CHRONO-PARALLEL OVERVIEW

Radu Serban
Chrono-Parallel
What is Chrono-Parallel?

A software library for OpenMP-based parallel simulation of Chrono models

- Middleware: can be embedded in third parties software
- Open source with BSD license
- Library developed in C++ (requires C++11)
- Cross-platform: compiles on GNU GCC, MSVC, etc.
Relationship to Chrono::Engine

- Chrono-Parallel relies on Chrono for all its modeling capabilities
- Supports a subset of Chrono modeling elements:
  - Rigid bodies with frictional contact (DEM-C or DEM-P)
  - Kinematic joints (revolute, spherical, translational, etc.)
  - Force elements (spring-dampers, actuators, etc.)
  - 1-D shafts and associated elements and constraints (shaft-body connection, gears, motors, etc.)
- No support for FEA
- Implements only the *Implicit Euler Linearized* time-stepper
- Chrono-Parallel uses different data structures and algorithms
Data structures and algorithms

- Chrono-Parallel uses structures of arrays (SOA) as opposed to arrays of structures (AOS)

- OpenMP parallel for loops

- Thrust (with OMP backend)
  - parallel algorithms library (sort, scan, reductions, etc.)
  - Interface similar to the C++ STL
  - [https://code.google.com/p/thrust/](https://code.google.com/p/thrust/)

- Blaze
  - High-performance (dense and) sparse matrix operations
  - Smart Expression Template implementation
  - [https://code.google.com/p/blaze-lib/](https://code.google.com/p/blaze-lib/)
Collision detection

• Broad phase
  • Based on binning with AABBs
  • Coming soon: adaptive 2-level hierarchical grid

• Narrow phase
  • Different options:
    • MPR (Minkovski Portal Refinement)
    • GJK (Gilbert-Johnson-Keerthi)
    • SAT (Separating Axis Theorem)
  • Hybrid option (SAT with fallback to MPR)
Visualization

• Run-time visualization with OpenGL
  • Optional support
  • Dependencies: GLFW (window inputs and events), GLEW, GLM (math)

• Post-processing with POV-Ray
Chrono-Parallel utilities

• Samplers for granular dynamics
  • Uniform grid, Poisson-disk sampling, hexagonal close packing
  • Sampling of different bounding volumes (box, sphere, cylinder)
  • Allows 2D sampling (rectangle, disk)

• Generators for granular dynamics
  • Create particles, using a given sampler, randomly selecting from a mixture of ingredients with prescribed expectation
  • Mixture ingredient: multivariate normal distributions for particle size and contact material properties

• Utility wrapper classes for interfacing with Chrono::Vehicle
• Utilities for file I/O, output for POV-Ray post-processing, etc.
• Convenience functions for creating bodies with simple geometry (contact and graphics)

• Some overlap with Chrono::Engine – common functionality will be eventually merged
User code

• New type of ChSystem:
  ChSystemParallelDVI system;
  or
  ChSystemParallelDEM system;

• Create elements in the mechanical system using Chrono::Engine functions

• Specify solver settings by directly accessing the structures
  ChSystemParallel::GetSettings()->solver
  and
  ChSystemParallel::GetSettings()->collision

• Advance simulation by invoking
  chrono::ChSystem::DoStepDynamics()
  or
  chrono::opengl::ChOpenGLWindow::DoStepDynamics()
system.GetSettings()->perform_thread_tuning = false;

system.GetSettings()->solver.solver_mode = SLIDING;
system.GetSettings()->solver.max_iteration_normal = 0;
system.GetSettings()->solver.max_iteration_sliding = 200;
system.GetSettings()->solver.max_iteration_spinning = 0;
system.GetSettings()->solver.max_iteration_bilateral = 50;
system.GetSettings()->solver.tolerance = 0.1;
system.GetSettings()->solver.alpha = 0;
system.GetSettings()->solver.contact_recovery_speed = 10000;

system.ChangeSolverType(APGD);

system.GetSettings()->collision.narrowphase_algorithm = NARROWPHASE_HYBRID_MPR;
system.GetSettings()->collision.collision_envelope = 0.01;
system.GetSettings()->collision.bins_per_axis = I3(10, 10, 10);
User code – solver settings DEM-P

```cpp
system.GetSettings()->perform_thread_tuning = false;

system.GetSettings()->solver.max_iteration_bilateral = 50;
system.GetSettings()->solver.tolerance = 0.1;

system.GetSettings()->solver.contact_force_model = HERTZ;
system.GetSettings()->solver.tangential_displ_mode = MULTI_STEP;

system.GetSettings()->collision.narrowphase_algorithm = NARROWPHASE_HYBRID_MPR;
system.GetSettings()->collision.bins_per_axis = I3(10, 10, 10);
```
Simulation results
Vehicle over cohesive granular terrain (DEM-C)

- Vehicle model
  - No steering
  - Simplified kinematic powertrain and 4WD driveline models
  - Rigid tire models (mesh)

- Granular material
  - Spheres (different sizes): approx. 300,000
  - $\mu = 0.4$, $F_c = 200$ N

- Integration time-step: $10^{-3}$ s, ~4.3 seconds per step
- APGD solver, 45 iterations
Vehicle over cohesive granular terrain (DEM-P)

- Chrono::Vehicle model
  - Double-wishbone independent suspension
  - Pitman arm steering
  - Simplified kinematic powertrain and 4WD driveline models
  - Rigid tire models (cylindrical or mesh)

- Granular material
  - Spheres (2 cm radius): approx. 153,000
  - $E = 10^8 \text{ Pa}$, $\nu = 0.3$, $cr = 0.1$, $\mu = 0.8$, $F_c = 100 \text{ N}$

- Integration time-step: $5 \cdot 10^{-5} \text{ s}$, ~0.8 seconds per step
Fording simulation – frictionless granular fluid

- Fording Setup: (dimensions reduced from the original physical model)
  - Depth 8ft, Width 14ft
  - Length of bottom 15ft

- Chrono-Vehicle HMMWV Model
  - 9 Body Model, 4WD, simple drivetrain, no driver
  - Chassis/Wheels converted to convex hulls from OBJ
  - Throttle set to 1 and kept constant

- 1,051,840 rigid spheres in an HCP grid

- Integration time-step: 0.001 s, ~50 seconds per step
Fording simulation – constraint fluid

- 1,050,225 fluid markers in an HCP grid
- Integration time-step: 0.001 s, ~20 seconds per step
Code availability

• Source code: https://github.com/projectchrono/chrono-parallel

• Bug reports, feature requests, pull requests → GitHub

• API documentation (doxygen) http://api.chrono-parallel.projectchrono.org/

• Animations http://sbel.wisc.edu/Animations/ https://vimeo.com/uwsbel/videos