Önvezető funkciók megvalósítására alkalmas jármű Unreal Engine 4 alapú szimulációja

Self-driving vehicle simulation based on Unreal Engine 4

KRECHT Rudolf[1], HAJDU Csaba[2], HOLLÓSI János[3]

[1]Széchenyi István University SZE-JKK Győr, Hungary krecht.rudolf@ga.sze.hu
[2]Széchenyi István University SZE-JKK Győr, Hungary herno@ga.sze.hu
3Széchenyi István University SZE-JKK Győr, Hungary hollosi.janos@ga.sze.hu

Összefoglaló

Az önvezető, vagy akár autonóm járművek fejlesztése napjaink hangsúlyos területévé vált. Ezen területen végzett fejlesztések fontos eleme sok egyéb mellett a jármű fedélzeti vizuális szenzoraiból kinyert képinformációk megfelelő feldolgozása és alkalmazása. A képinformációk feldolgozásához neurális hálókra, öntanuló algoritmusokra van szükség. A szenzorok kalibrálásához és a neurális hálók tanításához számos mérésre, képadatra van szükség, melyek előállítása költséges és időigényes feladat. Jelen cikk célja olyan számítógépes szimulációs eljárások létrehozásának bemutatása, amelyek által önvezető funkciók megvalósítására alkalmas járművek képi információkat rögzítő fedélzeti érzékelői pontosan, valósághűen szimulálhatók. A szimulációs eljárások alkalmazása egy meghatározott jármű és szenzor esetében, példákon keresztül kerül bemutatásra. A szimuláció alkalmazásának bemutatása során külön esetként kerül kezelésre a környezeti viszonyok szimulátoron belüli paraméterezhetősége. A szimulált jármű és szenzor által lehetőség nyílik a korábban említett időigényes és költséges folyamatok szimulációval történő kiváltására – a vizuális szenzorok kalibrációja és a képadatgyűjtés lehetővé válik szimulációk használatával.

Kulcsszavak: szimuláció, autonóm jármű, Unreal Engine 4, neurális háló

Abstract

The development of self-driving, autonomous vehicles is amongst the fastest-developing fields. One of the most important elements of developments in connection with this topic is the processing and the application of vision sensor data. In order to use vision sensor data for environmental perception, neural networks, self-learning algorithms are applied. The calibration of visual sensors and the training of neural networks requires measurements and visual sensor data. The process of sensor data acquisition and training image set creation is time-consuming and cannot be considered as cost-effective. The aim of this paper is to present the creation process of a computer simulation with the purpose of simulating a vehicle mounted with visual sensors. The result of the simulation process is presented through examples and use cases for a specific passenger car and visual sensor. The application of environmental parameters will be separately presented. By the use of the presented computer simulation method, it is possible to replace the time-consuming and expensive measurement and data acquisition processes by a simulated vehicle and sensor model.

1. BEVEZETŐ

Az önvezető járművek környezetpercepciójában fontos szerepet játszanak a vizuális szenzorok, kamerák. A képi információ nagy mennyiségű adatot tartalmaz, és ezen adatok megfelelő feldolgozásával gyorsan és könnyen készíthetünk modellt a jármű környezetéről. A képi információ feldolgozásához gyakori a neurális hálók alkalmazása, melyek tanítást igényelnek. A neurális hálók tanításához tanítómintákra, képcsomagokra van szükség. A képcsomagok előállítása alapvetően költséges és időigényes folyamat, amely adott területek szenzorokkal felszerelt járművekkel való bejárását igényli, valamit a bejárás során készült képi adatokat utólag szegmentálni kell, a neurális hálók

tanítására alkalmas csomagokat kell kialakítani. [1] Jelen cikk célja a képadatgyűjtési folyamat helyettesítése szimulációs környezet alkalmazásával generált képcsomagok által. Bemutatásra kerül az általunk választott szimulációs környezetben történő modellalkotás, a képcsomagok előállításához szükséges környezetek kialakítása, valamint a szegmentáció menete. A cikk a képalkotás menetét konkrét alkalmazási eseten keresztül mutatja be. Ez az alkalmazási eset egy önvezető funkciókkal ellátott Nissan Leaf jármű zárt pályán történő képgyűjtési folyamata.

A cikk során bemutatott önvezetési feladatok a Shell Eco-marathon hallgató verseny Autonóm Városi Koncepció (AUC) kategóriájának egyes versenyszámait veszik alapul. Így a Nissan Leaf jármű alkalmazásával kidolgozott önvezetést lehetővé tevő algoritmusok később átemelhetők a Széchenyi István Egyetem hallgatói versenycsapatának járművére is. A feladatkiírás ezen módja miatt a cikk során bemutatott önvezetési feladatok és szimulált környezetek a Shell Eco-marathon AUC kategóriájának szabályzata tiszteletben tartásával alakulnak.

2. A SZÁMÍTÓGÉPES SZIMULÁCIÓ

A számítógépes szimulációk célja egy rendszer, egy folyamat számítógépes modelljének létrehozása, valamint kísérletek elvégzése az elkészített modellel a rendszer viselkedésének tanulmányozása, megértése, vagy a rendszer működésére vonatkozó stratégiák felépítése céljából. [1] A szimulációs környezetek olyan eszközök, amelyek lehetővé teszik ezen modellalkotási folyamatot és a rá vonatkozó kísérletek elvégzését is. A legtöbb számítógépes szimuláció létrehozására alkalmas eszköz kezelőfelületen keresztül használható, a modellalkotás menete hasonlóan, konfigurálható modulok összeállításával, egymáshoz képesti viszonyának definiálásával történik.

A moduláris építőelemek egymáshoz képesti viselkedését könnyen és átláthatóan definiálja az elemek közti hierarchia. Ez a fa adatstruktúra a felhasználó által definiálható és módosítható. Ez abból adódik, hogy a játékokhoz, 3D CAD modellező szoftverekhez hasonlóan, a 3D szimulátorok is egy színtérgráfra (scene graph) épülnek. A szimulátorok, számítógépes szimulációk összeállítására alkalmas szoftverek a beépített moduláris elemeken kívül importált CAD modelleket is használhatnak alakzatokként. A legtöbb generikus CAD formátum (OBJ, DXF, FBX, STL, COLLADA, URDF) importálása megoldható. Az importált elemek rendelkezhetnek ugyanazokkal a tulajdonságokkal, mint a szimulátor beépített alakzatai (dinamikai tulajdonságok, ütközésdetektálás, szenzorok általi érzékelhetőség, renderelhetőség stb.), de legtöbb esetben a könnyebb kezelhetőség miatt a felsorolt tulajdonságokkal egy egyszerűsített alakzat rendelkezik, ehhez rendelhető hozzá az importált CAD modell a helyes vizuális megjelenés érdekében. Az importált és a beépített elemekhez egyaránt rendelhető textúra.

Az összeállított modellek szimulációját fizikai motorok segítik az elemek közti mechanikai interakciók gyors számításával. Gyakran alkalmazott fizikai motorok: Open Dynamics Engine (ODE), Newton, PhysX, Bullet, DART.

2.1. Játékmotor alkalmazása szimulációs feladatok elvégzésére

Kinematikai szimulációk készítésére számos eszköz áll rendelkezésünkre (pl. Gazebo-ROS, CoppeliaSim). Ezek az eszközök is lehetővé teszik a szimulátoron belüli modellalkotást, mégis az önvezető járművek fejlesztése során számos mérnöki feladat megoldása valósult meg a játékipar eszközeit alkalmazva. Erre az ad okot, hogy a játékipar rendelkezik a legnagyobb tapasztalattal az alacsony számítási kapacitást igénylő, kiemelkedően valósághű számítógépes megjelenítés megvalósításában. Továbbá számos játékmotor teszi lehetővé a gyors és egyszerű járműmodellezést, a jármű kinematikai és dinamikai paramétereinek figyelembevételével. Természetesen ezek a járműszimulációk pontosságban elmaradnak a kifejezetten szimulációs célokkal készített szoftverektől, de neurális hálók tanítására alkalmas képhalmazok előállítása szempontjából a realisztikus megjelenés és a gyors munkamenet fontosabb tényező, mint a kinematikai és dinamikai szempontból hibátlan szimuláció.

2.2. Az Unreal Engine 4 környezet

Az Unreal Engine egy játékmotor, amely lehetővé teszi növényzetet, domborzatot, épületeket tartalmazó világok létrehozását, amelyben saját 3D-modellen alapuló modellek mozoghatnak. [2]

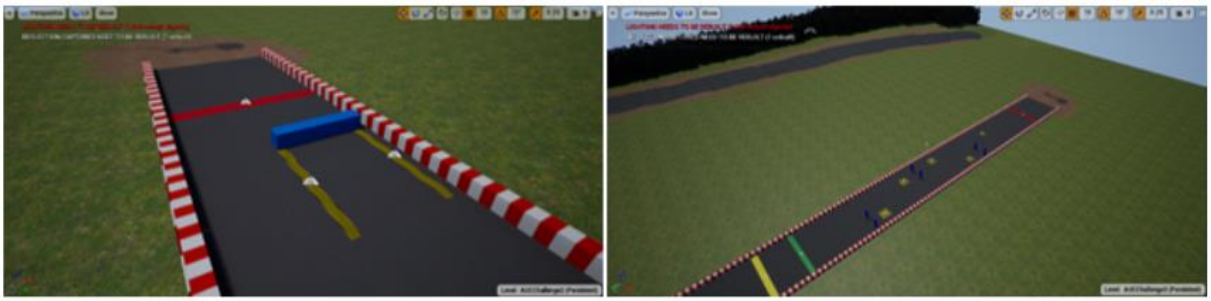
Megállapítható, hogy a vizuális szenzorokkal, kamerákkal foglalkozó mérnöki területeken a szimulációs szoftverekkel szemben támasztott követelmények nagy mértékben megegyeznek a játékiparban megfogalmazottakkal. Számítógépes játékok fejlesztése során fontos a valósághoz közel álló megjelenítés relatív kis hardverteljesítmény mellett. Hasonló igény fogalmazható meg kamerákkal felszerelt mobil robotok, járművek virtuális tesztelése során is. Valósághű környezetek alkalmazásával a kamerák szimulátoros kalibrálása lecsökkenti a valós tesztek költségeit, az átlagos rendszerigénynek köszönhetően a szimulációk könnyen kezelhetők maradnak. Fontos továbbá, hogy a kiterjedt fejlesztői bázis visszajelzéseinek, tapasztalatainak köszönhetően a játékmotoros szimulátoralkotás munkamenete gyorsabb lehet.

A játékmotoron belüli modellfejlesztés C++ kódokkal, vagy az Unreal Engine 4 saját grafikus programozói felületével (Blueprint Editor) végezhető.

Az Unreal Engine 4 kereskedelmi célokra díjköteles, nonprofit célokra ingyenesen használható Windows, Mac és Linux alapú rendszereken.

3. KÖRNYEZET MODELLEZÉSE

A jelenleg bemutatott feladat esetében célunk a neurális háló zárt pályán történő egyszerű feladatok végrehajtására történő felkészítése. A kijelölt feladatok a meghatározott, jelölt helyre történő parkolás és a szlalom. A feladatok végrehajtása előre megadott magasságú terelőelemekkel szegélyezett pályán történik, a szabályzat szerint a versenypálya határolóelemein kívül található tárgyak, járművek nem befolyásolhatják a pályán található jármű működését. A szabályzatnak megfelelően két pálya került kialakításra Unreal Engine 4 környezetben, CAD programból importált elemek alkalmazásával (1. ábra).

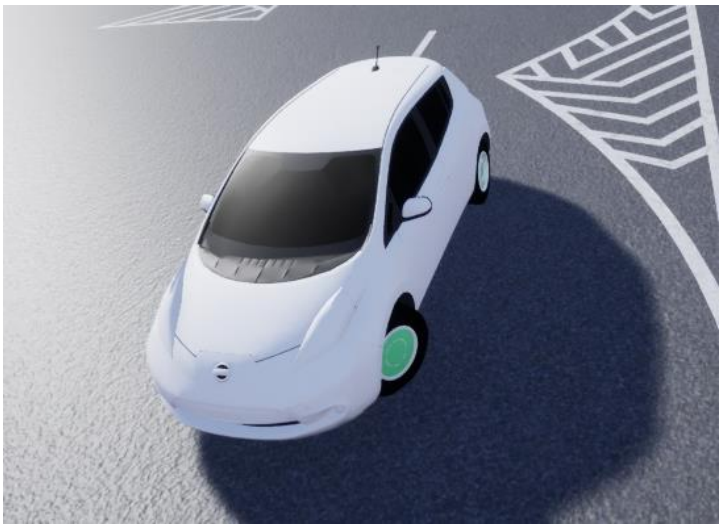


1. ábra *Virtuális környezetek*

Az Unreal Engine 4 lehetővé teszi az időjárás, a fényviszonyok változásának könnyű szimulációját. A környezeti paraméterek módosításával napszaknak megfelelő fényviszonyok adhatók meg. A köd szintén paraméterek módosításával szimulálható. Hó hozzáadása érdekében a környezet módosítása szükséges, a havazás szimulációja részecskeszimuláció segítségével készíthető el.

4. JÁRMŰ MODELLEZÉSE

A realisztikus képek érdekében érdemes a mérés tárgyát képező jármű teljes szimulációját elkészíteni. A modellezés folyamata a jármű CAD szoftverben történő előkészítésével kezdődik. A szükséges geometriai modellezés mellett ezen fázis során kerül meghatározásra a jármű karosszériája és a kerekek közötti hierarchia is. A modell importálását követően az Unreal Engine 4 környezetben elérhető négykerekű, Ackermann-kormányzású alapmodell paraméterezésével [4] elkészíthető a modellezendő jármű valóságot jól közelítő modellje (2. ábra).



2. ábra *Az elkészült járműmodell Unreal Engine 4 környezetben*

5. EREDMÉNYEK

A szimulációs folyamat eredménye több Unreal Engine 4 alapú szimulált környezet, amelyben az összeállított jármű használható. A szimulált járművön elhelyezett szimulált vizuális szenzorok alkalmazhatók neurális hálók tanítására alkalmas képhalmazok generálására.

Jelen cikk a neurális hálók tanítási módszereivel nem foglalkozik, de a modellalkotási módszer validálásának érdekében megemlíti, hogy a szimulátor segítségével generált, kis arányban valós képeket is tartalmazó képhalmazok hatásos tanítómintának minősültek.

KÖSZÖNETNYILVÁNÍTÁS

A cikk kutatásaihoz az Új Széchenyi Terv keretein belül a „Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén (EFOP-3.6.3-VEKOP-16-2017-00001)” projekt és a Széchenyi István Egyetem biztosított forrást. A kutatás az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

IRODALOMJEGYZÉK

[1] K. Zidek, P. Lazorík, J. Piteľ and A. Hošovský, "An Automated Training of Deep Learning Networks by 3D Virtual Models for Object Recognition," *Symmetry, vol. 11,* pp. 496-511, 2019.

[2] R. E. Shannon, "Introduction to the art and science of simulation," in *1998 Winter Simulation Conference. Proceedings*, Washington, DC, USA, 1998.

[3] A. C. A. Mól, C. A. F. Jorge and P. M. Couto, "Using a Game Engine for VR Simulations in Evacuation Planning," *IEEE Computer Graphics and Applications, vol. 28, no. 3,* pp. 6-12, 2008.

[4] C. Pepper, S. Balakirsky and C. Scrapper, "Robot simulation physics validation," in *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, New York, NY, USA, 2007.