Autonóm járművek komplex virtuális tesztkörnyezetének fejlesztése

Ormándi Tamás* Varga Balázs**
Tettamanti Tamás***

**MSc hallgató, BME, Közlekedés- és Járműirányítási Tanszék, Budapest, Stozcek u. 2. (e-mail: ormandi.tamas@edu.bme.hu),*
***PhD hallgató, BME, Közlekedés- és Járműirányítási Tanszék, Budapest, Stozcek u. 2. (e-mail: varga.balazs@mail.bme.hu),*
****Egyetemi docens, BME, Közlekedés- és Járműirányítási Tanszék, Budapest, Stozcek u. 2. (e-mail: tettamanti@mail.bme.hu).*

Absztrakt: Napjaink egyik kiemelt fejlesztési és kutatási területe az autonóm járművek fejlesztése és az ehhez elengedhetetlen tesztelés. Ezek komplex járműves tesztek (ViL – Vehicle in the Loop, SciL – Scenario in the Loop), melyek elvégzéséhez elengedhetetlen egy olyan virtuális környezet megalkotása, amely hűen reprezentálja a valóságot. Cikkünk egy ilyen komplex virtuális tesztkörnyezet rendszerelemeinek együttműködését és felépítését mutatja be, melynek fő elemei a SUMO mikroszkopikus forgalomszimulátor, a Matlab Simulink, a Unity 3D játékmotor és ezek interfészei. Az elkészült rendszer képes valós idejű kommunikációt megvalósítani a forgalomszimulátor és a játékmotor között a Simulink közvetítésével, lehetővé téve azt, hogy rugalmasan kezelve jelenítsük meg a forgalmat a járművek számától függetlenül és tetszőleges számú valós járművet helyezzünk el a szimulációban, akár valós idejű, akár korábban lemért adatok alapján. A Unity 3D játékmotor flexibilitásának köszönhetően képesek vagyunk járműdinamikát implementálni mind a SUMO által generált virtuális járművekre, mind a valós járműveket reprezentáló EGO járművekre, ezzel valósághűbbé téve a szimulációt.

1. BEVEZETÉS

Egy autonóm jármű fejlesztése hatalmas kihívást jelent a mérnökök számára. Rengeteg olyan változó játszik szerepet egy ilyen komplex rendszer létrehozása és tesztelése során, melyek egymásra hatása sokszor nem egyértelműen feltárható. Jó példa erre bármely olyan járműkomponens, melyben valamilyen mesterséges intelligencián alapú funkció van implementálva.

Ahhoz, hogy ezeket az összetett rendszereket kellő részletességgel megvizsgálhassuk komplex tesztkörnyezetre van szükség. Ha a tesztek azt kívánják meg, hogy azok valós járművel történjenek, de egy védett környezetben (pl. tesztpálya), remek lehetőséget kínálnak számunkra a SciL (Scenario in the Loop) és a ViL (Vehicle in the Loop) tesztek. A ViL tesztek koncepciója, hogy a tesztelni kívánt járművünk valós, de a forgalom többi eleme virtuális. A SciL tulajdonképpen ennek egy továbbfejlesztett változata, ahol már nem csak a járművünk fizikai tulajdonságai kerülnek tesztelésre, hanem a jármű egésze, beleértve a szenzorokat is egyfajta digitális másolatként reprezentálva azt. [1]

Segítségükkel nem csak időt és pénzt spórolhatunk, de olyan teszteket is elvégezhetünk, melyek végrehajtása a valós forgalomban túl nagy kockázattal járna vagy egyszerűen nem állnak rendelkezésre olyan körülmények az elérhető környezetünkben, melyekre szükségünk lenne.

A kutatók és fejlesztők számára rendkívül széleskörű eszköztár áll rendelkezésre az ilyen jellegű tesztkörnyezetek megalkotásához. A legnagyobb kihívást egy olyan tesztkörnyezet megalkotása jelenti, melyben több szimulációs szoftver működik együtt valamely komplex teszteset megvalósítására. Mint mindennek, a tesztkörnyezet összeállításához felhasználható szoftvereknek is megvannak a maga előnyei, hátrányai és korlátai. A létrehozott rendszernek kellő flexibilitással kell rendelkeznie ahhoz, hogy a jövőben felmerülő újabb és újabb mérnöki problémákra is megfelelő fejlesztéseket lehessen eszközölni.

Az általunk fejlesztett rendszer célja különböző valós és virtuális objektumok együttes szimulációja és realisztikus vizualizációja.

A rendszernek rugalmasan kell kezelnie a tesztekben résztvevő valós járműveket (EGO járművek), virtuális járműveket és más virtuális objektumokat. További kritérium, hogy a valós objektumokat (EGO jármű, tesztbábu stb.) valós időben vagy offline, valós mérés visszajátszásával szimulálhassuk. A valós és virtuális objektumok közös szimulációjával egy kevert valóság (mixed reality) hozható létre. Mindemellett törekszünk a lehető legvalósághűbb vizualizáció megteremtésére is, melynek természetesen nem csak esztétikai szerepe van, de szerepet játszhat olyan szituációkban, ahol a tesztelni kívánt funkció például kameraképen alapul. Ilyen esetekben nem elhanyagolható a vizualizáció valósághű megjelenése, legyen szó fényviszonyokról vagy különböző időjárásokról.

A cikk egy fő kontribúciója, hogy a rendszerben implementálásra került egy járműdinamika modell is, melynek segítségével a SUMO által generált forgalom járművei és az EGO járművek is egyenként paraméterezhető futóművet kaptak. Ezzel tovább haladunk a realizmus irányába és egyre széleskörűbb lehetőségek nyílnak meg

Online: ISBN 978-963-88875-6-6
Paper 19

Editor: Dr. Péter Tamás

előttünk a különböző teszt szcenáriók elkészítésekor, hiszen az egyes virtuális járművek ugyanúgy reagálni fognak a talaj által generált gerjesztésekre, mint ahogyan azt a valós járművek teszik. Ez az újítás azért lényeges, mert a SUMO egy 2-dimenziós forgalomszimulátor, amelyben a járműveknek csak a síkbeli koordinátái és állásszögei elérhetőek, függőleges mozgás itt nem értelmezhető. Hasonlóan, a rendszerbe bekötött EGO járművek lokalizációja is differenciál-GPS segítségével történik, amely ugyancsak nem ad pontos információt a kismértékű függőleges elmozdulásokról.

2. A RENDSZER FELÉPÍTÉSE, KOMMUNIKÁCIÓ

A megvalósult szimulációs rendszer 3 fő alappillére a SUMO forgalomszimulátor, a Matlab Simulink és a Unity 3D játékmotor. Ahhoz, hogy ezeket az önálló rendszereket együtt tudjuk használni, szükségünk van a különböző szoftverek közötti kommunikáció kialakítására, melynek megvalósítása az 1.ábrán látható.

2.1 Matlab Simulink

A kommunikáció középpontjában a Simulink helyezkedik el, ez az a rendszerelem, aki a szimulációs rendszer többi résztvevője között közvetíti az információkat. Ez a komponens felel a szimuláció indításáért és az egyes komponensek szinkronizációjáért. A Simulink – Unity 3D interfész bemenetei a SUMO és EGO járművek legfontosabb adatai, mint például a pozíciók, sebességek, headingek és kormányszögek. Ezenkívül olyan információkat is közvetít, mint a virtuális jelzőlámpák pozíciója és aktuális állapota.

A forgalomszimulátor a Simulinkkel a TraCI (Traffic Control Interface) segítségével kommunikál.

Alap esetben ez az interfész sem áll rendelkezésre Simulinkben, csak Matlabban, ezért a kommunikációhoz egyedi Simulink blokkokra van szükség, amely a TraCI Matlab függvényeit hívja meg.

Ennek a kommunikációnak a segítségével a SUMO-ban megjelennek az EGO járművek is és a saját maga generálta járművek érzékelik is ezeket, így a generált forgalom reagál a valós járműveink mozgására.

Miután sikeresen csatlakozik a TCP szerverként üzemelő saját fejlesztésű S-Function és a Unity 3D, a Simulink elindítja az általunk kiválasztott SUMO szimulációt és megkezdi a forgalomszimulátorral való kommunikációt is.

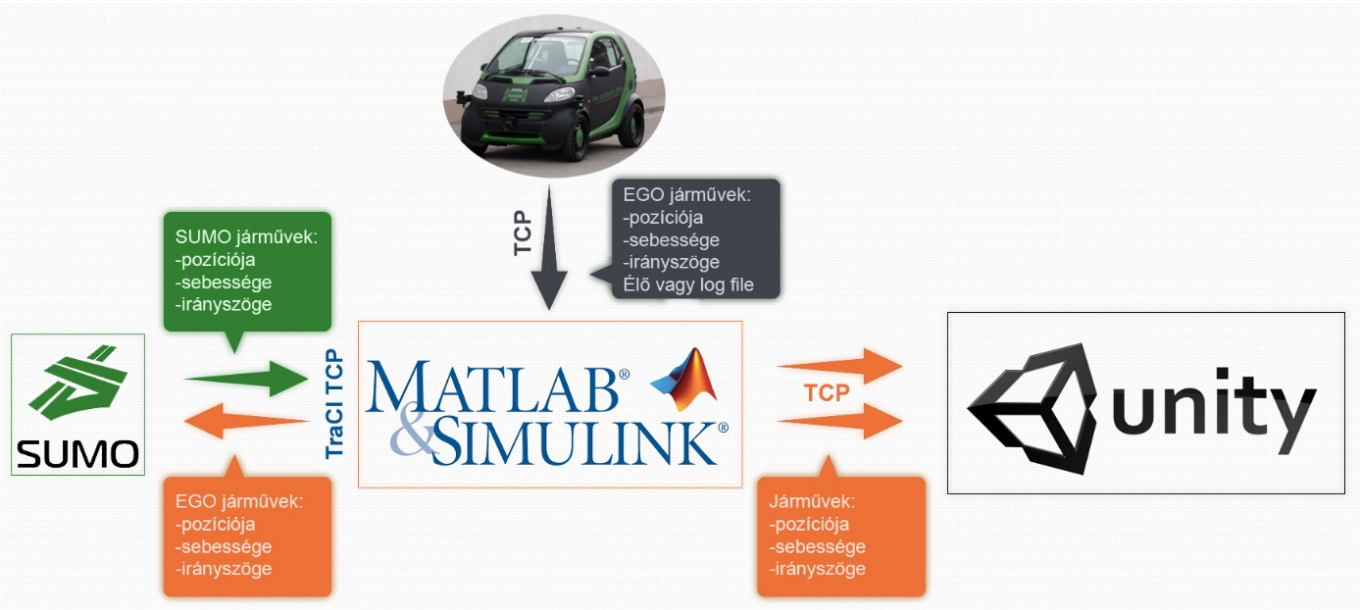
A Simulink szintén egy TCP interfész segítségével képes fogadni a valós járműből érkező adatokat vagy ha éppen arra van szükségünk, akkor képes feldolgozni a korábban rögzített méréseket egy egyszerű szöveges fájlból.

2.2 SUMO

A SUMO forgalomszimulátor a virtuális forgalom előállításáért és vezérléséért felelős. Ebben a szoftverben kell létrehozni azt a pályát, amin a járművek mozoghatnak és meg kell szabnunk a közlekedési szabályokat is, amelyeket a generált járművek betartanak. A korábban már említett TraCI segítségével könnyedén hozzáférünk a SUMO által generált járművek összes információjához, amelyek a 3D megjelenítéshez szükségesek.[2]

2.3 Unity 3D

A Unity 3D játékmotor alapvetően a 3D-s vizualizációért felelős, de mivel a teljes virtuális környezet egy komplett számítógépes játéknak felel meg, ezért ennél sokkal többre vagyunk képesek vele [3]. A Unity-ben található fizikai motor segítségével pedig egyszerű járműdinamikai



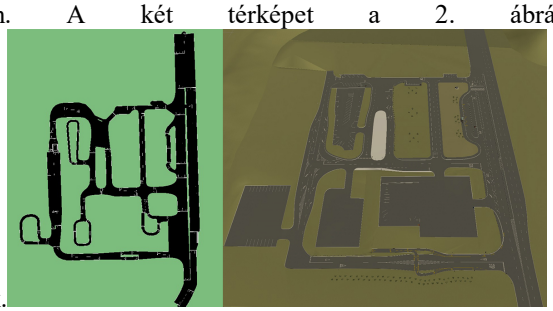
1. ábra A rendszerben megvalósuló kommunikáció

modellezést végezhetünk.

3. VIRTUÁLIS KÖRNYEZET FELÉPÍTÉSE

A virtuális környezet felépítéséért teljes mértékben a Unity 3D a felelős. Úgynevezett GameObject-ek segítségével vagyunk képesek a virtuális környezet elemeit felépíteni. A szimulációs rendszerben mi a ZalaZone tesztpálya lézerszkennelt változatát használjuk, amely GPS helyes és a pálya vertikális kiterjedéseit is tartalmazza. A szimulált környezetben bármely objektumnak adhatunk ún. collidereket, melyek segítségével a járművek képesek ütközni a környezeti elemekkel és a virtuális szenzorok segítségével is érzékelhetővé válnak.

A SUMO forgalomszimulátor NetEdit alkalmazásának segítségével (szabványos OpenDrive formátumból importálva) felépítésre került ennek a pályának a SUMO-s változata is, amely szintén GPS helyes, így már könnyedén összehangolható a két környezet és legenerálható a virtuális forgalom. A két térképet a 2. ábrán láthatjuk.



2. ábra SUMO (baloldalt) és Unity 3D (jobb oldalt) térképe

4. KOORDINÁTA TRANSZFORMÁCIÓK

A forgalomszimulátorban a járművek koordinátáit azok első lökhárítójának középpontja adja meg. Ahhoz, hogy a vizualizációban megjelenő járműveink pontosan kövessék a pozíciókat, ezt a koordinátát a Unity 3D-ben a jármű középpontjába kell helyeznünk.

Ugyanezt meg kell tennünk fordított esetben is, amikor az EGO jármű pozícióját helyezzük el a SUMO-ban. Ekkor a valós jármű GPS szenzorának adataiból kell kiindulni, melyeket először a szimulátor x-y koordináta rendszerébe kell vetíteni. Ehhez UTM (Universal Transverse Mercator) projekciót használtunk [6]. A forgalomszimulátorban modellezett hálózat nemcsak koordináta-helyes, de tartalmaz egy derékszögű koordináta rendszert is.

5. JÁRMŰDINAMIKA IMPLEMENTÁLÁSA

A GPS jel alapján modellezett járművek, illetve a forgalomszimulátor járművei nem rendelkeznek pontos függőleges irányú pozíciókkal, csak síkban (2D) mozognak. Ahhoz, hogy a 3D megjelenítőben ezek valósághűen szerepeljenek szükséges a hiányzó dinamika modell alapú számítása.

Ahhoz, hogy a virtuális járműveink számára működőképes dinamikai modelleket tudjunk implementálni, először meg kell értenünk, hogyan tudunk felépíteni egy olyan objektum rendszert, melyekre a dinamikai erők hatással lehetnek, illetve meg kell ismernünk az ehhez szükséges és rendelkezésre álló eszközöket a Unityben.

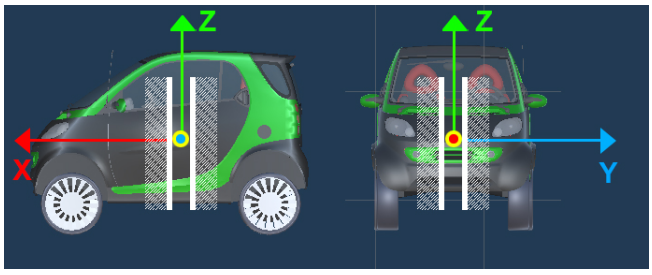
5.1 Raycasting és rigidbody

A raycasting a Unity 3D-ben egy olyan funkció, melynek segítségével képesek vagyunk egy sugarat kibocsájtani az általunk megadott pozícióból. Járműdinamika szempontjából a legfontosabb, hogy ezen sugarak végpontjaira erőket alkalmazhatunk, ezáltal adja magát a lehetőség, hogy a járművek kerekeit és a futóműveket egy-egy raycast sugár segítségével modellezzük. A raycast sugarak kezdőpontját a kerekeket reprezentáló objektumok középpontjába helyezzük el és vertikálisan eltoljuk felfelé, mintha csak a futómű kiinduló pontja lenne. A raycast sugaraknak 3 fontos paramétere van: a sugár kezdő koordinátája, a sugár hossza és a sugár végpontja. A felfüggesztés modellezésére ez a funkció azért is alkalmas, mert egyrészt a futómű gerjesztés hatására történő összenyomódását is tudjuk vele szemléltetni, másrészt a sugár végpontja, amely felütközik a talajra (illetve bármely colliderrel rendelkező felületre), tökéletesen megfelel arra, hogy a kerék talppontját reprezentálja számunkra. Ennek a pontnak a magasságát kiolvasva a gerjesztések mértéke is jól látható és rögzíthető a többi adattal együtt.[4]

A járművet reprezentáló objektum rendelkezik egy rigidbody (merevtest) komponenssel is. Ez azért fontos számunkra, mert a Unityben található fizika, például a gravitációs erő és minden egyéb általunk létrehozott erő, így a futómű erők is, ezekre a komponensekre képesek hatni. Ebben az összetevőben tudjuk beállítani a járművünk tömegét is és a Unity beépített fizikája automatikusan kiszámít számunkra olyan fontos paramétereket, mint a merevtest aktuális X, Y és Z irányú sebességei, valamint a szögsebességei.

5.2 Virtuális jármű szabadságfokai

A virtuális járműveket a beállított frekvenciával állítjuk mindig az aktuális pozícióba azok X és Y koordinátái, valamint irányszögük alapján, melyeket a SUMO küld ki, így tehát 3 szabadsági fokuk kötött. Az X és Y tengelyek körüli elfordulások és a Z tengely menti vertikális elmozdulás lehetővé teszi a dinamikai modellek alkalmazását.



3. ábra Szabadságfokok

5.3 Dinamikai modellek

Online: ISBN 978-963-88875-6-6
Paper 19

Editor: Dr. Péter Tamás

A virtuális járművünk felfüggesztését egytömegű egy szabadságfokú lengőrendszerként modellezzük. Ez a modell egy tömegpontszerűen kezelt test (tömeg) és a testet egy másik testtel (jelen esetben a talajjal) összekötő rugó és a lengéscsillapító együttes, absztrakt modellje. Azt feltételezve, hogy a vizsgált rendszer csak függőleges irányban képes mozgást végezni, a rendszer egy közönséges másodrendű differenciálegyenlet alkalmazásával írható le:

$$\frac{m}{4}\ddot{x} = -d\dot{x} + c(x_0 - x) + F(t) \quad (5)$$

ahol *m* a jármű tömege, $\ddot{x}$ a tömeg gyorsulása, $\dot{x}$ a tömeg sebessége, *x* a tömeg pillanatnyi pozíciója, x_0 a rugó terheletlen hossza, *c* a rugóállandó, *d* a csillapítási tényező és *F(t)* a külső gerjesztő erő az idő függvényében.

A modellben a rugóerő a rugó összenyomódásával, a csillapítóerő pedig a függőleges sebességével arányos.

A másik dinamikai modellünk, a jármű tömeg átterhelődésének hatását írja le a kerekeken ébredő vertikális erők formájában, melyeket a következő képletek segítségével számítunk:

$$F_z = \frac{1}{2}m\left(\frac{L_2}{L} - \frac{[(a]_x + g)h}{[(a]_z + g)L}\right)[(a]_z + g) - m\left(\frac{L_2}{L} - \frac{[(a]_x + g)h}{[(a]_z + g)L}\right)\left(\frac{[(a]_y + g)h}{E}\right) \quad (6)$$

ahol *L* a tengelytávolság, L_2 a jármű hátsó tengelyének távolsága a tömegközépponttól, a_x, a_y és a_z az X, Y és Z irányú gyorsulások, *g* a nehézségi gyorsulás, *E* a nyomtáv és *h* a jármű tömegközéppontjának magassága.[5]

A merevtest sebességeit felhasználva könnyedén kiszámíthatjuk az egyes tengelyek mentén fellépő gyorsulásokat numerikus differenciálással:

$$a = \frac{v_2 - v_1}{\Delta t} \quad (7)$$

ahol *a* a gyorsulás, v_2 az aktuális sebesség, v_1 az előző függvény futásban kiszámított sebesség és Δt a szimuláció lépésköze.

Ezeket az erőket a korábban említett raycast segítségével tudjuk alkalmazni a kerék és a talaj érintőpontjában, vertikálisan. A modellezett erők hatására megvalósul a jármű oldaldőlése, bólintása és rugózása. Fontos megjegyezni, hogy a járművön alkalmazott kényszerek miatt az csak egy adott pont körül képes elfordulni (3. ábra). Ezen pont megfelelő megválasztásával (momentán centrum) a dinamika jelleghelyes lesz ám nem elegendően pontos dinamikai mérések megvalósítására.

Mint ahogyan arról már korábban szó volt, az egyes virtuális járművek, melyek játékon belüli modelljei eltérőek, véletlenszerűen kerülnek legenerálásra, ezért is fontos, hogy képesek vagyunk minden egyes jármű paramétert egyenként beállítani, ezáltal növelve a virtuális forgalom diverzitását.

6. SZIMULÁCIÓS EREDMÉNYEK

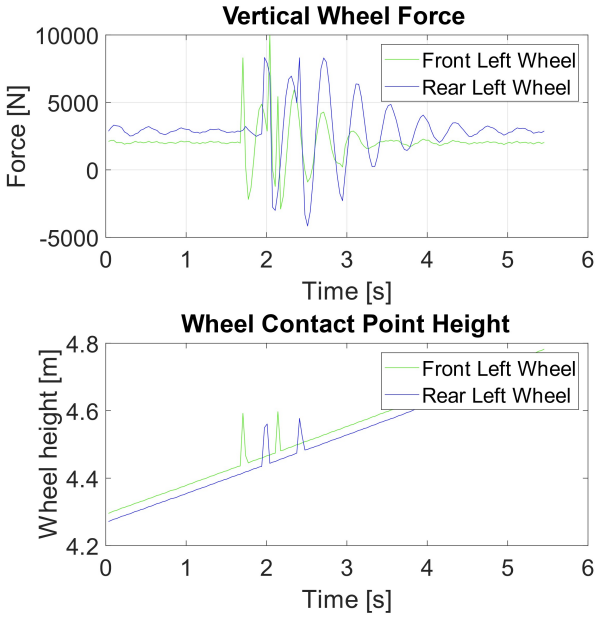
6.2 Fekvőrendőrön való áthaladás

A Unity 3D-ben használt lézerszkennelt virtuális környezet, ugyan tartalmazza a vertikális információkat, de a tesztpálya nem rendelkezik olyan gerjesztő elemekkel az úton, melyek a dinamikai modellek működését jól szemléltetnék. Ennek érdekében készítettünk egy egyszerű szimulációt, mely során 2 fekvőrendőrt helyeztünk el a virtuális jármű útvonalán.

4. ábra Fekvőrendőr szimulációja

A jármű geometriai paraméterei egy valós Smart Fortwo adatai alapján lettek beállítva. A futómű paramétereket empirikus úton állítottuk be, a lengések szemléltetéséhez megfelelően.

Először vizsgáljuk meg a fekvőrendőr által keletkezett gerjesztés hatására létrejövő kerékerőket.

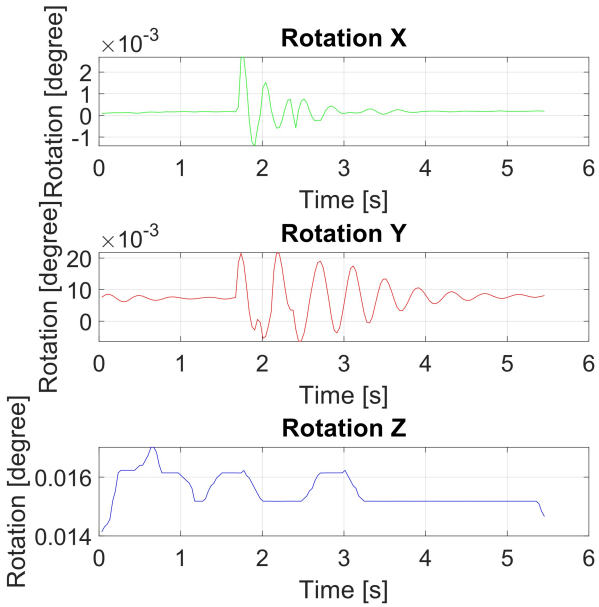


5. ábra Bal első és hátsó kerékerők és gerjesztések

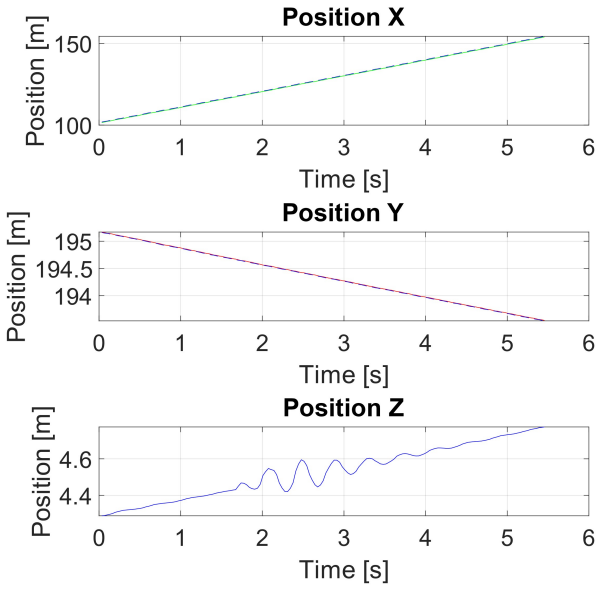
A 5. ábrán láthatjuk a bal első és a bal hátsó kerekeken mért erőket, illetve a két kerék vertikális pozícióját. Jól látható a vertikális pozícióból, hogy a pálya kb. 1.5%-ot lejt. Emellett jól látható a két fekvőrendőr okozta gerjesztés, melyek a két kerék esetében időben eltolva láthatóak. A kerekeken tapasztalható erőkön jól kivehető a rugó és a csillapítás

okozta lengés és annak lecsengése. A fekvőrendőrök okozta lengéseken kívül figyelembe kell vegyük, hogy a Unityben létrehozott talaj is egy colliderrel rendelkező több elemből álló rendszer, melyek illesztései nem feltétlenül pontosak és előfordulhatnak hibák rajta, ezáltal gerjesztéseket is okozhat bármely mérés során. Ez jól látható a fekvőrendőr előtti szakaszon, ahol egy kisebb lengés látható.

A merevtest rotációinak (6. ábra) vizsgálatából kiderül, hogy a járművünkre ható erők hatására létrejön a kocsiszekrény dőlése és bólintása is, melyek során szintén jól kivehető a lengés. Megfigyelhető a SUMO által kiküldött irányszög alakulása is.



6. ábra Rotációk



7. ábra X, Y és Z tengely menti pozíciók

A mérési adatok vizsgálata során egyértelműen megállapítható, hogy a futómű dinamikai modellje működik, hiszen a 7.ábrán látható függőleges pozíció adatai is rendkívül szemléletesen mutatják be a modellre jellemző jól ismert lengést.

6.3 Pozíciók differenciája

Megvizsgáltuk a SUMO által kiküldött X és Y pozíciókat és összehasonlítottuk a Unityben mért pozíciókkal. Az értékekben nem konstans eltérés figyelhető meg. Ennek valószínűleg az az oka, hogy a virtuális járművön működő dinamikai modellre ható erők miatt a pozíció egy Δt idő alatt elmozdulnak. Az eltérések nagysága függ a szimuláció beállításaitól, a járműparaméterektől és a sebességtől is. Az eltérés átlagosan 10-30 cm.

7. ÖSSZEFOGLALÁS

Munkánk során létrehoztunk egy olyan komplex szimulációs környezetet, amely napjaink ViL és SciL tesztjeinek követelményeit teljes mértékben kielégítik. A rendszer képes valós időben kommunikációt folytatni a szimulációban résztvevő szoftverek és valós járművek között, úgy, hogy az EGO járművek és a virtuális járművek látják egymást a virtuális környezetben és képesek reagálni egymásra. Mind a járművek mind a környezet virtuális másolatai hűen reprezentálják a valós megfelelőjüket. Az összeállított rendszer rugalmas maradt bármilyen irányú fejlesztésre. Az eredmények alátámasztják, hogy a szimulációban használt dinamikai modellek működőképesek a létrehozott korlátozott lehetőségek ellenére is.

A mérési eredményekben kapott adatokat a későbbiek folyamán validálni kell, és annak eredménye alapján lehet továbbfejleszteni a rendszer járműdinamikai komponenseit.

A szimulációs rendszerben további potenciális fejlesztési lehetőségek rejtőznek. A járműdinamikai implementáció a validáció nélkül a vizualizáció szempontjából teljesen megfelelő. Amennyiben a validálás megerősíti az eredményeket, akkor további koszimulációkat hozhatunk létre a Unity 3D és a Simulink között. A rendszerbe implementálhatóak más modellek is, legyen szó fékről, kerékmodellekről, ellenálláserőkről vagy akár egy teljesen virtuális EGO jármű teljes hajtásláncáról.

Játékmotorról lévén szó, akár egy olyan SciL teszt is elképzelhető, melyben a forgalmat valós emberek által irányított virtuális járművek alkotják, akik a valós vezetési stílusukat szem előtt tartva közlekednének a virtuális környezetben az EGO jármű közelében. Ebben az esetben olyan közlekedési helyzetek is megjelenhetnek a tesztelés során, amelyet egy forgalomszimulátor nem feltétlenül tud legenerálni.

KÖSZÖNETNYILVÁNÍTÁS

EFOP-3.6.3-VEKOP-16-2017-00001: Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási

technológiák területén - A projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

REFERENCIÁK

[1] Horváth, Márton and Lu, Qiong and Tettamanti, Tamás and Török, Árpád and Szalay, Zsolt. (2019). „Vehicle-In-The-Loop (VIL) and Scenario-In-The-Loop (SCIL) Automotive Simulation Concepts from the Perspectives of Traffic Simulation and Traffic Control.” *Transport and Telecommunication Journal.* **20**, 153-161. 10.2478/ttj-2019-0014.

[2] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," 2018 *21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2018, pp. 2575-2582, doi: 10.1109/ITSC.2018.8569938.

[3] M. Szalai, B. Varga, T. Tettamanti and V. Tihanyi, "Mixed reality test environment for autonomous cars using Unity 3D and SUMO," *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlany, Slovakia, 2020, pp. 73-78, doi: 10.1109/SAMI48414.2020.9108745.

[4] Unity Technologies, „Physics.Raycast”, *Unity User Manual (2020.2 alpha),* https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Physics.Raycast.html (Accessed: 2020)

[5] Kun Jiang, A. Pavelescu, A. Victorino and A. Charara, "Estimation of vehicle's vertical and lateral tire forces considering road angle and road irregularity," *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, 2014, pp. 342-347, doi: 10.1109/ITSC.2014.6957714.

[6] François Beauducel (2020). LL2UTM and UTM2LL (https://www.mathworks.com/matlabcentral/fileexchange/45699-ll2utm-and-utm2ll), MATLAB Central File Exchange. Retrieved July 6, 2020.