

Debreceni Egyetem
Informatikai Kar
Adattudomány és Vizualizáció Tanszék

Valós idejű regisztrált szimulációs és fizikai szenzor adatok
párhuzamos gyűjtése önvezető modell autók fejlesztéséhez.

Témavezető:

Dr. Kovács László
egyetemi adjunktus

Készítette:

Mészáros László
Programtervező informatikus
Vasváry Gábor
Mérnökinformatikus

Bírálatra való beküldéséhez hozzájárulok

Debrecen
2022

Tartalomjegyzék

Rövidítések listája:.....	3
Bevezetés.....	5
Rendelkezésre álló eszközök.....	7
Marvelmind GPS rendszer	7
Kommunikációs feladat leírás	8
Adattovábbítási megfontolások, követelmények.....	8
Lehetséges kommunikációs protokollok és összehasonlításuk	9
Továbbfejlesztési lehetőségek	11
MQTT és CoAP:.....	13
Önvezető jármű és tesztpálya 3 dimenziós rekonstrukciója.....	14
Alkalmazott 3D digitalizációs technológiák	15
Autó és pálya modellek utómunkái	20
Virtualizált környezet fejlesztése NVIDIA Isaac Sim segítségével	23
Virtualizált és valós környezet összekapcsolása	25
Szenzorok elhelyezése	26
Szenzor adatok regisztrálása generatív hálózat segítségével	27
Összefoglalás.....	28
Saját munka leírása:	29
Irodalomjegyzék	31

Rövidítések listája:

- UDP User Datagram Protocol
- GPS Global Positioning System
- Wi-Fi Wireless Fidelity
- OSI Open Systems Interconnection
- TCP Transmission Control Protocol
- IP Internet protocol
- HTTP Hypertext Transfer Protocol
- JSON JavaScript Object Notation
- M2M Machinet to Machine
- MQTT Message Query Telemetry Transport
- CoAP Constrained Application Protocol
- AMQP Advanced Message Queuing Protocol
- IoT Internet of Things
- TLS/SSL Transport Layer Security/Secure Socket Layer
- URI Uniform Resource Identifier
- DTLS Datagram Transport Layer Security
- ROS Robot Operating System
- LIDAR Light Detection and Ranging
- SfM Structure from Motion
- CUDA Compute Unified Device Architecture
- RTAB-MAP Real-time appearance-based mapping
- SLAM Simultaneous localization and mapping
- RGB-D RegGreenBlue-Depth
- IMU Inertial measurement unit

- GAN Generative Adversarial Network
- RTX Ray Tracing Texel eXtreme
- IA Inverse Architecture
- NIA Non Inverse Architecture

Bevezetés

A Society of Automotive Engineers International szerint vezetésautomatizálásnál öt szintet különböztetünk meg. A 0. szint a teljes manuális vezetés, ahol semmilyen eszköz nem segíti a vezetést. Az 1. szint a vezetés támogató rendszerek használata, ilyen például a sávtartó rendszer. A 2. szint a részleges vezetésautomatizálás, ezen a szinten a különböző vezetés támogató rendszerek segítségével az autó képes a vezető beavatkozása nélkül rövidebb utat is megtenni. A 3. szint a feltételes vezetésautomatizálás, amikor a jármű képes önmagát vezetni megadott útvonalakon. A 4. szint a magas vezetésautomatizálás, amikor az autó képes az összes vezetéssel kapcsolatos feladatot átvenni. Az 5. szint pedig a teljes önvezetést jelenti, ahol minden felmerülő körülmények között képes a jármű bármiféle emberi beavatkozás nélkül eljutni a kijelölt célhoz.¹ Ennek a megvalósításához a gépi látáson felül számos egyéb szenzor alkalmazására van szükség. Ezen szenzorokból nyert adatokat használjuk fel az önvezető modell tanítása során. Közlekedése során az önvezető jármű teljes mértékben a szenzorok adataira és a gépi látásra támaszkodik, így fel kell készítenünk a modellünket arra is, hogyan alkalmazkodjon az egyes szenzorok kiesésére, zajos szenzor adatokra vagy a különböző típusú meghibásodásokra. A dolgozat célja egy olyan rendszer létrehozása és bemutatása, amely valós időben képes szimulációs szenzor adatokat gyűjteni fizikai szenzorok alapján és képes ezen adatokat generatív hálózat segítségével módosítani.

A dolgozat célja egy olyan megoldási javaslat létrehozása és bemutatása, amely valós időben képes szimulációs szenzor adatokat gyűjteni fizikai szenzorok alapján felhasználva az NVIDIA Omniverse platform Isaac Sim nevezetű szoftverjét. Ebben a szoftverben elkészített szimuláció képes transzfer és reinforcement learning módszerek mellett, generatív eljárásokkal támogatott önvezető járművek nehéz körülmények közötti elemzésére, betanítására és adat augmentációjára hatékonyan alkalmazható. Különös tekintettel az egyes időjárési körülmények okozta szenzorjeltorzulásokra. A szimuláció lehetőséget nyújt olyan gépi tanulási folyamatok alkalmazására, mint a felügyelet nélküli tanulás, ami nem kivitelezhető belátható időn és

¹ Society of Automotive Engineers International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. https://www.sae.org/standards/content/j3016_202104/ [Letöltve: 2022. április]

erőforrásokon belül a valós környezetben. Ezen tanulási folyamatok bevezetésével például csökkenteni tudjuk a monoton adatelemzéssel és annotációval töltött munkaórákat, továbbá olyan esetek betanítása is lehetségessé válik, amik nem, vagy csak nehezen valósíthatóak felügyelt módszerekkel. Ezen felül a lehetségessé válik a szimulációval nyert adatokat a valós rendszerekbe integrálni és ezáltal tovább finomítani a járművek működését. A szimulációval képesek leszünk ritka események modellezésére, aminek eredményeként ezen események kezelésére is fel tudjuk készíteni az eszközöket. Továbbá a valós világ béli újonnan létrehozott folyamatok és tesztek virtuális térbeli ellenőrzésére is lehetőséget nyit a modellezés. Ezen virtuális tesztek alkalmazásával képesek leszünk előre nem látott hibák feltárására, továbbá az új folyamatok ellenőrzésére, amivel, akár baleseteket is elkerülhetünk. Nem utolsó sorban a szimuláció lehetőséget nyújt olyan eszközök tesztelésére, ami nem áll a rendelkezésünkre fizikálisan, vagy éppen az eszköz (jármű, pálya) mások által volna éppen használatban.

A szimulációt az NVIDIA Isaac Sim használatával készítettük el, ebben hoztuk létre egy a valósággal közel megegyező virtuális környezetet, ahol lemodelleztük az Autonom Járművek Kutató Labor² keretein belül fejlesztett önvezető járművet és teszt pályát. A modelleket 3D szkenneléssel hoztuk létre, majd modellező szoftver segítségével véglegesítettük azokat. Felhasználtuk a laborban már rendelkezésre álló belső pozicionáló rendszert is, amely alapján megfeleltetjük egymásnak a virtualizált és valós jármű koordinátáit. A pozíciós adatokat valós időben küldjük a szimulációs szoftvernek UDP csomagokkal, ami ez alapján pontosan arra a pozícióra helyezi el a teszt pályán a virtuális járművet, ahol a fizikai autó is található. Az így megfeleltetett szimulációs autón szenzorokat helyeztünk el, amelynek adatait feltudjuk használni a későbbiekben az adatgyűjtés során, valamint módosításokat végeztünk a szenzor adatokon generatív hálózatok segítségével, hogy a valós szenzorok közel megegyezők legyenek.

Köszönetnyilvánítás

Itt szeretnénk megragadni az alkalmat, hogy köszönetet mondjunk témavezetőnknek, Dr. Kovács Lászlónak a TDK elkészítésében nyújtott segítségével. Továbbá Kovács Kornél és

² Autonomous Vehicle Research Lab University of Debrecen. <https://autonomous-vehicle-research-group-ud.github.io/> [Letöltve: 2022. április]

Borsos Balázs mérnökinformatikus mester szakos hallgatóknak, akik aktívan hozzájárultak, ezen dolgozatban bemutatott eredmények elérésében.

Köszönettel: Mészáros László és Vasváry Gábor

Rendelkezésre álló eszközök

A dolgozat során felhasználtuk az Autonom Járművek Kutató Labor jelenleg rendelkezésre álló Dave-nek elnevezett önvezető járművet. Erre az eszközre készítettük el a kommunikációs szoftverünket, valamint erről a járműről készítettünk digitális 3D modellt. Felhasználtuk még az egyetemen található önvezető járművek számára létrehozott teszt pályát, mint a virtuális környezet alapját. Továbbá alkalmazásra került egy ipari környezetben is helytálló helymeghatározó rendszer, név szerint Marvelmind³, amely egy ultrahang technológiára épülő rendszer, robotok pozícionálásra fejlesztették ki és közel 2 cm-es pontosságot tesz lehetővé.

Marvelmind GPS rendszer

A Marvelmind egy széleskörben elérhető hardver/szoftver beltéri, akár kültéri precíz helymeghatározó rendszer. A rendszer 3 fő hardveres elemből épül fel, egy központi modemből, álló és mozgó jeladókból. A helymeghatározás alapja az ultrahangos „time-of-flight” mérés. Az eszközök sub-Ghz rádiós kommunikációval kommunikálnak egymással. A Marvelmind 2 fő úgy nevezett architektúrában alkalmazható, név szerint NIA (non inverse architecture) és IA (inverse architecture). Mi az előbbi megoldást alkalmaztuk, mivel, a hardveres eszközeink azt támogatják. NIA esetén a mozgó jeladók üzemelnek, mint adó és az állók, mint vevő és a központi modem által történik a helymeghatározás. Az IA jobban skálázható és lehetővé teszi több mozgó jeladó használatát egy rendszerben anélkül, hogy az negatívan befolyásolná a rendszer frissítési gyakoriságát. Emiatt a jövőben át fogjuk alakítani a rendszert ezen architektúra használatára.

³ Marvelmind weboldal. <https://marvelmind.com/> [Letöltve: 2022. április]

Kommunikációs feladat leírás

Felmerült az igény egy kliens-szerver alapú kommunikációs megoldás kialakítására, ami képes egy virtualizált környezethez sikeresen továbbítani a beltéri "GPS" rendszer által meghatározott térbeli koordinátákat és egyéb mozgási paramétereket, mint például a jármű iránya, sebessége, gyorsulása. Ezen szoftver-páros kliens oldali programját az önvezető autó vezérlő algoritmusába integráltuk, Python programozási nyelven. A szerver oldali szoftver implementálása esetében két lehetőség merült fel: (1) köztes kommunikációs továbbító ügynök segítségével, vagy (2) magát a virtualizációt futtató számítógépen futó alkalmazás fejlesztésével. A szerver oldali szoftvert a második lehetőséget alkalmazva készítettük el.

Adattovábbítási megfontolások, követelmények

Annak érdekében, hogy minimalizáljuk az adattovábbítás késleltetését a Marvelmind rendszer és a virtuális tér között, számos megoldási lehetőséget vetettünk össze több szempont alapján. Ezen opciók egy részét implementáltuk és teszteltük velük a rendszerünket. A kísérletek számos hasznos információval szolgáltak, ami rávilágított több nem előre látott problémára és egyértelműsítette a választást a kommunikációs opcióink között. A tervezés során az alábbi pontok voltak a fő kritériumok az adattovábbítással kapcsolatban:

1. A kommunikáció a lehető leggyorsabb legyen a késleltetés minimalizálására.
2. A különálló csomagok egyértelműen azonosíthatóak, csoportosíthatóak és sorba rendezhetőek legyenek.
3. A kommunikációs csatorna vagy csatornák széleskörűen legyenek elérhetőek.
4. Az adatcsomag lehetőleg, minél kisebbek legyenek.
5. Az üzeneteket legyenek titkosíthatóak.
6. Az üzenetek tartalma legyen egyértelmű és moduláris, későbbi továbbfejlesztés érdekében.
7. Az önvezető autó limitált erőforrásai miatt, az adatfeldolgozó és továbbító szoftver, minél gyorsabb és erőforrás kímélő legyen.
8. A kommunikációs rendszer átalakítható, skálázható legyen, annak érdekében, hogy a jövőben akár több jármű párhuzamos szimulációja is lehetségessé váljon.

Lehetséges kommunikációs protokollok és összehasonlításuk

A fentebb említett követelmények alapján közel egyértelmű volt, hogy internet kapcsolaton keresztül lesz a leghatékonyabb megoldani a két végpont összekötését. A kliens oldalon Wi-Fi vezeték nélküli csatlakozást használtunk fel, mivel a Marvelmind rendszer beltéri alkalmazási körében a legtöbb esetben biztosított a vezeték nélküli hálózati lefedettség. Továbbá ezen megoldás nem igényelt extra hardveres eszközöket, mivel az önvezető jármű központi számítógépe rendelkezik Wi-Fi-adapterrel. Ezzel a megoldással növeltük a jármű akkumulátor üzemidejét és csökkentettük a fejlesztési időt, továbbá csökkentettük a rendszer komplexitását az egyéb felhasználható technológiákhoz képest. Egyéb megoldások esetén, mint például Bluetooth, vagy Zigbee, mind kliens, mind szerver oldalon szükséges lett volna további eszközök beszerzése, felhasználása. Ezen kommunikációs protokollok továbbá, kisebb sávszélességet és adatátviteli sebességet biztosítanak és nehezebben skálázhatóak.

Internetes kommunikáció esetén az OSI, TCP/IP hibrid modell terminológiáját alkalmaztuk⁴. Első esetben a HTTP alkalmazási réteg protokollját használtuk fel, ami utólag elvetésre került több szempont alapján is. A HTTP alkalmazása kézenfekvő megoldásnak, tűnt mivel a virtualizációs környezet támogatja azt és széleskörűen elérhetőek hozzá, mind kliens, mind szerver oldali alkalmazások, függvénykönyvtárak. Ezen megoldás igényelt egy köztes szerver célt betöltő alkalmazást, ami képes fogadni az adatokat a kliensektől, ezeket az adatokat, osztályozni és http-get kéréseket kiszolgálni a virtualizációs környezet felé. Adatszerkezeti megoldásnak, pedig a JSON formátumot választottuk⁵. A fejlesztési idő csökkentése végett egy előre megírt Node.js csomagot használtunk fel, mint http szerver⁶, a kliens oldalon pedig a Python programozási nyelv requests⁷ nevű könyvtárát alkalmaztuk. Első tesztjeink során fény derült arra, hogy egy-egy üzenet továbbítása 2-3 másodpercet vesz igénybe, ami elfogadhatatlanul lassú a követelményekhez képest. A Marvelmind rendszer maximálisan 25 frissítést tud végezni másodpercenként, azaz a legmagasabb továbbítási idő csomagonként ideális körülmények között 40 milliszekundum kell, hogy legyen, de attól lehetőleg minél rövidebb. A probléma forrását főként az okozta, hogy az üzenetek között nem egy egységes

⁴ OSI, TCP/IP and Hybrid models - GeeksforGeeks. <https://www.geeksforgeeks.org/osi-tcp-ip-and-hybrid-models/> [Letöltve: 2022. április]

⁵ Introducing JSON. <https://www.json.org/json-en.html> [Letöltve: 2022. április]

⁶ JSON Server. <https://www.npmjs.com/package/json-server> [Letöltve: 2022. április]

⁷ Requests: HTTP for Humans™. <https://docs.python-requests.org/en/latest/> [Letöltve: 2022. április]

TCP munkamenet épül ki, hanem üzenetenként épül fel és bomlik le egy-egy kapcsolat. Ezt a problémát a requests könyvtár átállításával sikerült mérsékelni. Az úgy nevezett „session-mode” aktiválása révén, már egy kapcsolat került felhasználásra több üzenet továbbítása esetén is. Ezzel a megoldással sikerült redukálni az üzenetek célbajutattási idejét átlagosan 20-30 ms/üzenet-re. A következő probléma ezzel a megoldással szintén a http szerver egyik tulajdonságából adódott. Az üzenetek, mivel nem egyesével kerülnek fájlba írásra, hanem egy pufferelő folyamat után blokkoként íródnak ki másod rendű tárolóra, ezért a virtualizációs környezet nem egy folytonos adatfolyamhoz fér hozzá, emiatt alkalmatlan a http megközelítés a jelen követelmények kielégítésére. Ezen felül a másodrendű tár elérési ideje is lassítaná a kommunikációs folyamatot, ami ebben az esetben igen közel van a fentebb említett időkorláthoz. Tehát belátható, hogy a http protokoll igen hasznos olyan alkalmazási területeken, ahol perzisztens adatok nem időkritikus elérése a cél az interneten keresztül, viszont alkalmatlan M2M (machinet to machine) azonnali üzenet továbbításra.

A kliens oldalon egy másik függvénykönyvtár tesztelésére is sor került. Az úgy nevezett faster-than-requests⁸ könyvtár szintén alkalmatlannak bizonyult, mivel nem rendelkezik több http üzenet közti egységes munkamenet móddal, a request könyvtárral ellentétben.

Ezután arra a döntésre jutottunk, hogy a hibrid modellben egy szinttel alacsonyabban lévő protokollok, azaz a szállítási réteg protokolljait teszteljük és a megfelelőt alkalmazzuk. TCP kommunikáció esetén pozitívum, hogy a protokoll sajátosságai közé tartozik az adatok sorban történő továbbítása és a csomagok továbbításának az ellenőrzése és a munkamenet alapú kommunikáció. Viszont a kommunikációs követelmények időkritikussága miatt, az UDP protokollt találtuk megfelelőbbnek⁹¹⁰. Mivel a kis mértékű csomagvesztés nem kritikus hiba a rendszerben és a sorrendiség, igény szerint alkalmazási oldalon visszaállítható az üzenetek időbélyegjeinek segítségével, ezért nem szükséges TCP protokoll alkalmazása. UDP protokoll esetén továbbá, általánosan kijelenthető, hogy gyorsabb adatátviteli sebességet tesz lehetővé, mint a TCP. Az UDP alkalmazás drasztikus javulást hozott, ugyanis míg a http protokoll 1 darab üzenetet körül-belül 25 ms alatt jutott célba, addig az UDP módszer több, mint 100

⁸ Faster-than-Requests. <https://github.com/juancarlospace/faster-than-requests> [Letöltve: 2022. április]

⁹ Differences between TCP and UDP - GeeksforGeeks. <https://www.geeksforgeeks.org/differences-between-tcp-and-udp/> [Letöltve: 2022. április]

¹⁰ M. Masirap, M. H. Amaran, Y. M. Yussoff, R. A. Rahman, and H. Hashim, “Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT),” *ISCAIE 2016 - 2016 IEEE Symposium on Computer Applications and Industrial Electronics*, pp. 200–205, Sep. 2016, doi: 10.1109/ISCAIE.2016.7575063.

csomagot képes továbbítani kevesebb, mint 3 ms alatt. Ezen sebesség, többszörösen kielégíti a rendszerrel szemben támasztott követelményeket. Továbbá csökkenti a kliens program erőforrás igényét és futási idejét. A kliens program futási idejének csökkentése azért indokolt, mert az önvezető jármű fő esemény ciklusában kapott helyet közvetlenül a Marvelmind adatok feldolgozása után¹¹. Ezen fő ciklus pedig konfiguráció szerint másodpercenként egy minimális számban végre kell hajtódjon, a jármű megfelelő működése érdekében. A jelenleg alkalmazott Marvelmind nem inverz architektúra esetén lehetőség van a rendszer központi egységét vezérlő számítógépen keresztül az adatok lekérdezésére, ami lehetővé teszi, hogy a vezérlő számítógép kliensként is funkcionáljon. Viszont a jövőben át akarjuk alakítani a rendszert, az úgy nevezett inverz architektúrára, ami esetén maguk a mozgásban lévő jeladók számolják a saját pozíciójukat hasonlóan a GPS rendszerhez, emiatt célszerűbb a kliens alkalmazást a járműveken futtatni, hogy elkerüljük a Marvelmind rendszer rádiós kommunikációjából adódó többlet időt és előre nem látott komplikációkat.

A JSON adatformátumot utólag leváltottuk egy egyszerűbb saját adatszerkezetre, ami megfelelő tagolást biztosít az adatalemek között. Ezzel a módosítással még hatékonyabb lett a szoftverünk, mivel, így már nem volt igény a JSON Python függvénykönyvtár alkalmazására sem.

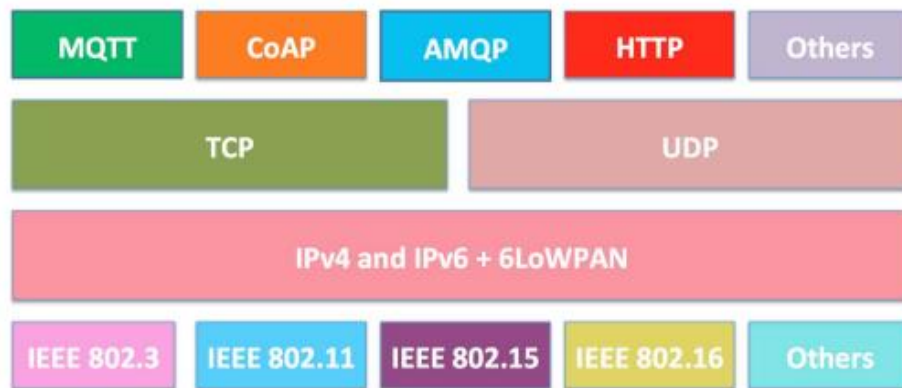
Továbbfejlesztési lehetőségek

Jelenleg a rendszer egy kliensből és egy szerverből épül fel. Ezt a jövőben szeretnénk kiterjeszteni úgy, hogy a virtuális térben akár több önvezető jármű vagy egyéb entitás adatai párhuzamosan megjelenjenek. Továbbá mindenféleképpen szükséges lesz a kommunikáció titkosítására. Ezen feltételek mellett már ideálisabb olyan alkalmazás rétegbéli protokollokat alkalmazni, amik kifejezetten instant és könnyűsúlyú kommunikációra lettek tervezve. Ilyen protokollok például az MQTT, CoAP, vagy a AMQP¹². Elképzelhető, hogyha felmerül rá az

¹¹ What is a Part. <https://docs.donkeycar.com/parts/about/> [Letöltve: 2022. április]

¹² M. Masirap, M. H. Amaran, Y. M. Yussoff, R. A. Rahman, and H. Hashim, "Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT)," *ISCAIE 2016 - 2016 IEEE Symposium on Computer Applications and Industrial Electronics*, pp. 200–205, Sep. 2016, doi: 10.1109/ISCAIE.2016.7575063.

igény, akkor lokális rádiókommunikációs IoT hálózatot is integrálunk a rendszerbe, viszont ezen megoldások mindenképp új hardveres elemeket feltételeznek.^{13 14}



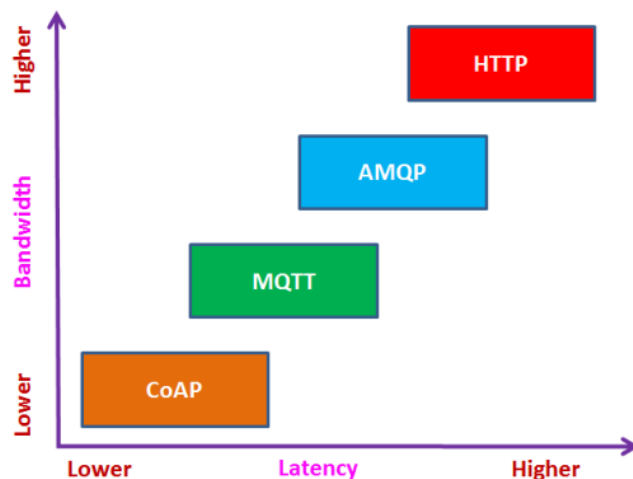
1. ábra protokollok¹⁵

¹³ 8 Most Popular IoT Protocols and Standards You Need to Know | SaM Solutions. <https://www.sam-solutions.com/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview/> [Letöltve: 2022. április]

¹⁴ A Complete Guide to IoT Protocols & Standards In 2021. <https://www.nabto.com/guide-iot-protocols-standards/> [Letöltve: 2022. április]

¹⁵ N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," 2017 *IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*, Oct. 2017, doi: 10.1109/SYSENG.2017.8088251.

MQTT és CoAP:



2. ábra sávszélesség és késleltetés¹⁶

A lehetséges jövőbeli megoldások közül ezen két szabványt emelnénk ki, ideális tulajdonságaik miatt. Az MQTT (Message Queuing Telemetry Transport Protocol) protokollt 1999-ben jelent meg először. A protokollt eredetileg távoli mérőállomások M2M gép-gép műholdas internet kommunikációjára fejlesztették ki, annak érdekében, hogy minél kisebb üzeneteken keresztül zajljon a kommunikáció, mivel a fejlesztésének az idejében igen költséges volt a műholdas internet kommunikáció és a forgalmazott adatmennyiség alapján történt a díjszabás. Azóta igen elterjedté vált a protokoll az IoT hálózatok alkalmazásánál és olyan megoldásoknál, ahol limitáltak az erőforrások, vagy kis méretű üzenetek gyors azonnali továbbítása szükséges. Egy MQTT üzenet igen kicsi 2 byte-os fejrészrel rendelkezik ezzel is csökkentve a továbbítási időt. Egy üzenet adatrészeének maximális hossza pedig 256 MB. Az MQTT hálózatok az úgy nevezett „publis/subscribe” elvet követik. Ennél a topológiánál van egy úgy nevezett központi szerver, amit az MQTT esetén broker-nek neveznek. Ezen broker-hez csatlakoznak különböző „témákra” (topic) a feliratkozók és kiadók. Az MQTT a TCP protokollt használja szállítási réteg szinten, viszont lehetséges akár UDP-vel is használni. A kommunikáció titkosítható TLS/SSL titkosítással.

¹⁶ Labbé, Mathieu, and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation." *Journal of Field Robotics* 36.2 (2019): 416-446.

A CoAP (Constrained Application Protocol) hasonlóan az MQTT-hez egy könnyűsúlyú M2M gép-gép kommunikációs protokoll. CoAP hasonlóan a http-hez URI segítségével továbbítja az üzeneteket kliens-szerver (request-response) architektúrát alkalmazva. Ellentétben az MQTT-vel a CoAP csatlakozás mentes protokoll, tehát UDP-n keresztül zajlik a kommunikáció, viszont lehetőséget nyújt az üzenetek vételének ellenőrzésére. A CoAP rendszerek az adó és vevő oldal függetlensége miatt jobban skálázhatóak az MQTT-nél. A kommunikáció DTLS módszerrel titkosítható. A CoAP előnyei közé sorolható még az MQTT-hez képest, hogy kisebb az erőforrásigénye, késleltetése, az üzenetek „overhead”-je, jobban standardizált és könnyebb az alkalmazása más technológiákkal összekapcsolva.¹⁷

A közeljövőben le fogjuk váltani a Donkeycar rendszert ROS2 (robot operating system) platformra. Az ROS2 széleskörű elterjedtsége és támogatása révén, gördülékenyebben tudjuk, majd továbbfejleszteni és skálázni a megoldásunkat. Például több szenzor adatai alapján, szenzor fúzió segítségével, pontosabb helymeghatározást, fogunk tudni elérni, magasabb frissítési gyakorisággal.

Önvezető jármű és tesztpálya 3 dimenziós rekonstrukciója

A virtuális tér életszerűségének növelése érdekében elkezdtünk foglalkozni a jármű és a tesztpálya 3 dimenziós digitalizálásának lehetőségével. Két fő irányvonal feltérképezésébe, tesztelésébe kezdtünk bele. A LiDAR szkennelés volt az első technológia, aminek elkezdtük a tesztelését. A módszer lényege, hogy a beolvasandó tárgyra egy speciális kamera lézer sugarakat vetít és méri a sugár kibocsátása és visszaverődése közti eltelt időt, amiből meg lehet állapítani, hogy az adott tárgy egyes pontjai milyen távolságban helyezkednek el a kamerától. Ezen pontokból utómunkálatok segítségével kialakítható egy poligon háló, ami már a virtuális térben reprezentálja a beolvasott tárgyat. Ezen hálóra további munkálatok segítségével van lehetőség textúrákat illeszteni, hogy a valós tárgyat minél jobban imitálja. A lézer szkennelés az Intel® RealSense™ LiDAR Camera L515 felhasználásával került tesztelésre. Míg a koncepció igen ígéretesnek hangzott, addig sajnos első próbálkozásaink során nem tudtuk megvalósítani szoftveres erőforrások hiányában. Az L515 kamera képes akár színes 3d lézer szkennelésre is, viszont a gyártó által hivatalosan támogatott szoftverek anyagi vonzatai miatt,

¹⁷ CoAP vs HTTP | difference between CoAP and HTTP. <https://www.rfwireless-world.com/Terminology/Difference-between-CoAP-and-HTTP.html> [Letöltve: 2022. április]

nem tudtuk kiaknázni a hardver által nyújtott lehetőségeket. Jelenleg is keressük annak lehetőségét, hogy a kamerát nyílt forráskódú ingyenes szkennelő szoftverekkel tudjuk integrálni a jövőben.

A második alkalmazott technológia a fotogrammetria volt. Ezen technika képes megfelelő 2 dimenziós fényképek sorozatából rekonstruálni 3 dimenziós tárgyakat. Az eljárás alapját az úgy nevezett mozgásból származó struktúra (SfM) képezi. Ezen eljárás képes az egyes képek egyedi kulcs attribútumainak (keypoints)¹⁸ a felhasználásával az egymást részben fedő kétdimenziós képek alapján térbeli koordinátákat előállítani és meghatározni a térbeli kamera pozícióját. Ezekből a származtatott koordinátákból épül fel a tárgy 3 dimenziós ponthálója. A folyamat további részei megegyeznek a LiDAR technológiánál említett lépésekkel. A fotogrammetria időigényesebb, kevésbé pontos és munkaintenzívebb folyamat, mint a LiDAR szkennelés, viszont sokkal költséghatékonyabb, mivel nincs szükség drága hardverekre, szoftverekre¹⁹²⁰. Elegendő hozzá egy kellően nagy felbontású kamera, amit a mai modern okostelefonok kamerái tökéletesen kielégítenek és egy kellően erős számítógép CUDA számításokra képes grafikus kártyával (szintén szükséges a LiDAR szkenneléshez). Fontos a megfelelő fényképek előállítása, ideértve szórt fényviszonyok alkalmazását és a reflektív felületek kerülését, vagy maszkolását valamilyen matt bevonattal. A tárgyat minél több pozícióból fényképezzük, annál jobb eredményt produkálnak a rekonstrukciós alkalmazások, viszont ezzel egyenesen arányosan nő a számítási idő is.

Alkalmazott 3D digitalizációs technológiák

A képek feldolgozására a Meshroom²¹ nevű fotogrammetriai 3D rekonstrukciós nyílt forráskódú, ingyenes szoftvert alkalmaztuk először. Az utómunkálatokhoz a MeshLab²² nyílt

¹⁸ Mapware's Photogrammetry Pipeline, Part 1 of 6: Keypoint Extraction | Mapware.

<https://mapware.ai/blog/photogrammetry-pipeline-part-1-of-6-keypoint-extraction/> [Letöltve: 2022. április]

¹⁹ E. P. Baltsavias, "A comparison between photogrammetry and laser scanning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 2–3, pp. 83–94, Jul. 1999, doi: 10.1016/S0924-2716(99)00014-3.

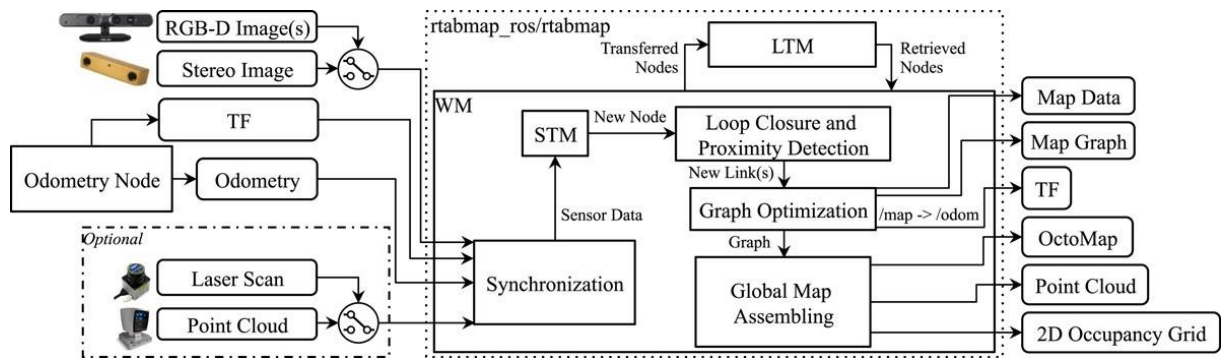
²⁰ D. Moon, S. Chung, S. Kwon, J. Seo, and J. Shin, "Comparison and utilization of point cloud generated from photogrammetry and laser scanning: 3D world model for smart heavy equipment planning," *Automation in Construction*, vol. 98, pp. 322–331, Feb. 2019, doi: 10.1016/J.AUTCON.2018.07.020.

²¹ AliceVision | Meshroom - 3D Reconstruction Software. <https://alicevision.org/#meshroom> [Letöltve: 2022. április]

²² MeshLab. <https://www.meshlab.net/> [Letöltve: 2022. április]

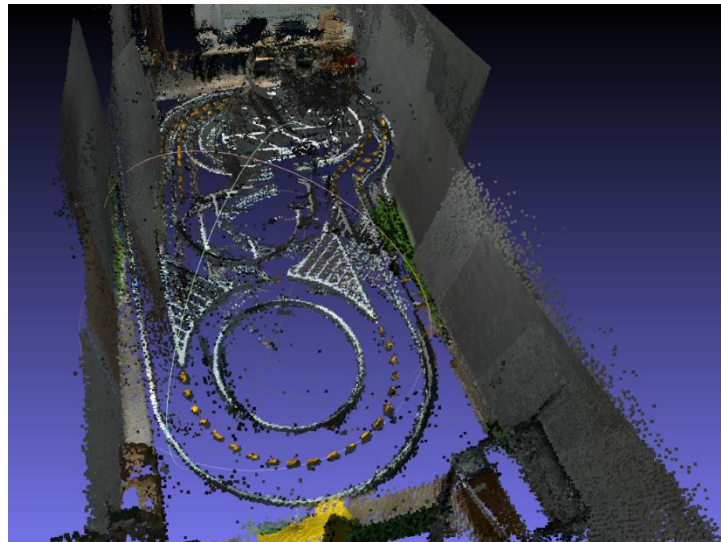
forráskódú térháló szerkesztő szoftvert alkalmaztuk. A Meshroom szoftver többszöri próbálkozásra sem produkált kellően jó eredményeket, emiatt áttértünk az Autodesk ReCap Pro elnevezésű programcsomag használatára. A ReCap és a ReCap Photo egy diákok számára az Autodesk által ingyenesen biztosított felhő alapú fotogrammetriai és 3D rekonstrukciós programcsomag. A felhő alapú számításoknak köszönhetően, jóval kevesebb időt vesz igénybe a modell generálása, mint egy átlagos laptopon, vagy asztali számítógépen. Az alkalmazás egyetlen megkötése, hogy projektenként maximum 100 db fényképet lehet felhasználni viszont, ahogy a képeken is látható, csak 100 fénykép felhasználásával is igen jó minőségű térbeli modell hozható létre, amit némi utólagos feldolgozással, tovább tökéletesíthető. Feltehetően a Meshroom alkalmazás extenzív dokumentációjának tanulmányozásával reprodukálhatóak lennének hasonló eredmények, mint a ReCap Photo esetén, de a rendelkezésünkre álló idő függvényében erre nem tudtunk egyelőre sort keríteni. Keresésünk az open source LiDAR szkennelő szoftverrel kapcsolatban eredményesnek bizonyultak, ugyanis találtunk egy nyílt forráskódú, ingyenes függvénykönyvtárat, ami támogatja az Intel RealSense kameráit. Név szerint RTAB-Map (Real-time appearance-based mapping).

Az RTAB-Map egy gráf-alapú SLAM (egyidejű helymeghatározás és feltérképezés) módszert felhasználó inkrementális megjelenés alapú hurok záródás detektor függvénykönyvtár, ami képes kezelni RGB-D, sztereó és LiDAR kamerákat. Lényegében a szoftver valós időben képes korlátozott rendszermemória felhasználásával, akár színes ponthálókat készíteni és azonosítani azokat a térbeli pontokat a már elkészült felvételek alapján, amiken, már a szenzor legalább egyszer tartózkodott (loop closure), és ezen azonosított felvételek alapján minimalizálni a generált térkép hibáját, ami az IMU és LiDAR szenzor hibáiból adódik össze. A hiba



3. ábra Az RTAB-Map működési blokk diagrammja

minimalizálása érdekében, minél több felvétel készül egy beolvasás során, annál pontosabb modell nyerhető vissza.^{23 2425262728}



4. ábra pontfelhő

²³ RTAB-Map | Real-Time Appearance-Based Mapping. <http://introlab.github.io/rtabmap/> [Letöltve: 2022. április]

²⁴ N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J. Y. Ertaud, "Benchmark of visual SLAM algorithms: ORB-SLAM2 vs RTAB-map," *2019 8th International Conference on Emerging Security Technologies, EST 2019*, Jul. 2019, doi: 10.1109/EST.2019.8806213.

²⁵ D. Droschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5000–5007, Sep. 2018, doi: 10.1109/ICRA.2018.8461000.

²⁶ W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 1271–1278, Jun. 2016, doi: 10.1109/ICRA.2016.7487258.

²⁷ X. Chen *et al.*, "OverlapNet: Loop Closing for LiDAR-based SLAM," May 2021, doi: 10.15607/RSS.2020.XVI.009.

²⁸ M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, Mar. 2019, doi: 10.1002/ROB.21831.

A rendszert a már előre lefordított CUDA támogatott alkalmazásával kezdtük el használni az Intel RealSense L515 kamerával és egy notebook felhasználásával. Az első beolvasások sikertelennek bizonyultak az L515-ös kamerával, feltehetően a tesztpálya sötétszürke alapja miatt: a szürke és fekete színű felületek jó fényelnyelő tulajdonságaik miatt, feltételezhetően nem verték vissza a kamera által kibocsájtott lézer sugarakat és/vagy a kamerába épített szenzor nem volt elég érzékeny a feladathoz.

Következő lépésként próba beolvasást végeztünk egy Intel RealSense D455 RGB-D kamerával. A D455 kamera az L515-tel ellentétben nem a LiDAR technológia alapján működik. Az eszköz sztereó infravörös kamerákkal rendelkezik és egy infravörös minta-projektorral. A két kamera képéből, továbbá a kivetített infravörös mintából állítja elő a 3 dimenziós pontfelhőt. Ezen felül a kamera rendelkezik széles kivetítésű infravörös projektorral és RGB kamerával. A D455 szenzorral készült felvételek, sokkal biztatóbbak voltak, mint az L515 készült társaik. Ugyan a felvételek nem voltak tökéletesek, de némi utómunkával korrigáltuk a pontatlanságokat.



5. ábra A D455 kamerával készített pontfelhő.



6. ábra RealSense L515-ös kamerával készült pontfelhő.

Az RTAB-Mapból exportált pontfelhőt a MeshLab nevű szoftverrel dolgoztuk fel. A 3d pontfelhőt Poission felület rekonstrukciós módszerrel ²⁹hoztuk létre. A Poission módszernél több problémába ütköztünk. Ha a módszer iterációs lépéseit túl alacsonyra állítottuk, egyenletes felületet kaptunk a pályáról, viszont a generált háló az alacsony poligon számosság miatt, igen elmosódottá vált. Ellenkező esetben pedig élesebb textúrát kaptunk, ellenben a pontfelhő pontatlansága miatt, a pályán több helyen is egyenletlenségek keletkeztek, amit részben szűrési eljárásokkal mérsékelünk, de megszüntetni nem tudtunk. Ezen felül kísérletezünk az alacsony poligon számú modell poligon számát felskálázni középpont módszerrel és a pontfelhőből származtatott textúrák korrigálásával helyrehozni az elmosódott színeket. Ezen próbálkozások nem produkáltak kielégítő eredményeket ezért további javításokat szükséges alkalmazni a

²⁹ Kazhdan, Michael, and Hugues Hoppe. "Screened poisson surface reconstruction." *ACM Transactions on Graphics (ToG)* 32.3 (2013): 1-13.

modellen, amit egy későbbi fejezetben részletezünk.

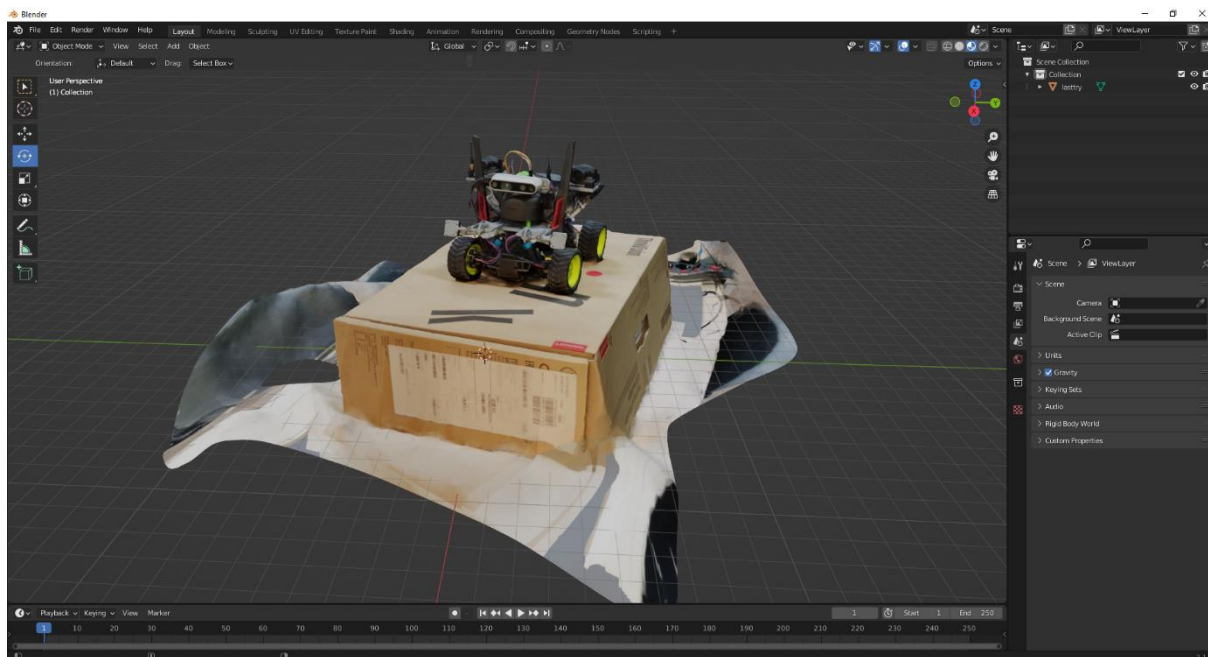


7. ábra végleges pálya képe

Autó és pálya modellek utómunkái

Az utómunkálatokra a Blender³⁰ nevű szoftver használtuk, ami kifejezetten 3D modellezésre, animálásra készült. A 3D szkennelés után rendelkezésre álló OBJ fájlt, beimportáltuk a már említett Blender szoftverbe, amit további munkák során tökéletesítettünk a későbbi használat céljából.

³⁰ Blender <https://www.blender.org/> [Letöltve: 2022. április]

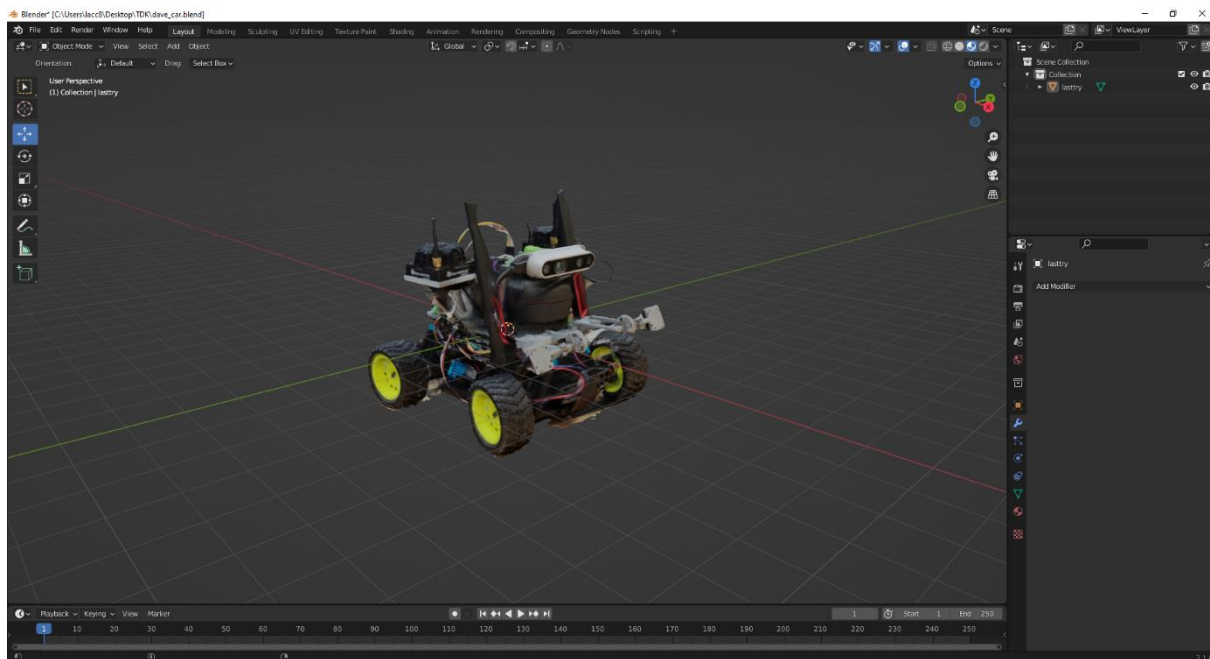


8. ábra jármű 3d szkennel változata Blenderben

A 8. ábra tartalmazza eredeti modell képét. Első lépésként a poligonok számát csökkentettük, mivel alap esetben a modell közel 1.000.000 darabot tartalmazott, aminek a megjelenítésének jelentős erőforrás igénye van. Erre a célra a Blenderben található egy módosító eszköz, aminek a neve decimate³¹. Ez az eszköz az egymás közelében lévő csúcspontokat olvasztja egybe így csökkentve a poligonok számát, bár ez információ veszteséhez vezethet, viszont az autó modell milliós poligon száma eleve túl részletes és szükségtelen számunkra. Végeredményként 10%-kal kevesebb, közel 100.000 poligonra csökkent a modell mérete.

Ezután a felesleges elemeket távolítottunk el, ilyen például a doboz, amin az autó szkennelésre került. Végül a négy kereket leválasztottam a főmodellről, ennek okára későbbi fejezetben térünk ki részletesen. A végeredményt a 9. ábra tartalmazza.

³¹ Decimate Modifier <https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html>
[Letöltve: 2022. április]



9. ábra javított jármű modell

Az elkészült pálya szkennelés eredményét is a Blender segítségével fejlesztettük tovább. Első lépésként itt is a poligon szám csökkentése volt a cél, amelyet a már ismertetett decimate eszköz segítségével oldottunk meg.



10. ábra pálya képe Blenderben

A 10. ábra mutatja, hogy az itt kapott eredménnyel még sok a probléma, többek között a padlón, a falon és a tárgyakon is megtalálhatóak lyukak és kitüremkedések, nincs egységes síkfelület. Ennek a javítására sculpting³² technikát alkalmaztuk, ami lényegében egy olyan eszköz, amely segítségével „agyagozhatunk” a modellel. A sculpting-on belül a flatten és smooth segítségével végeztük el a felületek tisztára simítását. Ezt követően a még problémás területek kitörölésre kerültek a modelltől, melyek helyettesítését egyszerű objektumokkal hajtottuk végre. Végül a megformázott modellen történt változások miatt a texturán is változtatni kellett, amit a texture baking³³ segítségével készítettünk el, ezzel a megoldással a modell jelenlegi mintázatát lehet képként lementeni. Utolsó lépésként mind két modellt FBX³⁴ formátumban került exportálásra.

Virtualizált környezet fejlesztése NVIDIA Isaac Sim segítségével

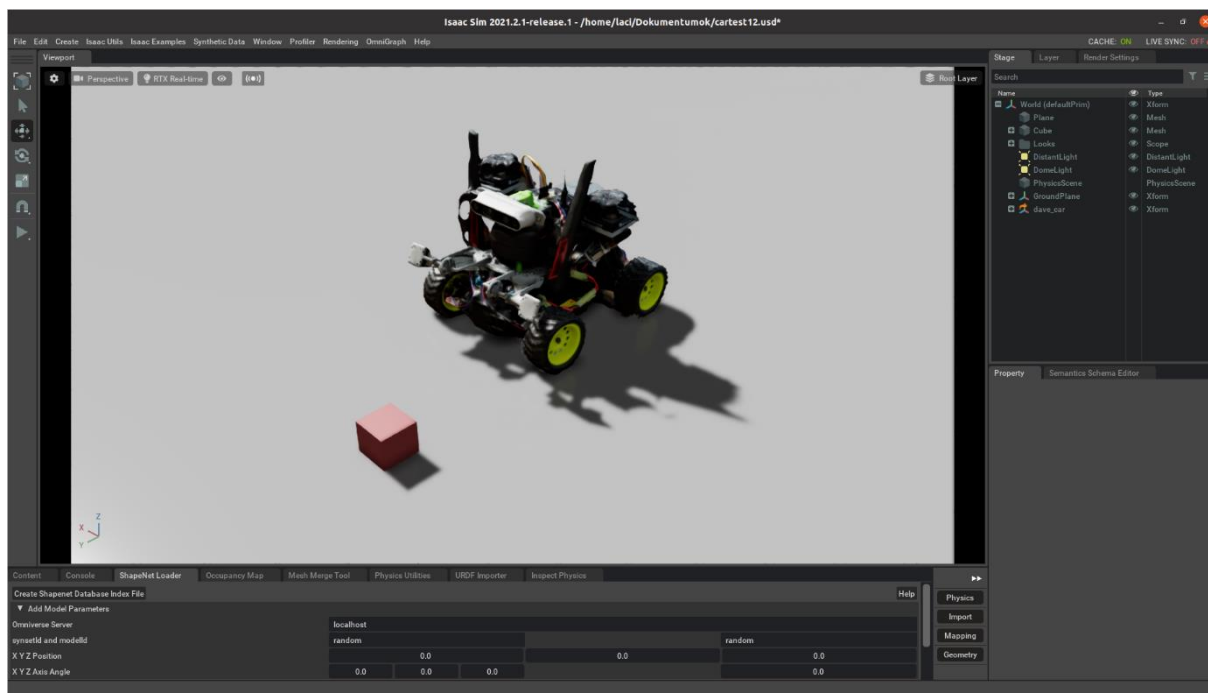
A környezet virtualizálására az NVIDIA által fejlesztett Isaac Sim³⁵ szoftver használtuk. Ezen program segítségével könnyedén készíthetünk fizikai szimulációkat, felhasználhatunk szenzorokat és támogatja a gépi tanuló módszereket is. Támogatja ezen felül az FBX fájl formátumot, így az elkészített modelleinket könnyen importálhatjuk a robotikai szimulátorba, a 11. ábra látható a jármű az importálás után.

³² Seamless Sculpting <https://www.blender.org/features/sculpting/> [Letöltve: 2022. április]

³³ Render Baking <https://docs.blender.org/manual/en/latest/render/cycles/baking.html> [Letöltve: 2022. április]

³⁴ FBX https://docs.blender.org/manual/en/2.80/addons/io_scene_fbx.html [Letöltve: 2022. április]

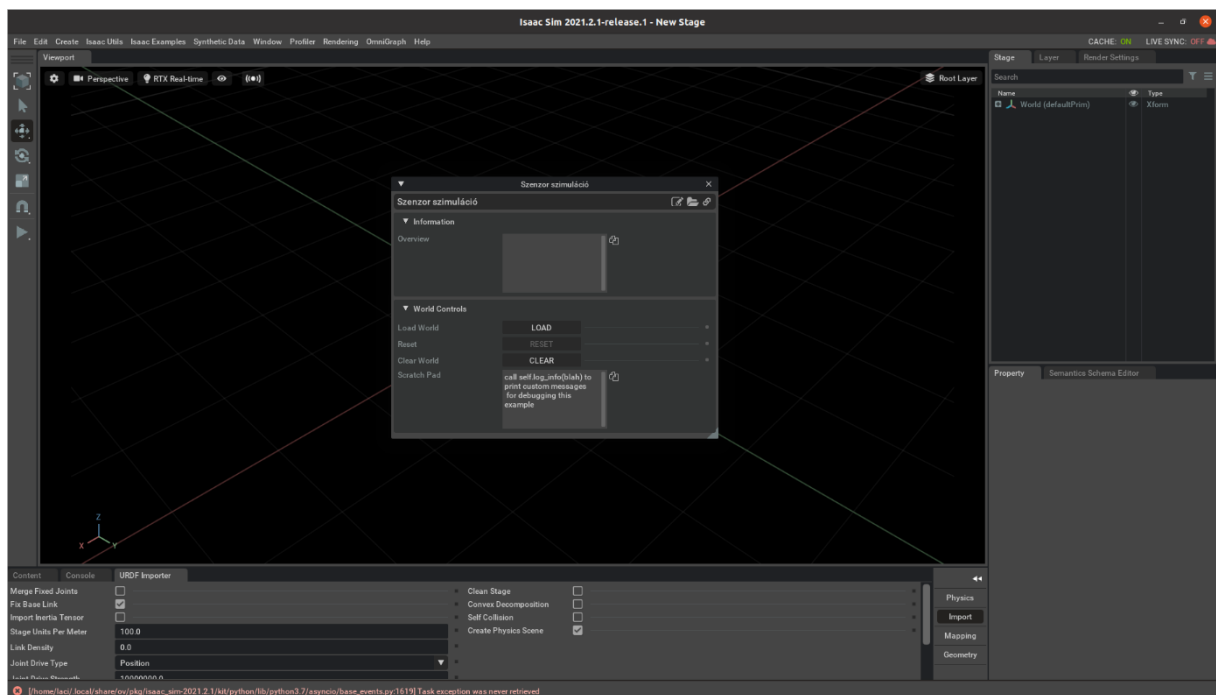
³⁵ NVIDIA Isaac Sim <https://developer.nvidia.com/isaac-sim> [Letöltve: 2022. április]



11. ábra beimportált jármű modell Isaac Simben

Az Isaac Sim programban szimulációkat extension³⁶ segítségével lehet létrehozni. Ehhez első lépésként létre kell hozni egy python alapú projektet. Ez a projekt bekerül az Isaac Sim menü sorába, innen kiválasztható és elindítható. Az extension teljes irányítást ad a szoftverhez, Python használatával betudjuk importálni az elkészült modelleket, el tudjuk helyezni őket a térben és beállíthatjuk a fizikai paramétereket, például a gravitáció mértékét is. Az általunk készített Python csomag jelenleg a következőkre képes, beimportálja és elhelyezi a pályát az origóban. Beimportálja és a pályára helyezi a járművet, létrehozza a fizikai szimulációhoz szükséges objektumokat. Beállítja, hogy a jármű fizikai szempontból aktív legyen, ezt egy *Rigidbody* paraméterrel teszi meg, ami arra szolgál, hogy jelezze a szimulátornak ez az modell egy véges nagyságú szilárd test. Ezután beállítja az autó súlyát a *mass* paraméter segítségével. Végül pedig importálja a 4 kereket hozzá csatolja az autókhoz és ennek is beállítja a fizikai paramétereit, kiegészítve súrlódás együtthatóval. Korábban azért volt szükség a kerék leválasztására a járműtől, hogy mint forgóalkatrészként tudjuk hozzákötni.

³⁶ Isaac Sim: Extensions API <https://docs.omniverse.nvidia.com/py/isaacsim/index.html> [Letöltve: 2022. április]

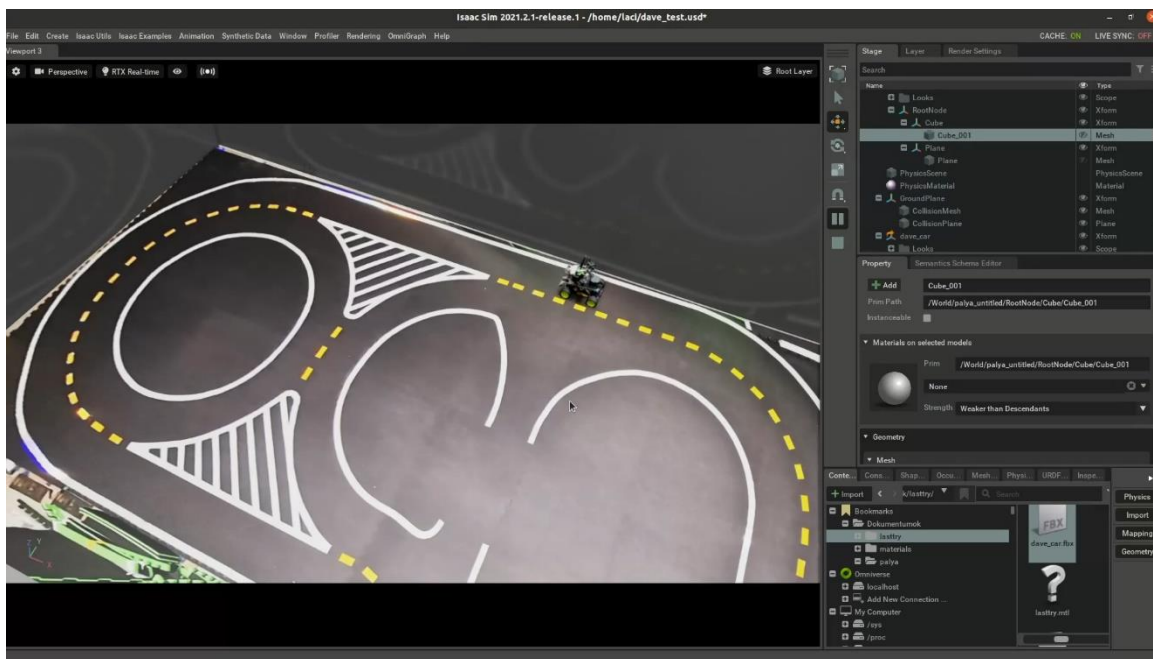


12. ábra Isaac Sim szenzor szimulációs extension

A végeredmény a 12. ábra látható, a LOAD gombbal tudjuk elindítani a szimulációt a RESET gombbal alaphelyzetbe állítani, a CLEAR gomb pedig törli az összes elemet a szimulációból és alaphelyzetbe állítja.

Virtualizált és valós környezet összekapcsolása

A következő lépés a virtualizált és valós környezet összekötése, ehhez tovább fejlesztettük az Isaac Sim-hez írt extensiont. Létre hoztunk egy UDP socketet, ami képes fogadni az adatokat a valós járműről. A kapott koordináták és vektorok alapján pedig elmozgatjuk a virtuális autót a pályán. Ehhez szükséges még a koordináták megfeleltetésének. Ehhez referencia pontokat határoztunk meg a két virtualizált és valós pályán. Ezekre a pontokra elhelyezve az autót a valós teszt pályán. A kapott koordinátákhoz viszonyítva kiszámoltuk a két koordináta rendszer, MarvelMind-os és a virtuális közötti különbségeket és ez alapján módosítottuk a bejövő adatokat. A végeredményként a valós jármű mozgásának tükrözését kaptuk a virtuális valóságban. A megfeleltetéssel azonban még akadnak problémák, mivel a MarvelMind beaconök 6 Hertz-es frissítéssel működnek, azért másodpercenként csak 6 koordinátát kapunk, ami többször a jármű remegéséhez és irányvesztéséhez vezet. Ennek a kiküszöbölésére a jövőben a GPS rendszereknél alkalmazott Kálmán-szűrő használatát tervezzük.



13. ábra Virtuális autó és pálya.

Szenzorok elhelyezése

Az utolsó lépés a virtualizált környezet tovább fejlesztésével kapcsolatban, a szenzorok elhelyezése a szimulációs járművön. Jelenleg három szenzort helyeztünk el a modellen. Egy darab RealSense³⁷ kamerát, ami a járműhöz van csatolva, tehát fix a lokális pozíciója a modellhez képest. Valamint két darab LiDAR szenzort. Egyet a jármű elejére és egyet a végéhez csatolva. Fontos megjegyezni, hogy ezeket a virtualizált szenzorokat be is kell kalibrálni, amit az általunk használt RPLiDAR³⁸ alapján készítettünk el.

A szenzor adatokat Python segítségével tudjuk lementeni a szimulációból.³⁹ Ezeket az adatok fogjuk felhasználni a továbbiakban az önvezető járművünk tanítása során.

³⁷ RealSense Camera <https://docs.nvidia.com/isaac/packages/sensors/doc/realsensecamera.html> [Letöltve: 2022. április]

³⁸ RPLIDAR A1 Datasheet. <https://www.generationrobots.com/media/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf> [Letöltve: 2022. április]

³⁹ Using Sensors: LIDAR https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/tutorial_advanced_range_sensor_lidar.html [Letöltve: 2022. április]

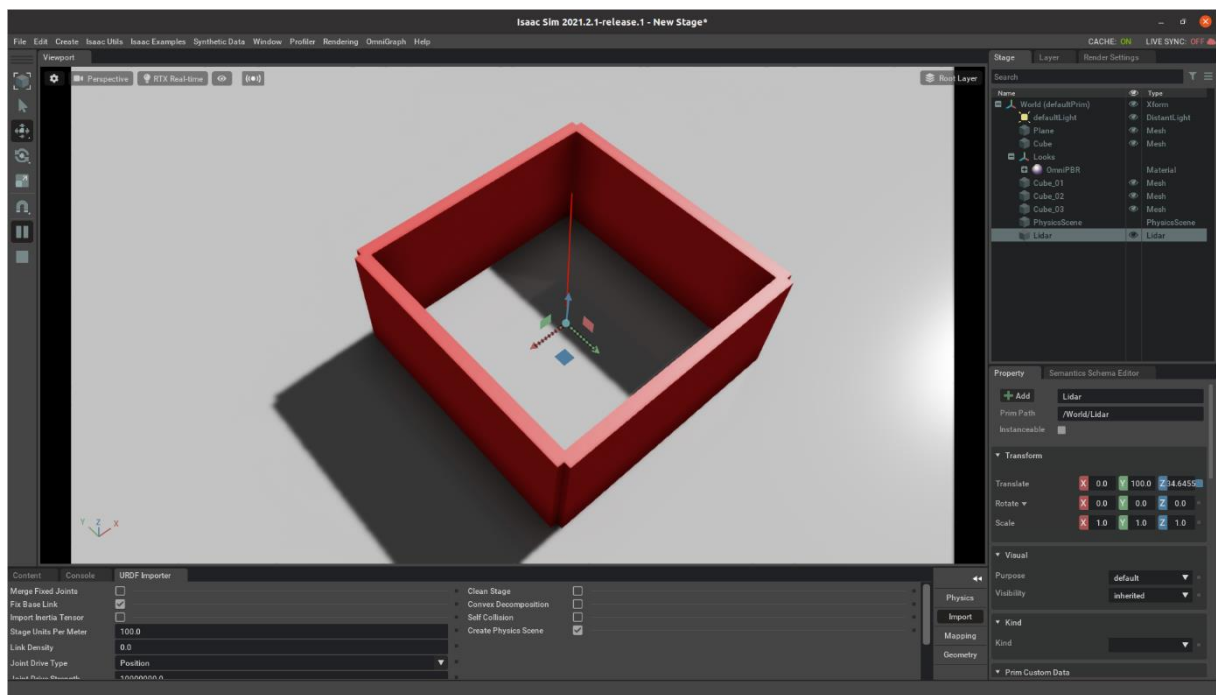
Szenzor adatok regisztrálása generatív hálózat segítségével

A virtuális és valós szenzor adatok eltérhetnek egymástól, többek között a virtuális környezetben fennálló steril környezet miatt. Több mérést is végeztünk és míg a valós LiDAR szenzorunk hibával terhelt, addig a virtualizált szenzor minden esetben közel tökéletes adatokat nyújt. Ahhoz, hogy a két szenzor adatait közelítsük egymáshoz lehetőségünk van véletlenszerű hibával terhelni az Isaac Sim szoftverben használt szenzort, ám ennél jobb megoldást nyújt, ha gépi tanulási módszert alkalmazunk és hamisítjuk az adatokat. Erre a célra megfelelő választás lehet a generatív hálózatok, amit például egy tanulmányban radar jelek generálására használtak⁴⁰. A szenzor adatok módosítására olyan hálózat választása szükséges, ahol nem generáljuk a jeleket, hanem módosítjuk azokat. Erre a célra megfelelő egy a Berkley Egyetemen készített hálózat, az úgynevezett CycleGAN⁴¹. Ebben a projektben képi adatokat módosítottak, például almákról készült képeket alakítottak át narancssá. A CycleGAN két generátor és két diszkriminátort használ. Az első G generátor megtanulja, hogy X adatból, hogyan jut el Y-ba, míg a második F generátor, azt, hogy hogyan jut el Y-ból X-be. A D_X diszkriminátor megtanulja a különbséget X és $F(X)$, míg D_Y diszkriminátor megtanulja a különbséget Y és $G(Y)$ között. A veszteségfüggvényünk pedig két részből fog állni. Az első eset, hogy X-et G segítségével Y-re alakítjuk majd F-fel vissza X-re és vesszük a különbségüket. A második eset pedig, hogy Y-ból indulunk ki ez röviden $Y-G(F(Y))$.⁴²

⁴⁰ T. Truong and S. Yanushkevich, "Generative Adversarial Network for Radar Signal Synthesis," 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-7, doi: 10.1109/IJCNN.2019.8851887.

⁴¹ Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
<https://junyanz.github.io/CycleGAN/> [Letöltve: 2022. április]

⁴² Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.



14. ábra Virtuális LiDAR tesz környezet

A generatív hálózathoz adatokra van szükség. Erre a célra a valós RPLiDAR szenzort belehelyeztük egy 1 méter oldalhosszú doboz középebe és rögzítettük a mérési adatokat. Ezután az Isaac Sim szoftverben is elkészítettük az 1 méter oldalhosszúságú dobozt - ez látható az 13. Ábrán - és ezen adatokat is rögzítettük. A két mérés közötti különbség 1% körüli. A tanításhoz létrehozott adathalmazomat két részből áll a virtuális és a valós szenzor adatok, mind a két halmazt további két részre osztottuk, 80% az adatoknak a Training és maradék 20% az adatoknak a Validation részbe került. A tanítást TensorFlow 2.6 és Keras 2.6 alatt végeztük egy NVIDIA 3060 RTX kártyán.

A végeredmény egy neurális hálózat, amellyel képesek vagyunk a virtualizált LiDAR szenzorokat megfeleltetni a valós szenzorokkal. Jelen esetben a virtuális adatok átalakítása után a különbség a korábbi 1% körüliről 0.1% körülire csökkent.

Összefoglalás

A dolgozat eredményeként létre jött egy szimulációs környezet az Isaac Sim szoftverben, ahol a valósággal közel megegyező körülmények között tudunk adatokat gyűjteni. Rendelkezésünkre állnak még ezen felül nagy részletességű modellek a járműről és a használt teszt pályáról, valamint maga a folyamat, hogy hogyan tudunk a jövőben 3D modelleket

készíteni a valós környezet szkennelésével. Ezen felül pedig elkészült egy olyan generatív hálózat, amely segítségével különböző LIDAR szenzor adatokat tudunk megfeleltetni egymásnak.

Az elkészült munka jelentőségét pedig a jövőbeli felhasználási területek adják. A virtualizáció segítségével lehetőségünk nyílik olyan eseteket is szimulálni és adatokat gyűjteni, amiket költséges vagy nehéz a valóságban előállítani, ilyen például a veszélyes közlekedési helyzetek szimulálása, amikor a szemből jövő jármű a mi sávunkban van, vagy a gyalogosok hirtelen irányváltása az útestre szabálytalanul. Ezen felül a szimulációban lehetséges különleges időjárási körülményeket létrehozni és ezen tesztelni az önvezető modellünket. Nem elhanyagolható az sem, hogy az önvezető járművek tovább fejlesztése előtt virtuális környezetben letudjuk tesztelni és eldönteni, az egyes szenzorok mely pozícióban nyújtanak a legjobb teljesítményt és így hova érdemes felszerelni azokat.

Érdeemes még megemlíteni a Reinforcement learning adta lehetőségeket, amely segítségével a virtuális környezetben lehetséges több milliószor elpróbálni egy-egy közlekedési problémát és tökéletesíteni az önvezetésre használt modellt a megfelelő jutalmakkal. Lehetséges a párhuzamosítás is, egyszerre akár több szimulációt is indíthatunk így gyorsítva fel a tanulás folyamatát.

Az elkészült kutatás egy hosszú út egyik fontos lépése, amellyel magasabbra tudunk jutni az önvezetés 5 szintjén belül.

Saját munka leírása:

Vasváry Gábor által elvégzett feladatok:

- Marvelmind rendszer alkalmazása, tesztelése, NIA, IA architektúrák felhasználása. Különböző beaconok tesztelése, különböző firmware-ekkel. Az alkalmazott megoldások összehasonlítása.
- Szerver-kliens szoftver kialakítása, Donkeyar rendszerrel való integráció fejlesztése. különböző kommunikációs lehetőség kutatása, fejlesztése, tesztelése.
- 3D digitalizációs módszerek kutatás, alkalmazása. Beolvasó és utómunkálatokhoz használt szoftverek tanulmányozása, felhasználása.

Mészáros László által elvégzett feladatok:

- 3D modellek Blender szoftver segítségével történő javítása, kiegészítése.
- NVIDIA Isaac Sim szoftverben virtuális környezet létrehozása a modellek alapján.
- Virtuális és valós környezet összekötése belső GPS koordináták alapján.
- NVIDIA Isaac Sim szoftverben a szenzorok létrehozása valós és virtuális adatok gyűjtése.
- CycleGAN elkészítése és a virtualizált és valós adatok megfeleltetése.

Irodalomjegyzék

- Society of Automotive Engineers International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*.
https://www.sae.org/standards/content/j3016_202104/ [Letöltve: 2022. április]
- Autonomous Vehicle Research Lab University of Debrecen. <https://autonomous-vehicle-research-group-ud.github.io/> [Letöltve: 2022. április]
- Marvelmind weboldal. <https://marvelmind.com/> [Letöltve: 2022. április]
- OSI, TCP/IP and Hybrid models - GeeksforGeeks.
<https://www.geeksforgeeks.org/osi-tcp-ip-and-hybrid-models/> [Letöltve: 2022. április]
- Introducing JSON. <https://www.json.org/json-en.html> [Letöltve: 2022. április]
- JSON Server. <https://www.npmjs.com/package/json-server> [Letöltve: 2022. április]
- Requests: HTTP for Humans™. <https://docs.python-requests.org/en/latest/> [Letöltve: 2022. április]
- Faster-than-Requests. <https://github.com/juancarlospaco/faster-than-requests> [Letöltve: 2022. április]
- Differences between TCP and UDP - GeeksforGeeks.
<https://www.geeksforgeeks.org/differences-between-tcp-and-udp/> [Letöltve: 2022. április]
- M. Masirap, M. H. Amaran, Y. M. Yussoff, R. A. Rahman, and H. Hashim,
“Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT),”
ISCAIE 2016 - 2016 IEEE Symposium on Computer Applications and Industrial Electronics, pp. 200–205, Sep. 2016, doi: 10.1109/ISCAIE.2016.7575063.
- What is a Part. <https://docs.donkeycar.com/parts/about/> [Letöltve: 2022. április]
- M. Masirap, M. H. Amaran, Y. M. Yussoff, R. A. Rahman, and H. Hashim,
“Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT),”
ISCAIE 2016 - 2016 IEEE Symposium on Computer Applications and Industrial Electronics, pp. 200–205, Sep. 2016, doi: 10.1109/ISCAIE.2016.7575063.
- 8 Most Popular IoT Protocols and Standards You Need to Know | SaM Solutions.
<https://www.sam-solutions.com/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview/> [Letöltve: 2022. április]

- A Complete Guide to IoT Protocols & Standards In 2021.
<https://www.nabto.com/guide-iot-protocols-standards/> [Letöltve: 2022. április]
- N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*, Oct. 2017, doi: 10.1109/SYSENG.2017.8088251.
- Labbé, Mathieu, and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation." *Journal of Field Robotics* 36.2 (2019): 416-446.
- CoAP vs HTTP | difference between CoAP and HTTP. <https://www.rfwireless-world.com/Terminology/Difference-between-CoAP-and-HTTP.html> [Letöltve: 2022. április]
- Mapware's Photogrammetry Pipeline, Part 1 of 6: Keypoint Extraction | Mapware.
<https://mapware.ai/blog/photogrammetry-pipeline-part-1-of-6-keypoint-extraction/> [Letöltve: 2022. április]
- E. P. Baltsavias, "A comparison between photogrammetry and laser scanning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 2–3, pp. 83–94, Jul. 1999, doi: 10.1016/S0924-2716(99)00014-3.
- D. Moon, S. Chung, S. Kwon, J. Seo, and J. Shin, "Comparison and utilization of point cloud generated from photogrammetry and laser scanning: 3D world model for smart heavy equipment planning," *Automation in Construction*, vol. 98, pp. 322–331, Feb. 2019, doi: 10.1016/J.AUTCON.2018.07.020.
- AliceVision | Meshroom - 3D Reconstruction Software.
<https://alicevision.org/#meshroom> [Letöltve: 2022. április]
- MeshLab. <https://www.meshlab.net/> [Letöltve: 2022. április]
- RTAB-Map | Real-Time Appearance-Based Mapping.
<http://introlab.github.io/rtabmap/> [Letöltve: 2022. április]
- Kazhdan, Michael, and Hugues Hoppe. "Screened poisson surface reconstruction." *ACM Transactions on Graphics (ToG)* 32.3 (2013): 1-13.
- N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J. Y. Ertaud, "Benchmark of visual SLAM algorithms: ORB-SLAM2 vs RTAB-map," *2019 8th International Conference on Emerging Security Technologies, EST 2019*, Jul. 2019, doi: 10.1109/EST.2019.8806213.

- D. Droeschel and S. Behnke, “Efficient continuous-time SLAM for 3D lidar-based online mapping,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5000–5007, Sep. 2018, doi: 10.1109/ICRA.2018.8461000.
- W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2D LIDAR SLAM,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 1271–1278, Jun. 2016, doi: 10.1109/ICRA.2016.7487258.
- X. Chen *et al.*, “OverlapNet: Loop Closing for LiDAR-based SLAM,” May 2021, doi: 10.15607/RSS.2020.XVI.009.
- M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, Mar. 2019, doi: 10.1002/ROB.21831.
- Blender <https://www.blender.org/> [Letöltve: 2022. április]
- Decimate Modifier
<https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html>
[Letöltve: 2022. április]
- Seamless Sculpting <https://www.blender.org/features/sculpting/> [Letöltve: 2022. április]
- Render Baking <https://docs.blender.org/manual/en/latest/render/cycles/baking.html>
[Letöltve: 2022. április]
- FBX https://docs.blender.org/manual/en/2.80/addons/io_scene_fbx.html [Letöltve: 2022. április]
- NVIDIA Isaac Sim <https://developer.nvidia.com/isaac-sim> [Letöltve: 2022. április]
- Isaac Sim: Extensions API <https://docs.omniverse.nvidia.com/py/isaacsim/index.html>
[Letöltve: 2022. április]
- RealSense Camera
<https://docs.nvidia.com/isaac/packages/sensors/doc/realsensecamera.html> [Letöltve: 2022. április]
- RPLIDAR A1 Datasheet. <https://www.generationrobots.com/media/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf> [Letöltve: 2022. április]

- Using Sensors: LIDAR
https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/tutorial_advanced_range_sensor_lidar.html [Letöltve: 2022. április]
- T. Truong and S. Yanushkevich, "Generative Adversarial Network for Radar Signal Synthesis," 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-7, doi: 10.1109/IJCNN.2019.8851887.
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
<https://junyanz.github.io/CycleGAN/> [Letöltve: 2022. április]
- Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.