RRT*-AR
Sampling-Based Alternate Routes Planning
with applications to
Autonomous Emergency Landing of a Helicopter

Sanjiban Choudhury, Sebastian Scherer and Sanjiv Singh
Motivation
Motivation
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary

- What do we wish to solve?
- Key challenges
- State of the Art
Problem Statement

Safely land a helicopter autonomously after engine failure

Specifically:
1. Using prior information about the terrain and ground below and a limited range laser sensor, guide the helicopter around mountains, buildings, trees and safely touchdown
2. Allow the pilot on board to choose the route
Detailed Problem Statement

Mission Specification

<table>
<thead>
<tr>
<th>Objective</th>
<th>Safely land a helicopter after engine failure while avoiding obstacles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Available</td>
<td></td>
</tr>
<tr>
<td>• Digital Elevation Map (DEM) of terrain</td>
<td></td>
</tr>
<tr>
<td>• Landing Zone (LZ) feasibility map</td>
<td></td>
</tr>
<tr>
<td>• Pilot can interact with HMI</td>
<td></td>
</tr>
</tbody>
</table>

Vehicle Specification

<table>
<thead>
<tr>
<th>Helicopter</th>
<th>UH-60 BlackHawk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward Speed</td>
<td>57 m/s</td>
</tr>
<tr>
<td>Descent Speed</td>
<td>10 m/s</td>
</tr>
<tr>
<td>Laser Range</td>
<td>700 m</td>
</tr>
</tbody>
</table>
Challenges: Hard Real Time Constraints

- Distance travelled = 2500 m
- Height descended = 600 m
- Total time = 60s
Challenges: Hard Real Time Constraints

Hard real-time constraints
Reachable landing zone exponentially decreases

Distance travelled = 2500 m
Height descended = 600 m
Total time = 60s
Challenges : Hard Real Time Constraints

Hard real-time constraints
- Reachable landing zone exponentially decreases

Limited Perception
- Reaction time less than 1 second

Distance travelled = 2500 m
Height descended = 600 m
Total time = 60s
Challenges : Hard Real Time Constraints

Hard real-time constraints
Reachable landing zone exponentially decreases

Limited Perception
Reaction time less than 1 second

Pilot Experience
Human can confirm safe path

Distance travelled = 2500 m
Height descended = 600 m
Total time = 60s
Lack of Suitable State of the Art

VP-400
The backup EFIS that flies your plane down in an emergency.

- Engine Failure?
- EFIS Failure?
- VFR into IMC?
- Pilot Incapacitation?

http://verticalpower.com/
Lack of Suitable State of the Art

VP-400
The backup EFIS that flies your plane down in an emergency.

Engine Failure?  
EFIS Failure?  
VFR into IMC?  
Pilot Incapacitation?

- Not for helicopter
- No terrain avoidance
- Only lands on airports
- No perception

http://verticalpower.com/
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary

• Overview of related work
• Brief look at some approaches
• Effectiveness of approaches
## Related Work

<table>
<thead>
<tr>
<th>Autorotation Trajectories</th>
<th>Flare</th>
<th>Full planning and control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal trajectories minimizing speed (Johnson et al., 1971, Lee et al. 1985)</td>
<td>Backward Reachability (Tierney, 2011)</td>
<td>Generation of trajectory and control for every phase (Yomchinda et al., 2012)</td>
</tr>
<tr>
<td>Optimal control for autorotation (Bachelder et al. 2003, Aponso et al. 2005)</td>
<td>Model Predictive Control (Dalamagkidis et al., 2010)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Imitation Learning (Abbeel et al., 2009)</td>
<td></td>
</tr>
</tbody>
</table>
Category 1: Autorotation Trajectories


Objective

Compute optimal autorotation trajectories and control, with a focus on failure at low altitudes

Approach

• Problem framed as non-linear optimal control problem - minimize velocity at touchdown
• Constrain actuators and state variables
• Sequential Gradient Restoration Technique used to solve for trajectories

Contribution

Determined the capabilities of a helicopter in autorotation and optimal strategy to land
Category 2: Flare


Objective

Compute backward reachable sets from safe landing states to determine safe flare initiation set

Approach

- Frame an optimization problem to minimize distance to safe landing state
- Constrain actuator and state variable
- Solve using interior point method
- List of starting points that reach landing state comprises safe flare initiation set

Contribution

Determined a set of safe flare start states and corresponding flare trajectories
Category 3: Full Planning and Control


Objective

To develop a complete autonomous autorotation system for helicopter

Approach

• Decouple the problem into Entry, Descent and Flare
• Descent is a Dubin’s curve
• Optimization problem framed to solve for control parameters using FMINCON

Contribution

Solves every stage of descent problem in terms of trajectory and control
### Comparison: Methods have Practical Limitations

<table>
<thead>
<tr>
<th>Method</th>
<th>Plan to a specified LZ</th>
<th>Plan in Real-time</th>
<th>Avoid Known Obstacles</th>
<th>Avoid Unknown Obstacles</th>
<th>Select a feasible LZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal Autorotation (Lee)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Flare trajectory and control (Tierney et al.)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Full planning and control (Yomchinda et al.)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Our approach</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Contribution

• Plan **alternate routes** as emergency backups / interesting options
• RRT*-AR algorithm **rapidly plans asymptotically optimal routes**
• Emergency landing **system safely lands vehicle**
Contribution

- Plan alternate routes as emergency backups / interesting options
- RRT*-AR algorithm rapidly plans asymptotically optimal routes
- Emergency landing system safely lands vehicle

<table>
<thead>
<tr>
<th>Planning time</th>
<th>1.0 s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost of path</td>
<td>Within 50% of global optimal</td>
</tr>
<tr>
<td>Reaction time to unexpected obstacles</td>
<td>0.1 s</td>
</tr>
<tr>
<td>Options provided to pilot</td>
<td>6 different options</td>
</tr>
<tr>
<td>LZ characteristic</td>
<td>Realistic LZ from digital elevation</td>
</tr>
<tr>
<td>Type of obstacles</td>
<td>Mountains, Hills, Trees, Rocks, Unmapped objects</td>
</tr>
<tr>
<td>Vehicle Speed</td>
<td>57 m/s</td>
</tr>
<tr>
<td>Sensor Range</td>
<td>700 m</td>
</tr>
</tbody>
</table>

Successfully land a helicopter in real-time in a high fidelity simulation for over 4500 trials
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary

- Formal planning problem
- Decoupling based approach
- Simplified planning problem
- Desired characteristics
Planning Problem

State Space

\[ x = \{ x^E, y^E, z^E, \dot{x}^E, \dot{y}^E, \dot{z}^E, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, \Omega \} \]

E subscript denotes North East Down frame
Planning Problem

State Space

\[ x = \{ x^E, y^E, z^E, \dot{x}^E, \dot{y}^E, \dot{z}^E, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, \Omega \} \]

Control Input

\[ u = \{ \delta_{col}, \delta_{lon}, \delta_{lat}, \delta_{pedal} \} \]
Planning Problem

State Space
\[ x = \{ x^E, y^E, z^E, \dot{x}^E, \dot{y}^E, \dot{z}^E, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, \Omega \} \]

Control Input
\[ u = \{ \delta_{\text{col}}, \delta_{\text{lon}}, \delta_{\text{lat}}, \delta_{\text{pedal}} \} \]

Optimization
\[ \text{minimize : } J = \int_0^{t_f} c(x(t)) dt + c(x(t_f)) \]
\[ \text{constraints : } \dot{x} = f(x(t), u, t) \]
\[ x(0) = x_0 \]
\[ x(t_f) \in X_{\text{LZ}} \]
\[ g \leq 0, J < \infty \]

E subscript denotes North East Down frame
Risk function
Autorotation dynamics
Engine Failure Point
Continuous Landing Zone
Actuator constraints

Landing zone feasibility is represented as a continuous mapping from X,Y to a metric. Risk function incorporates this metric.
Approach: Decouple Problem into Stages

Solve the entire constrained optimization problem?

✗ Planning problem difficulty differs from glide to flare
Approach: Decouple Problem into Stages

Solve the entire constrained optimization problem?

✗ Planning problem difficulty differs from glide to flare

Break it into separate planning problems?

✓ Decouple to Entry Glide, Glide, Entry Flare, Flare
Approach: Decouple Problem into Stages

Solve the entire constrained optimization problem?

× Planning problem difficulty differs from glide to flare

Break it into separate planning problems?

✔ Decouple to Entry Glide, Glide, Entry Flare, Flare

Assumptions

1. The engine failure occurs at a high enough altitude for decoupling

2. The vehicle has a minimum airspeed when the engine fails to attain glide

3. The vehicle isn’t immediately surrounded by obstacles till it is in glide
Emergency Landing System

Cruise Mode → Entry Glide → Glide → Entry Flare → Flare
Glide Planning Problem

Plan a path $P_c$ in command space such that a controller can track it

$$\begin{align*}
\text{minimize :} & \quad J = \int_0^{t_f} c(x(t))dt + c(x(t_f)) \\
\text{constraints :} & \quad \dot{x} = f(x(t), u(x, t, P_c, t)) \\
& \quad x(0) = x_0 \\
& \quad x(t_f) \in X_{end} \\
& \quad g \cdot \leq 0, J < \infty
\end{align*}$$
Glide Planning Problem

Plan a path \( P_c \) in command space such that a controller can track it.

\[
\begin{align*}
\text{minimize:} & \quad J = \int_0^{t_f} c(x(t)) dt + c(x(t_f)) \\
\text{constraints:} & \quad x = f(x(t), u(x, t, P_c), t) \\
& \quad x(0) = x_0 \\
& \quad x(t_f) \in X_{end} \cdot \\
& \quad g \cdot \leq 0, \ J \ < \ \infty
\end{align*}
\]

Risk function

\[ J_{obs} \quad \text{Cost decays with squared distance from obstacles} \]

\[ J_{curv} \quad \text{Cost increases as curvature reaches limits} \]

\[ J_{rpm} \quad \text{Cost penalizes deviation of rotor speed from 100\% rpm} \]

\[ J_{lz} \quad \text{Terminal cost depending on landing zone feasibility} \]
## Desired Planner Characteristics

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan quickly over large distances</td>
<td>Sampling based planner with optimality guarantees</td>
</tr>
<tr>
<td>Small reaction time to unknown obstacles</td>
<td>Plan alternate routes</td>
</tr>
<tr>
<td>Involve pilot in the decision process</td>
<td>Plan alternate routes</td>
</tr>
</tbody>
</table>
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary

- Relevance of alternate routes
- Desired properties
Why Alternate Routes (AR)?

Interesting set of path options for pilot

- Interesting, different options
- Spatially diverse
- Low Cost Options
- Bounded sub-optimality
Why Alternate Routes (AR)?

Interesting set of path options for pilot

- Interesting, different options
- Spatially diverse
- Low Cost Options
- Bounded sub-optimality

A set of emergency backup plans

- Backups should be different from current
- Spatially diverse
- Backups should be good candidates
- Bounded sub-optimality
Because we have a very detailed prior of the environment, our approach does not exactly address the same problem we are proposing. maximizes the expected survivability of a path considering a constraint. 

Green Kelly has a similar metric for sampling paths where the paths were selected according to "survivability," i.e., selecting a set that maximizes "dispersion," i.e., iteratively add paths that are most diversified. 

A backup set generator is required computations of this approach. Hence, the methods do not address the same problem we are proposing. We propose a way of sampling paths to maximize survivability by selecting a manageable subset of paths. 

The parent path set is a set of allowable landing zones, which is particularly unsuitable for our problem. 

We want to frame an optimization problem that returns as many alternate routes (AR) as possible. The alternate routes must be locally optimal and have costs close to the optimal cost and are spatially distinct. Abraham et al. (2010) propose a way of sampling paths to maximize survivability by selecting a manageable subset of paths. 

We define multiple cost functions to ensure all the solutions in the set have costs close to the optimal cost and are spatially distinct. 

The cost function can be created that penalizes proximity to obstacles. However, these methods only reason about collision avoidance and are agnostic of the desire to minimize risk. 

To compute AR, we use a discrete graph to solve for alternate routes in road networks. 

One such way to do it is to define multiple cost functions. The action is the state of the vehicle and is the set of environmental and hard constraints. 

This process is iterated for the second member in the set. The solution to this problem becomes the cost function representing the action. 

When the rubble is detected, the helicopter hasn't yet detected the rubble at the LZ, but already has an alternative plan pre-computed. (a) Helicopter instantaneously switches plan when the rubble is detected. (b) Helicopter Fig. 4. Having a backup plan allows the helicopter to instantly switch when the current path is infeasible. (a) Helicopter

\[
\begin{align*}
\text{find} : & \quad \sigma_i = (x(t), u(t)) \quad \forall i = 1 \cdots m \\
& \quad \sigma_i \in \Sigma^* \\
& \quad \text{AR DEFINITION} \\
& \quad \text{(Abraham et al., 2010)}
\end{align*}
\]
Desired properties of Alternate Routes

\[ \text{find: } \sigma_i = (x(t), u(t)) \quad \forall i = 1 \cdots m \quad \text{alternate routes (AR)} \]
\[ \sigma_i \in \Sigma^* \quad \text{AR DEFINITION} \]

(Abraham et al., 2010)

I. Limited sharing of path swath (Spatially Diverse)

\[ S(\sigma_i) \cap \{S(\sigma_0) \cup \cdots \cup S(\sigma_{i-1})\} \leq \gamma \|S(\sigma_i)\| \]
Desired properties of Alternate Routes

\[
\text{find : } \sigma_i = (x(t), u(t)) \quad \forall i = 1 \cdots m \quad \text{alternate routes (AR)} \\
\sigma_i \in \Sigma^* \quad \text{AR DEFINITION}\]

(Abraham et al., 2010)

1. Limited sharing of path swath (Spatially Diverse)

\[
S(\sigma_i) \cap \{S(\sigma_0) \cup \cdots \cup S(\sigma_{i-1})\} \leq \gamma \|S(\sigma_i)\|
\]

2. Bounded Sub-optimality

\[
J(\sigma_i) \leq (1 + \varepsilon)J(\sigma^*)
\]
Desired properties of Alternate Routes

\[
\text{find : } \quad \sigma_i = (x(t), u(t)) \quad \forall i = 1 \cdots m \quad \text{alternate routes (AR)} \nonumber
\]

\[
\sigma_i \in \Sigma^* \quad \text{AR DEFINITION} \nonumber
\]

(Abraham et al., 2010)

1. Limited sharing of path swath (Spatially Diverse)

\[
S(\sigma_i) \cap \{S(\sigma_0) \cup \cdots \cup S(\sigma_{i-1})\} \leq \gamma \|S(\sigma_i)\| \nonumber
\]

2. Bounded Sub-optimality

\[
J(\sigma_i) \leq (1 + \varepsilon)J(\sigma^*) \nonumber
\]

3. Locally Optimal
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary

- Relevance of RRT*
- Need for RRT*-AR
- Properties of RRT*-AR
- Guarantees of RRT*-AR
RRT*: Planning quickly over large distances

RRT* (Karaman and Frazolli, 2010) combines the speed of sampling based approaches with optimality guarantees
RRT*: Planning quickly over large distances

RRT* (Karaman and Frazolli, 2010) combines the speed of sampling based approaches with optimality guarantees.

At the $i^{th}$ iteration,

- end

start
RRT*: Planning quickly over large distances

RRT* (Karaman and Frazolli, 2010) combines the speed of sampling based approaches with optimality guarantees.

At the $i^{th}$ iteration,

- end

SAMPLE
RRT*: Planning quickly over large distances

RRT* (Karaman and Frazolli, 2010) combines the speed of sampling based approaches with optimality guarantees.

At the $i^{th}$ iteration,

- SAMPLE
- FIND BEST PARENT
- end

start
RRT*: Planning quickly over large distances

RRT* (Karaman and Frazolli, 2010) combines the speed of sampling based approaches with optimality guarantees

At the $i^{th}$ iteration,

- SAMPLE
- FIND BEST PARENT
- REWIRE TO CHILDREN
- end

start
RRT*: Planning quickly over large distances

RRT* (Karaman and Frazolli, 2010) combines the speed of sampling based approaches with optimality guarantees.

At the $i^{th}$ iteration,

- SAMPLE
- FIND BEST PARENT
- REWIRE TO CHILDREN

Run RRT* and select paths that satisfy AR
RRT*-AR: Encoding diversity and speed in RRT*

<table>
<thead>
<tr>
<th>Desired Attribute</th>
<th>RRT* property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produce diverse paths</td>
<td>Exploits cost function - dense tree with less diversity</td>
</tr>
<tr>
<td>Plan rapidly</td>
<td>Expensive procedure to determine parent / child</td>
</tr>
<tr>
<td>Re-plan efficiently</td>
<td>NA if vehicle is no longer on search tree</td>
</tr>
</tbody>
</table>
### RRT*-AR: Encoding diversity and speed in RRT*

<table>
<thead>
<tr>
<th>Desired Attribute</th>
<th>RRT* property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produce diverse paths</td>
<td>Exploits cost function - dense tree with less diversity</td>
</tr>
<tr>
<td>Plan rapidly</td>
<td>Expensive procedure to determine parent / child</td>
</tr>
<tr>
<td>Re-plan efficiently</td>
<td>NA if vehicle is no longer on search tree</td>
</tr>
</tbody>
</table>

**RRT*-AR (Accelerated Alternate Routes with Re-planning)**
- incorporates bounded diversity
- implements cost approximation to gain speed
- efficiently re-plans across cycles
Diversity: Relaxation of a greedy parent

**GOAL**

**START**

**RRT*** EXPLOITS

**RRT*-AR EXPLORES & EXPLOITS**

**Greedy Parent**

**d_{eq}**

**RETAINS ASYMPTOTIC OPTIMALITY**

**RRT***

**RRT*-AR**
Diversity: Relaxation of a greedy parent

RRT*

Greedy Parent

\( d_{eq} \)

\( d_{eq} \)

RRT*-AR

RETAINS ASYMPTOTIC OPTIMALITY

Optimal Cost: 1450

AR Cost: 1640

Optimal Cost: 1450

RRT* EXPLOITS

RRT*-AR EXPLORES & EXPLOITS
Diversity: Relaxation of a greedy parent

RRT* EXPLOITS

RRT*-AR EXPLORES & EXPLOITS

Optimal Cost: 1450

AR Cost: 1510

1640
Rapid Search: Leveraging cost approximation

RRT* searches slowly all the time.

RRT*-AR searches fast initially, slows down to ensure optimality.

RRT*-AR retains asymptotic optimality.
Rapid Search: Leveraging cost approximation

RRT*: RRT* searches slowly all the time.

RRT*-AR: RRT*-AR searches fast initially, slows down to ensure optimality.

Optimal Cost:
- RRT*: 1540 (vertices: 0, time: 0.000)
- RRT*-AR: 1480 (vertices: 0, time: 0.000)
Rapid Search: Leveraging cost approximation

RRT* searches slowly all the time.

RRT*-AR searches fast initially and slows down to ensure optimality.

Optimal Cost: 1540

Optimal Cost: 1480
Reuse Search Tree: Efficiently latch to old tree

Tree at time: $T$

Tree at time: $T+1$

Pruned Tree

Rewired Tree
Reuse Search Tree: Efficiently latch to old tree

Latching efficiently to old tree
- Find near neighbours around current position
- Sort them by depth
- Rewire to old tree and prune "dead" branches
Retention of Asymptotic Optimality Guarantees

1. Exploration turns to exploitation, as vertices increase

2. Approximation converges to exact as trajectories get smaller
RetentionPolicy  Asymptotic Optimality Guarantees

1. Exploration turns to exploitation, as vertices increase

For asymptotic guarantee

\[ \gamma_{RRT^* AR} \geq \frac{2}{(1 - \rho)} \left( (1 + \frac{1}{d}) \left( \frac{\mu(X_{\text{free}})}{\zeta d} \right) \right)^{\frac{1}{d}}, \quad \rho < 1 \]

fraction of near radius that is \( d_{eq} \)

2. Approximation converges to exact as trajectories get smaller
Retention of Asymptotic Optimality Guarantees

Choudhury et al., 2012

1. Exploration turns to exploitation, as vertices increase

For asymptotic guarantee

\[ \gamma_{\text{RRT}\ast AR} \geq \frac{2}{(1 - \rho)}((1 + \frac{1}{d})(\frac{\mu(X_{\text{free}})}{\zeta_d}))^{\frac{1}{d}}, \quad \rho < 1 \]

fraction of near radius that is \( d_{eq} \)

2. Approximation converges to exact as trajectories get smaller

Let \( p(n) \) be the mis-ranking error due to cost approximation. Then

\[ p(n) < n^{-(1+1/d)}, \quad \forall \ n \geq M \]

i.e mis-ranking error must monotonically decrease to 0
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
   - State machine of the system
   - Workflow of the system
7. Results
8. Summary
Emergency Landing System

Cruise Mode  Entry Glide  Glide  Entry Flare  Flare

Precomputed Maneuver  RRT*-AR  Stabilizing Controller  Trajectory Library
Workflow of the System

1. Engine failure occurs
2. Alternate routes planned
3. Pilot chooses path
4. Helicopter lands
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary

- Alternate routes
- Rapid planning
- System demonstration
### Results 1: RRT*-AR produces more routes

<table>
<thead>
<tr>
<th>Terrain</th>
<th>Colorado</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Point</td>
<td>900 m AGL</td>
</tr>
<tr>
<td>Iteration Time</td>
<td>Till convergence</td>
</tr>
<tr>
<td>#Routes</td>
<td>6</td>
</tr>
<tr>
<td>Gamma</td>
<td>2.7</td>
</tr>
<tr>
<td>Path Swath</td>
<td>500 m</td>
</tr>
<tr>
<td>Epsilon</td>
<td>4</td>
</tr>
<tr>
<td>Rho</td>
<td>0.2</td>
</tr>
</tbody>
</table>

RRT* computes **2 routes**

RRT*-AR computes **6 routes**
Results I: RRT*-AR produces more routes

RRT* computes 2 routes  
RRT*-AR computes 6 routes

Terrain: Colorado
Initial Point: 900 m AGL
Iteration Time: Till convergence
#Routes: 6
Gamma: 2.7
Path Swath: 500 m
Epsilon: 4
Rho: 0.2

Success Frequency (%) of computing AR over 791 trials

RRT*-AR on average computes 2.82 more routes
Results II: RRT*-AR computes routes rapidly

RRT*-AR optimizes rapidly indicating fast exploration because of cost approximation.

<table>
<thead>
<tr>
<th>Terrain</th>
<th>Colorado</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Point</td>
<td>900 m AGL</td>
</tr>
<tr>
<td>Iteration Time</td>
<td>Till convergence</td>
</tr>
<tr>
<td>#Routes</td>
<td>6</td>
</tr>
<tr>
<td>Gamma</td>
<td>2.7</td>
</tr>
<tr>
<td>Path Swath</td>
<td>500 m</td>
</tr>
<tr>
<td>Epsilon</td>
<td>4</td>
</tr>
<tr>
<td>Rho</td>
<td>0.2</td>
</tr>
</tbody>
</table>

![Graph](image_url)

Cost v/s Iteration Time over 3381 trials

**Normalized Cost**

- **RRT***
- **RRT*-AR**
Results II: RRT*-AR computes routes rapidly

RRT*-AR optimizes rapidly indicating fast exploration because of cost approximation
Results II: RRT*-AR computes routes rapidly

RRT*-AR optimizes rapidly indicating fast exploration because of cost approximation

If the search tree is reused across iterations, more paths are closer to global optimal than if a new tree is created at every iteration.
Demonstration of Emergency Landing System
Demonstration of Emergency Landing System
Outline

1. Problem Statement
2. Related Work
3. Problem Formulation and Approach
4. Alternate Routes
5. RRT*-AR
6. Emergency Landing System
7. Results
8. Summary
   - Summary
   - Conclusion
   - Future Work
Summary

Alternate Routes are important for emergency landing

- Useful backups to increase safety of system
- Useful option for pilots

RRT*-AR rapidly computes Alternate Routes

- Encodes spatial diversity
- Rapid computation by approximation and tree re-use

Emergency landing system

- Decouples problems into Entry glide, Glide, Entry flare, Flare
- Applies different techniques for different problem fidelity
Conclusion

1. RRT*-AR provides 3 times more backup than RRT*.

2. RRT*-AR provides 25% better quality paths than RRT* in the same planning time.

3. RRT*-AR provides acceptable plans under 1.0 s and can react to unexpected obstacles in 0.1 s, thus meeting real time criteria.
Future Work

1. How helpful are alternate routes - need for a quantitative assessment.

2. Emergency landing from low altitudes - a faster, more reactive approach

3. Wind effects on emergency landing - how does it impact the optimization problem
Take Home Message

RRT*-AR is a practical, guaranteed approach to planning that rapidly generates high quality alternate routes.

- It's faster than a RRT* trying to generate a single comparable route
ADDITIONAL SLIDES
RRT*-AR leverages cost approximation
Optimistic vs Pessimistic Cost

4.6.3 Conditions for Asymptotic Optimality

As $n \to \infty$, the maximum extension length, $\max$, reduces. This is because the maximum extension length is within the near neighbour radius, $\left| \max \right|_n = r_n$. For a given

Optimistic Cost

Pessimistic Cost
Optimistic Cost Approximation Algorithm

**Algorithm 9** \((x_{\text{min}}, \sigma_{\text{min}}) = \text{ChooseParentApprox}(X_{\text{near}}, x_{\text{rand}}, c_{\text{lb}})\)

**Input:** \(X_{\text{near}} = \text{set of near vertices}, x_{\text{rand}} = \text{a sampled point}, c_{\text{lb}} = \text{lower bound cost} \)

**Output:** \(x_{\text{min}} = \text{parent resulting in lowest cost}, \sigma_{\text{min}} = \text{trajectory from parent}\)

```plaintext
1  pSet ← \{\}, x_{\text{min}} ← \text{NULL}, \sigma_{\text{min}} ← \text{NULL}
2  for x_{\text{near}} ∈ X_{\text{near}} do
3      c_{\text{eval}} ← \text{SteerCostApproxOptim\(x_{\text{near}}, x_{\text{rand}}\)} + \text{Cost\(x_{\text{near}}\)}
4      if c_{\text{eval}} < (1 + \epsilon)c_{\text{lb}} then
5          pSet ← pSet ∪ \{c_{\text{eval}}, x_{\text{near}}\}
6      pSet ← \text{sort\(pSet\)}
7  for x_{\text{parent}} ∈ pSet do
8      \sigma ← \text{Steer\(x_{\text{parent}}, x_{\text{rand}}\)}
9      if \text{CollisionFree\(\sigma\)} then
10         c_{\text{min}} ← \text{Cost\(x_{\text{near}}\)} + \text{Cost\(\sigma\)}
11         x_{\text{min}} ← x_{\text{near}}, \sigma_{\text{min}} ← \sigma
12         break
13  return \((x_{\text{min}}, \sigma_{\text{min}})\)
```

The reasons for optimistic and pessimistic cost arise from the principle of branch and bound and rewiring respectively. During the extension process, the absolute magnitude of the cost is not very relevant as is the relative magnitude to determine the best parent. However, it is required to guess an upper bound above which we no longer consider a parent worth extending from. Let \(\hat{c}_{\text{optim}}(\uparrow) = c(\uparrow)\), \(\hat{c}_{\text{optim}}(\uparrow) > 0\) which means that an optimistic cost is always below that of the true cost. Then the optimistic cost from root of a new vertex for parent \(x_{1}\) is \(\hat{c}_{\text{optim}}(\uparrow) + c(x_{1})\). If this is more than an upper bound, \(\hat{c}_{\text{optim}}(\uparrow) + c(x_{1}) > c_{\text{upper}}\), then \(c(\uparrow) + c(x_{1}) > c_{\text{upper}} + \epsilon\). This implies that optimistic approximation should be used to ensure only when the true cost is definitely
Choose Best Parent Algorithm

Algorithm 7 \((x_{\text{min}}, \sigma_{\text{min}}) = \text{ChooseParentAlternate}(X_{\text{near}}, x_{\text{rand}}, r_{\text{near}}, c_{\text{lb}})\)

**Input:** \(X_{\text{near}} = \) set of near vertices, \(x_{\text{rand}} = \) a sampled point, \(r_{\text{near}} = \) radius of near neighbours, \(c_{\text{lb}} = \) lower bound cost

**Output:** \(x_{\text{min}} = \) parent resulting in lowest cost, \(\sigma_{\text{min}} = \) trajectory from parent

1. \(c_{\text{min}} \leftarrow \infty, x_{\text{min}} \leftarrow \text{NULL}, \sigma_{\text{min}} \leftarrow \text{NULL}\)
2. **for** \(x_{\text{near}} \in X_{\text{near}}** do
3. \(\sigma \leftarrow \text{Steer}(x_{\text{near}}, x_{\text{rand}}), d_{eq} \leftarrow \min(D_{eq}, \rho r_{\text{near}})\)
4. \(c \leftarrow \text{Cost}(x_{\text{near}}) + \text{Cost}(\sigma)\)
5. **if** \(\exists x_{eq} \text{ s.t. } ||x_{eq} - x_{\text{rand}}|| < d_{eq} \text{ and } x_{\text{near}} = \text{Parent}(x_{eq})\) **then**
6. \(c \leftarrow c + \epsilon c_{\text{lb}}\)
7. **if** \(c < c_{\text{min}}\) **then**
8. \(c_{\text{min}} \leftarrow c\)
9. \(x_{\text{min}} \leftarrow x_{\text{near}}, \sigma_{\text{min}} \leftarrow \sigma\)
10. **return** \((x_{\text{min}}, \sigma_{\text{min}})\)
Reuse Tree Results

![Diagram showing the number of cases for different ratios of cost to best cost. The x-axis represents the ratio of cost to best cost, ranging from 1 to 6. The y-axis represents the number of cases. Two bars are shown for each ratio, one for Reuse Tree and one for Fresh Plan. The Reuse Tree bar is generally higher than the Fresh Plan bar.](image-url)
Variance of normalized cost over time for different methods:

- **RRT***
- **Approx**
- **Constrained**

The graph shows the variance of normalized cost decreasing with time for each method, with **RRT*** having the highest variance initially but showing a faster decrease compared to the other methods.
RRT*-AR Part I

Algorithm 1 \( G = \text{RRT}^*-\text{AR}((V,E),N) \)

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{Input:} \( V = \) vertices, \( E = \) edges, \( N = \) number of iterations
\State \textbf{Output:} \( G \) is the tree returned
\For {\( i = 1, \ldots, N \)}
  \State \( x_{\text{rand}} \leftarrow \text{Sample} \)
  \State \( X_{\text{near}} \leftarrow \text{Near}(V,x_{\text{rand}},r_{\text{near}}) \)
  \State \( (x_{\text{min}}, \sigma_{\text{min}}) \leftarrow \text{ChooseParent}(X_{\text{near}},x_{\text{rand}}) \)
  \If {\text{CollisionFree}(\sigma_{\text{min}})}
    \State \( V \leftarrow V \cup \{x_{\text{rand}}\} \)
    \State \( E \leftarrow E \cup \{(x_{\text{min}},x_{\text{rand}})\} \)
    \State \( (V,E) \leftarrow \text{Rewire}((V,E),X_{\text{near}},x_{\text{rand}}) \)
  \EndIf
\EndFor
\State \textbf{return} \( G = (V,E) \)
\end{algorithmic}
\end{algorithm}
RRT*-AR Part 2

Algorithm 2 \((x_{min}, \sigma_{min}) = \text{ChooseParent}\) 
\((X_{near}, x_{rand}, r_{near}, c_{lb})\)

**Input:** \(X_{near}\) = set of near vertices, \(x_{rand}\) = a sampled point, \(r_{near}\) = radius of near neighbours, \(c_{lb}\) = lower bound cost

**Output:** \(x_{min}\) = parent resulting in lowest cost, \(\sigma_{min}\) = trajectory from parent

1. \(pSet \leftarrow \{\}\) , \(c_{min} \leftarrow \infty\) , \(x_{min} \leftarrow \text{NULL}\) , \(\sigma_{min} \leftarrow \text{NULL}\)
2. \(\text{for } x_{near} \in X_{near} \text{ do}\)
3. \(\quad d_{eq} \leftarrow \min(D_{eq}, \rho r_{near})\)
4. \(\quad c_{eval} \leftarrow \text{SteerCostApproxOptim}(x_{near}, x_{rand}) + \text{Cost}(x_{near})\)
5. \(\quad \text{if } c_{eval} < (1 + \varepsilon)c_{lb} \text{ then}\)
6. \(\quad \quad \text{if } \exists x_{eq} \text{ s.t. } ||x_{eq} - x_{rand}|| < d_{eq}\)
7. \(\quad \quad \quad \text{and } x_{near} = \text{Parent}(x_{eq}) \text{ then}\)
8. \(\quad \quad \quad c_{eval} \leftarrow c_{eval} + \varepsilon c_{lb}\)
9. \(\quad pSet \leftarrow pSet \cup \{c_{eval}, x_{near}\}\)
10. \(pSet \leftarrow \text{sort}(pSet)\)
11. \(\text{for } x_{parent} \in pSet \text{ do}\)
12. \(\quad \sigma \leftarrow \text{Steer}(x_{parent}, x_{rand})\)
13. \(\quad \text{if } \text{CollisionFree}(\sigma) \text{ then}\)
14. \(\quad \quad c_{min} \leftarrow \text{Cost}(x_{near}) + \text{Cost}(\sigma)\)
15. \(\quad x_{min} \leftarrow x_{near} , \sigma_{min} \leftarrow \sigma\)
16. \(\text{break}\)

\(\text{return } (x_{min}, \sigma_{min})\)
RRT*-AR Part 3

Algorithm 3  \( G = \text{Rewire}((V, E), X_{\text{near}}, x_{\text{rand}}, r_{\text{near}}, c_{\text{lb}}) \)

**Input:** \( V = \) vertices, \( E = \) edges, \( X_{\text{near}} = \) set of near vertices, \( x_{\text{rand}} = \) a sampled point, \( r_{\text{near}} = \) radius of near neighbours, \( c_{\text{lb}} = \) lower bound cost

**Output:** \( G \) is the tree returned

\[
\begin{align*}
1 & \text{for } x_{\text{near}} \in X_{\text{near}} \text{ do} \\
2 & \quad c_{\text{eval}} \leftarrow \text{SteerCostApproxPessim}(x_{\text{rand}}, x_{\text{near}}) + \text{Cost}(x_{\text{rand}}) \\
3 & \quad \text{if } \exists x_{\text{eq}} \text{ s.t. } \|x_{\text{eq}} - x_{\text{near}}\| < d_{\text{eq}} \text{ and } x_{\text{rand}} = \text{Parent}(x_{\text{eq}}) \text{ then} \\
4 & \quad \quad c_{\text{eval}} \leftarrow c_{\text{eval}} + \varepsilon c_{\text{lb}} \\
5 & \quad \quad \text{if } c_{\text{eval}} < (1 + \varepsilon)c_{\text{lb}} \text{ and } c_{\text{eval}} < \text{Cost}(x_{\text{near}}) \text{ then} \\
6 & \quad \quad \quad \sigma \leftarrow \text{Steer}(x_{\text{rand}}, x_{\text{near}}), d_{\text{eq}} \leftarrow \min(D_{\text{eq}}, \rho r_{\text{near}}) \\
7 & \quad \quad \quad \text{if } \text{Cost}(x_{\text{rand}}) + \text{Cost}(\sigma) < \text{Cost}(x_{\text{near}}) \text{ then} \\
8 & \quad \quad \quad \quad \text{if } \text{CollisionFree}(\sigma) \text{ then} \\
9 & \quad \quad \quad \quad \quad x_{\text{parent}} \leftarrow \text{Parent}(x_{\text{near}}) \\
10 & \quad \quad \quad \quad E \leftarrow E \setminus \{x_{\text{parent}}, x_{\text{near}}\} \\
11 & \quad \quad \quad E \leftarrow E \cup \{x_{\text{rand}}, x_{\text{near}}\} \\
12 & \text{return } G = (V, E)
\end{align*}
\]
Backup Path Results

(a) (b)

Fig. 11. AR for a failure over the mountains. (a) RRT* comes up with 2 routes - each going to a maximal feasible LZ (b) RRT*-AR coming with 6 routes including the routes of RRT* - each path having its own merit.

---

(a) (b)

Fig. 12. Alternate Routes allow quick reaction to clutter in LZ during a landing scenario in the valley. (a) The optimal path (blue) is followed by the helicopter while simultaneously planning alternate (yellow) routes (b) LZ-1 comes within the sensor range and is observed to be infeasible. The helicopter instantaneously switches to an alternate route.