ECE 573 Project Verification

Rory Scobie, Sahachel Flores, Michael Treiber

April 2021

1 Implemented verifiable requirements

1.1 B Requirements

- Requirement: The user shall have control over the following variables: a user-defined starting lat/lon coordinate or preset location from a list, number of chunks to load around car, car movement speed. This is dependent on the creation of a working GUI. The user-defined starting location shall correspond with the gazebo world origin at the beginning of program execution, where the vehicle will begin from.
- Test: Run tests on the created GUI to make sure that it is giving the user correct options, by seeing the data being received from the GUI when the user enters a location/speed or presses a button. Visually using Gazebo to see what happens is also a good test, because it will show the location in the world and the vehicle with its speed put into account. A generated world file and working URDF of a vehicle in the world must be made first, before proceeding with the UI. A semi-automatic test is provided to verify that the coordinates are published on the /chunk_coordinate topic after a user inputs a certain coordinate.
- Requirement: The program shall start the simulation from a reference coordinate (input by user). This is dependent on the URDF file being the center of the generated world file, set to those starting lat/lon coordinates
- Test: After the GUI node launches on ROS, a test node will subscribes to the chunck_coordinate topic. The goal of this node is to verify/test that the GUI node publishes the latitude and longitude values once the user clicks the start button. If the published array is empty, an exception will be raised to inform the user
- Requirement: The program shall generate the environment with a minimum real-world dimension of 1000x1000m for every OSM chunk. This is dependent on the creation of an SDF file using a .obj file generated from OSM meshpoint data.

- Test: Launch an OSM chunk in Gazebo and go to the World tab on the left-hand side. Open the link section in this tab, and then open the second visual section. Under that open the geometry section and then open the scale section. If X and Y both show 1000, then this requirement is verified, since Gazebo uses meters as the unit of measurement. Z also should show 1000 for 3D sizing. An automatic test is also provided to measure the generated model's size before use in gazebo, but this doesn't prove proper scaling in gazebo.

2 Unimplemented/Work in Progress but verifiable requirements

2.1 B Requirements

- Requirement: The simulation shall be generated on Gazebo.
- Test: Once the files all are created and everything looks correct, Gazebo will run the simulation to show that it works correctly. If Gazebo has problems generating any files or models are missing in Gazebo, then this test has failed. All other B requirements and tests should be done before proceeding with this test.
- Requirement: The program shall fulfill all other requirements in city block, neighborhood, and freeway environments.
- Test: Separate world files can be made in order to test the vehicle in different areas. These world files can be smaller in order to run tests in them. If the vehicle returns correct data, the GUI returns correct inputs from the user and everything looks correct in the Gazebo simulations, then this test is successful. World files with models, including the URDF file for the vehicle and the GUI must be made in order to test this case.
- Requirement: The program shall generate a world around the reference point such that the vehicle's sensors won't detect the unloaded world.
- Test: Generate a world file and see if the vehicle is the center/reference point of this world. Also verify that the sensors are not protruding outside the generated world, by giving the sensors a designated radius. If the car is centered and the sensors are not collecting empty chunk data, then this test is a success. A world file, with a model of the URDF vehicle file must be made, and a node to collect the sensor data of the world must be made before this test can be done.
- Requirement: During the simulation the vehicle shall stay on the roads limits.

- Test: Simulate the vehicle in Gazebo with the road models included. If the car steers into the road edge and does not go past it, then this test is successful. Limits with the vehicle URDF file and the generated world file must be taken into account, before proceeding with this test.
- Requirement: The speed of the vehicle shall be constant.
- Test: Once the world and vehicle are generated, and the constant input velocity in the GUI is set, view the topic that the vehicle velocity is subscribed to and verify that this velocity is relatively constant. If it is, then this test is successful. Nodes must be created, with the world file, and URDF file, and a node must be subscribed to the URDF file before running this test.

2.2 A Requirements

- Requirement: The car shall navigate between two GPS coordinates, following the roads where possible.
- Test: If the user inputs another set of GPS coordinates and the vehicle moves along the roads to those coordinates, and does not get stuck on any of roads to that location, then this test was successful. An updated GUI and URDF file that will move the vehicle accurately along roads to a second set of GPS coordinates must be created in order to verify this test.
- Requirement: Dynamic speed will change based on the environment.
- Test: Run the vehicle URDF file in the simulation and if the velocity slows down when the car is going to collide with anything, or if the vehicle speeds up when there's nothing in the way, then this test was successful. The URDF file must be updated in order to change velocity based on the collision settings with other objects and update the nodes receiving the URDF data to reflect this as well, before running this test.
- Requirement: The user shall control the car's movement with a keyboard.
- Test: Keybinding different keys to the nodes that control the movement of the vehicle can be used to move the car in the simulation. If all these keys are successful in moving the car around, then this test is successful. An updated URDF and updated vehicle control nodes that take these key bind events into account must be made before proceeding with this test.
- Requirement: The program will Generate obstacles along/on the road, such as people, cars, streetlights, etc.
- Test: The completed world file can be added onto with models that can move or interact with the environment, and if these models are included in the world when it is generated, then this test was successful. The completed world must be made before proceeding with this test.

- Requirement: The program shall generate a scenic environment. (water, trees, building aesthetics, cacti, etc.)
- Test: The completed world file can be added onto with more ascetic models, and if these ascetics are included in the world when it is generated, then this test was successful. The completed world must be made before proceeding with this test.
- Requirement: The generated world shall have common traffic control such as stop lights, crosswalks, and stop signs.
- Test: This test will be proven correct if the updated URDF vehicle file takes into account common traffic control with relative precision, when these models are put into the world. A completed world file that includes these new models and an updated URDF file that takes these models into account with updated nodes that correctly perceive the URDF data must be made before proceeding with this test.

3 unverifiable (non-functional) requirements

3.1 B Requirements

- Requirement: The program shall base its environments on real-world data.
- Test: Use OpenStreetMap to generate meshes of a given area and if the meshes do not come through the transition to an SDF file, then the process was not successful. A conversion from the OSM data to an SDF file must be made, before testing this case. A world file containing this SDF data must be made as well.
- Requirement: The generated world shall include world-accurate terrain heights.
- Test: Using Gazebo and OSM, one can take the OSM data and compare it to the visuals in Gazebo, using the Gazebo height map. OSM data must be generated in Gazebo, before proceeding with this test.
- Requirement: The generated world shall include roads with world-accurate positions, connections, and lengths.
- Test: Using OSM data and Gazebo, road SDF files can be generated and compared to real-life locations. If these real-life locations compare well with the generated Gazebo models, then this test is successful. OSM data needs to be converted to SDF files that Gazebo can use to generate the models before this test can be done.
- Requirement: The generated world shall include buildings with world-accurate footprints and heights.

- Test: The generated Gazebo simulation will be compared with real-life locations and if they compare well, then this test was a success. OSM data must be transitioned into SDF models for the buildings using different algorithms in order to test this case.

3.2 A Requirements

- Requirement: The quality of generating the world around the car will improve.
- Test: Update the world file with more improved models (i.e., if the model isn't a .dae file, try to make it a .dae file), and compare visuals of the world models before and after they change file types (if possible). If the new world generated has better quality of models than the original world, then this process succeeded. The complete, simulated world must be made before proceeding with this test.