Simulation of Tracked Vehicles on Granular Terrain Leveraging GPU Computing

Toby Heyn
Acknowledgments

- Professor Dan Negut
- Professor Mario Trujillo
- Professor Chris Rutland
- Dr. David Lamb

- Professor Alessandro Tasora
- SBEL colleagues
  - Justin Madsen, Hammad Mazhar
Outline

- Problem statement
- Parallel solution of dynamics problem
- Parallel collision detection
- Tracked vehicle model
- Granular terrain model
- Simulation results
Problem Statement
Simulation-Based Engineering

- Holy Grail in Computer Aided Engineering (CAE)
  - Do Virtual Prototyping before you build the first hardware prototype
  - Cut costs & time to market

Model → Simulate → Validate
● What’s pushing the need for high performance computing in multi-body dynamics simulation?
Rover on Granular Terrain

- Wheeled/tracked vehicle mobility on granular terrain

Simulation in Chrono::Engine
Track Simulation using Commercial Software
Frictional Contact Simulation
[Commercial Solution]

- Model Parameters:
  - Spheres: 60 mm diameter and mass 0.882 kg
  - Forces: smoothing with stiffness of 1E5, force exponent of 2.2, damping coefficient of 10.0, and a penetration depth of 0.1
  - Simulation length: 3 seconds

![Simulation](image)

**CPU time v. Number of Spheres**

\[ y = 0.8385x^2 - 7.2607x + 16.154 \]

\[ R^2 = 0.9985 \]
Frictional Contact: Two Faces of the Same Coin

- Long simulation times in off-the-shelf software traced back to the underlying formulation of the frictional contact problem
  - Draws on a “smoothing” (penalty) approach
    - Sophisticated but slow
    - General purpose tool

- Solution embraced draws on DVI (Differential Variational Inequalities)
  - A set of differential equations combined with inequality constraints
  - Less general than penalty approach
Dynamics Problem
Equations of Motion: DVI Approach

\[ \dot{q} = T(q)v \]

\[ M(q)\ddot{v} = f(t, q, v) - g_q^T(q, t)\lambda + \sum_{i=1}^{N_c} (\gamma_n^i D_n^T q^i + \gamma_u^i D_u^T q^i + \gamma_w^i D_w^T q^i) \]

\[ g(q, t) = 0 \]

\[ 0 \leq \Phi^j(q, t) \perp \gamma_n^i \geq 0 \quad i = 1, 2, \ldots, N_c \]

\[ \gamma_u^i, \gamma_w^i = \arg\min_{\gamma_n^i \geq \sqrt{(\gamma_u^i)^2 + (\gamma_w^i)^2}} \left( \gamma_n^i v^T D_n^T q^i + \gamma_u^i v^T D_u^T q^i + \gamma_w^i v^T D_w^T q^i \right) \]

\[ \text{Gap Function, for Contact } i \]

\[ \text{Friction Impulse Components, for Contact } i \]

\[ \text{Total Number of Contacts} \]

\[ \text{Friction Dissipation Energy} \]
Discretized EOMs

\begin{align*}
\mathbf{M}(\mathbf{v}^{(t+1)} - \mathbf{v}^{(t)}) &= h \mathbf{f}(\mathbf{r}^{(t)}, \mathbf{q}^{(t)}, \mathbf{v}^{(t)}) - \mathbf{g}_q^T(\mathbf{q}^{(t)}, t) \lambda + \sum_{i=1}^{N_c} \left( \gamma_{u}^i \mathbf{D}_{u}^{T,i} + \gamma_{w}^i \mathbf{D}_{w}^{T,i} \right)
\end{align*}

\text{Force Balance Equations}

\begin{align*}
\frac{1}{h} \mathbf{g}(\mathbf{q}^{(t)}, t) + \mathbf{g}_q^T \mathbf{v}^{(t+1)} + \mathbf{g}_r &= \mathbf{0}
\end{align*}

\text{Holonomic Kinematic Constraints}

\begin{align*}
0 \leq \frac{1}{h} \Phi(\mathbf{q}^{(t)}, t) + \mathbf{D}_{u}^{T,i} \mathbf{v}^{(t+1)} & \perp \gamma_{u}^i \geq 0 \\
& i = 1, 2, \ldots, N_c
\end{align*}

\text{Contact Complementarity Conditions}

\begin{align*}
(\gamma_{u}^i, \gamma_{w}^i) &= \arg \min_{\gamma_{u}^i, \gamma_{w}^i \geq 0} \left( \gamma_{u}^i \mathbf{v}^T \mathbf{D}_{u}^i + \gamma_{w}^i \mathbf{v}^T \mathbf{D}_{w}^i \right)
\end{align*}

\text{Coulomb Friction Model}

(Stewart, 1998)
Outer Loop (Time-Stepping)

1. Set $t = 0$, step counter $l = 0$, provide initial values for $q^{(l)}$ and $v^{(l)}$.

2. Perform collision detection between bodies. For each contact $i$, compute $D_{i,n}$, $D_{i,u}$, $D_{i,w}$.

3. For each body, compute forces $f(t^{(l)}$, $q^{(l)}$, $v^{(l)})$.

4. Use CCP Algorithm to solve the cone complementarity problem and obtain unknown impulse $\gamma$ and velocity $v^{(l+1)}$.

5. Update positions using $q^{(l+1)} = q^{(l)} + hL(q^{(l)})v^{(l+1)}$.

6. Increment $t := t + h$, $l := l + 1$, and repeat from step 2 until $t > t_{\text{end}}$. 
Inner Loop (CCP Algorithm)

1. For each contact $i$, evaluate $\eta_i = 3 / \text{Trace}(D_i^T M^{-1} D_i)$.

2. If some initial guess $\gamma^*$ is available for multipliers, then set $\gamma^0 = \gamma^*$, otherwise $\gamma^0 = 0$.

3. Initialize velocities: $v^0 = \sum_i M^{-1} D_i \gamma_i^0 + M^{-1} \dot{k}$.

4. For each contact $i$, compute changes in multipliers for contact constraints:
   
   $\gamma_i^{r+1} = \lambda \Pi_{\gamma_i} \left( \gamma_i^r - \omega \eta_i \left( D_i^T v^r + b_i \right) \right) + (1 - \lambda) \gamma_i^r$;
   
   $\Delta \gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$;
   
   $\Delta v_i = M^{-1} D_i \Delta \gamma_i^{r+1}$.

5. Apply updates to the velocity vector:
   
   $v^{r+1} = v^r + \sum_i \Delta v_i$.

6. $r := r + 1$. Repeat from 4 until convergence or $r > r_{\text{max}}$.
Parallelism, Opportunities
[at each integration time step]

1. Parallel Collision Detection
2. (Body parallel) Force kernel
3. (Contact parallel) Contact preprocessing kernel
4. (Contact parallel) CCP contact kernel
5. (Constraint parallel) CCP constraint kernel
6. (Reduction-slot parallel) Velocity reduction kernel
7. (Body parallel) Body velocity update kernel
8. (Body parallel) Time integration kernel
Collision Detection
Scalable Collision Detection (CD)

- 30,000 feet perspective:
  - Carry out spatial partitioning of the volume occupied by the bodies
    - Place bodies in bins (cubes, for instance)
  - Follow up by brute force search for all bodies touching each bin
    - Embarrassingly parallel
  - Similar in spirit to LeGrand algorithm (GPU Gems 3)
    - Yet capable of handling heterogeneous geometries (spheres, triangles, ellipsoids)
CD: Binning

- Example: 2D collision detection, bins are squares

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td></td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Body 4 touches bins A4, A5, B4, B5
- Body 7 touches bins A3, A4, A5, B3, B4, B5, C3, C4, C5
- In proposed algorithm, bodies 4 and 7 will be checked for collision by three threads (associated with bin A4, A5, B4)
Parallel Binning: Summary of Stages

- Stage 1: Find number of bins touched by each body, populate $T$ (body parallel)
- Stage 2: Parallel exclusive prefix scan of $T$ (length of $T$: $N$)
- Stage 3: Determine body-to-bin association, populate $B$ (body parallel)
- Stage 4: Parallel sort of $B$ (length of $B$: $M$)
- Stage 5: Find bin starting index, populate $C$ (bin parallel)
- Stage 6: Parallel sort of $C$ for pruning (length of $C$: $N_b$)
- Stage 7: Determine # of collisions in each bin, store in $D$ (bin parallel)
- Stage 8: Parallel prefix scan of $D$ (length of $D$: $N_b$)
- Stage 9: Run collision detection and populate $E$ with required collision info (bin parallel)

- $N$ – number of bodies
- $N_b$ – number of bins
- $M$ – total number of bins touched by the bodies present in the problem
Speedup - GPU vs. CPU (Bullet library)

GPU: NVIDIA Tesla C1060
CPU: AMD Phenom II Black X4 940 (3.0 GHz)
Spherical Decomposition

- Represent complex geometry as a union of spheres
- Allows use of fast parallel collision detection
Spherical Decomposition

- Create triangular surface mesh
- Fit a single sphere through the vertices of each triangle
  - Surface normal defined by right hand rule
  - Compute center of circumcircle
  - Use iterative method to achieve target center ratio $f = d/R$
Spherical Decomposition

Native Geometry
Export model as parasolid

SolidWorks
Translate model such that center of mass coincides with global origin

Cubit
Generate triangular surface mesh

Matlab
Translate to wavefront file

Matlab
Generate sphere-set, including refinement

Collision Geometry
Use in dynamic simulation

parasolid *.x_t
IGES *.igs
abaqus *.inp
Wavefront *.obj
sphere-set *.txt
Sphere-set Refinement
Examples…

- **Chain model**
  - 10 links
  - 7,797 spheres per link

- **Plow model**
  - 31,791 spheres in plow blade model
  - 15,000 spheres representing terrain
Redundant Contacts

- When compound bodies collide, many contacts may be identified between the same pair of bodies.
- Redundant contacts can lead to poor convergence for iterative CCP solver.
- Use random subset of contacts acting between a given pair of bodies.

Collision geometry: 20x20 grid of spheres
Subset: 8 contacts per pair
Tracked Vehicle Model
Track Components

1,594,908 spheres per track
Track Model
Track on Rigid Terrain
Granular Terrain Model
Terrain Representation

- Rigid plane
- Rigid sphere-set
- Discrete granular
Granular Terrain Model

- Represent terrain as collection of discrete particles
- Match terrain surface profile
- Capture changing granularity with depth
Random Filling

- Create particles at random locations in the domain

- Particle radius is $r \pm 0.1r$, uniformly distributed

- Preprocessing simulation allows particles to settle
Mesh-Based Terrain

- Mesh file gives elevation of terrain on regular grid
- Choose desired particle size, sample grid to place each particle
- Offset mesh in global y direction to create different layers
Heightmap-Based Terrain

- Heightmap image defines elevation by grayscale
  - Black=0=lowest
  - White=1=highest
- Must know maximum feature size
- Sample image for grayscale value – convert to elevation
- Offset particles to create layers
Terrain Model

- Perform preprocessing step to generate initial conditions for terrain particles

- Utilize ‘moving bounding box’ approach during simulation
  - No effort wasted simulating particles far from the vehicle
  - Allows larger terrain sets/smaller particles to be used within current memory limits
Moving Bounding Box

- terrainSphere
  - double x, y, z, r
  - bool active, bottomLayer
  - ChBody* tBody

- At each time step perform the following
  - Check position of each active body in box
    - If it is outside the box, remove it from the simulation
    - If it intersects the box, set it fixed
  - Check position of each inactive body in box
    - If it is inside the box, add it to the simulation
    - If it intersects the box, set if fixed
  - Update terrainSphere data for each body for persistence of terrain
Moving Bounding Box Demo
Simulation Results
Track Simulation 1

Parameters:
- Driving speed: 1.0 rad/sec
- Length: 12 seconds
- Time step: 0.005 sec
- Computation time: 18.5 hours
- Particle radius: 0.027273 m
- Terrain: 284,715 particles
Track Simulation 2

Parameters:

- Driving speed: 1.0 rad/sec
- Length: 10 seconds
- Time step: 0.005 sec
- Computation time: 17.8 hours
- Particle radius: .025±.0025 m
- Terrain: 467,100 particles
Track Simulation 3

Parameters:
- Driving speed: ±1.0 rad/sec
- Length: 6 seconds
- Time step: 0.005 sec
- Computation time: 15.4 hours
- Particle radius: .025±.0025 m
- Terrain: 434,886 particles
Results: Positions

Track Simulation 1
Results: Reaction forces

Track Simulation 2

Vertical Reaction in Forward-Most Road Wheel (after Gaussian Filter)

Driving Torque to maintain Constant Angular Velocity (after Gaussian Filter)
Results: Track ‘Footprint’
Validation at Microscale

- Sand flow rate measurements
- Approx. 1 million bodies
- Glass beads
- Diameter: 100-500 microns
Summary of Contributions

- Preliminary implementation of GPU collision detection
- Spherical decomposition method
- Reduction of redundant contacts to appropriate subset
- Three methods of generating granular terrain profiles from various input sources
- Moving bounding box approach for terrain simulation
- Extraction of reaction forces from GPU dynamics solver
Current limitations

- Large mass ratios lead to poor convergence
  - Mass of track shoe/mass of terrain particle ~100,000

- Handling of redundant constraints is not systematic

- Capability of CCP lags that of CD by three orders of magnitude
  - CCP: 1 million body simulation
  - CD: 1.4 billion contacts
Future Work

- Multi-GPU CCP solver
- Algebraic multi-grid for faster convergence
- Cluster of GPU machines
- Integration with smoothed-particle hydrodynamics (SPH) for multi-physics simulation
  - Interested in blast-worthiness of ground vehicles
Conclusions

- Achieved tracked vehicle simulation on granular terrain
  - Used GPU to speed up solution of CCP and CD problems
- Developed method for performing fast collision detection between complex geometries
- Developed method for representing complex granular terrain
  - Terrain preprocessing
  - Moving bounding box
- Demonstrated the capability to obtain reaction forces in track model
Thank you.