Design and Validation of a Whegs Robot in USARSim

B.K. Taylor
Case Western Reserve Univ.
Cleveland, OH, USA
brian.k.taylor@case.edu

S. Balakirsky, E. Messina
NIST
Gaithersburg, MD, USA
stephen.balakirsky@nist.gov,
elena.messina@nist.gov

R.D. Quinn
Case Western Reserve Univ.
Cleveland, OH, USA
roger.quinn@case.edu

***Abstract*— Simulation of robots and other vehicles in a virtual domain has multiple benefits. End users can employ the simulation as a training tool to increase their familiarity and skill with the vehicle without risking damage to the robot, potential bystanders, or the surrounding environment. Simulation allows researchers and developers to benchmark the robot's performance in a range of scenarios without needing to physically have the robot and or necessary environment(s) present. Beyond benchmarking current designs, researchers and developers can use the information gathered in the simulation to guide and generate new design concepts. USARSim (Urban Search and Rescue Simulation) is a high fidelity simulation tool that is being used to accomplish these goals within the realm of search and rescue. One particular family of robots that can benefit from simulation in the USARSim environment is the Whegs™* series of robots developed in the Biologically Inspired Robotics Laboratory at Case Western Reserve University. Whegs robots are highly mobile ground vehicles that use abstracted biological principles to achieve a robust level of terrestrial locomotion. This paper describes a Whegs robot model that was designed and added to USARSim's current array of robots. The model was configured to exhibit the same kind of behavioral characteristics found in the real Whegs vehicles. Once these traits were implemented, a preliminary validation study was performed to ensure that the robot interacted with its environment in the same way that the real-life robot would.**

***Keywords*:** *USARSim, Biologically Inspired Robotics, Whegs, Urban Search and Rescue, Simulation*

I. Introduction

A. Background on USARSim

Urban Search and Rescue Simulation (USARSim) is a high fidelity simulation tool that can be used to simulate robots in various environments [11]. USARSim is built on top of Epic Games' Unreal Tournament 2004* (UT2004) physics engine known as Unreal Engine 2.0. The Karma Physics Engine* [9] is utilized to simulate physics within the game. Unreal Script, the object oriented programming language for UT2004, is used to give robots functionality and to define how the robot will interact with its environment. Unreal Editor (UnrealEd) is used to create virtual worlds, or maps. It is also used to create 3D solid models (static meshes) that can be used to either construct a robot, or to construct obstacles and objects that are placed within a particular map.

The idea behind USARSim is as follows. A virtual robot is built by creating static meshes to represent its individual parts. The parts are connected to each other through a configuration file that specifies where and how parts are connected to each other (motors, hinges, ball-and-socket joints, etc). In addition to the robot, a map is created with obstacles that must be overcome, and objects and/or victims that need to be found. The physics engine handles the dynamics of how the robot should interact with the map that it is placed in. By adjusting parameters known as Karma Parameters [9,10], the performance of the robot in simulation can be changed. For example, changing the inertia tensor of a robot will affect its ability to rotate about particular body axes within a given world. For robots and maps, end users can select from the options available in a current release of USARSim [11], or design their own. Controller software is used to perform a range of tasks such as issuing simple drive commands, implementing autonomous features into the vehicle, and running multiple vehicles in a given environment [11,12]. This kind of setup allows an individual to build and simulate robots relatively quickly and inexpensively from both computational and monetary standpoints. USARSim currently has applications in end-user training for Urban Search and Rescue robots, and in the RoboCup Simulation League [2,3].

A disadvantage of USARSim is that the Karma Physics Engine is proprietary. This means that the exact mechanics behind how the engine uses the Karma Parameters cannot be obtained. Testing has been done to gain a better, more quantitative understanding of how the Karma Parameters affect the simulation, and how the parameters map to real-world quantities. For example, conversion factors between the simulation's length scales (Unreal Units and Karma Units) and real length scales (meters) have successfully been established and implemented in more recent releases of the software. However, there are still parameters

*Commercial equipment and materials are identified in this paper in order to adequately specify certain procedures. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

where this kind of understanding has not been reached. Ultimately, this means that iterative testing and comparisons must be done on a given real and virtual robot to determine the set of Karma Parameters that yields the most realistic performance of the virtual vehicle.

B. Background on Whegs

Whegs robots are highly mobile unmanned ground vehicles that were developed in the Biologically Inspired Robotics Laboratory at Case Western Reserve University. Their locomotion is based on abstracted biological principles observed in cockroach locomotion [1]. Unlike RHex which is a biologically inspired robot that predates Whegs [7], Whegs robots employ an appendage called a wheel-leg, which is made up of a hub with spokes equally spaced about the hub's central axis (Fig. 1).

Fig. 1. A three spoke wheel-leg appendage

Most wheel-legs have three spokes. Rotating the wheel-legs about their central axes at a constant speed allows a given Whegs robot to move in the same way that a wheeled vehicle would be driven. In addition, the spokes allow the robot to obtain discontinuous footholds on irregular terrain, similar to legs [7]. Furthermore, the spokes also allow the wheel-leg to reach footholds that are taller than the wheel-leg radius. These wheel-leg features allow Whegs robots to be propelled in a similar manner to wheeled vehicles. They also enable Whegs robots to climb over and negotiate terrain that may be impassable to wheeled vehicles (Fig. 2).

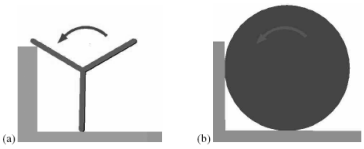

Fig. 2. (a) A wheel leg is able to obtain footholds on obstacles that are taller than the wheel-leg radius. (b) A wheel is unable to reach footholds of equal height

Cockroaches have six legs and typically walk in a tripod gait, meaning that the front and rear legs on one side move in phase with the middle leg on the opposite side. Contralateral pairs of legs move out of phase with each other (e.g. when the front left leg is in swing, the front right leg is in stance) [5]. When the animal comes to a large barrier, it moves its contralateral legs into phase to aid in surmounting the obstacle [6]. Similarly, Whegs robots employ six wheel-legs with contralateral pairs being placed out of phase such that the vehicle walks in a nominal tripod gait. Each axle features a compliant mechanism that allows the robot to passively bring its wheel-leg pairs into phase. This feature aids the robot in surmounting obstacles, and allows it to passively adapt its gait to changing and irregular terrain (Fig. 3).

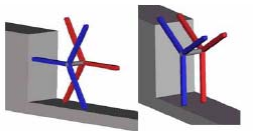

Fig. 3 Compliant mechanisms in the axles allow wheel-leg spokes to come into phase which aids in climbing obstacles

Also, cockroaches have a body flexion joint. They use the joint pitch the front of their bodies down to avoid high centering and to allow their front legs to reach the substrate [6]. More recent Whegs robots have been outfitted with body flexion joints or similar reasons. In addition, the joint also allows the vehicle to pitch its body upward to get a foothold on an obstacle during a climb (Fig. 4) [8].

Fig. 4. Whegs robot with a body flexion joint surmounting a larger step than the robot's body height

In addition to Whegs robots, a series called Mini-Whegs™* has also been developed (Fig. 5).

Fig. 5. Mini-Whegs and its relative size as compared to a cockroach

These are intended to be smaller, more compact versions of Whegs robots. They are on the order of 0.09 m long (9 cm), and have a top speed of about 10 bodylengths/second (0.9 m/s). Because of their small size, Mini-Whegs robots only possess four wheel-legs instead of six [4]. The four wheel-legs move in a diagonal gait. While some work has been done with implementing compliance into the axles of these robots, for simplicity, Mini-Whegs typically lack both torsional compliance and body flexion joints [4].

While different types of Whegs robots have been constructed and tested, there is formalized method for developers to gauge a robot's performance limits or test the viability of design ideas before construction begins. Also, the only current way to learn how to operate a Whegs robot is to drive a real robot. A Whegs simulation would allow robot designers to test their ideas before construction begins, allowing them to make design changes that will improve performance. The simulation can also be used by developers to test robots in environments that are not readily available, or potentially damaging. This would allow designers to gauge a given robot's performance limits. For robots in development, performance enhancing changes could then be

implemented. In addition to design work, a Whegs robot simulation would allow end users to become skilled in operating Whegs vehicles in numerous environment(s) without having the robot or environment(s) physically present. This can reduce the risk of damaging the robot. If a simulated robot is incapacitated, the simulation can be restarted rather than having to repair or rebuild the vehicle.

In this paper, a virtual Whegs robot was created and given the same behavioral characteristics as a real robot. The virtual robot's performance was then benchmarked against the real robot. Section II describes the approach and methods used to impart functionality to the virtual robot and benchmark its performance. Section III describes the results obtained during testing. It also describes some of the problems that were encountered during testing and how these issues were resolved. The final section summarizes the work presented in this study and discusses future work.

II. Methods

To perform this study, a generic Whegs robot model was first created in USARSim by adding a "Whegs" class. This class and its base classes were given functionality to enable the virtual robot with the same behavioral characteristics that are found in a real Whegs vehicle. After the virtual robot had the necessary behaviors, it was run through several tests to gain an understanding of how particular Karma Parameters affected its performance. Once the effects of these parameters were known, the virtual and real robots were placed in test scenarios with the same conditions. The results of these tests were compared and used to make changes to the virtual robot's Karma Parameters to improve its performance.

For the purposes of this study, the virtual vehicle was modeled after a Mini-Whegs robot with torsional compliance. This was done in an effort to lay the ground work for creating any given Whegs vehicle while still maintaining a degree of simplicity during modeling and testing. As stated above, Mini-Whegs robots typically lack a body flexion joint and only use four wheel-legs. These features make Mini-Whegs robots easier to simulate because there are fewer features to control and less wheel-legs to monitor during testing. Torsional compliance, while not present on all Mini-Whegs vehicles, was not a feature present in USARSim. Because this feature is found on many of the Whegs vehicles, it was felt that successfully modeling and implementing it would aid in laying the fundamental groundwork necessary for building more specific and accurate Mini-Whegs and full size Whegs models.

A. Developing a Whegs™ Robot Model

Modeling a Whegs vehicle can be broken down into two main steps: creating the appropriate static meshes, and writing and modifying classes to give the virtual vehicle the same types of behavioral characteristics as the real vehicle. The static meshes were created using UnrealEd. For the purposes of simplicity, the chassis was modeled as a rectangular block. The wheel-legs were modeled as cylinders (the hub of the wheel-leg) with three rectangular blocks (the spokes of the wheel-leg) attached to them and placed 120° apart (Fig. 6).

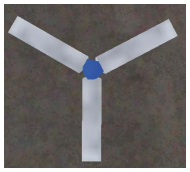

Fig. 6. Wheel-Leg appendage created in UnrealEd

Two separate wheel-leg meshes were made. One resembled a "Y" shape (Fig. 6) while the other was an inverted "Y". The two meshes were used as contralateral leg pairs to achieve proper wheel-leg phasing.

Incorporating the appropriate behavioral characteristics into the robot involved adding new functionality into USARSim. As mentioned above, there were no native USARSim features that allowed for passive torsional compliance. To solve this problem, a new class called "KDSpringy" was created. This class tells USARSim to make a hinge joint (KHinge) whose hinge type is set to a spring (KHingeType=Springy). Physically, this is like connecting two objects together with a torsional spring that is able to have stiffness and damping about a particular axis. The spring attempts to maintain an input angle (KDesiredAngle) between two objects placed in an Unreal map (Actors). In USARSim, this corresponds to maintaining a desired angle between the current part and its parent. Once the KDSpringy class was created and the appropriate base classes were modified, it was implemented in the following way.

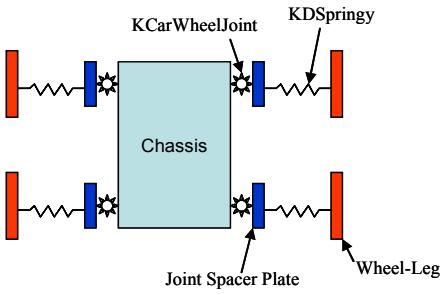

Fig. 7. Illustration of how passive torsional compliance is implemented

A static mesh that is used as a spacer plate was created. As can be seen in Fig. 7, the chassis was connected to the spacer plate via a KCarWheelJoint. A KCarWheelJoint is a joint that has a spin axis that is driven by a motor, and a steering axis that is driven by a controlled motor that attempts to achieve a specified orientation, similar to a servo. The spacer plate was connected to the wheel-leg via KDSpringy. This setup effectively made the spacer plate the actual "wheel" that drove the vehicle. However, because the wheel-leg's parent is the spacer plate, the two parts rotate together. Differences in their rotational speeds come from the reaction torques and forces that are experienced by the wheel-leg from the terrain, and from the parameter used to set

the stiffness of KDSpringy. A large stiffness allows the spring to withstand large reaction torques before deflecting, thus allowing the wheel-leg and spacer plate to move at more closely matching speeds (this corresponds to the wheel-leg functioning like a normal wheel). A lower stiffness means that the spring is easier to displace and must be deflected to the point where it is able to exert a large enough torque to spin the wheel-leg. This enables the spacer plate to wind up the spring and build torque when a particular wheel-leg is unable to move, which allows the contralateral wheel-leg to come into phase with it. This is precisely how Whegs robots behave in reality when surmounting obstacles.

A disadvantage to this method is that UT2004 appears to only use tire properties for a tire (a KTire in UT2004) that is connected to a KCarWheelJoint. KTire properties allow the user to control the following properties of the tire: Rolling Friction, Lateral Friction, Rolling Slip, Lateral Slip, Minimum Slip, Slip Rate, Tire Softness, Tire Restitution, and Tire Adhesion. Even though the wheel-legs are defined as KTires, since they are connected via KDSpringy, it appears that they only have what are known as KActor properties. KActor properties allow the user to control the following parameters: KFriction and KRestitution. As can be seen, a KTire is the ideal case because more control is allowed over how the tire will interact with the environment. This problem can be rectified by altering the static mesh so that individual wheel-leg spokes are added to the hub as tires. However, this solution requires a larger number of parts and more class functionality to make the robot function properly. Also, this approach gave adverse preliminarily results, which are discussed in the next section. Because real Mini-Whegs robots do not use formal tires, the preliminary validation was performed with the wheel-legs as KActors. It was felt that this approach would provide a good first approximation of the appropriate set of Karma Parameters that would yield realistic virtual performance while narrowing the search space at the same time.

B) Virtual Test Maps and Validation Testing

After the virtual robot was developed, two maps were created to test the vehicle's performance. One of these maps was a large empty room to test the vehicle in walking and running while minimizing its chances of hitting a wall. The other map included basic obstacles such as: ramps, standard 2x4 boards (3.81 cm by 8.89 cm actual cross sectional dimensions) and textbooks for climbing, a straightaway for walking/running testing, and stairs and large drops for falling and impact testing. These worlds were used to compare the virtual robot's performance to that of the real vehicle. In this study, attention was focused on walking/running and basic climbing over textbooks. The following metrics were used to evaluate the virtual robot's performance:

- Top speed of about 0.9 m/s without significant end-over-end rotation (~25 rad/s wheel-leg drive speed)
- End-over-end rotation when attempting to climb up a wall at higher wheel-leg drive speeds
- The ability to surmount obstacles (textbooks in this study) that are 0.04 m tall in head on and oblique angle (30^o) approaches (Fig. 8)

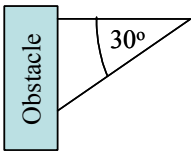

Fig. 8. Top view of head on and oblique approaches

Fig. 9 provides an illustration of how robots are currently validated

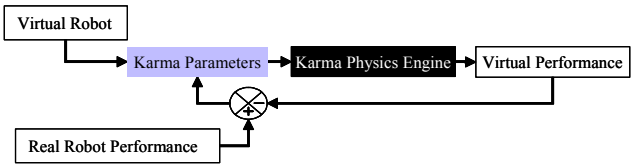

Fig. 9. Illustration of how robots are currently validated

As stated above, the Karma Physics Engine is proprietary, which means that the exact way in which the engine uses the Karma Parameters to affect the simulation cannot be directly obtained. In addition, the engine uses its own unit system (e.g. lengths are in Unreal Units or Karma Units depending on the context). The mapping between the Unreal Unit System and real world quantities is known for some parameters (e.g. length and time). However, conversions for other parameters (e.g. force and torque) are still under investigation. Due to the proprietary nature of the engine, to perform a validation, a robot is run through the engine with an initial set of Karma Parameters, as illustrated in Fig. 9. This yields the robot's virtual performance, which is then compared with real robot performance through the use of video data and any other relevant performance metrics for the robot in question. The information learned from the comparison is used to modify the Karma Parameters. After parameter modification, the virtual robot is run through the engine again for comparison with the real robot. This cycle is repeated until the virtual performance meets a desired level. For actual validation testing, this method was combined with the following procedure:

1) Baseline Run: First, the virtual robot was run through a range of drive speeds with an initial set of Karma Parameters. The goal of this run was to obtain performance data for an initial set of parameters for the purposes of comparison. In the open room, vehicles were run through the following wheel leg drive speeds: {0, 2, 10, 15, 20, 25, 30} rad/s.

2) Individual Karma Parameter Variation: After the baseline performance test, individual Karma Parameters were varied through a range of values while leaving all other parameters at their initial settings. For each value, the virtual robot was run through the same set of drive speeds used in the baseline run. The purpose of these runs was to obtain data that illustrated how each parameter affected the vehicle's performance.

3) Physical Reasoning: At this stage, physical reasoning was used to determine what conditions were required for the virtual robot to behave in a particular way in order to explain its performance and the effects of individual Karma Parameters.

4) Karma Parameter Search: At this point, with an understanding of the effects of different Karma Parameters, the method illustrated in Fig. 9 was used to improve the virtual robot's performance.

The virtual robot was compared to physical observation of a real Mini-Whegs robot. All simulation trials were recorded using FRAPS* [13] video capturing software. In addition, the vehicle's instantaneous velocity, position in the world, orientation with respect to the world coordinate frame, time, and speed change commands were logged to various files. The logged parameters were used to plot the vehicle's speed and velocity components against time. Velocities were reported in both world (fixed) and vehicle (moving) coordinates to help quantify the robot's behaviors. Steering was not used in any of the tests.

III. Results

A. Dimensions Used in the Simulation

Initial testing (phases 1-3 of the above described procedure), was done with a slightly larger vehicle. Phase 4 was performed with a smaller vehicle (Table 1).

		Initial Dimension	Final Dimension
Body	Length (m)	0.1143	0.09
	Width (m)	0.09144	0.068
	Height (m)	0.01905	0.02
Wheel-Legs	Diameter (m)	0.096	0.0762

Table 1. Initial and final dimensions used in the simulation

The latter dimensions were chosen because they more accurately reflect the size and performance basis of current Mini-Whegs robots. For example, the 10 bodylength/second speed listed above corresponds to a 0.09 m bodylength, so for this performance metric, the smaller vehicle size is more appropriate. The study could have been performed with the larger size vehicle since a real vehicle could be created that has larger dimensions. The dimensional change is only used here for convenience in comparing virtual and real performance.

B. Functionality for Maintaining Proper Wheel-Leg Phasing

During testing, it became apparent that new functionality would need to be added to the "Whegs" class to ensure that the virtual robot maintained proper wheel-leg phasing. Initially, when the vehicle spawned into a world, it would spawn properly with its wheel-legs out of phase, but then "fall" due to its mass such that the wheel-legs were in phase (Fig. 10).

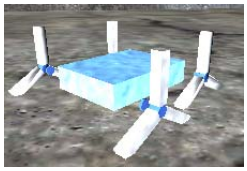

Fig. 10. Wheel-Legs are unable to maintain proper phasing without extra class functionality

Occasionally, all of the wheel-legs would stay out of phase upon spawning such that the robot could be tested. However, if the robot was not stopped with the correct orientation and speed, the wheel-legs would "fall" out of phase. This behavior appeared to be independent of the torsional stiffness that was provided, and even occurred when the wheel-legs were connected directly to the chassis via a KCarWheelJoint. This was problematic because real Whegs robots maintain proper phasing even when stopped. It was determined that this problem was due to the nature of the KCarWheelJoint class. This class rotates a given Actor about a spin axis by applying a torque to overcome external torques. If the motor applies no torque, then the actor will be rotated about the motor's spin axis by all other external torques. Physically, a KCarWheelJoint is analogous to having an axle that rotates a wheel mounted in a perfectly frictionless bearing and motor. Real Whegs robots have a drive train that connects each wheel leg to a single drive motor and torsional springs that are pretensioned which causes them to maintain proper phasing even when the motor is not running. To fix this problem, a member function was added to the "Whegs" class that forces the KCarWheelJoint to achieve zero angular velocity by using a preset torque value when the robot is stationary (drive speed = 0 rad/s). This solution appeared to solve the problem.

C. Wheel-Legs Behaving as KActors Vs. Tires

Initially, the wheel-legs were implemented as KTires. However, when attempts were made to run the robot, the wheel-legs would rotate but not cause the robot to translate, resulting in the wheel-legs spinning in place and the robot itself not making any forward progress. It was determined that because the wheel-legs were not connected to KCarWheelJoints, KActor properties were used instead of KTire properties. The default KActor friction value is zero, which led to the wheel-legs perfectly sliding on a given substrate. An attempt to rectify this problem was made by implementing the solution proposed above: making each spoke a KTire that is connected to the central hub via a KCarWheelJoint. Because of the problems experienced in maintaining proper wheel-leg phasing mentioned in the section above, a function was added to the "Whegs" class that forces the spokes to maintain their initial orientation relative to their parent hub. This solution resulted in the wheel-leg spokes drifting into altered positions over time, particularly at higher drive speeds. An attempt to remedy this problem was made by increasing the torque used to maintain the spoke orientation. This resulted in the vehicle going through seemingly nonphysical end-over-end rotation at higher wheel

leg drive speeds (about 15 rad/s and higher) while still translating forward, and did not appear to remove the drifting problem. It was found that the only apparent way to influence the problem was to increase the vehicle's inertia tensor or angular velocity resistance (KAngularDamping) Karma Parameters. These parameters only seemed to slow down the rotation. They also had to be raised to levels much higher than any other vehicle in the USARSim, including vehicles that are more massive such as the Hummer.

The nonphysical nature of this behavior and its solution prompted performing the validation with the wheel-legs behaving as KActors instead of KTires. This approach led to behavior that appeared to be more physically relevant when compared to the real vehicle. Also, as stated above, because this approach offers a narrower parameter search and because the wheel-legs on Mini-Whegs vehicles are not formal tires, it was felt that using the wheel-legs as KActors would provide a reasonable approximation that would allow for relatively simple but effective preliminary validation.

D. Effects of Individual Karma Parameters

The robot's speed and velocity components were plotted in both world and vehicle coordinates. These plots were used as a tool to help examine the effects of individual Karma Parameters on the robot's performance. Example plots are shown in Fig. 11.

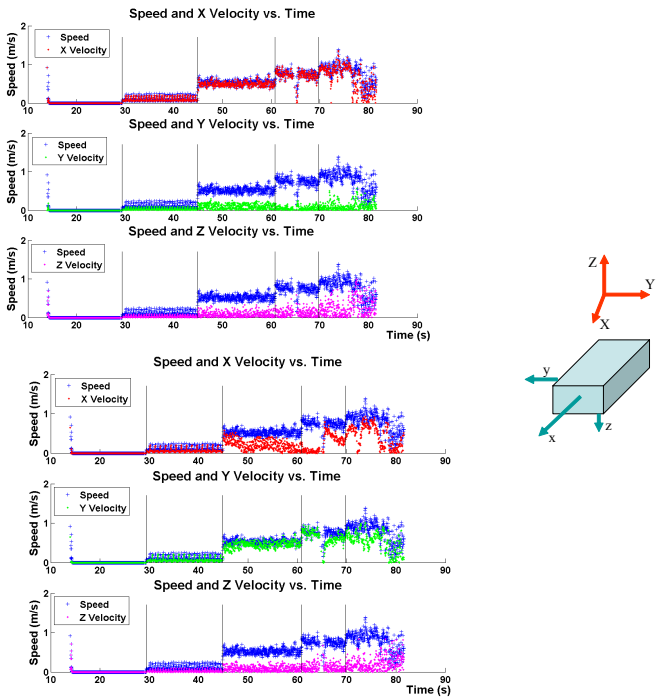

Fig. 11. Velocity in vehicle coordinates (top) and world coordinates (bottom). In vehicle coordinates, x is forward, y is starboard, and z is out the bottom of the vehicle. The absolute value of the velocity components is plotted here for comparison with the speed.

The vertical black lines are the times at which speed change commands were issued. The vehicle's initial speed (which occurs around 15 s in the plots shown above) is due to it spawning into the world. By looking at the body-fixed coordinate plot, it can be seen that the X-component of velocity in the vehicle coordinates is nearly identical to the speed, meaning that the vehicle is making forward progress and not translating to its left or right. In addition, there is a point around 65 s where the X-velocity and the speed both drop while the Z-velocity spikes. Upon comparison with the video data, it was observed that this was the location at which the robot flipped over, or end-over-ended. This kind of feature was present in all cases where the robot flipped over. The world coordinate plot illustrates the vehicle's tendency to move in a particular direction within the world. In this plot, the robot is initially heading mostly in the Y direction. However, after it flips over at around 65 s, it also begins to have motion in the X-direction as well. In addition to these kinds of observations, the data in the plots can be used to look at other phenomena such as the average speed of the vehicle for a given time interval and the variability of the data about the average.

The following Karma Parameters were singly modified while leaving all other parameters at their initial settings to determine their effects on the performance of the vehicle:

1) ChassisMass: This is the mass of the chassis. With the initial parameters that were set, it was found that altering this parameter did not appear to have a large impact on the overall performance of the vehicle in terms of being able to reach a top speed, or end-over-ending.

2) KMass: This is the mass of the Spacer Plates. This mass was varied to determine the effects of raising and lowering mass that is not coincident with the vehicle's center of mass. Initally, the mass of the chassis was very small, both compared to the spacerplates, and in an absolute sense, so this test also revealed how the vehicle's overall mass affected performance. When these masses were lowered to a value of 0.002 in the Unreal Unit System, the vehicle immediately began to end over end at higher drive speeds (10 rad/s and over). At mass values of 2 in the Unreal Unit System, the vehicle appeared to perform in a relatively predictable manner with only occasional end-over-end instances occurring at drive speeds between 25 rad/s and 30 rad/s.

3) KFriction: This is the friction present in the wheel legs. It was found that, as one would expect, higher values of friction (10 in this study) resulted in the wheel-legs not slipping on the substrate as much during walking. Visible slippage occurred with lower friction values (0.5 in the Unreal Unit System). Both of these friction values yielded roughly the same vehicle average speeds for a given drive speed. However, the variation about the average was much higher for the larger friction value. This was attributed to stronger braking forces in the step cycle. The wheel-legs provide both propulsive and braking forces, where braking occurs in the beginning of the stance phase, and propulsion occurs towards the end. Because the friction value is higher, both the brake and propulsion forces are increased. Therefore, when a wheel leg touches down, it is able to provide better traction to propel

the vehicle, but also has a greater tendency to retard its motion initially.

4) KRestitution: This parameter is similar in concept to the coefficient of restitution used in collision analysis. A value of 1 corresponds to an elastic collision between two objects. Values less than one result in increasingly inelastic collisions. At a KRestitution value of 1 in the Unreal Unit System, as the vehicle's drive speed was increased, it appeared to have increasingly continuous elastic collisions with the ground while its forward speed appeared to reach a relatively constant value that became independent of the input drive speed. As a result, the vehicle's average speed at higher drive speeds seems to flat-line when compared to other tests. The average speed also had a great deal of variability for each drive speed.

5) Torsional Stiffness: The torsional stiffness of the rear wheel-legs was set to 250 in the Unreal Unit system for all runs. This value appeared to make the back wheel-legs rotate with the spacer plates under all circumstances. The front torsional stiffness of the front wheel-legs was adjusted to see how adjusting the stiffness affected their motion. As expected, higher values of stiffness led the wheel-legs to behave more like conventional wheels, where low stiffness values allowed the spring to "wind up" before rotating the wheel-legs. Excessively low values of stiffness cause wheel-legs to fall out of phase when the vehicle is spawned. At these low values, when a drive command was issued, the wheel-legs would not rotate at first. However, after the springs were deflected sufficiently, they would rotate forward to release the tension as one would expect.

E. Karma Parameter Search

After the effects of the above mentioned parameters were understood from the single parameter variations, testing was done to move the virtual robot towards matching real performance. Table 2 indicates the parameters that were changed.

	Karma Parameter (Unreal Unit System)	Initial Value	Current Value
Chassis	ChassisMass	0.00342	0.75
	MaxTorque	32000	50
	MotorTorque	2400	50
	KCOMOffset (X, Y, Z)	0	0.04464
	KCOMOffset (X, Y, Z)	0	0
	KCOMOffset (X, Y, Z)	0	0
Wheel Legs	Wheel-Leg Kfriction	1	0.75
	Wheel-Leg KRestitution	0	0.1
	Wheel-Leg Kmass	0.0008	0.08
Spacer Plate	Spacer Plate Kmass	1	0
	Spacer Plate KInertiaTensor(0) ~ Ixx	0.0035	0
	Spacer Plate KInertiaTensor(3) ~ Iyy	0.0066	0
	Spacer Plate KInertiaTensor(5) ~ Izz	0.0035	0

Table 2. Initial and Current Karma Parameter Values

Column 1 is a listing of the initial values of the Karma Parameters. Column 2 represents the current values that they have been adjusted to. As can be seen from Table 2, the following general changes were made. First, because spacer plates are not found on the real robot, their mass and inertia values were set to zero so that they would have no effect on the dynamic characteristics of the vehicle. Based on the results obtained from varying individual Karma Parameters, lowering the mass of the Spacer Plates caused severe end-over-ending of the vehicle. This prompted raising the ChassisMass property of the vehicle to 1 in the Unreal Unit System, which drastically reduced this problem. Based on testing of an actual Whegs robot on tile, it was observed that the wheel-legs slip during walking at higher speeds, similar to what can occur with lower values of the KFriction parameter. The vehicle also appeared to have a degree of elasticity with the ground, similar to when the KRestitution values were raised. Accordingly, the KFriction and KRestitution values were lowered and raised respectively. The center of mass of the vehicle (KCOMOffset) was not varied in the single parameter variation study. However, after examining a particular Mini-Whegs vehicle, it was found that many of the components such as the steering servo, steering mechanism, and battery are located towards the front of the vehicle. Also, the virtual vehicle still went into end-over-ending behavior more than was desired. Therefore, the KCOMOffset value was adjusted to bring the center of mass of the vehicle forward. This reduced the end-over-ending behavior slightly, but did not completely remove the problem. While the center of mass is not typically in the forward section of a Mini-Whegs vehicle, the change was made here to judge its impact on the performance. In addition to these parameters, the KCarWheelJoint motor torque and wheel-leg masses were also changed. The motor torque was ultimately lowered from its initial value of 2400 to 50 in the Unreal Unit System to give the vehicle a more realistic level of drive torque (for comparison, the Hummer uses a motor torque value of 2400). The wheel-leg masses were raised from 0.0008 to 0.08 (Unreal Unit System).

F. Performance Results

The parameter changes that were made appeared to move the virtual vehicle towards matching the performance of the real vehicle. With the final set of parameters given above, the robot was able to walk at near top speeds with occasional end-over-ending. It was also only able to end-over-end at walls with higher wheel-leg drive speeds (10 rad/s and up), which is normal Mini-Whegs behavior. In climbing tests, the robot was able to successfully surmount a 0.04 m obstacle with head-on and 30^{o} approaches.

IV. DISCUSSION

Basic functionality modifications of existing USARSim base classes along with functionality implemented in the newly defined "Whegs" class appears to successfully replicate

the general behaviors of torsional compliance and wheel-leg phasing found in Whegs robots. Karma Parameter modification based on physical reasoning and observation of real robots through videos and direct interaction appeared to result in improved, more realistic performance of the virtual robot in walking, running and basic climbing. In addition, the procedure used in the parameter modification yielded insight into how each individual parameter contributes to the overall performance of the vehicle. The procedure also yielded the velocity history of the vehicle in world and body coordinates, along with the speed of the vehicle, and the average speed for a given time increment. This data was used to determine if the vehicle is able to attain a particular drive speed, the degree to which the vehicle collides with the ground, and the vehicle's tendency to end-over-end.

While the accomplishments listed above are significant first steps towards creating an accurate representation of a Whegs robot in USARSim, there are many steps that can be taken to improve the virtual robot's performance. With respect to the real robot, high speed video capture methods can be used to obtain data and establish performance metrics that can also be measured within the simulation for more accurate benchmarking. In terms of the virtual robot, several steps can be taken including: making the wheel-legs function as actual tires, refining the behavioral characteristics of the "Whegs" class and investigating the use of more detailed and accurate static meshes. In terms of the Karma Parameters, the above tests can be conducted in more depth and expanded to better understand the effects of individual parameters on robot performance. More testing can also be done to better understand how individual Karma Parameters map to real world quantities. Also, while individual Karma Parameters can be varied, they are not necessarily independent, so the coupling between Karma Parameters needs to be understood. For the validation, a number of steps can be taken. Several map substrate surfaces can be made from KActors and tuned to match the performance of real surfaces such as concrete, tile, wood, etc. The virtual robot can then be tested and compared to the real robot on each of these surfaces, which would yield a better representation of the appropriate Karma Parameters for the robot. With more rigorous performance metrics defined, numerical methods could be employed to help determine how well the virtual robot matches the real robot's performance. All of these steps would lead to a more reliable and repeatable simulation.

ACKNOWLEDGEMENTS

We would like to thank Ben Balaguer and Richard Bachmann for their contributions in developing this work.

This research was performed under an appointment to the U.S. Department of Homeland Security (DHS) Scholarship and Fellowship Program, administrered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. The opinions expressed here do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE.

REFERENCES

[1] Allen, T.J, Quinn, R.D., Bachmann, R.J., and Ritzmann, R.E. (2003) "Abstracted Biological Principles Applied with Reduced Actuation Improve Mobility of Legged Vehicles," IEEE International Conference on Intelligent Robots and Systems (IROS 2003), Las Vegas.

[2] Balakirsky S, Scrapper C, Carpin S and Lewis M. (2006) "USARSim: Providing a Framework for Multi-Robot Performance Evaluation," Performance Metrics for Intelligent Systems, Gaithersburg MD, USA., August 21-23, 2006.

[3] Balakirsky S, Scrapper C, Carpin S and Lewis M. "USARSim: A RoboCup Virtual Urban Search and Rescue Competition," Proceedings of SPIE, 2007.

[4] Morrey, J.M., Lambrecht, B., Horchler, A.D., Ritzmann, R.E., and Quinn, R.D. (2003) "Highly Mobile and Robust Small Quadruped Robots", IEEE International Conference on Intelligent Robots and Systems (IROS 2003), Las Vegas.

[5] Quinn, R.D., Kingsley, D.A., Offi, J.T. and Ritzmann, R.E., (2002), "Improved Mobility Through Abstracted Biological Principles," IEEE Int. Conf. On Intelligent Robots and Systems (IROS'02), Lausanne, Switzerland.

[6] Quinn, R.D., Nelson, G.M., Ritzmann, R.E., Bachmann, R.J., Kingsley, D.A., Offi, J.T. and Allen, T.J. (2003), "Parallel Complimentary Strategies For Implementing Biological Principles Into Mobile Robots," Int. Journal of Robotics Research, Vol. 22 (3) pp. 169-186.

[7] Saranli, U., Buehler, M. and Koditschek, D. (2001) "RHex A Simple and Highly Mobile Hexapod Robot". International Journal of Robotics Research, 20(7): 616-631.

[8 Schroer, R.T., Boggess, M.J., Bachmann, R.J., Quinn, R.D., and Ritzmann, R.E. (2004) "Comparing Cockroach and Whegs Robot Body Motions," IEEE Conference on Robotics and Automation (ICRA '04), New Orleans.

[9] "Unreal Developer Network Karma Reference." Unreal Developer Network. (9/25/2007) http://wiki.beyondunreal.com/wiki/

[10"Unreal Wiki: The Unreal Engine Documentation Site." (DATE HERE) http://wiki.beyondunreal.com/wiki/

[11] USARSim (9/25/2007) http://sourceforge.net/projects/usarsim

[12] MOAST (9/25/2007) http://sourceforge.net/projects/moast

[13] FRAPS (9/25/2007) http://www.fraps.com/