Lecture 8 – Constraints and articulated rigid bodies

Constrained = “tvingad”
Articulated = “förenad genom leder”

Joints, hinges – ways to constrain body motion
Example: Ball-and-socket or spherical (“kulled”)

Constrains one point on body i
to a point on body j.
In theory, no rotational constraints,
but in reality, there are angular limits.

Holonomic and Non-holonomic constraints

Kinematic constraints
Joint Modeling
Effectors/Motors
Applications

Holonomic and Non-Holonomic Constraints

• A rigid body has 6 degrees of freedom \((r, \Omega)\)
• (7 if we use quaternions)
• Two rigid bodies obviously have 12 DOF’s.
• We introduce the generalized position vector of bodies i and j,
  \[ s = [r_i, q_i, r_j, q_j]^T \]
• Constraints and joints reduce the number of degrees of freedom, by
  constraining some of the DOF’s

  \[ \Phi(t,s) = 0 \]  (bi-lateral, holonomic constraint, removes DOF’s)
  \[ \Psi(t,s) \geq 0 \]  (uni-lateral, non-holonomic constraint)
  \[ \Phi(s) = 0 \]  (holonomic and scleronemous)

  (rhonomorous - with explicit time dependence, i.e.
in addition to time dependence of \(s\))

4/10/2006
Holonomic Kinematic Constraints

- A joint is typically described by \( m \) holonomic constraints, each removing one degree of freedom.
- Example: ball-and-socket
- In world coordinates there is a common anchor point,
  \[
  \Phi_1(s) = x_i - x_j = 0 \\
  \Phi_2(s) = y_i - y_j = 0 \\
  \Phi_3(s) = z_i - z_j = 0 \\
  \Phi(s) = 0 \quad \text{(as a vector)}
  \]
- Each holonomic constraint removes one degree of freedom, and thus reduces the size of the generalized position vector.
- When two bodies are rigidly attached to each other, 6 DOF’s are removed, and the constrained system is effectively reduced to one rigid body – a composite body.

Holonomic Kinematic constraints

Read sections 7.5 and 18.5 carefully
We introduce the generalized velocity vector of the constraint,
\[
\dot{s} = Su
\]
where \( u = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T \)
and \( S \) that fulfils this is a transformation matrix,
\[
S = \begin{bmatrix}
1 & 0 \\
Q_x & 1 \\
0 & Q_y
\end{bmatrix}
\]
with \( Q \) rotation matrices, \( \frac{1}{2} \omega_x q_i = Q_i \omega_i \).
Given that \( q_i = [s, x_i, y_i, z_i]^T \) (don’t confuse the different \( s \)’s!)
\[
Q = \begin{bmatrix}
-x_i & -y_i & -z_i \\
-x_i & -y_i & -z_i \\
-x_i & -y_i & -z_i \\
-x_i & -y_i & -z_i \\
-x_i & -y_i & -z_i \\
-x_i & -y_i & -z_i
\end{bmatrix}
\]
Holonomic Kinematic constraints

- Each holonomic constraint can be differentiated into a kinematic constraint,

\[
\frac{d\Phi(s)}{dt} = \frac{d\Phi}{ds} \frac{ds}{dt} = \frac{d\Phi}{ds} S u = J_\phi u = 0
\]

- by which we have defined the Jacobian matrix, \( J_\phi \in \mathbb{R}^{m \times 12} \) with \( m \) the number of holonomic constraints. The Jacobian describes how the degrees of freedom are coupled/constrained/reduced.

- The reactive constraint force can always be written as

\[
F_\phi = J_\phi^T \lambda_\phi
\]

- Where \( \lambda_\phi \) are called Lagrange multipliers - which is typically what we solve for. This is also how to compute joint impulses – which is what we will do!

Non-Holonomic Kinematic Constraints

- Non-holonomic, unilateral, constraint with no explicit time dependence,

\[ \Psi(s) \geq 0 \]

- Just as with holonomic constraints, we can differentiate with respect to time to obtain a kinematic constraint,

\[ J_\psi u \geq 0 \]

- With constraint forces,

\[ F_\psi = J_\psi \lambda_\psi \]

- These constraints typically describe contacts, i.e. a point on body i is not allowed to penetrate body j.
Ball-and socket joint

When modeling constraints, we do not model the actual physical realization, so the theoretical model for joining two boxes with a ball-and-socket typically looks like:

Thus, the anchor point can be anywhere in space, and doesn’t need to relate to a real physical, or be inside or on the surface of the body. We need to be careful though, so that the joint doesn’t violate other constraints, e.g. force contacts/intersections.

Ball-and socket joint

We previously just looked at world coordinates. We need transform from body coordinates.

Now compute,

\[ J = \frac{d\Phi}{ds} \]

This is straightforward but involves some algebra. Refer to section 18.5. See also [http://www.ode.org/joints.pdf](http://www.ode.org/joints.pdf) for a velocity variant of this derivation and some useful identities.
Joint Modeling

- The Jacobian can be written as,
  \[ J = \{ J_p^i, J_p^j, J_o^i, J_o^j \} \]
  so that,
  \[ \mathbf{J}_u = [J_p^i, J_o^i, J_p^j, J_o^j] \begin{bmatrix} v_i & \omega_i & v_j & \omega_j \end{bmatrix}^T \]
  \[ = J_p^i v_i + J_o^i \omega_i + J_p^j v_j + J_o^j \omega_j \]
  \[ = b \quad \text{(velocity correction)} \]

- \( J_p^i v_i + J_o^i \omega_i \) velocity of body \( i \) joint bearings
- \( J_p^j v_j + J_o^j \omega_j \) velocity of body \( j \) joint bearings

i.e. relative joint bearing velocity must be zero – or what is required to correct the joint violation (constraint impulse).

Solving for kinematic constraints

Kinematic constraints are velocity constraints, not position constraints so we solve for velocity updates that correct the positions and thus the process is coupled with the time integration.

Example: Two particles can move along a line, where they are jointed together:
\[ J_u = v_i - v_j = 0 \]

However, if there is an error,
\[ r_i - r_j = r_e \]
we need to adjust velocities by \( b = r_e / \Delta t \) during a timestep \( \Delta t \) in order to correct for this error, i.e. we need,
\[ J_u = v_i - v_j = b \]

For more complex cases, one must solve the matrix equation to find the desired velocity and the corresponding impulse.
Solving for kinematic constraints

- In practise, a parameter is used to control the rate of correction of the constraint violation, which basically mixes the old solution with the new at each timestep to get gradual convergence. A typical value is 0.7-0.9 (ODE uses 0.8).
- Another trick is to mix in the constraint forces on the right hand side (“constraint force mixing”) with another parameter. This parameter behaves like a material parameter (spring constant) that makes it possible to model the constraint violation (i.e. large constraint force gives large error correction velocity, while small forces give smaller correction).
- Constraint force mixing can also give faster convergence since it “smoothes” or decouples the system. This can sometimes fail badly though ...
- Alternatives: Full variable set – don’t reduce degrees of freedom in the system because of constraints. Reduced variable set: Remove degrees that are removed by constraints. With reduced set, rigid bodies can be “partly composite”, i.e. stiffly constrained with no dynamics at all in the constraint. This goes beyond a pair-wise solver, since e.g. impulses on i must travel through the constraint and affect j too.

Iterative impulsive constraint solver

- Compute constraint impulses just as before for rigid bodies.
- Use Jacobian framework and info about constraint violation to find impulses in a pair-wise fashion.
- Use full variable set.
- Use a prestabilization for positions and poststabilization for velocities, i.e. find a desired “constraint collision impulse” together with other collision impulses. Next integrate velocities, and after that iterate over “contact impulses” so that also “constraint contact impulse” approaches the desired value.
- The method iterates over collisions, contacts as well as constraints in a unified manner.
- Corresponds to “projected Gauss-Seidel/SOR/SUR” (as usual, but published in many disguises). Novelty is how collisions and contacts are separated.


And lab notes.

In the book, this iterative method is only dealt with for contacts/collisions.
Iterative impulsive constraint solver

Net
Tank
Bridge

See web page:
http://graphics.stanford.edu/~rachellw/prestabilization.html

Constraints in the solver framework

- Problems:
  - Constraints may be degenerate, i.e. redundant, and there might exist several feasible solutions, which may cause solver problems.
  - Constraints may be contradictory, i.e. there is no feasible solution satisfying all constraints.
  - Errors, i.e. constraints violation because solver doesn’t find a good solution, or just an approximate solution.
  - Constraints may introduce strong coupling and loops in the system, that make it much harder to find a good solution and errors can introduce mal-functioning simulators (i.e. a broken vehicle drive train, or wheels on the loose...).
  - Constraints are often highly “visible” so joint violation is easily spotted by the user.
  - Lots of “interesting” details to deal with, i.e. joint types, signs, winding numbers, joint limits, joint motors, joint friction, relaxation parameters, conservative stabilization, breakable joints, attachable joints, error projection, error limits, etc.
  - Lots and lots of code for modeling and for solver optimization and pre-/post-processing.
Other joint types

- Ball-and-socket joint
- Hinge joint
- Slider joint
- Car-wheel joint (or hinge-2 joint)
- Universal
- Fixed
- Etc...
- Contact point (same, same but different...)
- Three-body joints

Joint limits

- Similar to unilateral constraints, i.e. contact points for rigid bodies, but now for dealing with joint limits
- Results in a linear complementarity constraint for each limit – or alternatively modeled as a "joint-limit collision/resting impulse"
Joint effectors and engines

- Control the relative motion of two bodies in a joint
- Add torque or force to the rotational or translational degrees of freedom of the joint
- Two modeling parameters (to start with)
  - Desired speed
  - Maximum force or torque allowed
- Use the error correction term to model the motor
  \[ J_{\text{motor}} u \geq h_{\text{motor}} \]
- And the constraint force/Lagrange parameters
  \[ F_{\text{motor}} = J_{\text{motor}} \lambda_{\text{motor}} \quad \text{with} \quad \lambda_{\text{motor}} > |\lambda_{\text{max}}| \]
- In general, e.g. for a car engine torque depends on r.p.m.
- Joint effectors can also be used as a way to model joint friction

Applications

- Machines
- Robotics
- Bio-mechanics
- Vehicle dynamics
- Complex objects, materials:
  - Cloth
  - Cables, wires, chains
  - Elastic materials