UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Allan Mitt

Simulation of Tartu City Centre for Testing Autonomous Vehicles

Bachelor's Thesis (9 ECTS)

Supervisor: Tambet Matiisen, MSc

Tartu 2021

Simulation of Tartu City Centre for Testing Autonomous Vehicles

Abstract:

Before testing an autonomous vehicle in the real world, it would be more efficient to try it in the simulation representing the real-life setting. This thesis aims to describe the designing process behind creating an adequate representation of Tartu city centre within the simulation.

Keywords:

Simulations, environments, autonomous vehicles, computer graphics, Unity, Blender, RoadRunner, SVL Simulator, high definition map, orthographic map, point cloud

CERCS: P175 Informatics

Tartu kesklinna simulatsioon autonoomsete sõidukite testimiseks

Lühikokkuvõte:

Enne autonoomse sõiduki testimist reaalses maailmas oleks tõhusam katsetada seda simulatsiooni keskkonnas. Selle lõputöö eesmärk on kirjeldada töökäiku, kuidas loodi adekvaatselt toimiv maailm simulatsiooni tarbeks, mis imiteeris Tartu kesklinna.

Võtmesõnad:

Simulatsioonid, keskkonnad, autonoomsed sõidukid, arvutigraafika, Unity, Blender, RoadRunner, SVL Simulaator, kõrglahutusega kaart, ortograafiline kaart, punktipilv

CERCS: P175 Informaatika

Tale of Contents

1. Introduction

An autonomous vehicle is a car that can drive itself from a starting point to a predefined destination automatically with little or no human input. It uses various technologies and sensors such as global satellite navigation technology, lidar, and radar to recognise its environment and move safely.

According to the information gathered from online articles [1][2][3], making sure that the autonomous vehicle behaviour algorithms work adequately requires a lot of testing through a wide range of test scenarios. It is not financially and practically feasible to perform all of these tests in a real-world setting. Using virtual testing instead can protect the developers from costly crashes and incidents out on the road that can potentially happen. The simulation will be a key here. By using advanced simulation tools in a virtual setting, they can run these scenarios quickly and cost-effectively. Doing that will accelerate development time and limit the number of physical road tests needed. It is the most practical way to evaluate autonomous vehicles performance over the billions of kilometres of test driving to make sure they are safe.

There are many simulations available to fulfil that role. The top performers on the commercial side are NVIDIA DRIVE Sim[1] and rFpro[2]. When it comes to the open source market, one has a lot to pick from, including CARLA[3], AirSim[4], SVL Simulator[5] and Deepdrive[6]. For this project, SVL Simulator was selected as the platform to run the simulation due to its good integration with autonomous driving software used in the Autonomous Driving Lab at the University of Tartu.

Every simulator needs an environment that resembles the real-life setting as closely as possible to run the test scenarios. That real-time digital counterpart of a physical object is called a digital twin. This thesis describes the design process behind creating an adequate representation of Tartu city centre within the simulation where autonomous driving technology (Autoware[7], Apollo[8]) can be used to evaluate the test vehicle's performance in a virtual environment. In the process, various software applications and input data were used to produce realistic results.

This thesis is divided into eight chapters. Chapter two, Background, provides a more in-depth look into autonomous vehicles, simulations and what is a digital twin. Chapter three, Software Used, describes the software used to create the digital twin and the selection process. Chapter four, Input Data, reviews the input data used and the goal behind it. In chapter five, Methodology, the actual design process is described in detail. In chapter six, Results, different test results and various facts about the digital twin are reported. Chapter seven, Conclusion and Future Work, briefly visits what was done good, what could be done better, and the plans associated with the future.

[1] https://www.nvidia.com/en-us/self-driving-cars/simulation/
[2] https://www.rfpro.com/
[3] http://carla.org/
[4] https://github.com/Microsoft/AirSim
[5] https://www.svlsimulator.com/
[6] https://deepdrive.voyage.auto/
[7] https://www.autoware.ai/
[8] https://3data.io/product-apollo/

2. Background

The idea is that a self-driving car can drive itself with basically no human input. Right now, companies worldwide are working hard to have a version of transportation available that requires no human interaction to make driving safer for everyone.

Many would never think a car without having a human in control can be safe. Who is going to make all these difficult decisions? How can a machine have all the experience of a person who has been driving for years? Contrary to everyone's belief and based on data collected, humans are actually terrible drivers and prone to accidents. More than 90% of accidents on the roads are caused by human error (due to increased cell phone usage and in-car entertainment system options, it is unlikely to change anytime soon) [4]. These accidents could have been avoided if the person in charge of the wheel had paid more attention. That is something that the autonomous vehicle has no trouble doing. Looking at the data, it is evident that self-driving cars are by far a safer option compared to a human counterpart when it comes down to who is in the driving seat [5][6]. Based on the data gathered by Google self-driving car project[9], none of their accidents were a fault caused by an actual car [7]. Pictures painted by that fact should give a real push when it comes to using autonomous vehicles. Even the billionaire Elon Musk has stated that self-driving cars can be a reason to ban manual driving in the coming future [8].

Summarising data found on the topic [2][9][10], to teach a human to drive, one just needs to learn basic vehicle functions. There is no need to teach how to distinguish between an old-looking sign and a new one or if the sign obscured by fog is conveying the same information as before. When developing autonomous systems, one must assume it doesn't initially know anything. Based on that, the training process will take a long time and needs a lot of road-testing. The widely accepted number among the industry is one billion miles or 1.6 billion kilometres to develop an autonomous driving system [9]. The world's leading autonomous driving company Waymo has accumulated 9 million miles in road testing through nine years by investing millions of dollars [9]. Even by increasing that tenfold, it still would take around 100 years to complete the validation process solely relying on road-testing [9]. Also, real-world road tests don't actually provide genuine validation for autonomous systems because two different test cases can never have the same conditions. In a world of science, two plus two is always four, but no two situations can ever be precisely equivalent in the real world. How many kilometres an autonomous vehicle has driven on the road or virtually is not instantly a metric of success when it comes to training. Ensuring that a car is safe and roadworthy depends on how many „smart miles" it has covered and how many scenarios were tested while at it [10]. Deciding which methods are needed to test the sensor perception, AI, and general vehicle design is essential. For example, if the ball bounces into the road, a human driver most likely predicts that a child might be chasing the ball and slows down in advance. An autonomous system might not understand that connection. That is one of many reasons also to include virtual training instead of just driving a vehicle on a street where it could potentially injure somebody before the connection between seeing a ball and slowing down is made. Testing through simulations in addition to testing on the road is the only practical way to accelerate the time needed to market, reduce costs and improve product quality.

Testing all these scenarios in a simulator needs an environment to play them out. Here is where the concept of digital twin comes in. It is a digital representation of a real-life location

[9] https://waymo.com/

(in this project's scope) where different elements such as roads, buildings, vegetation, and traffic infrastructure are designed to mimic the real-life counterparts. Everything is in sync location wise. When one checks GPS coordinates in simulation, they correspond to precisely the exact location on an actual map. Realistic traffic rules are also applied along with dynamic variables such as the number of road users, weather effects and time of day that can be adjusted. Having control over various road situations, traffic volumes, traffic lights, and other nuances gives the testers the ability to simulate an autonomous vehicle's behaviour in different conditions. By doing all that, one can obtain high-quality information about the road safety of the autonomous vehicle tested before even a single kilometre is driven on the actual road.

3. Software Used

The current project was created using various types of input data and specific programs to combat each challenge. Because of that, different software environments were utilised in the process. This chapter describes the used software programs, what they did and why they were selected.

3.1 Blender[10]

According to the Blender manual's introduction page [11], it is a free and open-source 3D creation suite. It has a wide variety of tools, making it a suitable choice for almost any media production. The key features are modelling, rendering, animation and rigging, video editing, VFX, compositing, texturing, and many types of simulations. It is also cross-platform, with an OpenGL GUI, has high-quality 3D architecture and active community support.

Blender was selected as the prefered modelling software for the project due to the fact of having previous experience with the product. Blender is also the most popular free modelling tool, runs on almost any machine and is easy to use. The software was used to create (model and texture) different objects located in Tartu city centre (bridges, buildings etc.).

3.2 RoadRunner[11]

As stated on MathWorks's RoadRunner description page [12], the RoadRunner is an interactive editor for designing 3D scenes to simulate and test automated driving systems. One can insert signs, signals, guardrails, and road damage, as well as foliage, buildings, and other 3D models. RoadRunner also supports the visualisation of the lidar point cloud, aerial imagery, and Geographic Information Systems data. Scenes built in the RoadRunner environment can be exported into several formats. The exported scenes can then be used in different automated driving simulators and game engines.

When it comes down to creating the road network from scratch, there is no better alternative to RoadRunner as the mentioned software is precisely designed for that purpose. It was used to create the road network, add the vegetation, position, and develop structures situated in the Tartu city centre. All the development was done based on the point clouds, aerial imagery, GIS (geographic information system) and vector data.

3.3 Unity[12]

Unity is a game engine and robust cross-platform integrated development environment for developers [13]. As a game engine, Unity provides many important built-in features, like physics, 3D rendering and collision detection [13].

Unity was selected as the default game engine for the project because of having previous experience in the environment and the software being more suited to small scale projects. It was used to merge all the different parts of the project and add some additional features (traffic and traffic lights, street lights, bus stops, etc.). The result was a digital map of the Tartu city centre usable in the SVL Simulator environment.

[10] https://www.blender.org/
[11] https://uk.mathworks.com/products/roadrunner.html
[12] https://unity.com/

3.4 SVL Simulator[13]

From SVL Simulator's introduction page [14], the SVL Simulator is a platform used for autonomous vehicle development. SVL Simulator includes simulation software, content and plugins that enable various use cases and the cloud environment that allows large scale testing. It can help to simulate a virtual environment, autonomous systems and their sensors, traffic, and other dynamic objects.

SVL Simulator was the choice because of being the most popular simulation platform based on the Unity game engine and having integration with Autoware software. A simulator was used to assess the performance and the results.

3.5 Other Software

This chapter describes the other used software programs, what they did and why they were selected.

3.5.1 Cloud Compare[14]

CloudCompare is a point cloud editing and processing software allowing automatic creation of meshes from them. It was picked as the prefered software because of the recommendations.

3.5.2 Inkscape[15]

Inkscape is a vector graphics editor used to create vector images. As it was very beginner-friendly, it was the pick for that task.

3.5.3 Java OpenStreetMap Editor[16]

Java OpenStreetMap Editor is an editor for OpenStreetMap (OSM). It supports loading GPX exchange format tracks, background imagery, and OSM and allows editing of the OSM data (nodes, ways, and relations) and their metadata tags. It was used as the preferred software because of the recommendations.

[13] https://www.svlsimulator.com/
[14] http://www.cloudcompare.org/
[15] https://inkscape.org/
[16] https://josm.openstreetmap.de/

4. Input Data

All the base data for the project was gathered from the Estonian Land Board website[17]. The elevation data, topographic data and orthophotos were used.

4.1 The Elevation Data

According to the Geoportal's Elevation Data page [15], the Estonian territory has been covered by LIDAR collected elevation data. LiDAR material was in LAZ format, and the data structure was a point cloud. An example of the point cloud data in the RoadRunner environment is shown in Figure 1.

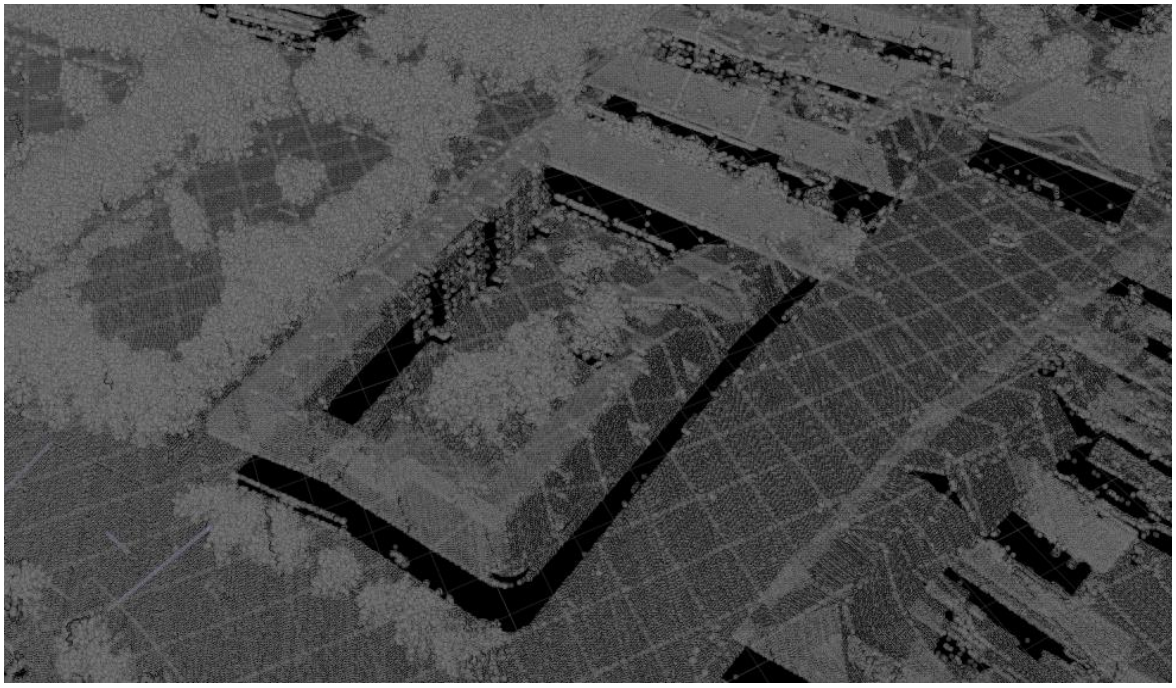

Figure 1: Example of a point cloud structure in LAZ format from Tartu's Town Hall Square. All the tiny dots represent the elevation at that point.

The whole area used for the project had the point cloud data available.

4.2 The Orthographic Data

Based on the info on Geoportal's orthophotos page [16], an orthophoto is an aerial photo from which distortions caused by terrain relief, camera tilt relative to the ground at the moment of exposure and camera central projection are removed. Digital orthophoto's pixel size was 10 cm. An example of orthographic data is presented in Figure 2.

4.3 The Topographic Data

The topographic data was used to set the ground and structure boundaries on top of the orthographic map. An example is presented in Figure 2, where the topographic information is the white line bordering the roads and buildings. It was layered on top of the orthographic photo of the same region.

[17] https://geoportaal.maaamet.ee/eng/

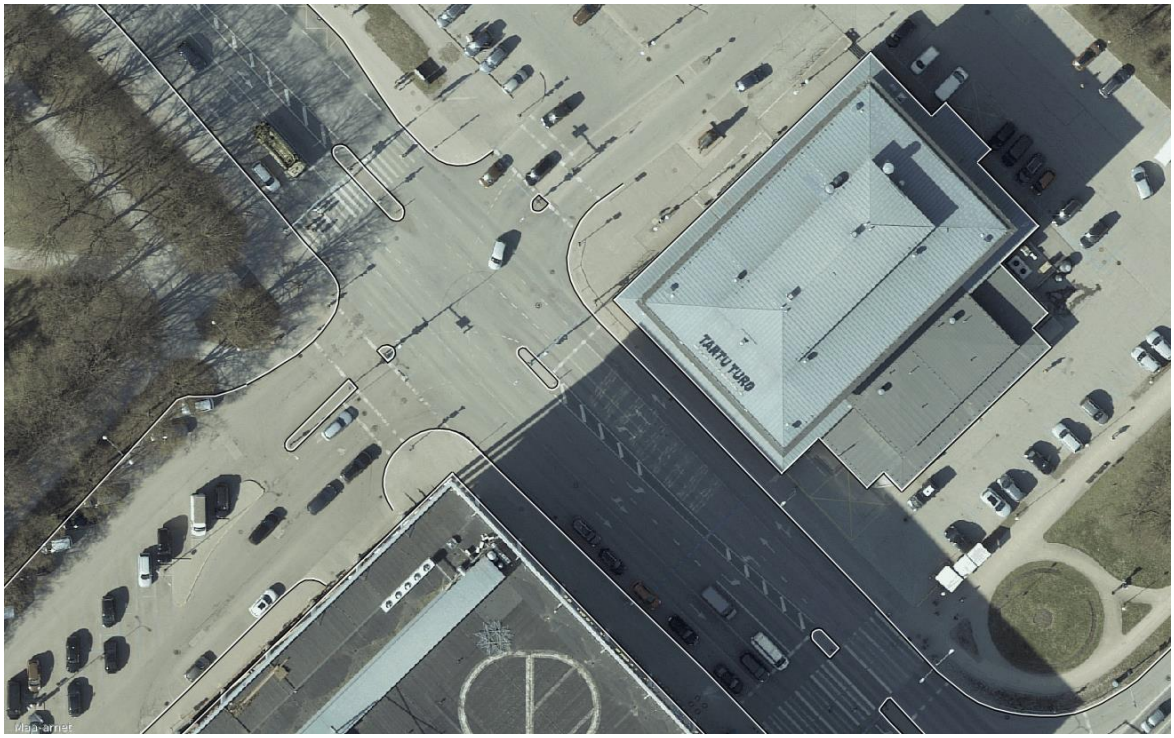

Figure 2: Example of topographic data on top of the orthographic map.

All the data was downloaded from the Estonian Land Board Geoportal[18].

4.4 Google Street View[19]

Google Street View is a virtual representation of the surroundings on Google Maps[20] that includes millions of panoramic images. Figure 3 illustrates Google Street View's look towards Tartu's Town Hall Square.

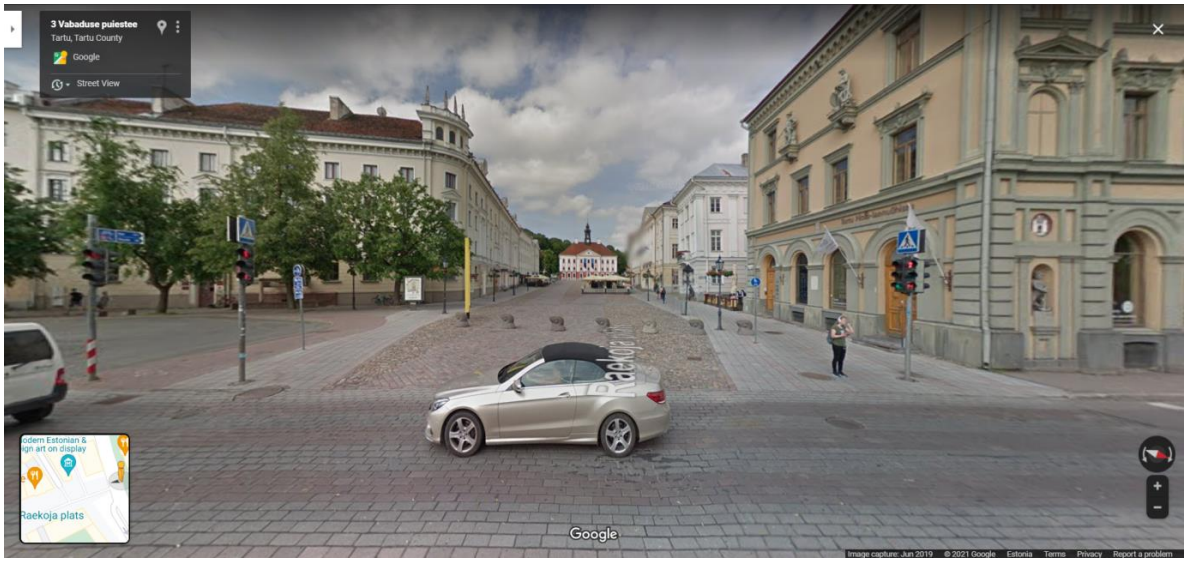

Figure 3: Example of Google Street View's look towards Tartu's Town Hall Square.

Information gathered with it was used to help complete various tasks in this project.

[18] https://geoportaal.maaamet.ee/eng/Maps-and-Data/Estonian-Topographic-Database/Download-Topographic-Data-p618.html
[19] https://www.google.com/streetview/
[20] https://www.google.com/maps

5. Methodology

This chapter describes the process of developing the digital twin of the Tartu city centre.

5.1 Creation of Structures in the City Centre

Structures were built in three different ways:

- Creating the structure through Blender software based on the point clouds.
- Creating the structure through RoadRunner software.
- Using objects in RoadRunner's asset library.

5.1.1 Structure Creation Process in Blender

The elevation data in LAZ format was downloaded from Geoportal. CloudCompare software was used to single out the specific structure, and the Mesh->Delaunay 2.5D feature used to convert the point cloud to mesh. The result was exported in Wavefront (.obj) format and imported into Blender. Figure 3 shows an imported mesh in the Blender environment after it was converted from the point cloud data using CloudCompare.

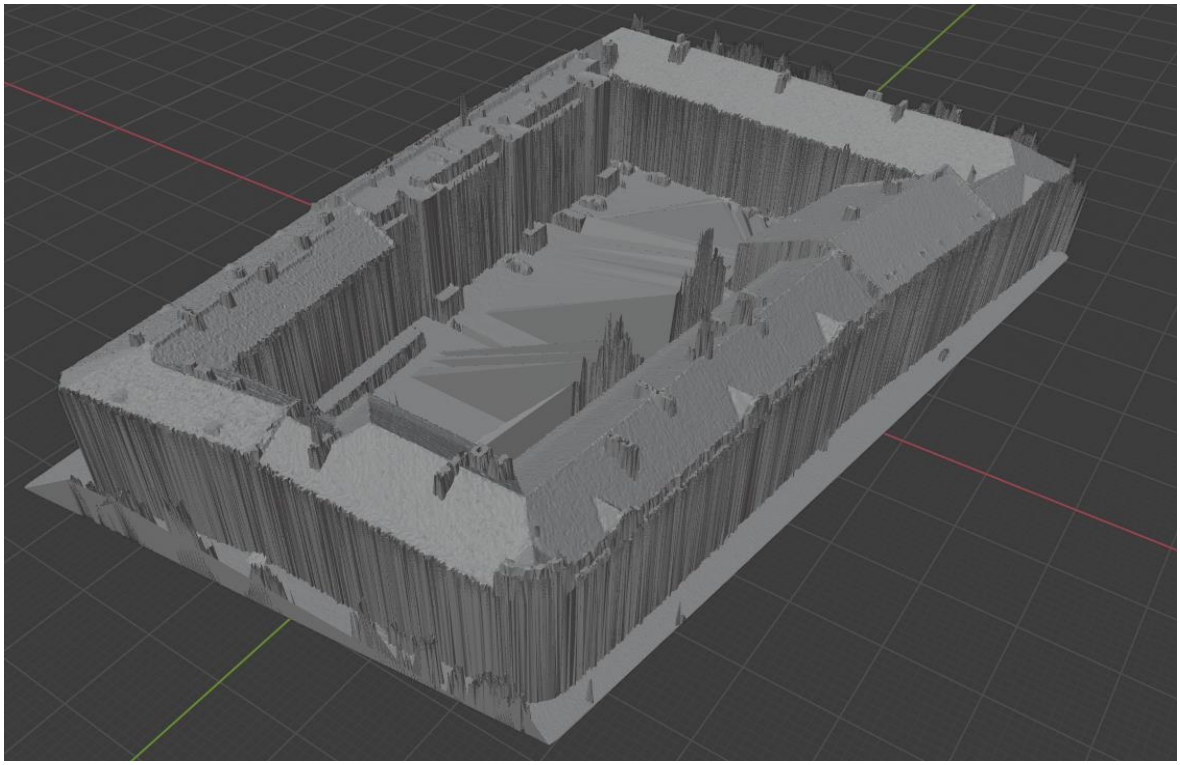

Figure 3: Example of imported mesh in Blender after the conversion from a point cloud in CloudCompare.

In the Blender environment, the new 3D structure was built on top of the existing mesh using it as a base measuring tool. The new object was textured by adding new materials in Blender or using the UV mapping [17] feature. In Figure 4, one can see a building that is constructed and textured using the UV mapping technique.

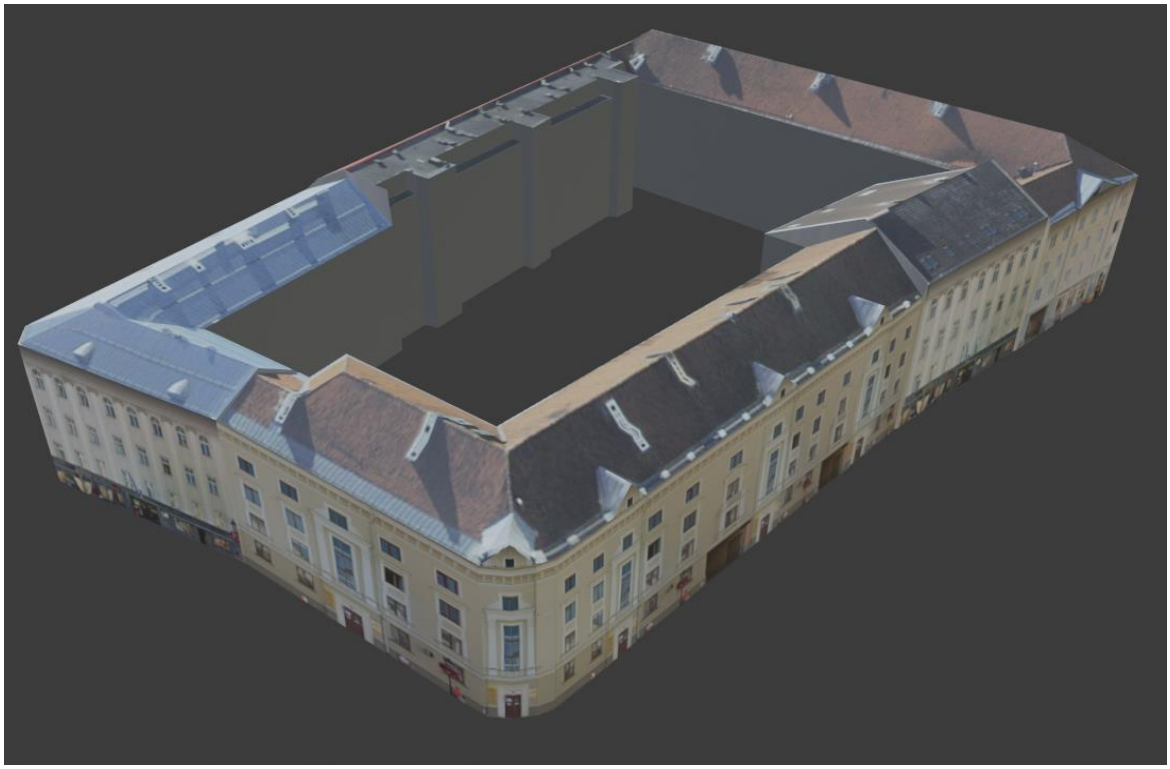

Figure 4: Example of a mesh created in Blender based on the point cloud base and textured using the UV mapping.

Textures for the UV mapping were taken from Google Street View.

Two additional models were used in the scope of the project created by other authors. The first one was the facade of the Tartu Kaubamaja and Kvartal[21] (two buildings in Tartu city centre) designed by Ali Abdollahi Dehaghani[22]. The second model was the Delta Centre (building in Tartu city centre), designed by Madis Vasser[23]. Both models were heavily modified and updated in Blender to be suitable for the project. Each author granted his permission to use their work in the context of the project.

This structure building process in Blender was time-consuming, especially when paying attention to detail. It was impossible to create every building using this method in the scope of this project. Due to that, an alternative option in RoadRunner was used where needed.

5.1.2 Structure Creation Process in RoadRunner

To create a structure in RoadRunner, one needs to select the base borders of the object (object borders are put in place based on the topographic data) at the ground level, then adjust the height of the surface inside the bounds (the height is set based on the point cloud data) and add the proper texture. In Figure 5, the mentioned technique is on display.

[21] https://sketchfab.com/3d-models/riia-st-tartu-estonia-4f6eafe0211c42f7afd5e4a24b956480
[22] ali.abdollahi.dehaghani@ut.ee
[23] Madis.Vasser@gmail.com

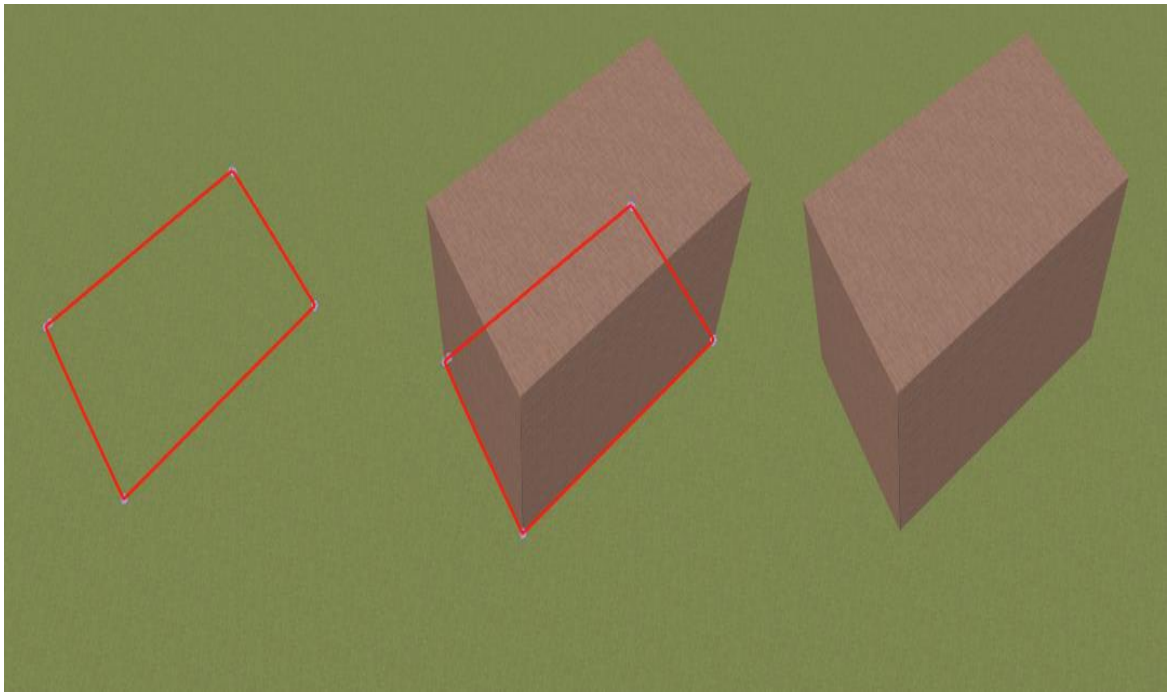

Figure 5: Example of structure creation in RoadRunner.

Structures created using the mentioned method lacked the details on the top-level (roof), but the ground floor's placement was accurate according to orthographic and topographic data.

For the creation of the road signs not present in the RoadRunner's asset library, a built-in feature called 2D editor was used. It took previously created vector graphic [18] designs and allowed to combine, recolour and scale them in the sign creation process. Vector graphic designs were created by using Inkscape software.

5.1.3 Using Objects in RoadRunner's Asset Library.

RoadRunner comes with a rich asset library, including traffic lights, road signs, guardrails etc. Assets were used where possible when the visual impression was close to resembling the real-life counterparts in Tartu.

5.2 Development of the Road Network

All the roads were created and positioned using the RoadRunner. Elevation, orthographic and topographic data was used in the process by importing orthophotos, topographic vector data (in the form of shapefiles from the Autonomous Driving Lab at the University of Tartu) and point clouds into the RoadRunner environment. It was the base for assessing road boundaries during the creation. Figure 6 shows how topographic vector data (blue lines) on top of the orthographic information allowed precise placement of the roads.

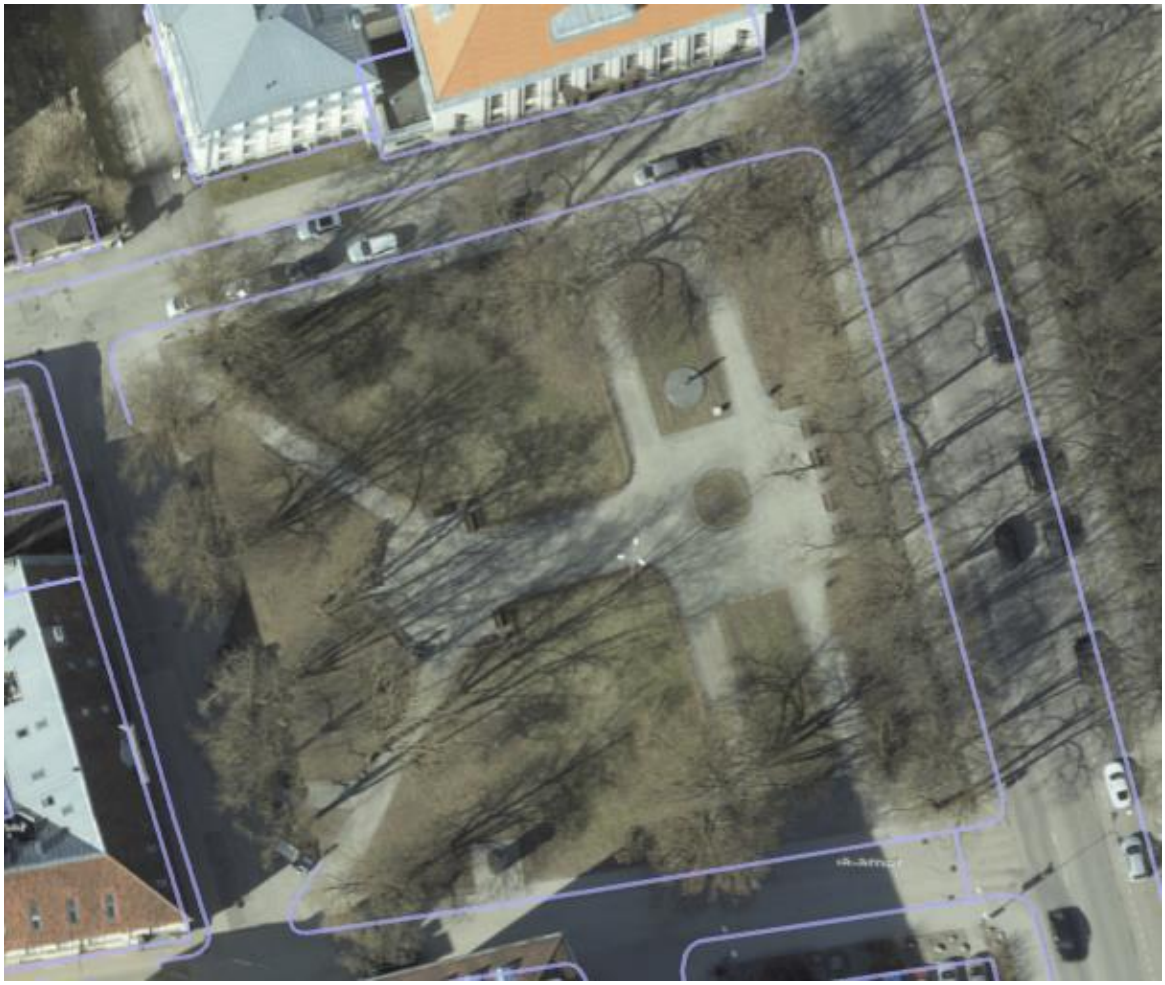

Figure 6: Example of vector data on top of the orthophoto in RoadRunner showing boundaries of the road.

Elevation of the road network was put in place by using the point cloud data.

Road surface marking was done using RoadRunner's built-in feature by applying materials from its asset library. The layout was designed based on the info from orthophotos, vector data and Google Street View to get a realistic presentation.

5.3 Positioning of the Structures

All the positioning was done in the RoadRunner environment. Objects that were

- created with Blender,
- taken from the RoadRunner's asset library,
- made with the RoadRunner,

were positioned using imported point cloud data and information gathered from Google Street View.

5.4 Assembling the Project in Unity

All the project data was exported from the RoadRunner environment in Filmbox format (.fbx) plus textures. All of it was then imported into the Unity environment, where some additional objects were added and settings configured to make sure the digital twin of the Tartu city centre is compatible with the SVL Simulator.

5.4.1 Adding Additional Objects and Environment Settings

In the Unity environment, all the street lights were added. The street light models were previously created in Blender and imported to Unity. After that, the light source was added along with the script that controlled its activation time.

Additionally, the following changes were made:

- All the road texture shaders [19] were changed to allow environmental effects to work on the road surfaces.
- Mesh colliders were added to all meshes that needed physics collisions.
- A semantic tag was added to each object in the scene (vegetation, road, building, obstacle).
- Reflections on textures were changed. Surfaces like asphalt were made matte while the glass was made reflective.

5.4.2 High Definition Map Annotation

The source of the high definition map [20] used in this project is a map created by Karl-Johan Pilve[24]. The original high definition map had more coverage than needed in the scope of this project. Additional parts were removed using Java OpenStreetMap Editor. In Figure 7, the mentioned map is displayed before and after the removal of the unneeded data.

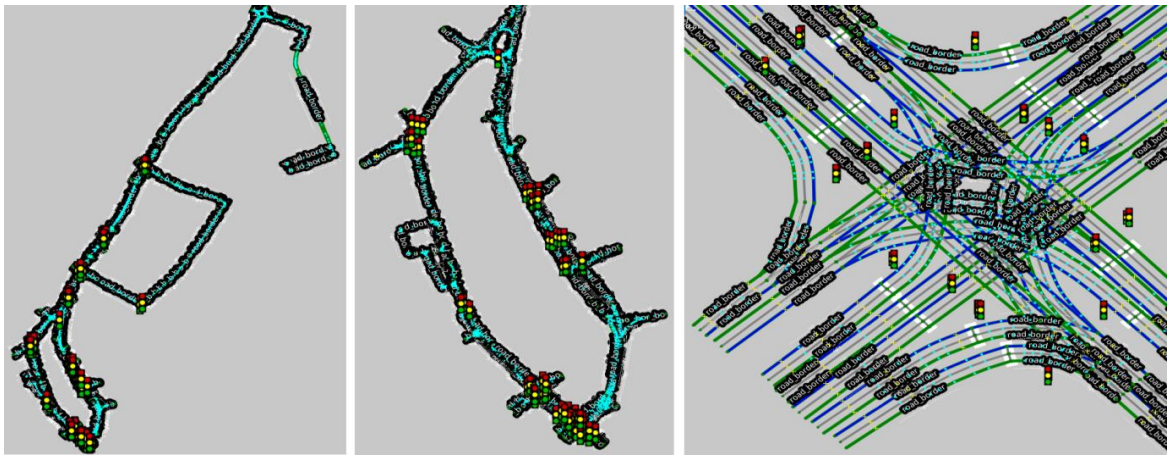

Figure 7: Example of Karl-Johan Pilve's high definition map in Java OpenStreetMap Editor. From the left: original high definition map in full size, part of the map used for the project, close up view of the map lanes.

The high definition map was imported into the Unity environment. Some additional traffic lights and stop lines were added. Traffic lights were moved into position according to the data gathered from Google Street View. All the junctions were updated to work with the SVL Simulator environment. In Figure 8, an imported high definition map is shown on top of the original mesh and a closeup view displaying the traffic lanes, stop lanes, traffic lights, and junction holders.

[24] karljohan30@gmail.com

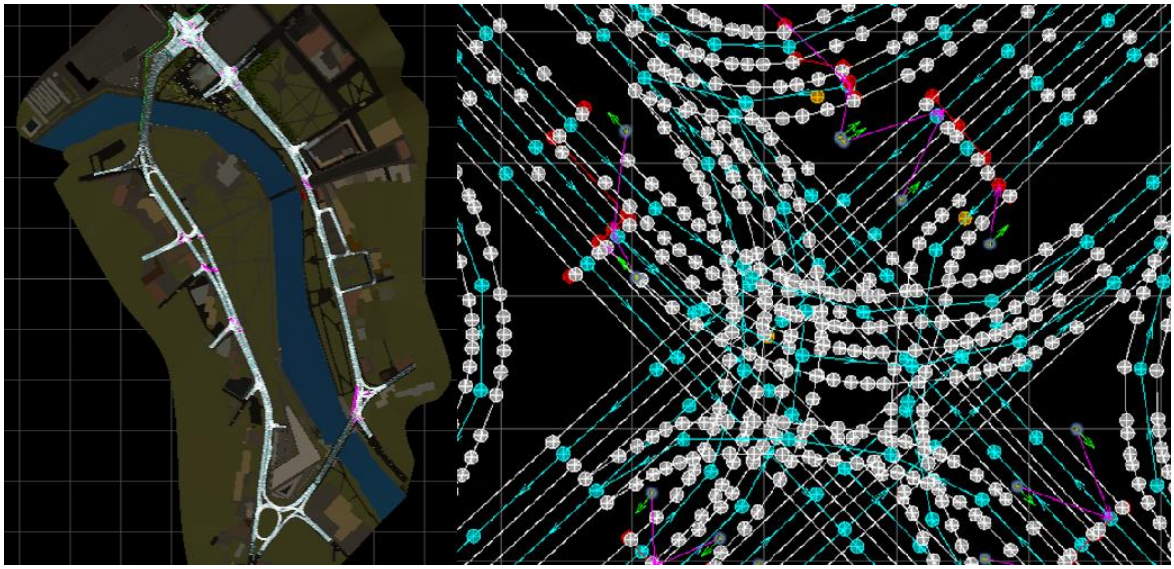

Figure 8: From the left: high definition map on top of the original mesh in Unity, close up view of the high definition map lanes in Unity.

In this project's scope, the high definition map is used by SVL Simulator to control the traffic. Traffic cars follow the traffic lanes assigned to them by high definition map, pause at the stop lanes when the traffic light related to it is red and yield to a different route if so instructed. As the SVL Simulator is still in development and not perfect, complicated junctions might give some trouble when it comes to traffic stopping with the red light. Instead, they only yield to the oncoming traffic and, when the road is clear, still make the turn with the red light. Further development on the simulator side will remove those issues in the future.

5.5 Setting the Digital Twin of Tartu City Centre Up for Testing in SVL Simulator

After using the Simulator->Build command in Unity to create the new map file, one needs to go to SVL Simulator's webpage and download the client. After setting it up and starting the client, access is granted to the web user interface. Through that, one can upload their map file and set up desired simulator options, as shown in Figure 9.

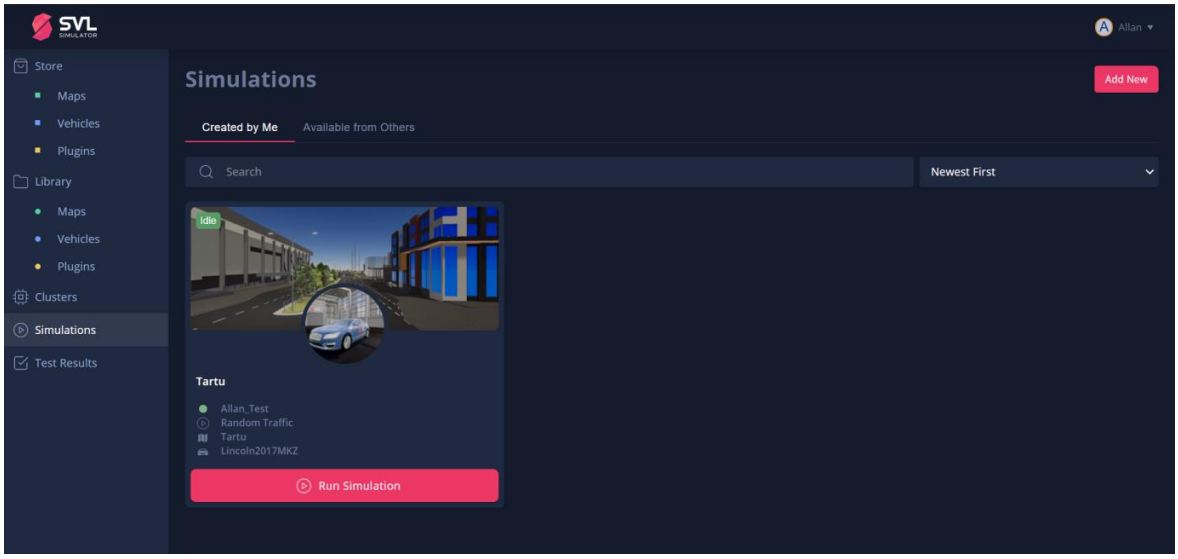

Figure 9: Example of SVL Simulator's web user interface.

Following those steps enables the testing process for the new map.

6. Results

This chapter describes how realistic and authentic the virtual world was, the results of using Apollo 5.0[25] autonomous driving technology to control the vehicle in the simulation and the estimated time spent on each task.

6.1 Virtual World Characteristics

The area covered by the digital twin of the Tartu City centre was around 640,000 square meters. Figure 10 shows the area on the actual map and its counterpart in the RoadRunner's environment.

Figure 10: Area covered by digital twin on the actual map and in RoadRunner.

The whole digitised road network ran 4500 meters. The time of day cycle worked perfectly. When it got dark, the street lights turned on and illuminated the road section of the map. Figure 11 shows the Kaubamaja junction during the day, in the evening and during the night. Weather effects were on point as well. It was possible to adjust the thickness of the clouds and fog. Set the amount of rainfall and wetness of the road surface. Adjusting the wetness level forced the actual road textures to change, allowing little ponds to form. An example is shown in Figure 12, where the Kaubamaja junction is on display during heavy rain.

[25] https://github.com/ApolloAuto/apollo

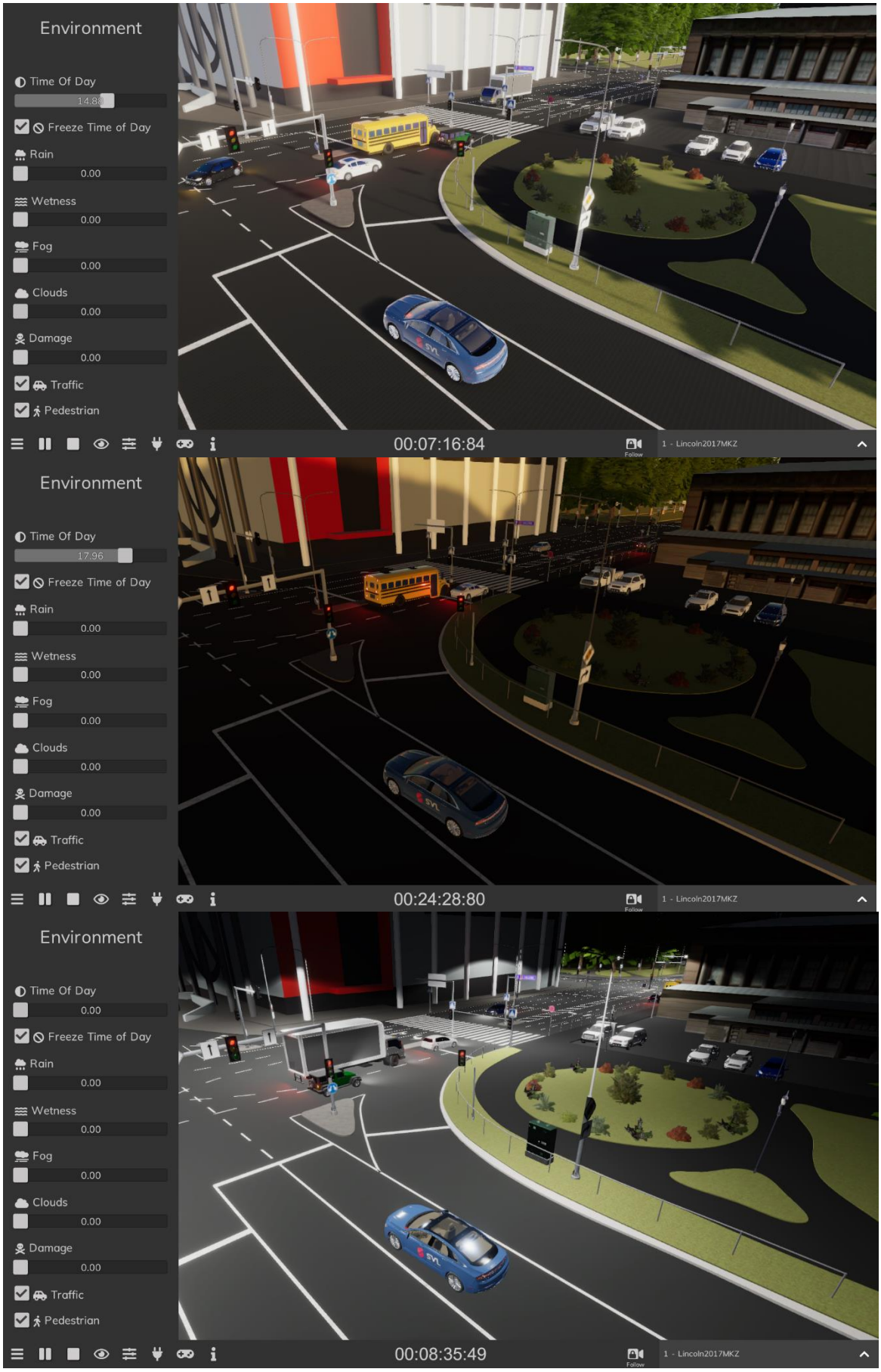

Figure 11: From the top: Kaubamaja junction during the day, in the evening and during the night.

Figure 12: Kaubamaja junction during the heavy rainfall.

Traffic on the road followed the driving lanes correctly. Traffic lights were working as was intended. It was impossible to set them up exactly as the real-life counterparts without rewriting the traffic light source code of the simulator. On some complex junctions like Kaubamaja, cars that were waiting to make a turn ignored the red light. As the high definition map was set up correctly, SVL Simulator's side must have caused the error.

6.2 Results from Using Apollo 5.0 Autonomous Driving Technology

Testing with Apollo 5.0 was done by Navid Bamdad Roshan[26] from the Autonomous Driving Lab at the University of Tartu. The main issue was the fact that some traffic lights in the simulator did not publish their status to Apollo when the car approached the junction. An example of it is shown in Figure 13.

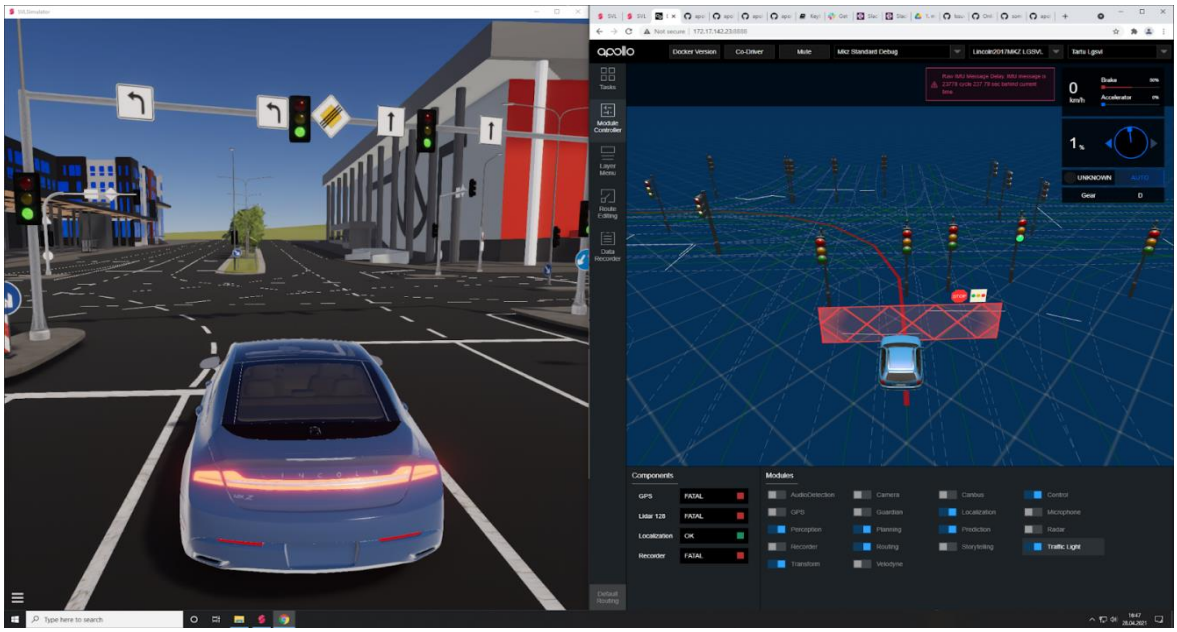

Figure 13: Apollo's interface on the right only picks up the status of one traffic light.

[26] navid.bamdad.roshan@ut.ee

It was suggested that it must be a high definition map problem. Further research showed that additional components were needed to be added at every junction on the high definition map when it came to an intersection and Apollo. The map was updated, but further testing was still pending.

It was also mentioned that pedestrian crossings and some traffic signs were missing from the high definition map even though they did show up in the virtual world. Figure 14 shows an example of it.

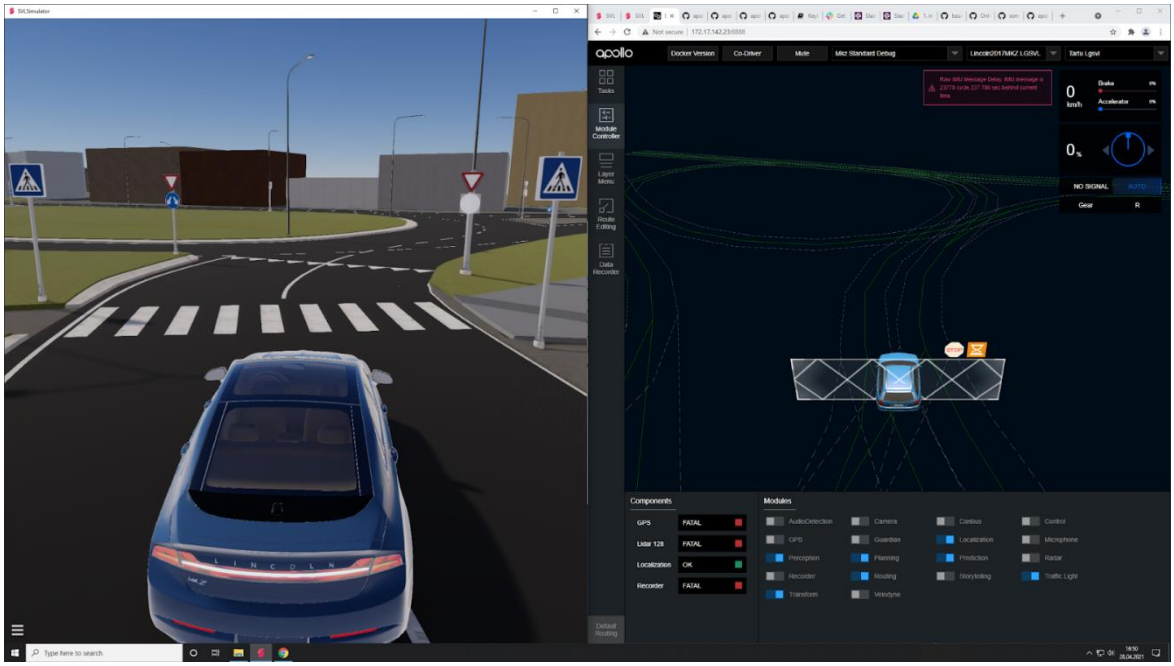

Figure 14: Apollo's interface on the right does not pick up traffic signs nor pedestrian crossroads.

As the original high definition map used for this simulation did not include any, they were not in the project's scope.

Trying to reroute and swap to a different lane made Apollo suggest travelling the whole lap and changing at the intersection. That meant that the adjacent lanes were not connected in the high definition map, thus not a fault of a simulation. Figure 15 illustrates the situation.

Being in a situation where cars just out of nowhere jump on the road was said to be problematic and potentially cause a crash (Figure 16 portrays this happening). Some traffic spawn points were moved further back away from the main road.

Finally, the GPS coordinates were a few hundred meters off compared to real-life data. It was adjusted and now positions match.

Outside of that, the vehicle followed the road correctly and completed the turns where traffic light status was published with no difficulty.

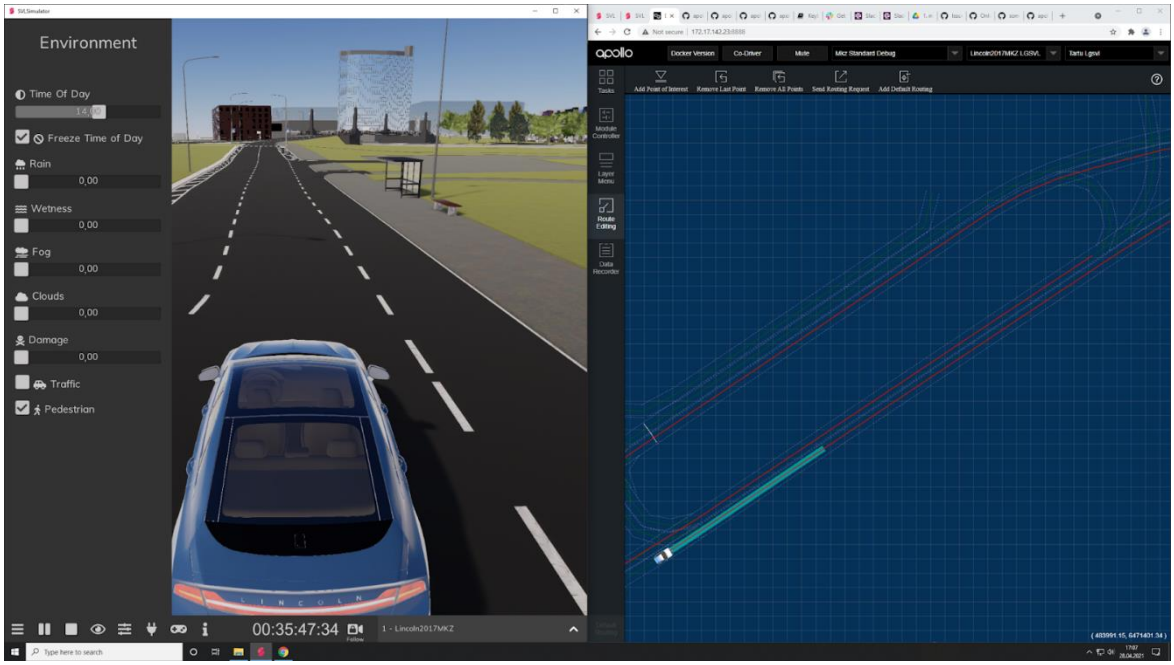

Figure 15: Apollo's interface on the right showing the route when one tries to change to the left lane.

Figure 16: Cars jumping on to the road out of nowhere.

There are also two videos[27] showing the test process with Apollo.

6.3 Estimated Time Spent on Each Task

It would be really challenging to single out the timeframe for each task individually. Table 1 gives a general idea of how many objects were created and positioned into the digital world during that timeframe. In general, a lot of time was spent on learning how to use the software needed before the real work could begin. After a month or two, as the skillset had improved, understanding started to form concerning how badly some of the tasks were

[27] https://drive.google.com/drive/folders/1cKVLpLy_yUKf0B7QpeP6d2ycxytkDrY3?usp=sharing

completed in the past. That led to redoing a lot of work. This cycle happened a couple of times during the project's lifecycle and did have a setback effect time-wise.

Tabel 1. Amount of structures created within the virtual world

Structure	Amount
Road network	4528 meters
Traffic lights	68
Traffic signs	234
Pedestrian crossings	27
Street lights	158
Structures created with Blender	11
Structures created with RoadRunner	85
Cars parking on the street	374
Trees and bushes	1239

Another major issue was the fact that the software always did not complete the task it was assigned to in a way it should have, even though the manual was followed line by line. That led to improvising through finding a different path to the same goal. It was always a very time consuming and frustrating road to take but essential nevertheless.

Overall, the time investment was definitely higher than expected to complete the task, but there were no regrets as the process was gratifying.

7. Conclusion and Future Work

The main goal of the thesis was achieved by designing a digital twin of a Tartu city centre that allowed virtual testing of autonomous vehicles at that location. The resemblance to the real-life counterpart was on point, given the timeframe allocated to the project. Of course, with additional time investment, the results could have been improved. A big part of the project was on the art side, and due to that, there was no actual finish line. There is always something one can constantly improve upon, even though all the required functionality has been achieved.

Some additional sub-goals were not hit one-hundred per cent. Traffic cars had trouble with traffic lights on some complex junctions. Attempt to design those hotspots by following the manual line by line did not deliver the promised effects. The result was total chaos. Improvisation was needed, and a lot of time was spent trying to find an adequate fix to get it to work as it presented. Hopefully, further updates to the simulation software will alleviate the problem. As the game world was quite detailed in some places and graphics are not optimised as of yet, it needed up-to-date computer hardware to run smoothly. Finally, the pedestrians were missing from the streets. Again following the manual gave zero results in making them appear. Brainstorming and rule-bending were required to get something going, but still, the results were not good enough to be included with the release.

Work does not end here as it is in the interests of the Autonomous Driving Lab at the University of Tartu to include digital twin as an addition to a different project. Due to that, some improvements are required. All the traffic light problems have to be solved. It might require editing the runtime functionality of the simulator through Python's application programming interface. A lot of modelling has to be done in Blender to replace the currently low detail objects on the map. Textures are to be updated to give a more realistic look to buildings. Foot-travellers have to be added to the streets to simulate the activity on pedestrian crossings. High definition map updates might be needed to ensure no complications ensue when it comes to Apollo and Autoware software compatibility. Finally, further optimisation needs to occur to improve the application's framerate by adding levels of detail (LOD)[28] to different game objects in the world and taking advantage of the oculus culling[29] technique.

Additional knowledge was acquired by the author when it came to geography in the forms of topography, orthography and elevation data (point clouds). The ability to compose art in vector graphics form was learned. Expertise was gained in using RoadRunner software to design road structures. The chance of getting to know what high definition maps are and how they are used in the simulation environment allowed me to peek into the world of autonomous vehicles. The ability to use SVL Simulator introduced me to how potential testing and training might look in said environment. Finally, substantial improvements were made in understanding how to work with Unity and Blender to design a digital world.

[28] https://www.techopedia.com/definition/11791/level-of-detail-lod
[29] https://docs.unity3d.com/Manual/OcclusionCulling.html

8. References

[1] Rastogi V. Virtual Reality Based Simulation Testbed for Evaluation of Autonomous Vehicle Behavior Algorithms, *Clemson University*, 2017 https://core.ac.uk/download/pdf/268664049.pdf (28.04.2021)

[2] Hunsley J. Why simulation is the cornerstone of autonomous vehicle development, *Automotive World*, 2020. https://www.automotiveworld.com/articles/why-simulation-is-the-cornerstone-of-autonomous-vehicle-development/ (28.04.2021)

[3] FEA Information Engineering Solutions, 2020. https://www.dynalook.com/fea-newsletters/fea-newsletter/06_feaies_june_2020.pdf (21.04.2021)

[4] Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey, *National Highway Traffic Safety Administration*, 2015 https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115 (07.05.2021)

[5 Automated Vehicles for Safety, *National Highway Traffic Safety Administration* https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety (05.05.2021)

[6] Walker A. Are self-driving cars safe for our cities? *CURBED*, 2020 https://archive.curbed.com/2016/9/21/12991696/driverless-cars-safety-pros-cons (05.05.2021)

[7] Scanlon J. Waymo Simulated Driving Behavior in Reconstructed Fatal Crashes within an Autonomous Vehicle Operating Domain, *Waymo, LLC* https://storage.googleapis.com/waymo-uploads/files/documents/Waymo-Simulated-Driving-Behavior-in-Reconstructed-Collisions.pdf (07.05.2021)

[8] Badger E. Elon Musk says we'll outlaw human drivers in a world of driverless cars. Really? *Washington Post*, 2015 https://www.washingtonpost.com/news/wonk/wp/2015/03/18/should-we-outlaw-human-drivers-in-a-world-of-driverless-cars/ (05.05.2021)

[9] Castignani L. Achieving Autonomous Driving with Simulation & Testing, *MSC Software* https://www.mscsoftware.com/sites/default/files/Achieving-Autonomous-Driving-with-Simulation-and-Testing.pdf (05.05.2021)

[10] Castignani L. Simulation is key to real-world autonomous driving, *The Engineer*, 2020 https://www.theengineer.co.uk/autonomous-vehicles-simulation-real-word-road/ (05.05.2021)

[11] Introduction, *Blender Manual* https://docs.blender.org/manual/en/latest/getting_started/about/introduction.html (21.04.2021)

[12] RoadRunner, *Mathworks* https://uk.mathworks.com/help/roadrunner/index.html (21.04.2021)

[13] Sinicki A. What is Unity? Everything you need to know, *Android Authority*, 2021. https://www.androidauthority.com/what-is-unity-1131558/ (21.04.2021)

[14] Introduction, *SVL Simulator* https://www.svlsimulator.com/docs/getting-started/introduction/ (21.04.2021)

[15] Elevation Data, *Geoportal* https://geoportaal.maaamet.ee/eng/Spatial-Data/Elevation-data-p308.html (21.04.2021)

[16] Orthophotos, *Geoportal* https://geoportaal.maaamet.ee/eng/Spatial-Data/Orthophotos-p309.html (21.04.2021)

[17] Denham T. What is UV Mapping & Unwrapping? *Concept Art Empire* https://conceptartempire.com/uv-mapping-unwrapping/ (21.04.2021)

[18] What Are Vector Graphics? *Vectr* https://conceptartempire.com/uv-mapping-unwrapping/ (23.04.2021)

[19] Shehata O. A Beginner's Guide to Coding Graphics, *ENVATO TUTS+*, 2015. https://gamedevelopment.tutsplus.com/tutorials/a-beginners-guide-to-coding-graphics-shaders--cms-23313 (24.04.2021)

[20] Vardhan H. HD Maps: New age maps powering autonomous vehicles, *Geospatial World*, 2017. https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/ (24.04.2021)

Appendix

I. Additional Files

Recommended system requirements according to SVL Simulator's official webpage to run the SVL Simulator are:

- at least 4 GHz Quad core CPU,
- NVIDIA GTX 1080 (8GB memory) or higher,
- Windows 10 (64-bit), Ubuntu 18.04 (64-bit), or Ubuntu 20.04 (64-bit).

It is possible to run the application with lower-specification hardware, but there may be issues with the performance.

As the project is still a work in progress, the digital twin of Tartu City centre is not yet available to the public through the SVL web interface. To gain access to the track and instructions on running it, contact the author of the thesis, A. Mitt (allan.mitt@gmail.com).

II. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Allan Mitt**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Simulation of Tartu City Centre for Testing Autonomous Vehicles,
supervised by Tambet Matiisen.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available tothe public via the web environment of the University of Tartu, including via the DSpacedigital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work andcommunicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Allan Mitt

07.05.2021