Differentiable Physics and Stable Modes for Tool Use and Manipulation Planning

Presenter: Steven Patrick

11/9/21
Task and Manipulation Planning (TAMP)

- The act of planning multiple tasks interacting with the environment to accomplish a goal

- Examples:
  - Cleaning a room
  - Putting furniture together
  - Preparing food
Differentiable Physics Engines

- If physics are fully differentiable, then “in-theory” you could start at the end position and back-propagate your way to the initial condition and known control inputs.
Differentiable Physics Engines

- Not enough computation to do this or time steps would be too coarse
- Multiple solutions result in the same final configuration
- Therefore, optimization approaches are promising since most rely heavily on gradient information
Optimization

$$\min_u J(u) := K(x(T)) + \int_0^T L(t, x(t), u(t)) dt$$

s.t. $x(t) \in X, \forall t \in [0, T]$

$x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t)$

$u(t) \in U, \forall t \in [0, T]$

$g_i(x(t)) \leq 0 \forall i \in [0..L], \forall t \in [0, T]$  

$h_i(x(t)) = 0 \forall i \in [0..L], \forall t \in [0, T]$
Optimization

\[
\min_u J(u) := K(x(T)) + \int_0^T L(t, x(t), u(t)) \, dt
\]

s. t. \( x(t) \in X, \forall t \in [0, T] \)
\( x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t) \)
\( u(t) \in U, \forall t \in [0, T] \)
\( g_i(x(t)) \leq 0 \forall i \in [0..L], \forall t \in [0, T] \)
\( h_i(x(t)) = 0 \forall i \in [0..L], \forall t \in [0, T] \)

Terminal Cost
Optimization

\[
\min_{u} J(u) := K(x(T)) + \int_{0}^{T} L(t, x(t), u(t)) \, dt
\]

s.t. \( x(t) \in X, \forall t \in [0, T] \)
\( x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t) \)
\( u(t) \in U, \forall t \in [0, T] \)
\( g_i(x(t)) \leq 0 \ \forall i \in [0..L], \forall t \in [0, T] \)
\( h_i(x(t)) = 0 \ \forall i \in [0..L], \forall t \in [0, T] \)

Running Cost
Optimization

$$\min_u J(u) := K(x(T)) + \int_0^T L(t, x(t), u(t)) \, dt$$

s.t. $x(t) \in X, \forall t \in [0, T]$ $x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t)$ $u(t) \in U, \forall t \in [0, T]$ $g_i(x(t)) \leq 0 \forall i \in [0..L], \forall t \in [0, T]$ $h_i(x(t)) = 0 \forall i \in [0..L], \forall t \in [0, T]$
Optimization

\[
\min_u J(u) := K(x(T)) + \int_0^T L(t, x(t), u(t)) \, dt
\]

s.t. \( x(t) \in X, \forall t \in [0, T] \)

\[
x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t)
\]

\[
u(t) \in U, \forall t \in [0, T]
\]

\[
g_i(x(t)) \leq 0 \quad \forall i \in [0..L], \forall t \in [0, T]
\]

\[
h_i(x(t)) = 0 \quad \forall i \in [0..L], \forall t \in [0, T]
\]

Initial Condition
Optimization

\[
\min_u J(u) := K(x(T)) + \int_0^T L(t, x(t), u(t)) dt
\]

s.t.

\[
x(t) \in X, \forall t \in [0, T]
\]
\[
x(0) = x_0,
\]
\[
x(t + \Delta t) = f(x, u, \Delta t)
\]
\[
u(t) \in U, \forall t \in [0, T]
\]
\[
g_i(x(t)) \leq 0 \quad \forall i \in [0..L], \forall t \in [0, T]
\]
\[
h_i(x(t)) = 0 \quad \forall i \in [0..L], \forall t \in [0, T]
\]

Dynamics Constraints
Optimization

\[
\min_u J(u) := K(x(T)) + \int_0^T L(t, x(t), u(t)) \, dt \\
\text{s.t. } x(t) \in X, \forall t \in [0, T] \\
x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t) \\
u(t) \in U, \forall t \in [0, T]
\]

Control Constraints

\[
g_i(x(t)) \leq 0 \quad \forall i \in [0..L], \forall t \in [0, T] \\
h_i(x(t)) = 0 \quad \forall i \in [0..L], \forall t \in [0, T]
\]
Optimization

\[ \min_{u} J(u) := K(x(T)) + \int_{0}^{T} L(t, x(t), u(t)) dt \]

s.t. \( x(t) \in X, \forall t \in [0, T] \)
\( x(0) = x_{0}, \quad x(t + \Delta t) = f(x, u, \Delta t) \)
\( u(t) \in U, \forall t \in [0, T] \)

Inequality Constraints

\[ g_i(x(t)) \leq 0 \forall i \in [0..L], \forall t \in [0, T] \]
\[ h_i(x(t)) = 0 \forall i \in [0..L], \forall t \in [0, T] \]
Optimization

\[
\min_{u} J(u) := K(x(T)) + \int_{0}^{T} L(t, x(t), u(t)) \, dt \\
\text{s. t. } x(t) \in X, \forall t \in [0, T] \\
x(0) = x_0, \quad x(t + \Delta t) = f(x, u, \Delta t) \\
u(t) \in U, \forall t \in [0, T] \\
g_i(x(t)) \leq 0 \forall i \in [0..L], \forall t \in [0, T] \\
h_i(x(t)) = 0 \forall i \in [0..L], \forall t \in [0, T]
\]

Equality Constraints
Mixed-Integer Programming

- Some or all the optimization variables must have integer values.
- Typically solved with **Branch and Bound** techniques using constraint relaxation.

1. Solve Non-integer problem
2. Branch result into two cases with most “non-integer” value set above and below the nearest integers of the solution
3. Continue branching until all values are integers or no feasible solution
Contact Invariant Optimization

- Signify contacting a surface with a binary variable
- Add a cost function that penalizes forces when contact is not active

\[
L'(x) = L(x) + \sum_i \frac{\|f_i\|^2}{c_i + \epsilon}
\]

- \(c_i\) is the contact variable, \(f_i\) is the force being applied, \(\epsilon \sim 10^{-6}\)

[*] Discovery of complex behaviors through contact-invariant optimization
Logic-Geometric Program

\[
\min_u J(u) := K(x(T)) + \int_0^T L(\ddot{x}(t)) \, dt
\]

s.t. \[ x(0) = x_0, h_{goal}(x(T)) = 0, g_{goal}(x(T)) \leq 0 \]
\[ \forall t \in [0, T]: h_{path}(\ddot{x}(t), s_{k(t)}) = 0 \]
\[ g_{path}(\ddot{x}(t), s_{k(t)}) \leq 0 \]

\[ \forall k \in [1, \ldots, K]: h_{switch}(\ddot{x}(t_k), a_k) = 0 \]
\[ g_{switch}(\ddot{x}(t_k), a_k) \leq 0 \]

\[ s_k = f(s_{k-1}, a_k) \]

\[ \ddot{x} = (x, \dot{x}, \ddot{x}) \]
\[ \dot{x} = (x, \dot{x}, \dot{x}_{new}) \]
\[ a_{1:K} \Rightarrow \text{skeleton} \]
Multi-Bound Tree Search

- 3 Optimization Steps
  - P1: Evaluates initial and final pose without considering costs
  - P2: Two time-step evaluations per action operator.
  - P3: Fine time-step evaluations over entire trajectory.
Path Constraint Predicates

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(touch X Y)</td>
<td>distance between X and Y equal 0</td>
</tr>
<tr>
<td>[impulse X Y]</td>
<td>ImpulseExchange eq &amp; skip smoothness constraints on X Y</td>
</tr>
<tr>
<td>(staFree X Y)</td>
<td>create stable (constrained to zero velocity) free (7D) joint from X to Y</td>
</tr>
<tr>
<td>(staOn X Y)</td>
<td>create stable 3D $xy\phi$ joint from X to Y</td>
</tr>
<tr>
<td>(dynFree X)</td>
<td>create dynamic (constrained to gravitational inertial motion) free joint from world to X</td>
</tr>
<tr>
<td>(dynOn X Y)</td>
<td>create dynamic 3D $xy\phi$ joint from X to Y</td>
</tr>
<tr>
<td>(inside X Y)</td>
<td>point X is inside object Y $\rightarrow$ inequalities</td>
</tr>
<tr>
<td>(above X Y)</td>
<td>Y supports X to not fall $\rightarrow$ inequalities</td>
</tr>
<tr>
<td>(push X Y Z)</td>
<td>(see text)</td>
</tr>
</tbody>
</table>

\[
\forall t \in [0, T]: h_{path}(\bar{x}(t), s_{k(t)}) = 0 \\
g_{path}(\bar{x}(t), s_{k(t)}) \leq 0
\]
Path Constraint Predicates

Resting and Stable Relations

| (staFree X Y) | create stable (constrained to zero velocity) free (7D) joint from X to Y |
| (staOn X Y)   | create stable 3D $x y \phi$ joint from X to Y |
Path Constraint Predicates

Inertial Motion

(dynFree X) | create dynamic (constrained to gravitational inertial motion) free joint from world to X
(dynOn X Y) | create dynamic 3D $xyz\phi$ joint from X to Y
Path Constraint Predicates

Impulse Exchange

\[ \text{ImpulseExchange eq & skip smoothness constraints on X Y} \]
## Path Constraint Predicates

### Geometric

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(touch X Y)</td>
<td>distance between X and Y equal 0</td>
</tr>
<tr>
<td>(inside X Y)</td>
<td>point X is inside object Y ( \rightarrow ) inequalities</td>
</tr>
<tr>
<td>(above X Y)</td>
<td>Y supports X to not fall ( \rightarrow ) inequalities</td>
</tr>
<tr>
<td>(push X Y Z)</td>
<td>(see text)</td>
</tr>
</tbody>
</table>
Action Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>grasp(X Y)</td>
<td>inside X Y (staFree X Y)</td>
</tr>
<tr>
<td>handover(X Y Z)</td>
<td>inside Z Y (staFree Z Y)</td>
</tr>
<tr>
<td>place(X Y Z)</td>
<td>above Y Z (staOn Z Y)</td>
</tr>
<tr>
<td>throw(X Y)</td>
<td>(dynFree Y)</td>
</tr>
<tr>
<td>hit(X Y)</td>
<td>[touch X Y] [impulse X Y] (dynFree Y)</td>
</tr>
<tr>
<td>hitSlide(X Y Z)</td>
<td>[touch X Y] [impulse X Y] (above Y Z) (dynOn Y Z)</td>
</tr>
<tr>
<td>hitSlideSit(X Y Z)</td>
<td>“hitSlide(X Y Z)” “place(X Z)”</td>
</tr>
<tr>
<td>push(X, Y, Z)</td>
<td>komo(push X Y Z)</td>
</tr>
</tbody>
</table>

\[
\forall k \in [1, \ldots, K]: h_{\text{switch}}(\hat{x}(t_k), a_k) = 0 \\
g_{\text{switch}}(\hat{x}(t_k), a_k) \leq 0
\]
Results
### Results

<table>
<thead>
<tr>
<th>problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree size</td>
<td>12916</td>
<td>34564</td>
<td>7312</td>
<td>12242</td>
<td>12242</td>
<td>3386</td>
</tr>
<tr>
<td>branching</td>
<td>10.66</td>
<td>13.63</td>
<td>9.25</td>
<td>10.52</td>
<td>10.52</td>
<td>7.63</td>
</tr>
</tbody>
</table>
Results
Results (Computation Time)
Critiques

- Not real-time. Takes a minute or two to generate a trajectory

- Have to hard code knowledge base

- Finds the optimal solution which is usually the least robust
Conclusions

- This is meant as a new method to explore rather than a total replacement of current manipulation planners

- Has a lot of potential if algorithms are created to reduce computation times and add robustness to solutions
References

[1] Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning
[4] Integrated Task and Motion Planning
[5] Discovery of complex behaviors through contact-invariant optimization
Thank You!

QUESTIONS?