S7316: Real-Time Robotics Control and Simulation for Deformable Terrain Applications Using the GPU

Daniel Melanz
Overview

1. Who are we?
2. What do we do?
3. How do we do it?
4. How can you do it?
Overview

1. Who are we?
2. What do we do?
3. How do we do it?
4. How can you do it?
Energid develops software for simulation and control of any robotic system.

**Actin** software is at the core of our business:

- Actin SDK
- Actin Applications
- Integration Services
Energid Technology Development

Core Robot Software Development

Software Productization

Commercial Market Penetration
Overview

1. Who are we?
2. What do we do?
3. How do we do it?
4. How can you do it?
Actin Software

Actin is a powerful constraint/optimization engine.
Actin Software

Simulation and control software for any robotic system

- Multi-robot coordination
- Dynamic collision avoidance
- Singularity avoidance
- Kinematically redundant mechanisms
- Complex kinematic chains
- Global path planning
- Real-time dynamic simulation
- IO and sensor feedback
- Easy integration of new hardware components
- Integration with CAD
- Desktop applications for Windows, Linux, OS X
- Control on VxWorks, Real-Time Linux and RTOS32
- Distributed processing over DDS
Overview

1. Who are we?
2. What do we do?
3. How do we do it?
   • System Description
   • Control Theory
   • Simulation (Sensors, Contact, Dynamics)
4. How can you do it?
Top-Level Simulation Hierarchy

- Simulation
  - Dynamic Simulation
  - Interface or Executive
  - Miscellaneous
    - Visualization Randomization
    - High-Level Control Etc.
  - Control System
    - Vector of Position Control Systems
    - Vector of Velocity Control Systems
  - Stated System
    - System
    - Configuration
    - State
The model is organized into a constant system, a slow-changing configuration, and a fast-changing state.
Generic Motion Control

- Any number and type of joint.
- Any number of bifurcations.
- Open and closed kinematics.
- Any number and type of constraint:
  - Pose (3D positioning and orienting)
  - Point positioning (1D, 2D, or 3D)
  - Orientation
  - Pose with one free axis
  - Distance
  - Joint positions
  - Linear constraints
  - Center of mass
  - Momentum
Velocity Control Framework

- Jacobian Equation

\[ V = J(q) \dot{q} \]

- \( V \) represents hand (or other constraint) motion
- \( q \) represents the configuration
- \( J \) is the manipulator Jacobian
Solution Method

- Using a matrix \((W)\), a vector \((F)\), and a scalar \((\alpha)\):

\[
\Phi = \left[ \frac{J}{N_J^T W} \right]^{-1} \left[ \frac{V}{-\alpha N_J^T F} \right]
\]

where

\[
J N_J = 0
\]

- This minimizes

\[
g = \frac{1}{2} \Phi^T W \Phi + \alpha \Phi^T F
\]
Filtering

- Scale the joint rates:

\[ \dot{q} = \beta(q, V) \left[ \frac{J}{N_J^T W} \right]^{-1} \left[ \frac{V}{-\alpha N_J^T F} \right] \]

- This allows robust navigation near kinematic and algorithmic singularities (and general positive behavior).
Control Expression Tree

- The control system has a tree structure.
- This allows a flexible description of the control system and supporting parameters (scalar, vector, matrix, and function).
- It also supports dynamic programming (tabulation of subproblem solutions).
Singularity Prevention

- Practical problems occur when
  \[
  \det \left[ \frac{\mathbf{J}}{\mathbf{N}_J^T \mathbf{W}} \right]
  \]
  changes sign from one time step to another.
- When this happens, assumptions regarding the linearized equations are violated.
- Actin uses \( N_J \) to detect determinate sign changes and stop the manipulator.
- Actin can also shift constraints into optimization and use redundancy to singular proximity.
Position Control

- Position control uses velocity control.

- Trapezoidal, intermediate, or proportional:

\[
\mathbf{v}_d = k_\lambda \cdot (\mathbf{P}_d - \mathbf{P}_a) \\
\mathbf{\omega}_d = k_a \cdot \Theta \cdot \hat{u}
\]
Force Control Optimization

- Force control uses position control, as position control uses velocity control.

This allows the use of kinematic redundancy to achieve secondary criteria.

It also allows position control to be combined with force control.

\[ V_F = D(k_p(F_d - F_m) + k_i \int (F_d - F_m) dt) \]
Mobile Kinematics

- All components of the toolkit apply to both fixed-based and mobile mechanisms.
Mechanism Dynamics

Articulation Dynamics: describes internal interaction

Impact Dynamics: describes physical contact
Composite Rigid Body Inertia

- Mobile-base dynamics equations:

\[ \tau = M(q) \dot{q} + C(q, \dot{q}) \dot{q} + D(q) A_b + B \]

\[ D^T \dot{q} + I_b A_b = F \]

- CRBI mobile-base simulation algorithm:

\[
\begin{bmatrix}
I^C_b & D(q)^T \\
D(q) & M(q)
\end{bmatrix}
\begin{bmatrix}
A_b \\
& \tau - C(q) \dot{q} - G(q) + B
\end{bmatrix}
= \begin{bmatrix}
F
\end{bmatrix}
\]

Cost: order(N^3)
Articulated Body Inertia

\[ \mathbf{F} = \mathbf{I}^A \mathbf{A} + \mathbf{B} \]

Articulated Body

Cost: order(N)
Contact & Particle Simulation

Approach: Use the Discrete Element Method (DEM)

- Individual particle interactions are modeled
  - Hertz/Hooke Normal Contact Model
  - Full Tangential Displacement History (Coulomb Friction)
  - Elastic/Plastic Rolling Resistance
  - Adhesion/Cohesion Forces

- Implement particle system interaction with rigid bodies (Manipulators) in Actin
The contact models currently implemented in Actin follow those of LIGGGHTS:

- **Hookean Contact Model**

  \[
  k_n = \frac{16}{15} E \sqrt{R} \left( \frac{15 m V^2}{16 E \sqrt{R}} \right)^{1/5} \\
  k_t = k_n \\
  \gamma_n = \sqrt{\frac{4 m k_n}{1 + \left( \frac{\pi}{\ln(\text{COR})} \right)^2}} \geq 0 \\
  \gamma_t = \gamma_n,
  \]

- **Hertzian Contact Model**

  \[
  k_n = \frac{4}{3} E \sqrt{R \delta_n} \\
  k_t = 8 G \sqrt{R \delta_n} \\
  \gamma_n = -2 \sqrt{\frac{5}{6}} \beta \sqrt{\frac{3}{2}} m k_n \geq 0 \\
  \gamma_t = -2 \sqrt{\frac{5}{6}} \beta \sqrt{m k_t} \geq 0,
  \]

where

\[
\bar{E} = \left( \frac{1 - \nu_i^2}{E_i} + \frac{1 - \nu_j^2}{E_j} \right)^{-1} \\
\bar{R} = \left( \frac{1}{R_i} + \frac{1}{R_j} \right)^{-1} \\
\bar{m} = \left( \frac{1}{m_i} + \frac{1}{m_j} \right)^{-1} \\
\beta = \frac{\ln(\text{COR})}{\sqrt{\ln^2(\text{COR}) + \pi^2}}
\]
Parallel (GPU) Implementation

**CPU**
- Allocate GPU memory
- Update Rigid Body States
- Integrate particle positions
- Create uniform grid and sort
- Process interactions
- Reduce Rigid Body Forces and Torques
- Render Particles

**GPU**
- Global memory – particle arrays
- Constant memory – Rigid Body States
- Integration Kernel
- Sorting Kernel
- Collision Processing Kernel
- Reduction Kernel
- Global memory – particle arrays

---

From Actin Simulation Loop

Simulation Loop

**Copy**
Shapes Demonstration
Simulating Interaction over Terrain

Simulate terrain in specified zone around a mobile robot as a particle system and transition to/from height field as the robot traverses the terrain

A. Define a zone of influence around the robot
B. Convert height field into particles
C. Simulate particle/rigid body interaction until particles move out of the zone
D. Update height field with new surface location based on particle locations
Simulating Interaction over Terrain
Simulating Interaction over Terrain
Simulating Interaction over Terrain (Full-Scale Military Vehicles)
Sensors

Sensor models are implemented using CPU & GPU programming.
Radar

- Uses size and surface properties to model radar response.
Camera Views

- Shadows can be rendered in camera views
LiDAR Sensors

- Simulate point cloud data
  - Use OpenGL rendered depth buffer to generate a point cloud for each scan in real-time
  - Use texture map warping to simulate non-linear scan projections

- Parameter space:
  - Gaussian noise in angle and distance
  - Scan rate
  - Motion blur
  - Dropout rate
HDR Rendering & Effects

- Generalized standard (OpenGL) rendering pipeline
- Added support for full-scene effects
- High Dynamic Range Rendering, Tone Mapping and Light Bloom implemented as examples
Ray Tracing

- Integration completed with NVIDIA Optix Ray tracing engine
  - Automated conversion of scene graph from OpenSceneGraph to Optix
  - Enables global illumination-based rendering, with emergent effects such as shadow
- Leverages GPU-based parallelization (using single or multiple cards)
Ray Tracing Demonstration

- Ray traced resolution: 1280x960 @
  - 10 fps on typical engineering laptop w/Quadro K1100M
  - 27 fps on engineering workstation w/Tesla K40
Automated Convoy Studies

- Convoy Study Configuration GUI was created to configure the studies
- Currently user can create a default study and then edit:
  - Randomizing optimizer
  - Input Variable Map
  - Output Variable Map
  - Randomizer
  - Optimizer

- With the convoy editor, the user can configure the convoy by clicking on vehicles
- With the path editor, the user can view/edit a path.
- The user can view the study metric results as each evaluation completes
Data Distribution Service (DDS)

- Typical multi-robot deployments
  - One robot controller machine per robot
  - Communication between the controllers
    - Commands
    - State synchronization
    - Sensor feedback
    - Hardware status

- Single-robot deployments
  - Increasing demand for “remote” capability

- DDS is a high-level communication architecture

- High-level communication is a natural fit for Actin’s high-level robotic control

- High-level communication is ideally fitted to distributed robotic control architectures
Distributed Robotic Control Architecture

**command primitives** (joint frames, end-effector pose, …)

- **actinViewer** (Control)
  - ddsCommonPlugin
  - ddsSimSynchronizationPlugin
  - ddsSensorManagerPlugin
  - ddsHardwareControlRequesterGUIPlugin

- **actinRT** (Robot 1)
  - ddsCommonPlugin
  - ddsSimSynchronizationPlugin
  - ddsSensorManagerPlugin
  - ddsHardwareControlProviderPlugin
  - hardwarePlugin

- **actinRT** (Robot N)
  - ddsCommonPlugin
  - ddsSimSynchronizationPlugin
  - ddsSensorManagerPlugin
  - ddsHardwareControlProviderPlugin
  - hardwarePlugin

**robot states**

- **sub**
  - actinViewer (Control)
  - actinRT (Robot 1)
  - actinRT (Robot N)

- **pub**
  - command primitives

- **sub**
  - robot states
Universal Robots Demo

- Two machines: UR3 and UR10
- UR10 machine
  - **Publish**: UR3 command and UR10 state
  - **Subscribe**: UR3 state
- UR3 machine
  - **Publish**: UR3 state
  - **Subscribe**: UR3 command and UR10 state
Universal Robots Master Slave Demo
Overview

1. Who are we?
2. What do we do?
3. How do we do it?
4. How can you do it?
Flexible Programming System

- Programs can be saved, restored, edited and archived with ease. Generic programs allow users to future proof their robotic programming investment.
- Programs are composed of lower level ‘tasks’ that can be reused.
- Actin works with any robotic hardware to maximize new designs or reanimate legacy robots.
- With run-time configurable kinematics, advanced features such as collision avoidance can be added to traditional robots with ease.
Flexible Programming System
Interface Enables New Control Approaches
Summary

1. Energid develops software for simulation and control of any robotic system.
2. The Actin software, a powerful constraint optimization engine, is at the core of our business.
3. Actin uses novel algorithms and the latest hardware for robotic system control and simulation.
4. Actin provides a flexible programming system.
We want to work with you!

For more information, visit our website: energid.com

Daniel Melanz
Robotics Research Engineer
melanz@energid.com