The OpenSTORM Project: A Proposal for an Open Simulation Tool for Robotics, Automation, and Manufacturing

Todd D. Murphey
University of Colorado

Ken Goldberg
UC Berkeley

Vijay Kumar
University of Pennsylvania

Kevin M. Lynch
Northwestern University

A. Frank van der Stappen
Utrecht University

Computation for complex, mechanical systems has traditionally focused on minimizing computational complexity. This is the result of scalable simulation typically focusing on computer graphics needs, such as those required when producing a feature-length animated film. However, these techniques are often hand-tuned by experienced animators, with the only goal being plausibility rather than validation of a physical mechanism in simulation. As a result, we have well-established simulation techniques for systems that operate in virtual worlds. Simulation of manufacturing systems must be physically meaningful, and must be so without much input or tuning from the user. Even finite element methods (FEMs) often require hand-tuning of parametric inputs to the model, and the computational complexity of FEM techniques often make them inappropriate for computations that require multiple simulations. Hence, we propose to develop a common software infrastructure to be used both for evaluating existing software simulation tools as well as eventually providing a platform for robust simulation. *The goal is to allow more seamless sharing of algorithms to enable demonstration and comparison.* This code will be open source, and will be specifically designed with manufacturing systems in mind, such as assembly lines where multiple physical processes are occurring simultaneously. An assembly line can include conveyor belts, grippers, and blades, and would moreover come equipped with sensors of various sorts. Each of these components has its own modeling needs that would be modularized and incorporated into `OpenSTORM`'s specification language. Without a package such as the one that would result from the `OpenSTORM` project, we cannot virtually prototype, verify controllers, or develop planners for a wide range of systems.

This proposal stems from the recent IEEE International Conference on Robotics and Automation (ICRA 2008) *Workshop on Algorithmic Automation*, the ICRA 2008 *Workshop on Contact Models for Locomotion and Manipulation*, and the upcoming IEEE Conference on Automation Science and Engineering (CASE 2008) *Workshop on Algorithmic Automation*. The ICRA workshops were unusually well-attended–with over 80 people attending at times–indicating recognition in the research community of the importance of the topics. All three of these workshops illustrate the need for physically reliable simulation techniques for manufacturing systems. Some of the realizations from these workshops are very high-level in nature–that successful next-generation manufacturing requires advanced technologies and that the United States' continuing global competitiveness requires investment in our manufacturing infrastructure. In particular, reliability in manufacturing is key to consistent product quality. Reliable simulation is necessarily only one component of what is needed, but will play an important role in improving manufacturing state of the art by improving our rapid-prototyping and system verification capabilities.

Our view is that a single simulation environment, independent of its intended application, is not feasible. Hence, a key goal of the `OpenSTORM` project would be to provide a common specification for simulation that involves rigid body systems with constraints and impacts–both elastic and plastic–for polygonal bodies and special nonpolygonal bodies (e.g., circles, cylinders). This is similar to the `Collada` project [1], but `Collada` was primarily intended for animation specification rather than physical simulation. The simulation specification environment would need to support:

1. both two and three dimensional problems;

2. constraints, including closed-kinematic chains;

3. impacts, by only specifying the geometry of bodies and type of impact–plastic or elastic with coefficient of restitution;

4. friction–stick/slip interactions based on Coulomb conditions and arbitrary friction algebraic descriptions;

5. finite-dimensional representations of elasticity.

Many simulators require the use of certain "magic" parameters–parameters that do not have any natural physical interpretation but nevertheless are essential to the underlying algorithm. For instance, Open Dynamics Engine (`ODE`) [4] has error reduction parameters (`erp`) that are used by animators to correct for non-physical behavior. These parameters should be included as a separate input to the simulation environment, partially to make them as explicit as possible to the end-user. Such parameters typically include time step Δt and various tolerances, but different simulation packages will have different needs. For instance, some simulators, such as those that use splines, do not explicitly use the notion of a time step. In the case of a package like the `trep` package [7, 2, 3], these parameters would be time step, constraint tolerance, and root-solver tolerance. The goal is to force simulators to be explicit about what heuristics are being used.

The package should also provide a standard for input/output. Our view of this is that the input specification should be in generalized coordinates represented in a tree-form data structure while the output trajectory should be in free-body representation (a copy of $SE(3)$ for every rigid body). This choice is natural because the constrained free-body representation (another standard input for dynamic simulation) is a special choice of generalized coordinates. Visualization techniques typically require free-body representations (such as OpenGL), leading to that being a natural representation of output. As a consequence, we anticipate using OpenGL as the visualization environment. There is a question of whether every choice of specification should have a method of converting back to the original generalized coordinates directly from the free-body representation, since this is what many control strategies require. At minimum, `OpenSTORM` will be compatible with motion planning software such as `OOPSMP` [5].

The simplest version–and the most immediately attainable–of `OpenSTORM` is to input a system in generalized coordinates and output its trajectory in constrained free-body representation. This requires a language choice–we have found s-expressions to be a particularly compact representation that still do not leave any ambiguity. Our opinion is that at first `OpenSTORM` should *not* be CAD compatible. CAD models create too much ambiguity in terms of the underlying algorithm, which may not lead to rigorous comparisons of simulation techniques. At some later time it will make sense to put a CAD software layer on top of `OpenSTORM`.

Error handling should be a standard part of `OpenSTORM`, and should include checking that constraints are being satisfied as well as various known symmetries (energy and momenta). Other types of error handling that is mechanically relevant should be incorporated as well–one of the goals of this project would be to determine what other type of error handling should be present.

Lastly, `OpenSTORM` should be a place where we can build a library of canonical examples that can be used to test simulation techniques. The scissor lift example [3] appears to illustrate limitations and strengths of various simulation techniques and is therefore an example of what we are looking for. These benchmark examples should encompass the types of situations we expect to see in manufacturing examples, and should be one of three types:

1. Academic examples with analytic or near-analytic solutions
   (a) scissor lift [3];
   (b) bouncing ball or polygon;
   (c) elastic impact between simple end-effector and surface.

2. Complex examples of practical manufacturing situations

   (a) pushing many impacting polygonal parts on a frictional surface;

   (b) an industrial bowl feeder, having as its input a polyhedron and a distribution of initial conditions and its output a mechanically correct distribution of outcomes.

These canonical examples will illustrate what capabilities every simulator should have (e.g., attaching a revolute joint to a stationary frame, handling plastic impacts, etcetera) and force the simulator to specify canonical situations it cannot handle.

We anticipate the initial `OpenSTORM` simulation modules being `trep` [7] and `daVinci` [6]. We will encourage others to contribute as well. We anticipate that this project will start with a representative from each simulation package contributing to the development of a common dynamic simulation mark up language (DSML), and getting consensus on the semantics of this mark up language will be nontrivial. However, once that mark up language is in place, `OpenSTORM` would simply be a front-end for that file format.

References

[1] COLLADA. http://www.collada.org, 2008.

[2] E. Johnson and T. D. Murphey. Discrete and continuous mechanics for tree representations of mechanical systems. In *IEEE Int. Conf. on Robotics and Automation*, 2008.

[3] E. Johnson and T. D. Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *IEEE Transactions on Robotics*, Submitted.

[4] Open Dynamics Engine (ODE). http://www.ode.org, 2008.

[5] OOPSMP: An Object-Oriented Programming System for Motion Planning. http://www.kavrakilab.org/software/OOPSMP, 2008.

[6] daVinci Code: Accurate Physical Simulation. http://www.robotics.cs.rpi.edu/dvc/, 2008.

[7] TREP. http://trep.sourceforge.net, 2008.