Vehicle Modeling with Chronos

A. Giavaras

Contents

1 Vehicle Modeling with Chronos

In this section we will develop and simulate vehicle model using the open source physics engine chronos

1.1 The Chrono::Vehicle library

The Chrono::Vehicle is a C++ middleware library for the modeling, simulation, and visualization of wheeled and tracked ground vehicles. It consists of two core modules:

- The ChronoEngine_vehicle
  - Defines the system and subsystem base classes
  - Provides concrete, derived classes for instantiating templates from JSON specification files
  - Provides miscellaneous utility classes and free functions for file I/O, Irrlicht vehicle visualization, steering and speed controllers, vehicle and subsystem test rigs, etc.
- The ChronoModels_vehicle
  - Provides concrete classes for instantiating templates to model specific vehicle models

The following dependencies should be satisfied in order to use the library.

- The Chrono::Engine required
- The Chrono:: Irrlicht and the Irrlicht library, Chrono::OpenGL and its dependencies. Both are optional
- The Chrono::FEA and Chrono::MKL (optional)

The Chrono::Engine supports the notion of a system. In our case, the following components are considered a system

- Powertrain

Chrono::Vehicle encapsulates templates for systemsand subsystems in polymorphic C++ classes:

- A base abstract class for the system/subsystem type (e.g. Chrono::ChSuspension)
- A derived, still abstract class for the system/subsystem template (e.g. Chrono::ChDoubleWishbone)
- Concrete class that particularize a given system/subsystem template (e.g. Chrono::HMMWV_DoubleWishboneFront)

1.2 Setup simulation

1.2.1 Setup the vehicel chrono::vehicle::sedan::Sedan

Now that we went over the basics of the Chrono::Vehicle library let's try to set up a basic simulation; namely a vehicle that move in straight line. Concretely, we will use an instance of the chrono::vehicle::sedan::Sedan class. The following code initializes the vehicle instance for the simulation

```
// Create the vehicle, set parameters, and initialize
    Sedan vehicle;
    vehicle.SetContactMethod(contact_method);
    vehicle.SetChassisFixed(false);
    vehicle.SetInitPosition(ChCoordsys<>(initLoc, initRot));

    vehicle.SetTireType(tire_model);
    vehicle.SetTireStepSize(tire_step_size);
    vehicle.SetVehicleStepSize(step_size);
    vehicle.Initialize();

    vehicle.SetChassisVisualizationType(chassis_vis_type);
    vehicle.SetSuspensionVisualizationType(suspension_vis_type);
    vehicle.SetSteeringVisualizationType(steering_vis_type);
    vehicle.SetWheelVisualizationType(wheel_vis_type);
    vehicle.SetTireVisualizationType(tire_vis_type);
```

1.2.2 Create the application

```
// Create the vehicle Irrlicht application
ChVehicleIrrApp app(&vehicle.GetVehicle(), &vehicle.GetPowertrain(),
                    L"Steering_XT_Controller_Demo",
        irr::core::dimension2d<irr::u32>(800, 640));

app.SetHUDLocation(500, 20);
app.SetSkyBox();
app.AddTypicalLogo();

irr::core::vector3df v1(-150.f, -150.f, 200.f);
irr::core::vector3df v2(-150.f, 150.f, 200.f);
irr::core::vector3df v3(150.f, -150.f, 200.f);
irr::core::vector3df v4(150.0f, 150.f, 200.f);
app.AddTypicalLights(v1, v2, 100, 100);
app.AddTypicalLights(v3, v4, 100, 100);
app.EnableGrid(false);
app.SetChaseCamera(trackPoint, 6.0, 0.5);
app.SetTimestep(step_size);
```

The following link can be used to consult for further information http://api.projectchrono.org/ tutorial_install_project .

References

[1] Åström K. J., Murray R. M. *Feedback Systems. An Introduction for Scientists and Engineers*

[2] Philip , Florent Altche1, Brigitte dAndrea-Novel, and Arnaud de La Fortelle *The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?* HAL Id: hal-01520869, https://hal-polytechnique.archives-ouvertes.fr/hal-01520869

[3] Marcos R. O., A. Maximo *Model Predictive Controller for Trajectory Tracking by Differential Drive Robot with Actuation constraints*