1. Introduction

As robotics research advances, robots are becoming increasingly complex in terms of kinematics and dynamics, with greater emphasis on walking or balancing robots. Concurrently, there is an increasing use of vision, whether monocular, stereo, and/or color based, as a primary sensing modality. Developing the algorithms and techniques to control such complex robot platforms operating in dynamic worlds becomes a difficult challenge. For expensive, large, or otherwise fragile robots, this is especially true.

It has been well established within the agents and robotics literature that simulation can be a powerful tool for speeding up the development cycle for robot control systems. For a simulation to be useful, however, it must capture the important characteristics of the physical world, where importance is a function of the problem in question.

We consider the problem of highly dynamic, multi-robot environments where vision is the primary sensor. In our prior research, we have spent considerable time investigating such domains with a particular emphasis on the use of simulation as a mechanism to improve the speed of the development cycle. We have found that to have impact upon development in such environments, a number of key features are desirable, which will be discussed in the following section. Although there have been numerous simulation engines developed within the community, with varying levels of realism, to the authors knowledge no simulator successfully contains all of these features for complex, potentially dynamically balancing robots operating in dynamic multi-robot domains.

In this paper, we report on our progress developing ÜberSim, a simulation engine targeted to address these problems. We report on the techniques we have developed to address the challenges for physically accurate simulation of a robot operating in a dynamic environment with vision as its primary sensor. Concretely, we have implemented our techniques to create a simulator for the Segway RMP robot, a dynamically balancing robot based on the commercialized Segway scooter that uses vision as its only external sensor.

2. Key Challenges

When building a simulation environment that is useful and realistic for vision-centric robots such as the Segway RMP, several main challenges must be considered. These issues include: realistic robot dynamics and simulation accuracy, support for high-frequency control loops, accurate vision synthesis with occlusion and artifacts, latency modeling, and flexible and extensible robot specification.

By default, the Segway RMP exists in an unstable state and is only kept stable via a tight control loop that solves the inverted-pendulum control problem in real-time. As this behavior is intricately tied to the dynamics of the robots, simulating this behavior requires both the ability to simulate accurate dynamics while also accurately simulating the effects of changes in actuation at high frequencies.

Vision synthesis is also equally important as it allows for correct occlusion and visual artifacts, unlike other methods of passing scene data directly to the simulated robot. Furthermore, the simulation must take into account the effects of latency, which are often omitted in simulations. Latency in actuators and sensors can have many effects on large and unstable robots, such as the Segway RMP.

Another challenge in building an effective simulation environment relates to the interface for building virtual robots and integrating their control code with the simulation. Ideally, the integration should be seamless and should maximize reuse of existing control code in the simulation. Furthermore, the specification of the virtual agent’s composition should be done in a simple and intuitive manner.

3. High-Fidelity Simulation – Our Approach

ÜberSim addresses the various challenges with a combination of mature, proven technologies as the cornerstones for the simulation framework as shown in Figure 1. Realistic dynamics and simulation accuracy are provided through the use of a rigid-body physics engine known as the Open Dynamics Engine (ODE). ODE provides the ÜberSim server with accurate simulation of rigid-body dynamics, including varying surface and material properties. Given the set of properties for each
material, contact joints are created in ODE and the collisions are resolved appropriately. The library also provides fast, two-phase collision detection routines which we make use of in ÜberSim.

**Figure 1. Components of ÜberSim.**

High-frequency control loops are handled in ÜberSim by the simulation and network protocols, which simulate forward conservatively so as to ensure that the client and server components of the simulation remain synchronized at all times. The simulation can also be run at a configurable rate which can be tuned to match with the frequency of sensors in the simulation, allowing users to select a balance between accuracy and speed depending on the requirements of their simulated environment.

In order to address the issue of latency, ÜberSim provides several latency models which can be easily configured for each simulated robot and allow for correct simulation of sensor and actuator latency. This further reduces translation time between simulation and real-world control as the latency is built into the simulator and does not need to be added to the client control code.

Finally, in order to present a streamlined interface, ÜberSim provides an XML description language for specifying the composition of a virtual robot. Modifying the robot’s composition requires only small changes to the robot’s XML definition and does not require recompilation of the server. Furthermore, modification and recompilation of client code is also unnecessary if the changes do not affect the existing control algorithms, such as is the case for most modifications to physical structure.

### 4. Results

The ÜberSim server was implemented in C++ and a simple client was written in C#. The client simulates a virtual Segway RMP with an inclinometer and a camera affixed to the front of the unit. Dynamic balance is maintained using a simple PD controller where the controller gains were determined through experimentation. The message processing loop implemented in the client requires only a few lines of code and is extremely compact, owing to the simple network protocol between client and server. An example image from the server visualization is given in Figure 2.

Camera vision is also synthesized on the server and sent to the client as raw bits over a TCP connection. This makes the vision data available in a format similar to that of physical camera. The camera primitives allow for the user to specify the camera’s frame rate and resolution. The motion generated by the simulation generally matches well with that of the real Segway, with the exception that our PD loop is most likely less sophisticated than that of the real Segway RMP.

**Figure 2. ÜberSim Segway Simulation.**

Our next focus is to integrate the actual Segway RMP balance code with the simulated Segway and to examine the similarity between the simulated and actual motion under various conditions. Conversely, we are also working to develop control code on the simulated Segway and then transfer it to the actual Segway RMP to analyze the simulation applicability in greater detail. We also intend to continue work to enhance the graphical realism of the simulator’s rendered output, as well as examine the scalability of the system with larger numbers of simulated robots and more complex actuator and sensor primitives.

### References


