voor het vinden van de beste voertuigcommando's om bepaalde doelen te behalen. Voor de SDC zijn deze doelen de uitdagingen die door de RDW zijn opgegeven. Een doel is bijvoorbeeld het volgen van het wegdek of veilig stoppen voor een verkeerslicht. Om deze optimale voertuigcommando's te vinden, kozen we voor de *Sense*, *Think*, *Act*-architectuur (**Figuur 3**), hoewel we de nadelen kennen [6]. Deze veelgebruikte architectuur wordt ook toegepast in gerelateerd roboticaonderzoek zoals bij zelfrijdende voertuigen in [7] en [8].

Deze architectuur splitst navigatie op in drie afzonderlijke taken. De *Sense*-taak verzamelt informatie over de omgeving van het voertuig,



Figuur 4. De ingebouwde voetgangersdetector van de ZED SDK.



Figuur 5. Oorspronkelijk systeemoverzicht.

bijvoorbeeld de afstand tot een voertuig of de locatie van het wegdek. Deze informatie kan komen van allerlei sensoren op of buiten het voertuig zoals camera's of kaartdata. De *Think*-taak berekent het beste pad om de doelen van het navigatiesysteem te halen. De *Act*-taak zorgt ervoor dat het voertuig fysiek het beste pad volgt, bijvoorbeeld via sturen of het gaspedaal. Samen vormen deze modules een navigatie-pijplijn die in een continue lus wordt verwerkt. We behaalden een gecombineerde pijplijn-snelheid van 0,2 s, resulterend in een update-frequentie van 5 Hz.

De sense taak

Dit begint met het vastleggen van data over de omgeving van het voertuig met het ZED X stereo visionsysteem. Dit systeem gebruikt twee camera's voor stereovisie en behaalt diepte-perceptie met minder dan 2 cm nauwkeurigheid in helder buitenlicht. Het levert ook 2K RGB-beelden van de linkercamera, via de ZED SDK, met diepteverwerking op het Jetson Orin-apparaat. Om het systeem simpel te houden, kozen we ervoor geen extra sensoren zoals extra camera's, Lidar of wielodometrie toe te voegen.

De beelddata van de ZED X-camera wordt geanalyseerd met een set parallel draaiende detectoren om autonomie-gerelateerde informatie uit de ruwe beelddata te halen. Al deze detectoren zijn gebaseerd op Machine Learning (ML), omdat ML zich goed aanpast aan allerlei externe factoren zoals schaduwen en bewolking die het beeld veranderen [9]. Traditionele computer vision-technieken focussen op beeldanalyse zonder context en falen als externe factoren niet gecontroleerd kunnen worden zoals bij de SDC.

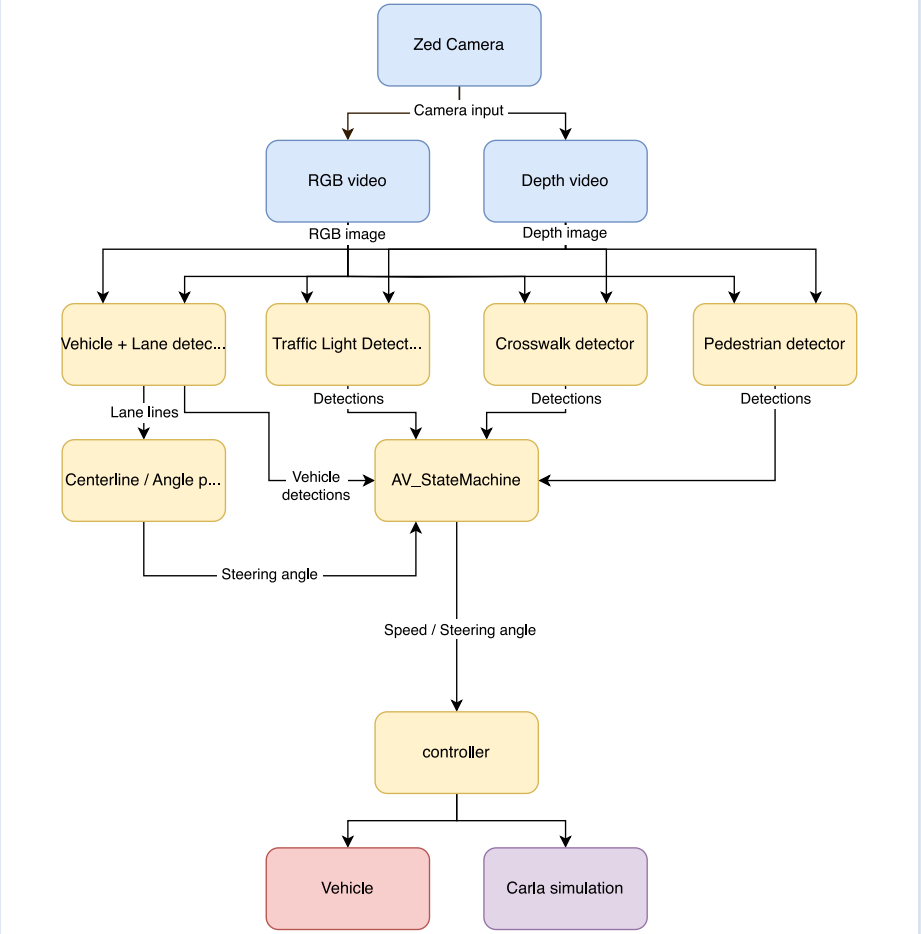
Voetgangersdetectie

Voetgangersdetectie (**Figuur 4**) is gedaan met de ingebouwde voetgangersdetector en tracker van de ZED SDK. Dit gebruikt een ML-model van Stereolabs dat diepte- en RGB-beelden combineert voor volledige positiedetectie van mensen. Zo kun je locatie en afstand van voetgangers bepalen.

Voertuig- en rijstrookdetectie

In ons originele systeemoverzicht (**Figuur 5**) zijn voertuigdetectie en rijstrookdetectie twee aparte modules. Tijdens de implementatie vonden we echter een model genaamd YOLOPv2 dat beide taken combineert in één ML-model dat state of the art resultaten haalt [10]. Daardoor konden we de architectuur aanpassen, de uiteindelijke versie staat in **Figuur 6**.

Het YOLOPv2-model gebruikt de populaire YOLO-architectuur voor de encoder-backbone. Het onderscheidt zich door meerdere



Figuur 6. Definitief systeemoverzicht.

taken te combineren in één model. Dit heet een hydranet en combineert meerdere decoders met dezelfde embeddingsruimte ("heads") om verschillende taken uit te voeren zoals weg-segmentatie, lijn-detectie en obstakeldetectie. Hydranets zijn een trend in de wereld van zelfrijdende voertuigen; Tesla zet hier bijvoorbeeld vol op in met hun nieuwe end-to-end-aanpak [11].

Voertuig- en rijstrookdetectie zijn te zien in **Figuur 7**. De rode lijnen zijn het rijstrookresultaat van het YOLOPv2-model. Het gele kader toont de voertuigdetectie in actie. Dit gele kader representeert de x- en y-coördinaten in het beeld. De afstand tot het voertuig (z) is ook nodig voor veilige navigatie. Om deze afstand te bepalen worden de bounding box-coördinaten van de ML-detectie in het RGB-beeld gebruikt als masker in het dieptebeeld van de ZED X-sensor. Zo kun je de z-coördinaten halen om de afstand te krijgen. De RGB-coördinaten als masker gebruiken werkt, omdat diepte- en RGB-beeld van de ZED X hetzelfde coördinatensysteem en oorsprong hebben.

Verkeerslichten

De verkeerslichtdetectie (**Figuur 8**) wordt uitgevoerd met een YOLOv8 ML-model uit [12]. Dit model kan de locatie (x, y) en status van het verkeerslicht detecteren. De afstand tot het verkeerslicht (z) werd bepaald met een bounding box masker over de dieptekaart. Het model leverde ruwe detecties. Om een verkeerslicht over meerdere frames te volgen was een tracker nodig. Een KCF (Kernelized Correlation Filters) tracker werd gebruikt om elk verkeerslicht een ID te geven, wat nodig is voor goede planning. Nuttige voorbeelden staan bij [13].

Zebrapaden

Een geschikt zebrapad-detectiemodel vinden was lastig. In plaats daarvan is een eigen model getraind op basis van de YOLOv8-architectuur met de Ultralytics-bibliotheek [14]. Een dataset is gemaakt met testvideo's van het parcours. Een dataset maken van één locatie is normaal slechte praktijk omdat dit overfitting geeft [15]. Voor de SDC bleek overfitting hier juist handig, omdat het voertuig maar één specifieke locatie hoeft te rijden. Afstand tot het zebrapad (z) werd berekend met een bounding box masker over de dieptekaart.

Figuur 7. YOLOPv2-lijndetectie plus onze stuurvectorberekening.

Figuur 8. YOLOv8 Rode verkeerslichtdetector met weergave van het betrouwbaarheidsniveau.

Figuur 9. YOLOv8 verkeersborddetectie.

Tot slot werd verkeersborddetectie gedaan met een ander YOLOv8-model. Dit model is getraind op een grote dataset van Europese verkeersborden. Het geeft de locatie (x, y) en het bordtype. Bijvoorbeeld, het kan onderscheid maken tussen een 10 km/h en een 20 km/h bord, zoals te zien in **Figuur 9**. Afstand tot het verkeersbord (z) werd bepaald met een bounding box masker over de dieptekaart.

De think taak

De Think-taak krijgt informatie zoals de locatie van objecten zoals verkeerslichten of andere voertuigen om beslissingen te nemen. Oorspronkelijk werd een occupancy grid planner gemaakt door alle informatie uit het sense-model samen te voegen in een "wereldmodel". Maar een accuraat wereldmodel maken bleek lastig met ons kleine team en beperkte tijd. Daarom werd een simpelere planner gebruikt die beslissingen per frame maakt zonder verleden of toekomst te gebruiken. Het systeem werkt door een state-machine die hoog-niveau gedrag definieert te combineren met een centerline tracker om op een berijdbaar oppervlak te blijven.

De centerline tracker werkt door eerst de rijstrookgrenzen te zoeken door vanaf het midden naar buiten te scannen, en dan het midden ertussen te berekenen. Dit midden wordt gebruikt om een hoek te bepalen tussen het midden van het voertuig en het midden van de rijstrook. Deze hoek wordt aangepast op basis van rijstrookpositie en gezichtsveld (FOV), en helpt de juiste stuurhoek te schatten om gecentreerd te blijven.

De state-machine definieert hoog-niveau gedrag met toestanden zoals `AV_DRIVING`, `AV_WAITING` en `AV_CROSSWALK`. Elke toestand en overgang heeft bijbehorend gedrag. Bijvoorbeeld, `AV_DRIVING` houdt een constante snelheid aan, gegeven door de verkeersborddetector en gebruikt de centerline tracker om op de weg te blijven. Overgang van `AV_DRIVING` naar `AV_CROSSWALK` vereist remmen tot 0 km/h. In `AV_CROSSWALK` wacht het voertuig tot de voetganger wegloopt. Daarna gaat het voertuig weer naar `AV_DRIVING`, wat gas geven betekent om weer op de snelheidslimiet te komen. De state-machine bevat zes verschillende toestanden met overgangen om het gedrag van het voertuig te bepalen in de verschillende SDC-uitdagingen.

Act

De Think-taak geeft gewenste voertuigcommando's door in de vorm van een gewenste snelheid en stuurhoek om de doelen van het navigatiesysteem veilig te behalen. De Act-taak voert deze gewenste commando's daadwerkelijk uit op het voertuig. Deze verantwoordelijkheid ligt volledig bij de low-level vehicle controller. Het navigatiesysteem stuurt de gewenste snelheid en stuurhoek over het Zenoh-protocol naar de low-level controller, die de acties uitvoert en de status van het voertuig terug-rapporteert.

Groot succes, met een paar problemen

Een aantal van de veiligheids- en low-level logica-onderdelen zijn afzonderlijk getest en gevalideerd met de CARLA-simulator [16]. Na succesvolle tests werd het hele systeem geëvalueerd tijdens de SDC-finale op het RDW-testparcours in Lelystad (**Figuur 10**). Het voertuig presteerde goed in de meeste uitdagingen, en werd uiteindelijk eerste. Functies zoals afstandsbediening (RC) hielpen het voertuig snel te verplaatsen na een mislukte poging. Ook de state-machine liet correct gedrag zien binnen elke toestand en overgang. Toch waren er ook enkele beperkingen; die bespreken we hieronder.

- De ML-gebaseerde detectoren van de sense-taak gaven soms false positives of false negatives. Vooral het publiek rond het parcours tijdens de finale, dat er tijdens de tests niet was, was lastig. De stoplichtdetector gaf bijvoorbeeld valse rode lichtdetecties op basis van rode kleding van toeschouwers. Dit leidde tot verkeerde toestands-overgangen, zoals niet naar de `AV_DRIVING`-toestand gaan als het licht op groen sprong. Vanwege betrouwbaarheidsproblemen is de verkeersborddetector niet gebruikt; in plaats daarvan werd 15 km/h op rechte stukken en 5 km/h in bochten gereden.
- Sommige ML-detectoren hadden een hoge inference-tijd (reactietijd) [17], wat leidde tot een trage update. Vooral de verkeerslichtdetector was traag, waardoor de besturing niet vloeiend was en het voertuig bijvoorbeeld niet in de rijstrook bleef.
- Het eenvoudige planningssysteem met centerline tracker kon in complexe situaties niet op koers blijven. Het werkte goed op

Figuur 10. Tijdens de finale stopt ons voertuig bij het rode licht.

het rechte stuk met twee lichte bochten, maar bij splitsingen en bochten vertraagde en versnelde het voertuig onregelmatig en ging het uit de rijstrook. Ook was stoppen voor het verkeerslicht op de juiste plek vaak een probleem door een te simpele afbouwfunctie.
- Het draadloze veiligheidssysteem had te weinig bereik, waardoor het voertuig voortijdig stopte.
- De centerline tracker en rijstrookdetector waren gevoelig voor de camerapositie op het voertuig, waardoor de ZED X meerdere keren verplaatst moest worden met een niet-optimale montage tot gevolg.

Aanbevelingen voor verbetering

Het huidige systeem vormt een solide basis, zoals blijkt uit de winst van de 2024 SDC open-categorie. Op basis van testdata en resultaten op de dag van de challenge hebben we een lijst aanbevelingen voor volgende SDC-edities. We verwachten dat deze aanbevelingen ons systeem verbeteren.

De eerste twee problemen hierboven zijn beide gerelateerd aan prestatiebeperkingen van de ML-detectoren. Reactietijd kan worden verbeterd door nieuwere modelarchitecturen te gebruiken met kleinere modellen en vergelijkbare nauwkeurigheid. Foute detecties kunnen worden verbeterd door bestaande modellen te fine-tunen op meer representatieve data. Deze updates kunnen het navigatiesysteem verbeteren, maar we verwachten vooral winst door het huidige per-frame planningssysteem van de Think-taak (zie punt drie) te verbeteren. Door temporale informatie op te nemen in een wereldmodel wordt de besturing soepeler en consistenter. Ook kun je met het global-model valse detecties filteren, wat nu eenmaal bij ML hoort [18].

Een planningssysteem op basis van een occupancy grid kan gebruikt worden: alle informatie uit de Sense-module wordt in één wereldmodel gebracht, geprojecteerd op een grid waarin elke cel een kostenwaarde heeft. Een planningsalgoritme plant een route met de laagste kosten om de navigatiedoelen te optimaliseren. Zo'n occupancy grid maakt het navigatiesysteem robuuster doordat verleden en heden samen gebruikt worden, in plaats van alleen de huidige tijdstap. Dit biedt sterkere navigatie in moeilijke situaties.

Probleem 4 kan worden opgelost door het noodstopsysteem te vervangen door een versie met minimaal 400 m bereik. De camerapositie (probleem 5) kan worden opgelost met een nieuwe camerabeugel die in de rijrichting verstelbaar is.

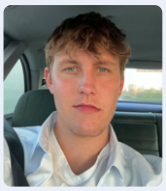
Richting nieuwere systemen

Belangrijke spelers zoals Tesla bewegen steeds meer richting leer-gebaseerde in plaats van regel-gebaseerde beslissingen. Werk als [19] en [20] vervangt de volledige Sense-taak door een enkele hydranet ML om een robuust global-model en occupancy grid te maken. Ook end-to-end methoden die de hele Sense, Think, Act-architectuur vervangen door één hydranet ML-model (zoals die van NVIDIA) laten zien dat betrouwbaar autonoom rijden mogelijk is in veel scenario's [21][22][23]. Toekomstige SDC-teams kunnen deze trends onderzoeken om het huidige expliciete systeem sterk te verbeteren. In dit artikel hebben we de ingrediënten gegeven voor een redelijke kans op succes. Maar het blijft veel werk, expertise en inzet van het team.

240677-03

Over de Auteur

Edwin van den Oetelaar is technicus en docent aan Fontys ICT in Nederland. Hij maakt deel uit van de onderzoeksgroep High Tech Embedded Software onder leiding van prof. Teade Punter. Hij is specialist in hardware- en softwareontwikkeling en werkt graag samen met studenten aan uitdagende projecten zoals zelfrijdende auto's en robots.



Sieuwe Elferink is gastonderzoeker bij de onderzoeksgroep High Tech Embedded Software en oprichter/eigenaar van MultiRotorResearch / MRR Drones. Hij behaalde een master in robotica en AI, met specialisatie in slimme navigatiesystemen en AI-gebaseerde besluitvorming. Zijn onderzoek richt zich zowel op autonome drones als zelfrijdende voertuigen.

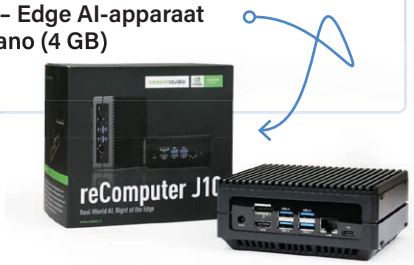
Teade Punter is lector High Tech Embedded Software aan Fontys ICT in Nederland. Zijn onderzoeksgroep doet toegepast onderzoek naar data-engineering, digital twinning en AI voor slimme systeemontwikkeling, met toepassingen in smart industry en demontage, zelfrijdende laboratoria voor chemie, energiesystemen en autonoom rijden.

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de redactie van Elektor via redactie@elektor.com.

Gerelateerd Product

- **reComputer J1020 v2 – Edge AI-apparaat met NVIDIA Jetson Nano (4 GB)**
  www.elektor.nl/20968



WEBLINKS

[1] Wikipedia, Citron / CC-BY-SA-3.0: https://commons.wikimedia.org/wiki/File:Renault_Twizy.jpg
[2] "Jetson Orin AGX: High-performance AI computing for edge devices," 2024, NVIDIA: https://tinyurl.com/nvidia-jetson-orin
[3] "MAQ473: 9-Bit to 14-Bit MagAlpha Automotive Angle Sensor with ABZ Incremental and PWM Outputs," 2024, Monolithic Power Systems: https://www.monolithicpower.com/en/products/automotive-aecq-grade/sensors/maq473.html
[4] Eclipse Foundation, "Eclipse Zenoh: Zero overhead pub/sub, store/query and compute," 2024, GitHub: https://github.com/eclipse-zenoh/zenoh
[5] ZED X: Industriële stereocamera voor ruimtelijke perceptie, Stereolabs Inc.: https://www.stereolabs.com/en-nl/products/zed-x
[6] Siegel, M., "The sense-think-act paradigm revisited," 1st International Workshop on Robotic Sensing, 2003, IEEE: https://doi.org/10.1109/ROSE.2003.1218700
[7] Claussmann, L., Revilloud, M., Gruyer, D., & Glaser, S., A review of motion planning for highway autonomous driving, IEEE Transactions on Intelligent Transportation Systems: https://doi.org/10.1109/TITS.2019.2913998
[8] Pendleton, S. D., et al., "Perception, planning, control, and coordination for autonomous vehicles," Machines, 5(1), Article 6: https://doi.org/10.3390/machines5010006
[9] O'Mahony, N., et al., "Deep learning vs. traditional computer vision," arXiv : https://doi.org/10.48550/arXiv.1910.13796
[10] Han, C., et al., "YOLOPv2: Better, faster, stronger for panoptic driving perception," 2022, arXiv: https://doi.org/10.48550/arXiv.2208.11434
[11] Tesla's HydraNet - How Tesla's Autopilot works, Think Autonomous: https://www.thinkautonomous.ai/blog/how-tesla-autopilot-works/
[12] Syazvinski, Traffic Light Detection and Color Classification Using Yolo v8, GitHub: https://github.com/Syazvinski/Traffic-Light-Detection-Color-Classification
[13] Flucus, Traffic Sign Detection AI, GitHub: https://github.com/Flucus/Traffic-Sign-Detection-AI
[14] AI vision for everyone, Ultralytics: https://ultralytics.com
[15] Xue, Y., "An overview of overfitting and its solutions," Journal of Physics: Conference Series, 1168(2), 022022, 2019, IOPScience: https://doi.org/10.1088/1742-6596/1168/2/022022
[16] Dosovitskiy, A., et al., "CARLA: An open urban driving simulator," 2017, arXiv: https://doi.org/10.48550/arXiv.1711.03938
[17] What is machine learning inference, Hazelcast: https://tinyurl.com/hazelcast-machine-learning
[18] Sapkota, R., et al., "Comprehensive performance evaluation of YOLO11, YOLOv10, YOLOv9, and YOLOv8 on detecting and counting fruitlet in complex orchard environments," 2024, arXiv: https://doi.org/10.48550/arXiv.2407.12040
[19] Xu, H., et al., "A survey on occupancy perception for autonomous driving: The information fusion perspective," 2024, arXiv: https://arxiv.org/abs/2405.05173
[20] OpenDriveLab, OccNet: Scene as Occupancy [Code], GitHub: https://github.com/OpenDriveLab/OccNet
[21] Zhou, H., et al., "Review of learning-based longitudinal motion planning for autonomous vehicles: Research gaps between self-driving and traffic congestion," 2021, arXiv: https://doi.org/10.48550/arXiv.1910.06070
[22] Li, Z., et al., "Hydra-MDP: End-to-end multimodal planning with multi-target Hydra-distillation," 2024, arXiv: https://doi.org/10.48550/arXiv.2406.06978
[23] Singh, K., "Tesla FSD V12.5.5 adds end-to-end highway stack; Tesla teases upcoming features," 2024, Not A Tesla App: https://tinyurl.com/tesla-fsd-v1255