CHAPTER V. GENERAL CONFIGURATION PRACTICES AND PROCEDURES

V.A. Common practices

In addition to the tutorials and instructions available from online documentation ([38]), the author developed practices which proved to be useful while attempting to determine the causes of various errors encountered while configuring an arbitrary simulation:

V.A.1. Use valid "model_name::interface_name" addressing

For Player to communicate successfully with Gazebo, Player must know what interfaces Gazebo is providing. Determining the valid address for interfaces was more difficult than anticipated. However, valid values for "model_name::interface_name" addressing in Player configuration files may be determined by reviewing the available interfaces in directory "`/tmp/gazebo...`" corresponding to the user's running instance of Gazebo, which are defined by the world file. For example, loading the model used to evaluate rollover of a representative challenge vehicle entering 2004 GCE course segment 2570-2571-2572 (see Appendix F) creates a "`position.cv_model::position_iface_0`" interface in directory "`/tmp/gazebo...`". The corresponding Player configuration file "`gz_id`" for this interface is therefore "`cv_model::position_iface_0`", and this "`gz_id`" must be defined by the Player configuration file before launching Player for Player to communicate successfully with Gazebo.

V.A.2. Increase the controller update rate to increase the quality of logged data

A comment in file `playerv.c` (the `playerv` utility) states: "20 Hz update rate

is good for user interaction". However, when the author began to log output generated by the improved steering controller ("improved controller") updates were being generated at 10 Hz.

File `Controller.cc` attempts to set parameter `updateRate` when a controller is loaded. Parameter `updateRate` is not in use by any packaged controllers, models, or world files. As a result, the author was unaware parameter `updateRate` could be declared until he reviewed the Gazebo codebase to identify parameters which could be declared. See paragraph V.A.3. below.

An `<updateRate>` declaration was included in the `<controller>` declaration to increase the update rate of the improved steering controller and the quality of logged data.

An update rate of 50 Hz gave good results for logging data, with few missed intervals, and no observable impact on the ratio of simulation time to real time. Increasing the update rate beyond 50 Hz did not result in a significant increase in the quality of logged data, and resulted in a greater number of missed intervals. Decreasing the update rate to 10 Hz resulted in instability when the model was traveling at high speed.

V.A.3. Review the Gazebo codebase to identify parameters which may be declared

The author was unaware parameter `updateRate` could be declared because it was not in use by any packaged controllers, models, or world files. Review of file `Controller.cc` identified two other parameters which may be declared: `name` and

`alwaysOn`. Parameter `alwaysOn` was not in use by any packaged controllers, models, or world files. Parameter `name` was in common use.

The author reviewed the Gazebo codebase for occurrences of "`new ParamT`" in files to identify parameters which may be declared. Some files, for example, the packaged steering controller, use private class member variables in lieu of parameters.

V.B. Common procedures

The author developed common procedures, some of which were based on tutorials and instructions available from online documentation:

V.B.1. Use of the `ffmpeg` utility to create movies from captured images

The "Save Frames" command in Gazebo was used to capture images during simulation, then movies were created from the captured images using the following commands:

```
ffmpeg -f image2 -i UserCamera_0-%04d.jpg [destination]
ffmpeg -i UserCamera_0%04d.jpg [destination]
```

However, captured images were skewed to the right. See Figure 2. Although it was possible to create movies from the captured images, the resulting movies were also skewed to the right, making it difficult to effectively visualize the simulation. After several attempts, the author abandoned the use of the `ffmpeg` utility to create movies from captured images.

KSnapshot, a screenshot utility packaged with the K Desktop Environment (KDE), was used to capture images during simulation, and these images are the images included herein.

V.B.2. Patch generation

The following command was used to prepare patches submitted as a result of this research:

```
diff -rup /path/to/unmodified/source /path/to/modified/source
```

V.C. Model creation

Models of a representative challenge vehicle and obstacles DARPA identified as typical of obstacles challenge vehicles would encounter during the 2004 GCE were developed during this research.

V.C.1. Representative challenge vehicle

To maximize the re-usability of the model, the author selected a representative challenge vehicle which was:

- Successful.

  Teams 2005-06, 2005-13, 2005-14, and 2005-16 successfully completed the 2005 GCE.

- A commercially-available SUV.

  A commercially-available SUV was the most common platform selected by teams which participated in the 2004 or 2005 GCE. Commercially-available SUVs were in use by Teams 2005-06, 2005-14, and 2005-16.

- Described in sufficient technical detail in published records to model in simulation.

  Neither challenge vehicle SSF nor height of vehicle CG were reported by published records for Team 2005-14 or 2005-16 challenge vehicles.

As a result, the Team 2005-06 challenge vehicle was selected as representative. A model was created using five bodies (“chassis_body”, “left_front_wheel”, “right_front_wheel”, “left_rear_wheel”, and “right_rear_wheel”) and associated geoms with the physical dimensions and other characteristics of the representative challenge vehicle. The representative challenge vehicle model is described in detail in Appendix F.

Because the selected simulation targets included an evaluation of the rollover condition, realistic physical dimensions and other characteristics of the model were selected to ensure the track width, height of vehicle CG, and curb weight in simulation were identical to those of the representative challenge vehicle.

By default, Gazebo places the CG of a body at its center. The author did not alter the default behavior. Realistically, the rollover condition is dependent on the location of vehicle CG, which may not be at the geometric center of the model's “chassis_body”. However, the author considers it likely the representative challenge vehicle CG was very close to the left-right centerline of the vehicle, although he acknowledges it may have been forward of the front-back centerline of the vehicle due to the weight of the engine.

The distance of the CG from the front-back centerline of the vehicle may affect vehicle dynamics, including rollover, but the effect will be much less than that of the distance of the CG from the left-right centerline due to the difference between the wheelbase and track width dimensions. For the representative challenge vehicle, the distance between front and rear axles (wheelbase) was 1.7 times the track width. The author is confident the contribution to vehicle dynamics, including rollover, of the distance between the CG and left-right centerline of the representative challenge vehicle

is greater than the contribution of the distance between the CG and front-back centerline of the vehicle, and considers the model to be accurate enough to evaluate the selected simulation targets.

A mesh was created to provide the model with a visual similarity to the representative challenge vehicle. See Figures 3 and 4 for a visual comparison of the representative challenge vehicle to the model. Packaged meshes were used for the wheels of the model.

V.C.2. Representative obstacles

DARPA published a description of obstacles teams participating in the 2004 GCE could expect to encounter during the 2004 QID and which were representative of obstacles teams could expect to encounter during the 2004 GCE: “Dirt Hills”, “Tower Obstacle”, “Car Obstacle”, “Steep Hill”, “Sand Trap”, “Ditch”, “Cattle Guard”, “Overpass”, “Boulders”, “Moving Car Obstacle”, and “Washboard” ([10]).

To effectively evaluate the simulation targets, two obstacles were selected as representative: “Tower Obstacle” and “Car Obstacle”. The obstacles were modeled using the Player Project Model Creation Tutorial ([40]). The “Car Obstacle” model was based on the dimensions and weight of a 2009 Honda Accord. Meshes were created to provide the models with a visual similarity to the representative obstacles. Unlike the representative challenge vehicle, the representative obstacles were modeled using a “trimesh” geom primitive[11] to provide the most accurate interaction with sensors possible[12].

See Figures 5, 6, 7, and 8 for a visual comparison of the representative obstacles

to the models created by the author.

V.D. Mesh creation

The representative challenge vehicle, representative obstacles, and guides used to visually evaluate the interaction of the representative challenge vehicle model with the environment required the creation of meshes having arbitrary shapes. As a result, the author created several custom meshes during this research. The author found the Player Project Mesh Creation Tutorial ([41]) to be a useful starting point when creating meshes, but it would have required the author to learn to use Blender or another 3D rendering application with which the author had no familiarity. However, the author determined Blender ([42]) could be used as an intermediate application.

The author installed `blender-2.49a-4.5` using YaST. Packages `openal-soft 1.9.616-1.1.1` and `libopenal1-soft 1.9.616-1.1.1` were installed by YaST to resolve dependencies. The author then installed the Blender Exporter ([43]).

The author created models using TurboCAD Mac Deluxe, an application with which the author had some familiarity, exported them, and imported them into Blender. To determine which file format provided the best compatibility, the author attempted to import several different file formats exported from TurboCAD Mac Deluxe (DXF, DWG, AI, RAW, WRL, and STL) into Blender, with varying results:

- Several files caused Python script errors.
- Attempting to import a DWG file caused a “DWG-Importer cant find external DWG-converter (DConvertCon.exe) in Blender script directory” error

("DConvertCon.exe" is a Windows executable).

- WRL files were imported "one-sided".
- Attempting to import AI files resulted in a "Not a valid file or an empty file" error.

The Autocad file format (DXF) had the best compatibility. As a result, all models created by the author were created using TurboCAD Mac Deluxe, exported as Autocad Revision 12 (R12) files, and imported into Blender using the DXF importer.

Because Blender was installed on a desktop computer running openSUSE 11.2, the author had to specify "Unix (CR)" line-end characters when exporting models from TurboCAD Mac Deluxe. All models created using TurboCAD Mac Deluxe were created using metric units.

The author experienced unexpected behavior when importing DXF files into Blender because the "origin point" in Blender does not necessarily correspond to the origin in TurboCAD Mac Deluxe. The Player Project Mesh Creation Tutorial states: "Move the mesh to the origin." Although this is straightforward, importing an Autocad file into Blender causes the origin point to shift, even though the apparent origin of the model does not appear to have changed from the intersection of the x-, y-, and z-axes. The author used Blender's "Center" or "Center Cursor" functions to align the origin point with the apparent origin of the model, resolving the problem.

The models were then exported as meshes using the OGRE Mesh Exporter. When exporting the meshes using the OGRE Mesh Exporter, the author disabled options

“Export Materials”, “Fix Up Axis to Y”, or “Require Materials”, enabled option “OgreXMLConverter”, and clicked “Export”.

The resulting OGRE mesh files were then copied to the `/gazebo/Media/models` directory or `models` subdirectory of one of three test directories for use.