Small Aerial Vehicle (SAV) Trajectory Planning in 3D
A Hybrid Randomized/Nonlinear Programming Technique

Patrick Bouffard  Steven Waslander

University of California, Berkeley
University of Waterloo

Oct. 11, 2009
Outline

1. Introduction
2. Planning Algorithm
3. Results
Outline

1. Introduction
2. Planning Algorithm
3. Results
Introduction

Planning Algorithm

Results

[1] Aeryon Systems
[3] Raytheon
Path Planning

- Various techniques proposed for this problem, among them:

**Randomized methods**
- quickly explore large parameter space
- circuitous paths from randomized sampling of control inputs

**Optimization methods**
- locally/globally optimal control inputs
- more computationally intensive as problem size increases

Complementarity suggests use of a hybrid algorithm
Path Planning

- Various techniques proposed for this problem, among them:

<table>
<thead>
<tr>
<th>Randomized methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>quickly explore large parameter space</td>
</tr>
<tr>
<td>circuitous paths from randomized sampling of control inputs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>locally/globally optimal control inputs</td>
</tr>
<tr>
<td>more computationally intensive as problem size increases</td>
</tr>
</tbody>
</table>

Complementarity suggests use of a hybrid algorithm
Path Planning

Various techniques proposed for this problem, among them:

Randomized methods
- quickly explore large parameter space
- circuitous paths from randomized sampling of control inputs

Optimization methods
- locally/globally optimal control inputs
- more computationally intensive as problem size increases

Complementarity suggests use of a hybrid algorithm
Path Planning

Various techniques proposed for this problem, among them:

Randomized methods
- quickly explore large parameter space
- circuitous paths from randomized sampling of control inputs

Optimization methods
- locally/globally optimal control inputs
- more computationally intensive as problem size increases

Complementarity suggests use of a hybrid algorithm
Prior Work

<table>
<thead>
<tr>
<th>Randomized Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic Roadmaps (PRM) widely used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed Integer-Linear Programming (MILP)</td>
</tr>
<tr>
<td>Nonlinear Programming (NLP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapidly-Exploring Random Tree (RRT), NLP, RRT +NLP [Karatas &amp; Bullo 2001]</td>
</tr>
<tr>
<td>combined algorithm better than either approach alone</td>
</tr>
</tbody>
</table>
### Prior Work

#### Randomized Methods
- Probabilistic Roadmaps (PRM) widely used

#### Optimization Methods
- Mixed Integer-Linear Programming (MILP)
- Nonlinear Programming (NLP)

**Hybrid**
- Rapidly-Exploring Random Tree (RRT), NLP, RRT + NLP [Karatas & Bullo 2001]
  - combined algorithm better than either approach alone
Prior Work

Randomized Methods
- Probabilistic Roadmaps (PRM) widely used

Optimization Methods
- Mixed Integer-Linear Programming (MILP)
- Nonlinear Programming (NLP)

Hybrid
Rapidly-Exploring Random Tree (RRT), NLP, RRT + NLP [Karatas & Bullo 2001]
- combined algorithm better than either approach alone
Quadrotor Helicopters

STARMAC - Stanford Testbed of Rotorcraft for Multi Agent Control

Aeryon Systems

MicroDrones GmbH
Outline

1 Introduction

2 Planning Algorithm

3 Results
Vehicle Model

State

\[ y = (x, y, z, v_x, v_y, v_z, \theta, \phi, \psi, T) \in \mathbb{R}^{10} \]

Control

\[ u = (u_\theta, u_\phi, u_T) \in \mathbb{R}^{3} \]

Dynamics

- Point-mass translation
- 1st order integrator attitude
- 1st order integrator thrust
- Constants based on actual quadrotor platform (STARMAC)
Environment Model

- Surfaces that vehicle’s trajectory must not intersect
- Set $E = \{\Delta_k \mid k \in 1, \ldots, N_{tri}\}$ of triangles
- Can represent objects of any shape; convex, nonconvex
Problem Statement

Given
- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{dest}$
- Final time $t_f$, number of timesteps $N$

Want
$u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
Problem Statement

Given

- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{\text{dest}}$
- Final time $t_f$, number of timesteps $N$

Want

$u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
Problem Statement

Given

- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{dest}$
- Final time $t_f$, number of timesteps $N$

Want

$u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
Problem Statement

Given
- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{\text{dest}}$
- Final time $t_f$, number of timesteps $N$

Want
- $u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
Problem Statement

Given
- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{dest}$
- Final time $t_f$, number of timesteps $N$

Want
$u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
Problem Statement

Given

- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{dest}$
- Final time $t_f$, number of timesteps $N$

Want

$u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
Problem Statement

Given

- Dynamics $\dot{y}(t) = f(y, u, t)$
- Obstacles $E$
- Start and destination states $y_0$ and $y_{dest}$
- Final time $t_f$, number of timesteps $N$

Want

$u_1, \ldots, u_N$ so that vehicle is driven from $y_0$ to $y_f$
**Algorithm Overview**

**Planning Algorithm**

1. **Construct PRM**
   - Captures connectivity of free space
   - Doors, windows, hallways, etc.

2. **Generate pre-path**
   - Gives basis for initial guess for NLP solver

3. **Solve NLP**

**Output**

Sequence of control inputs guiding vehicle from start to end
Roadmap Construction

Algorithm
- Select point at random
- Connect to existing *visible* points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Construction

Algorithm
- Select point at random
- Connect to existing visible points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Construction

Algorithm
- Select point at random
- Connect to existing visible points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Construction

Algorithm

- Select point at random
- Connect to existing *visible* points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Construction

Algorithm

- Select point at random
- Connect to existing *visible* points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Construction

**Algorithm**

- Select point at random
- Connect to existing *visible* points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Construction

Algorithm
- Select point at random
- Connect to existing visible points
- Repeat
- Extensions (e.g., Gaussian Sampling, Bridge Sampling) exist to improve coverage
Roadmap Query

Algorithm
- Define $y_0$ and $y_{dest}$
- Attach to roadmap
- Discrete planner (Dijkstra, A*, etc) finds shortest path
- Refinement - line of sight heuristic
Roadmap Query

**Algorithm**

- Define $y_0$ and $y_{dest}$
- Attach to roadmap
- Discrete planner (Dijkstra, A*, etc) finds shortest path
- Refinement - line of sight heuristic
Roadmap Query

Algorithm

- Define $y_0$ and $y_{dest}$
- Attach to roadmap
- Discrete planner (Dijkstra, A*, etc) finds shortest path
- Refinement - line of sight heuristic
Roadmap Query

Algorithm

- Define $y_0$ and $y_{dest}$
- Attach to roadmap
- Discrete planner (Dijkstra, A*, etc) finds shortest path
- Refinement - line of sight heuristic
Roadmap Query

Algorithm
- Define $y_0$ and $y_{dest}$
- Attach to roadmap
- Discrete planner (Dijkstra, A*, etc) finds shortest path
- Refinement - line of sight heuristic
Roadmap Query

**Algorithm**

- Define $y_0$ and $y_{dest}$
- Attach to roadmap
- Discrete planner (Dijkstra, A*, etc) finds shortest path
- Refinement - line of sight heuristic
Nonlinear Program

Recap
- Pre-path now defined
- Not dynamically feasible
- Not optimal
- Use NLP to resolve this
Nonlinear Program

Recap
- Pre-path now defined
- Not dynamically feasible
- Not optimal
- Use NLP to resolve this
Nonlinear Program

NLP Formulation

minimize \( J(x) \)
subject to \( c(x) \leq 0 \)
\( h(x) = 0 \)
\( x_{min} \leq x \leq x_{max} \)

Variables of optimization

\( x = [u_1, \ldots, u_N, y_1, \ldots, y_{N-1}] \)
\( x \) has dimension \( N_{opt} = 13N - 10 \)
Cost Function

\[ J(x) = \frac{1}{N} \sum_{i=1}^{N} (u_i - u_h)^T W (u_i - u_h) \]

- Quadratic sum of deviation of control from hover condition \( u_h \) at each step
- \( W \) is a weighting matrix
Inequality constraints $c(x) \leq 0$

$c(x) = (c_1(x), \ldots, c_N(x))^T \leq 0$

where $c_i(x) = d_{\text{min}} - \frac{\text{dist}(L_{ab,i})}{L_{\text{ref},i}}$

Keep trajectory segments at least $d_{\text{min}}$ from all of $E$
Distance Function

- Clearance $> 0$: use clearance value $d_\alpha$
- Clearance $\leq 0$: use heuristic $-d_\beta$
  - (negative of) minimum distance from segment $k$ midpoint to pre-path
- Pathological case
Distance Function

- Clearance > 0: use clearance value $d_\alpha$
- Clearance ≤ 0: use heuristic $-d_\beta$
  - (negative of) minimum distance from segment $k$ midpoint to pre-path

Pathological case
Distance Function

- Clearance $> 0$: use clearance value $d_\alpha$
- Clearance $\leq 0$: use heuristic $-d_\beta$
  - (negative of) minimum distance from segment $k$ midpoint to pre-path
- Pathological case
Equality constraints $h(x) = 0$

$$h_i(x) = y_{i-1} - y_i + \int_{t=t_{i-1}}^{t_i} f(y, u_i) dt = 0,$$

$\forall i \in \{1, \ldots, N\}$

Enforces boundary conditions between timesteps

Simple Bounds

$$y_{min} \leq y_i \leq y_{max}, u_{min} \leq u_i \leq u_{max} \forall i \in \{1, \ldots, N\}$$
Outline

1. Introduction
2. Planning Algorithm
3. Results
Implementation

- Algorithm implemented in software
- Ran simulations for some test cases

Free/Open Source Components

- PQP - Proximity Query Package
- IPOPT (Interior Point OPTimizer) - NLP Solver
  - Using OpenOpt interface
- Panda3D - visualization
- Python - main program
Implementation

- Algorithm implemented in software
- Ran simulations for some test cases

Free/Open Source Components

- PQP - Proximity Query Package
- IPOPT (Interior Point OPTimizer) - NLP Solver
  - Using OpenOpt interface
- Panda3D - visualization
- Python - main program
Scenarios

1. Simple
2. Maze
3. Twostorey
4. Cave
Scenarios

1. Simple
2. Maze
3. Twostorey
4. Cave
Scenarios

1. Simple
2. Maze
3. Twostorey
4. Cave
Scenarios

1. Simple
2. Maze
3. Twostorey
4. Cave
## Test Scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>$N_{tri}$</th>
<th>PRM Nodes</th>
<th>$N$</th>
<th>$t_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple</td>
<td>256</td>
<td>200</td>
<td>10</td>
<td>40 s</td>
</tr>
<tr>
<td>maze</td>
<td>1238</td>
<td>500</td>
<td>18</td>
<td>76 s</td>
</tr>
<tr>
<td>twostorey</td>
<td>2301</td>
<td>500</td>
<td>22</td>
<td>88 s</td>
</tr>
<tr>
<td>cave</td>
<td>65536</td>
<td>250</td>
<td>10</td>
<td>40 s</td>
</tr>
</tbody>
</table>
Simulations performed on commodity laptop (2.80 GHz, Core2 Duo)
Results
Summary

Hybrid Planning Algorithm

- PRM rapidly explores free space to discover connectivity
- NLP generates locally optimal trajectories
- Combined strengths of both methods

Simulation Results

- Successful plans for realistic scenarios
- On commodity PC, planning time less than flight time
- Hybrid algorithm shows promise
Summary

Hybrid Planning Algorithm

- PRM rapidly explores free space to discover connectivity
- NLP generates locally optimal trajectories
- Combined strengths of both methods

Simulation Results

- Successful plans for realistic scenarios
- On commodity PC, planning time less than flight time
- Hybrid algorithm shows promise
Future Work

- Improved $\text{dist}_\beta$ function to avoid/minimize pathological cases
- Better initial guess (all states)
- Variations: minimum-time, maximum-distance
- Implementation onboard hardware platform in receding-horizon framework
Thank You

- Prof. Steve Waslander
- Prof. Claire Tomlin