Longitudinal Motion Planning for Slung-Loads Using Simplified Models and Rapidly-Exploring Random Trees

Eric N. Johnson
eric.johnson@ae.gatech.edu
Lockheed-Martin Associate Professor of Avionics
Georgia Institute of Technology
Atlanta, Georgia, USA

John G. Mooney
john.g.mooney@gatech.edu
Graduate Research Assistant
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

A randomized motion-planning approach to providing guidance for helicopters with under-slung loads is presented. Rapidly-exploring Random Trees are adapted to plan trajectories for simplified helicopter-load models. Four different planning models are tested against four different representative tasks. The poor performance of the baseline planner, and subsequent efforts to improve that performance shows the sensitivity of the RRT to proper sizing of the sampling area and amount of computation available. Further lines of potential research into optimizing planner performance and reducing computational cost are identified.

INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are rapidly becoming more important and relevant to military and civil aviation operations. However, there are a number of tasks which, thus far, still require the skill, adaptability, and experience of a human pilot. One task in particular that requires an especially skilled pilot is helicopter slung-load operations.

A helicopter's capability to carry large and heavy loads externally, as well as pick them up and drop them off in unprepared sites, is one of its primary advantages over fixed-wing aircraft. However, the addition of an external load changes the stability and handling qualities of the system, typically making it more difficult to fly. As a result, most pilots flying with slung-loads limit their speed and aggressiveness to keep the load and the aircraft well within the flight envelope. Moreover, pilots need the ability to see the load, meaning flight in degraded visual environments is very limited. Finally, the lack of maneuverability means that pilots will tend to fly well above any obstacles, and the aircraft will be more exposed to observation and threat (in a military context).

Despite the difficulties of flying with a slung-load, aggressive maneuvers are possible. Because of factors unique to the local Christmas tree market, Christmas tree farms in the US Pacific Northwest have used helicopters with slings to load their trucks for transport for at least 30 years (Figure 1). These pilots fly hundreds of loads every day, 8-10 hours per day, during the harvest. Because of the urgency of the operation, the pilots have learned how to employ very aggressive maneuvers to pick up and drop the trees at very precise locations in a minimum of time. It appears that part of the reason the pilots have developed this capability is that they are able to anticipate the load's motion and plan their inputs to drive the load to a precise position. Likewise, it seems that there is potential for applying this principle to the way that UAVs handle slung-loads. Motion planning has long been studied for a wide range of robotics applications, and could enable unmanned slung-load operations as well as augment human pilots in this difficult task.

Fig. 1. Helicopter loading Christmas trees for transport.

This paper reports on the continued investigation of the

Presented at the Sixth AHS International Specialists' Meeting On Unmanned Rotorcraft Systems, Chandler, Arizona, January 20-22, 2015.

suitability of RRTs to motion-plan for aggressive slung-load operations (Ref. 1). In particular, it focuses on longitudinal motion planning, using four simplified propagation models to accomplish four tasks of increasing complexity. This work provides an assessment the ability of the algorithm to compute a feasible solution, as well as assessing the sensitivity of results to certain parameters of the planner. Since RRTs are non-deterministic, solutions are sampled in order to find a distribution of performance results. This work is the only known application of RRTs to helicopter slung-loads.

Review of Relevant Literature

To date, most stability and controls research for helicopters with slung-loads has focused on stabilizing the load or limiting load swing (for example, (Refs. 2–6)). Only two sources were found during review (Refs. 7, 8) where guidance and/or motion planning methods were applied to this problem. However, there is no shortage of capable motion-planning methodologies.

One method in particular which seems well-suited for planning for slung-loads is Rapidly-Exploring Random Trees (RRTs), due to their ability to easily handle nonlinear models with arbitrary constraints (Refs. 9, 10). Since its introduction in 1998, the RRT has seen several modifications and improvements. LaValle and Kuffner (Ref. 11) themselves introduced the ideas of reverse RRTs, which grow from the goal configuration backward, and RRT-Connect, a version that grows forward and reverse trees simultaneously, attempting to connect the two in the middle. Other researchers have found ways to improve or speed the solution by caching parts of previously solved trees (Ref. 12), dynamically adjusting the sampling domain size (Ref. 13), optimizing nearest-neighbor searches (Ref. 14), and pruning or re-wiring the tree to seek the optimal solution (Refs. 15, 16)

PROBLEM FORMULATION

Guidance Algorithm

In the context of this problem, a tree is a set of possible state trajectories, and associated control inputs, with a common starting point. Where the trajectories diverge from one another forms the branches of the tree. Several methods exist for developing trajectory tree, which primarily differ from each other in how they determine which element of the tree to grow from at each iteration, and in how new branches are grown.

Below is the basic algorithm for constructing a Rapidly-exploring Random Tree. The algorithm shown here (Algorithm 1) is a slight modification of that published in (Ref. 9). In essence, the RRT selects the node to grow from the tree by generating a random state configuration, then searching the tree for the node which is "closest" in some sense. The algorithm then grows the tree from that node by selecting a control input through arbitrary means—in this case, at random—and uses an incremental simulator to find the resultant trajectory segment. This incremental simulator will be referred to as the planning model. An example tree where a two dimensional path is found around an obstacle without dynamic constraints is shown below in Figure 2.

The computational complexity of the RRT depends upon the planning model complexity, the method of NEAREST-NEIGHBOR searches, the distance metric, and the control selection technique used. For this work, brute-force lookups and randomized control selection is used. The distance metric and planning models are detailed below. Analysis has shown computation to be $\mathcal{O}(c_2K^2 + c_1K)$, where K is the total number of iterations (and nodes in the tree) and the coefficents c_i are functions of model, integrator, and distance function complexity.

Algorithm 1 Generate RRT (adapted from (Ref. 9))

procedure GENERATERRT($x_{init}, x_{goal}, K, \Delta t, bias, \delta$)
  $v_0 \leftarrow$ **new** Node(x_{init},0)
  $T.init(v_0)$
  for $k = 1$ to K **do**
    $x_{rand} \leftarrow$ RANDOMSTATE($x_{goal}, bias$)
    $v_{near} \leftarrow$ NEARESTNEIGHBOR(T, x_{rand})
    $u \leftarrow$ SELECTINPUT($x_{rand}, v_{near}.x$)
    $x_{new} \leftarrow$ MODEL.NEWSTATE($v_{near}.x, u, \Delta t$)
    if MODEL.COLLISIONFREE($x_{new}, v_{leaf}.x$) **then**
      $v_{new} \leftarrow$ **new** Node(x_{new}, u)
      $T.addNode(v_{new})$
    end if
  end for
  $v_{nearest} \leftarrow$ NEARESTNEIGHBOR(T, x_{goal})
  $U \leftarrow$ ASSEMBLECONTROLSEQUENCE($T, v_{nearest}$)
  return U
end procedure

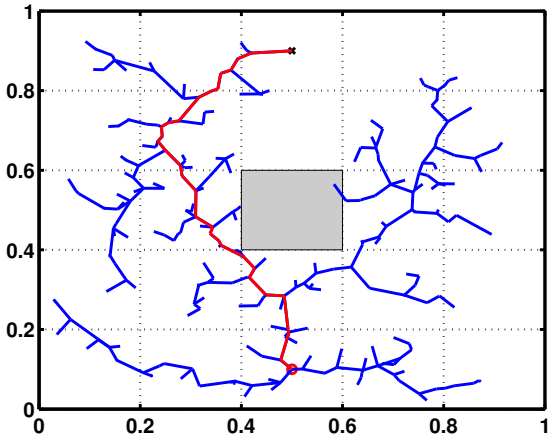

Fig. 2. Sample 2-D geometric RRT.

Planning Models

The incremental simulator of the helicopter-load system is modeled as a pendulum attached to, alternatively, a constant-altitude double integrator (abbreviated "CP"), a two-degree-of-freedom double integrator ("DI") (Equations (1) - (10)), and point mass with first-order pitch and thrust dynamics ("PP")(Equations (11) - (12)). A fourth model will represent the load directly as a point mass with first-order thrust dynamics and second-order pitch dynamics ("LL"). These dynamics are formulated using multibody equations of motion, with a virtual spring-mass-damper in the linkage to help enforce the constraint between the pivot and the mass. Again, these models are drastic simplifications of real aircraft with slung-loads, but are being examined to determine if they are "good enough" for planning purposes. Note that these models are generic enough to be adapted to three dimensions, by adding a third dimension to $\vec{r}_P$, $\dot{\vec{r}}_P$, $\vec{r}_M$, $\dot{\vec{r}}_M$, and $\vec{u}$. Figure 3 shows the DI model, with the CP and PP models differing only in how the pivot acceleration is defined. The LL model is depicted in Figure 4.

Pendula with differential pivot motion

$$\vec{x} = \left\{ \vec{r}_P^T,\ \dot{\vec{r}}_P^T,\ \vec{r}_M^T,\ \dot{\vec{r}}_M^T \right\}^T \tag{1}$$

$$\dot{\vec{x}} = \begin{Bmatrix} \dot{\vec{r}}_P \\ \ddot{\vec{r}}_P \\ \dot{\vec{r}}_M \\ \vec{a}_{EXT} + \vec{a}_C + \vec{a}_{AP} + \frac{\vec{F}_{CE}}{m} \end{Bmatrix} \tag{2}$$

where

$$l = \left\| \vec{r}_M - \vec{r}_P \right\| \tag{3}$$

$$\vec{\eta} = \frac{\left(\vec{r}_M - \vec{r}_P \right)}{l} \tag{4}$$

$$\vec{a}_C = \frac{\left\| \dot{\vec{r}}_M - \left(\dot{\vec{r}}_M^T \vec{\eta} \right) \vec{\eta} \right\|^2}{l_0} \vec{\eta} \tag{5}$$

$$\vec{a}_{AP} = \left(\ddot{\vec{r}}_P^T \vec{\eta} \right) \vec{\eta} \tag{6}$$

$$\vec{D} = -\frac{\rho f}{2} \left\| \dot{\vec{r}}_M \right\| \dot{\vec{r}}_M \tag{7}$$

$$\vec{a}_{EXT} = \left(\frac{\vec{D}}{m} + \vec{g} \right) - \left[\left(\frac{\vec{D}}{m} + \vec{g} \right)^T \vec{\eta} \right] \vec{\eta} \tag{8}$$

$$\vec{F}_{CE} = -k_{CE} (l - l_0) \vec{\eta} - c_{CE} \left[\left(\dot{\vec{r}}_M^T \vec{\eta} \right) - \left(\dot{\vec{r}}_P^T \vec{\eta} \right) \right] \vec{\eta} \tag{9}$$

$$\ddot{\vec{r}}_P = \vec{u} \tag{10}$$

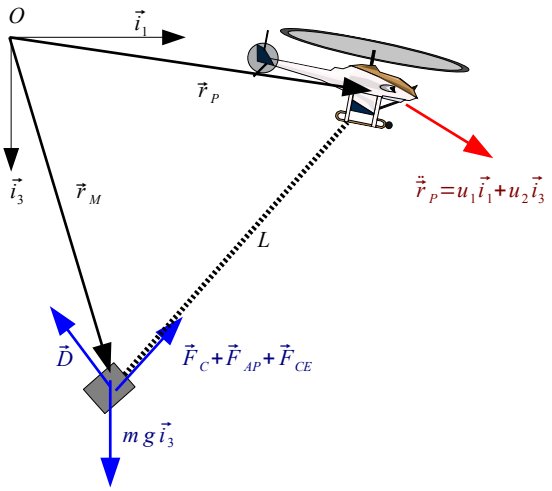

Fig. 3. Pendulum with Kinematic Pivot

The coefficients on the constraint enforcement are set so that the spring-mass-damper frequency is an order of magnitude faster the pendulum frequency, but the period is less than half of the numerical integration time step size.

This same set of equations applies to the pitching particle model, with T and θ added to the state vector, and the following modifications to equation (10):

$$\ddot{\vec{r}}_P = -T \begin{Bmatrix} \sin\theta \\ \cos\theta \end{Bmatrix} \tag{11}$$

$$\begin{Bmatrix} T \\ \dot{\theta} \end{Bmatrix} = \vec{u} \tag{12}$$

Load modeled as point mass with double integrator attitude dynamics A slightly different, simpler, approach is to directly plan the load trajectory by modeling it as a point mass with a defined attitude and applying slower dynamics. The position and velocity commands for the aircraft are then solved kinematically as a function of the attitude and angular velocity of the load.

$$\vec{x} = \left\{ \vec{r}_M^T,\ \theta,\ \dot{\vec{r}}_M^T,\ \dot{\theta},\ T \right\}^T \tag{13}$$

$$\dot{\vec{x}} = \begin{Bmatrix} \dot{\vec{r}}_M \\ \dot{\theta} \\ -T \begin{Bmatrix} \sin\theta \\ \cos\theta \end{Bmatrix} \\ u_2 \\ u_1 \end{Bmatrix} \tag{14}$$

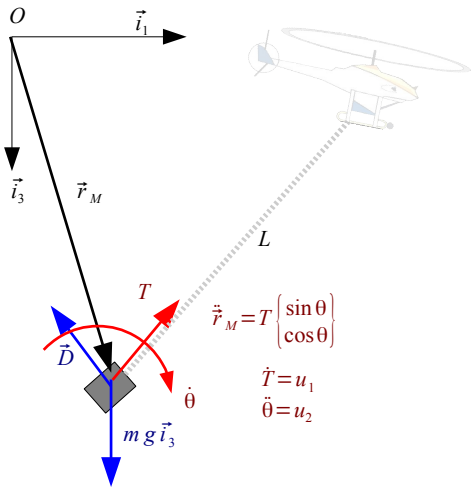

Fig. 4. Longitudinal Load-based Model

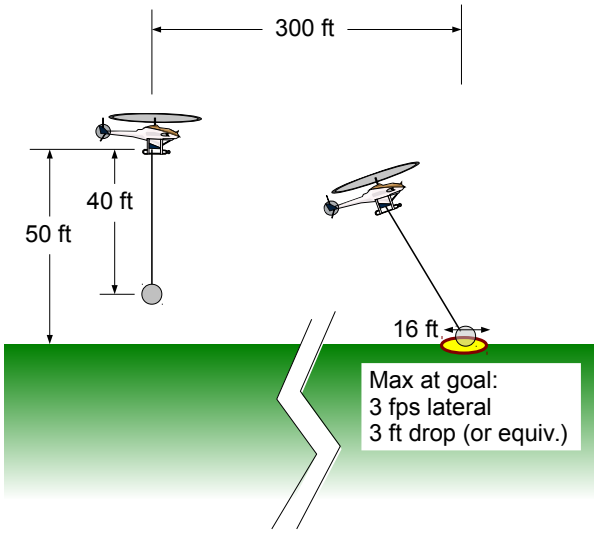

Fig. 5. Task 4.

Prototype Slingload Tasks

A final element of this investigation is to define tasks which adequately represent the more generic problem of aggressive precision slung-load delivery. To that end, four prototype tasks have been developed, partly inspired by the attempt described in (Ref. 17) to quantify obstacle avoidance performance. Criteria for successful completion of each task are described below; the numbers are somewhat arbitrary but may be changed depending upon application.

Task 1: Trim-Hover to Trim-Hover The first scenario is an "easy" reposition task to help establish a baseline measure upon which all the other tasks are based. The load and the aircraft start and end at or near rest, and both aircraft and load position are included in the goal state. The aircraft begins from a point 50 ft AGL, with a 40 ft line, and moves a distance of 300 ft. The goal region is defined as both load and aircraft being within a 10 ft by 10 ft box centered on the goal position, and moving less than 3 feet per second.

Task 2: Christmas Tree Drop The second scenario mimics the Christmas tree harvesting operation referenced in the introduction. This task specifies a final position and velocity of the load, but not on the aircraft. The remainder of the task geometry is the same as task 1. The goal region is defined as load being within a 8 ft of the goal position, and moving less than 3 feet per second.

Task 3: Linear Wall Obstacle Task 3 is identical to task 1, except for the inclusion of a 25 foot high wall obstacle placed at the midpoint between start and goal. This task is the first to require the load, line, and aircraft to avoid an obstacle other than the ground.

Task 4: Precision Load Placement on Ground This task is similar to task 2, but is likely the most difficult of the longitudinal tasks, as it requires precision load placement with minimal impact velocity and the additional constraint of not being able to approach the goal from the underside (Figure 5). Success in this task is defined as the load being within 16 ft laterally of the goal, and moving less than 3 feet per second laterally. Vertically, the combination of vertical speed and drop height should cause an impact velocity less than the equivalent of a 3 foot drop from rest.

TEST CONDITIONS

The RRT algorithm was tested by applying it, with each of the four planning models, to each of the four tasks. The testing environment consisted of simply executing the planner in a single query for a single initial condition. The RRT was 75 times with each model performing each task in order to find the distribution and tendencies of the planner's performance. This number was selected based on a trade study showing that larger sample sizes gave little to no additional refinement to the sample distributions.

The baseline parameters used to obtain these results are included in Tables 1 and 2, and the cost/distance function used is given in Equations 15 and 16. The system parameters were designed to mimic the scale of a Yamaha R-MAX, the most likely testbed for a future implementation of this planner. The simulations were carried out in a C++ implementation running under Ubuntu 12.04.4 LTS (32-bit) on an 8-core Intel®Xeon™3.20 GHz CPU with 4 GB of memory.

$$\alpha = 2^{\left(-\left\|\vec{r}_{M,goal} - \vec{r}_M\right\|/l_0\right)} \tag{15}$$

Table 1. Dynamic System Parameters.

Parameter	Value
Line Length	40 ft
Load Mass	20 lbm
Load Equiv. Flat Plate Area	1 ft^2
Air Density	0.002378 slug/ ft^3

Table 2. Planner Parameters.

Parameter	Value
Number of Nodes	1000
Control Space (linear)	$[-X,0,X]$, $X \sim U(0,3)\ ft/s^2$
Control Space (angular)	$[-Y,0,Y]$, $Y \sim U(0,1.4)\ rad/s$
Branch Length	3 s
Integration Time Step	0.05 s
Goal Bias	0.9
Sampling Domain	
Load Position	$\vec{R}_M \sim [U(-150,450), U(-290,310)]^T\ ft$

$$J = (1-\alpha)\left\|\vec{r}_{M,goal} - \vec{r}_M\right\|^2 + \alpha\left\|\dot{\vec{r}}_{M,goal} - \dot{\vec{r}}_M\right\|^2 \qquad (16)$$

Each sample provided the following outputs, with the scalar outputs aggregated into a distribution: percentage of plans that successfully find the goal (as defined above); duration of planned trajectory; computation time; trees and trajectories from the "best" runs—e.g. the shortest trajectory duration or least final error, etc.; and trees and trajectories from a randomly selected run. Several other quantities were captured, such as load swing and aggregate control input, but did not produce results which merit reporting.

RESULTS AND DISCUSSION

A baseline comparison of the four longitudinal planning models is presented here. Again, each of the four longitudinal tasks were given to the planner with the four longitudinal planning models using the parameters listed in Tables 1 and 2, generating an RRT 75 times, using controls selected from a uniform random distribution, with one exception. Using a uniform random distribution with the LL model so often resulted in unusable trees that the possible inputs were restricted to five discrete control vectors from which the actual control was randomly selected.

All four planning models had difficulty achieving success at each of the four tasks. The only combination with greater than 50% success rate was the CP performing Task 2—to be expected, as the model merely needs to find the right horizontal position with the right amount of swing (Table 3).

Table 3. Longitudinal Model Base Success Rates (%).

	CP	DI	PP	LL
Task 1	5.3	0	0	0
Task 2	54.6	1.3	0	0
Task 3	0	0	0	0
Task 4	0	1.3	0	0

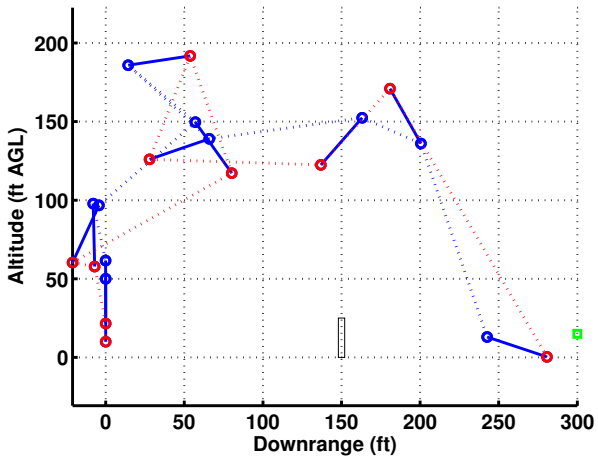

Fig. 6. Trajectory and tree for randomly selected run in trial, model PP, Task 3.

Baseline Task Performance

This experiment generated a number of general observations, and the comparative performance of the four planning models was similar for all four tasks, with a few exceptions to be described in detail. An important common characteristic each of the models performing their tasks was the unreliability of getting a "good" trajectory in any arbitrarily selected run of the planner, for example the result shown in Figure 6. This unreliability will be addressed later.

The CP model outperformed the other others in Tasks 1 and 2, presumably due to the fact that the starting altitude of the load was very close to the target altitude. For Task 1, the median trajectory duration was about 44 seconds (Figure 7), an average speed of 6.8 ft/s, which is very docile for this scale aircraft. This model, however, was the only one that had a chance to put the load inside the target at low speed—typically within 10 ft of the target (Figure 8) and less than 10 ft/s (Figure 9). The fastest trajectory was completed in just under 28 seconds, which is an average of 10.5 ft/s, a bit faster clip. For Task 2, the CP model was able to result in less than 8 ft of position error most of the time—expected, since the load will start and stay at nearly the goal altitude. The performance of CP at Tasks 3 and 4 was far worse, as the goal state was not reachable by virtue of the model's fixed altitude. A sample trajectory is shown in Figure 10.

The DI model provided the best overall performance in that it could reach every goal state, but still produced trajectories which were quite reasonable if not achieving the threshold required for success. It generated trajectories which were both more aggressive and less able to precisely place the system in the target location. Examination of the trees produced by the planner not coincidentally showed that the DI model's trees more thoroughly explored the state space than the more complex PP and LL (examples shown in Figures 11). The median time of flight for Task 1 was about 44 seconds, identical to the CP; however it had far more variability. The final position and

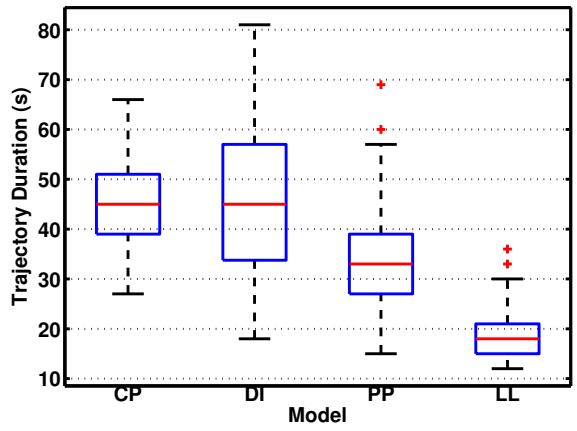

Fig. 7. Boxplot comparison of trajectory duration for four planning models, Task 1

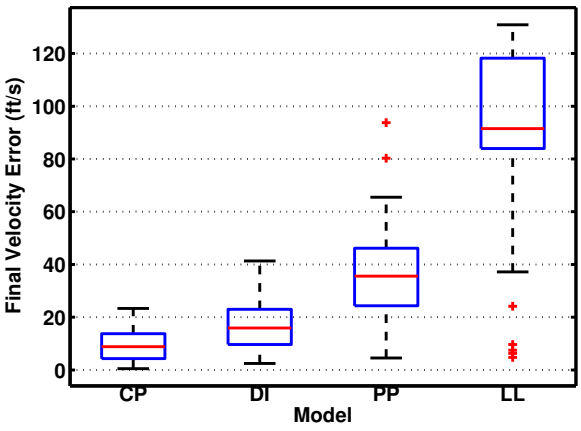

Fig. 9. Boxplot comparison of final velocity error for four planning models, Task 1

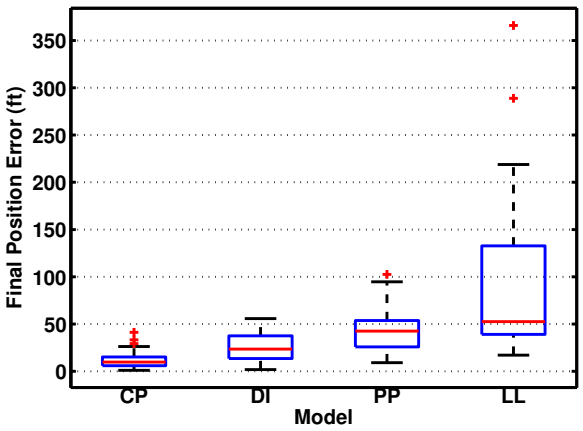

Fig. 8. Boxplot comparison of final position error for four planning models, Task 1

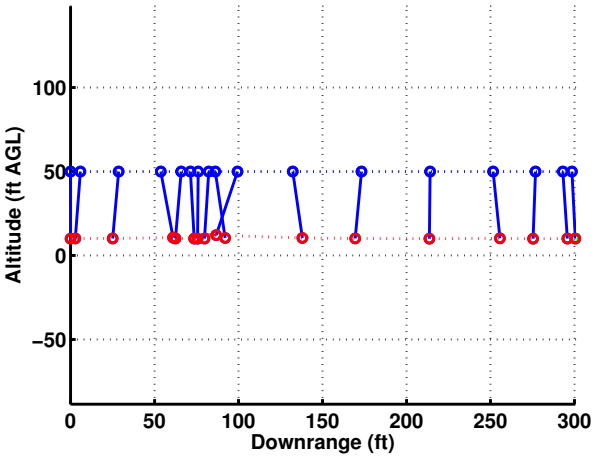

Fig. 10. Trajectory for least final error in trial, model CP, Task 1.

velocity errors were about twice as much as for the CP. For all of the tasks the DI (and PP), showed "zeroing" behavior at the end of a trajectory (Figures 12 and 13). The DI model, notably, was the only of the three to generate a successful trajectory for Task 4.

The PP model produced fast trajectories, but resulted in even less precision hitting the goal state. The median flight time for Task 1 was just under 35 seconds, a speed of 8.5 ft/s, but the system missed the target by an median of around 45 feet, and was moving at an average of 20 ft/s at the trajectory end. Results for the other three tasks were similar. A sample trajectory for this model is shown in Figure 14.

The LL model produced the fastest trajectories, but this is at least partly to a large fraction of the trees having failed to explore enough of the state space. This resulted in terrible precision in hitting the goal state–a median error of over 50 ft, with some plans showing 200 ft or more of error—barely covering half of the course. The LL model's trees (for example, Figure 15) show very distinct bunching of branches, with very clear indications that the state space is not being well-explored (also a problem with the PP model). This suggests a fundamental problem between this type of model and the cost function that is used to grow the tree. During the NEARESTNEIGHBOR phase of the RRT algorithm (see Algorithm 1) the growth node is selected based on Euclidean distance and speed, while the model is controlled by (effectively) specifying jerk. One positive note is that the LL model required about half of the computation needed for the other three, which explicitly model the pendulum of the load (Figure 16).

Effects of Computation Budget

The lack of success detailed above led to a search for ways to improve the success rate of the planner. The first attempt was to examine the effect of increasing or decreasing the computa-

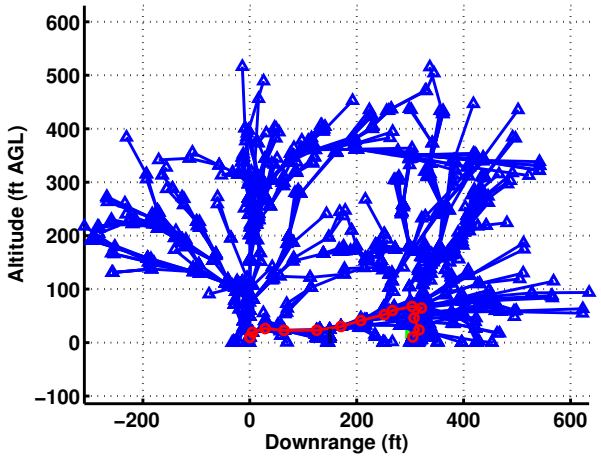

Fig. 11. Tree for least final error in trial, model DI, Task 3.

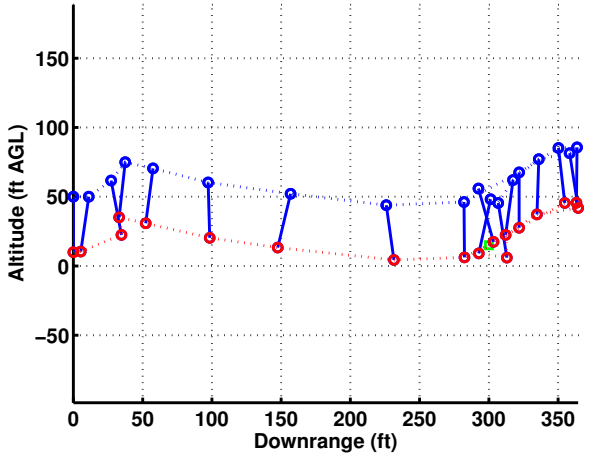

Fig. 12. Trajectory for least final error in trial, model DI, Task 2.

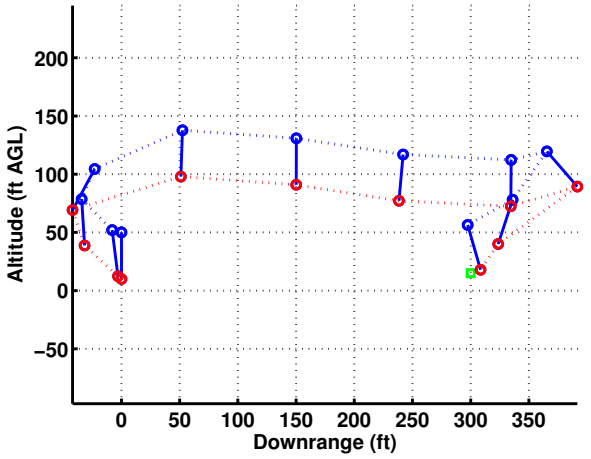

Fig. 13. Trajectory for least final error in trial, model PP, Task 2.

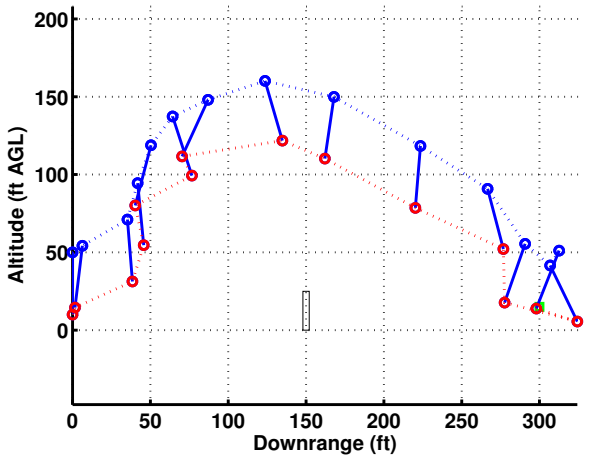

Fig. 14. Trajectory for least final error in trial, model PP, Task 3.

tional budget. A single task-model combination (DI performing Task 2 using the standard parameters) was run 75 times with each of the following set of maximum number of iterations: $\{50, 100, 200, 400, 800, 1600, 3200, 6400\}$. This experiment was repeated using Task 4, with virtually identical results.

Figure 17 shows a comparison between the median actual computation time for each size computational budget, and the theoretical growth of the compute time based on the order of complexity, using the actual compute time at 50 nodes to calibrate the estimate. Interestingly, the actual growth is less than predicted, and appears to be nearly linear—perhaps $\mathcal{O}(n \log n)$ rather than quadratic. This is probably a combination of two effects: first, the comparison operation which scales with number of nodes squared is probably cheaper than in the estimate; and second, also because comparison is cheap compared to numerical integration, the number of nodes needs to be much larger for the square term to start dominating the equation.

Figure 18 shows the change in success rate with computational budget. Though the jagged nature of the graph indicates this measurement probably variable from trial to trial, the initial indications is that increased computation after 800 or 1600 nodes provides diminishing returns, and still not quite to the level required to make the planning method reliable. Additionally, Figure 19 shows how final cost-to-go drops drastically each time the number of nodes is doubled until about 800, when nearly all variability is wiped out. The best final error solutions for each tree size were compared for quality. Ironically, the 50 node solution looks superior to the path planned by either 800 node or 6400 node trees. This is likely to be a fluke, as the sparsity of the 50 node tree means very little of the state space is explored. It does, however, suggest that repeatedly running the planner with a small number of nodes may produce as good or better plans than simply scaling up

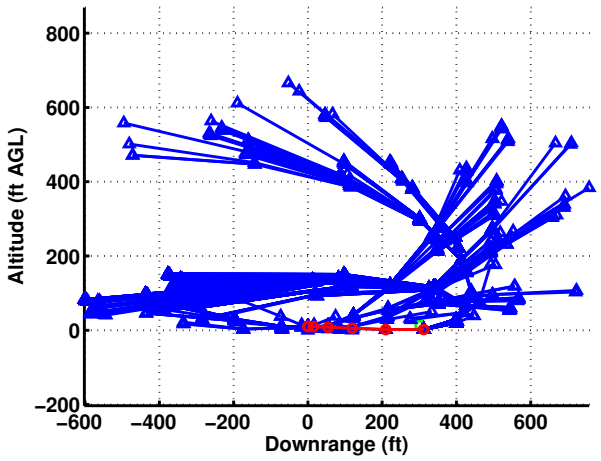

Fig. 15. Tree for least final error in trial, model LL, Task 2.

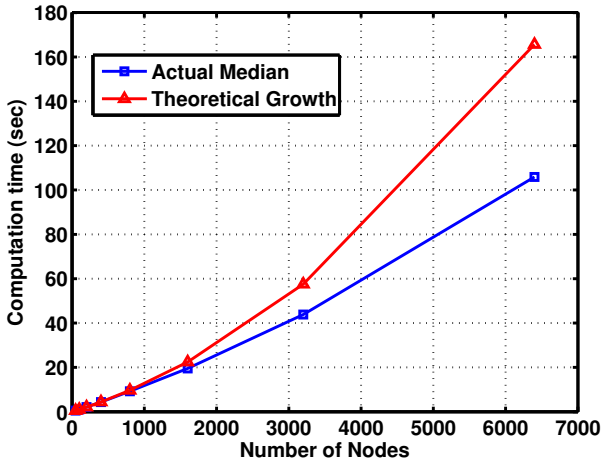

Fig. 17. Sensitivity of computation time to number of nodes for RRT.

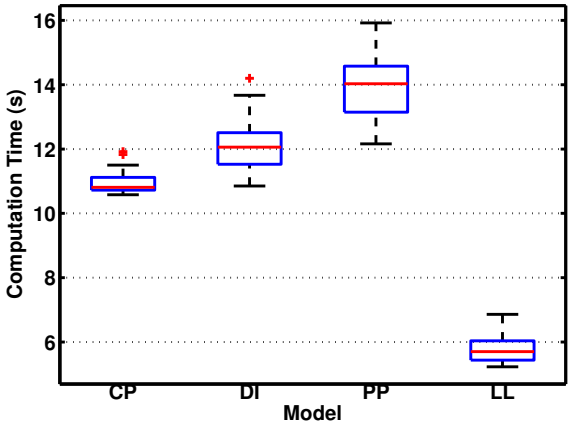

Fig. 16. Boxplot comparison of computation time for four planning models, Task 1

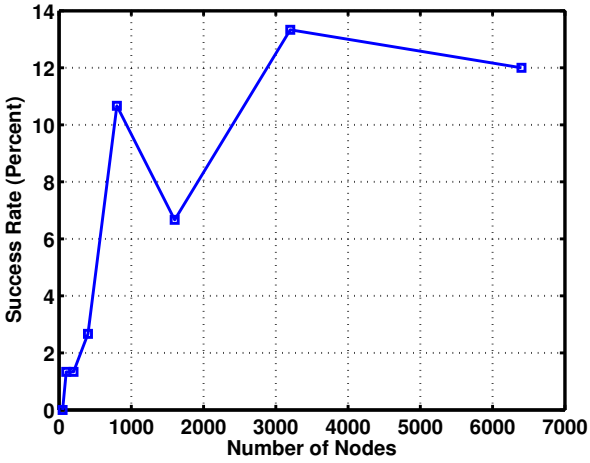

Fig. 18. Sensitivity of plan success rate to number of nodes for RRT.

the number of nodes in a single run.

Effects of Sampling Area Size

Several additional experiments were conducted to search for a method by which the success rate could be boosted to reasonable levels with modest computation and approach 100% with increased computation. The only factor which made a discernible difference in the success of the basic RRT with randomly selected inputs and step size was shrinking of the sampling area.

Two experiments show this result. First, a study was done using varying manually-selected sampling area sizes ranging from extra small (the smallest possible box which includes both the start and goal states, about 300 ft long and 50 ft tall) to extra large (900 ft x 900 ft). A graphical comparison of the standard sampling area used in the baseline results to the optimal sampling area size is shown in Figure 20. The step size was selected on a uniform distribution $U\ (0,3]$. The rest of the planning system parameters were as shown in Table 2. The most successful configuration used a sampling area which was a bit bigger than smallest (340 ft x 100 ft) (Figure 21). This result makes intuitive sense—in the extreme, if the sampling area is limited only to the states which lie on the optimal trajectory, then the solution will be optimal as well. It stands to reason that a larger sampling area with the same number of nodes, particularly with limited step size and random inputs, would limit exploration of the space.

A second experiment was conducted to find if the benefits of smaller sampling spaces would continue to be realized with increased computing budget. The optimal sampling area size from above was tested with 800, 1600, 3200, and 6400 nodes, and the results indeed show an asymptotic approach to 100 % success rate with increased computation, Figure 22.

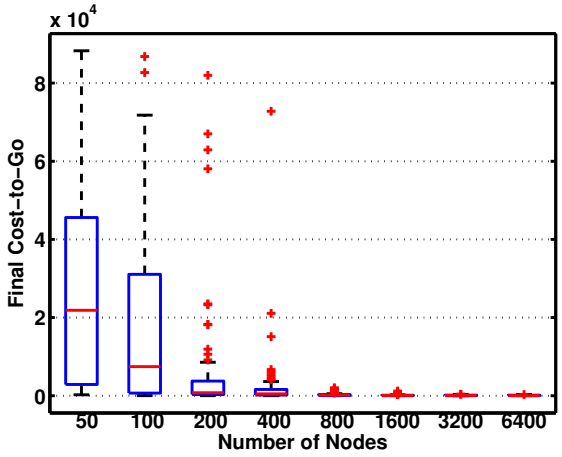

Fig. 19. Sensitivity of final cost-to-go to number of nodes for RRT.

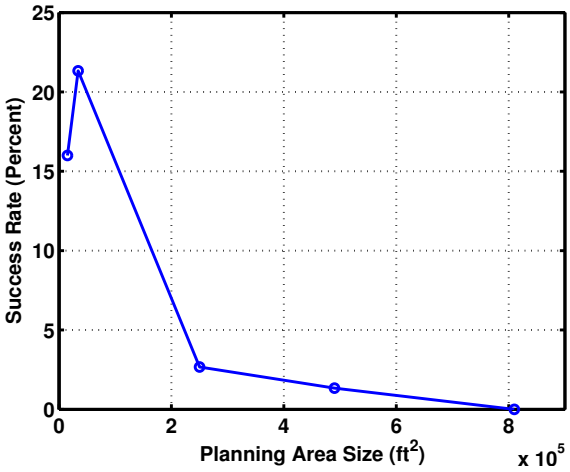

Fig. 21. Success rates of varying sampling area sizes, 1000-node tree.

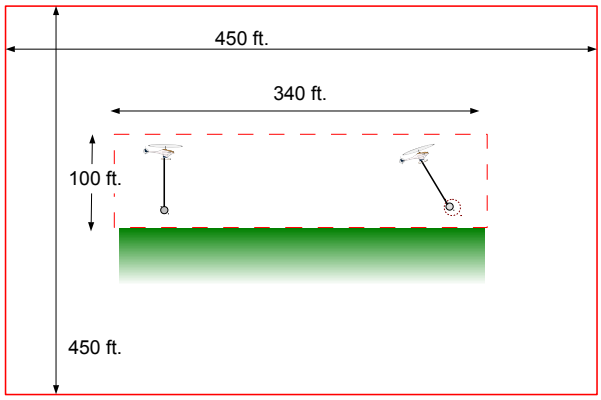

Fig. 20. Comparison of the sampling area sizes.

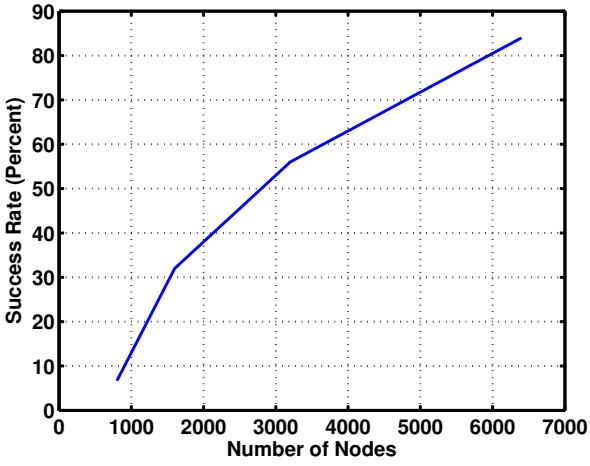

Fig. 22. Success rates of varying computation for small sampling area.

Potential Improvements and Further Targets of Research

The results presented here suggest several further lines of investigation. LaValle and Kuffner's follow-up publication (Ref. 10) to the original introduction of RRTs suggests two relevant ways to make direct improvements. First, they suggest that changing from a simple goal bias in sampling to a method that starts as a uniform distribution and begins to focus on the goal region as the tree grows. The second approach is designing a better distance metric, one that does not necessarily have relation to Euclidian distance, but is a measure of the input required to drive from x_{near} to x_{rand}.

Another approach could be to design a planning model that combines positive characteristics of the above models. For example, the LL model has low computational cost, while DI was best performing. Perhaps using a double integrator to directly model to load's motion would be a strong improvement—though would likely need some kind of jerk and acceleration limits to make the resulting trajectory feasible for a real aircraft and load. In addition to having better exploration of the state space, it may also make it possible to directly connect states (a difficult problem for the planning models used here), allowing some other variations of RRTs to be employed.

The availability of a node-connecting control means that the optimal variation of RRT, RRT* (Ref. 18), is possible. RRT* grows a tree similarly to RRT, but re-wires the tree for greatest optimality in the neighborhood of the newly added node.

Even without the availability of a connecting control, methods have been recently introduced (Ref. 16) which produce near-optimal trees and trajectories by pruning clearly sub-optimal branches to make way for more optimal branch growth. The method is called Sparse-Stable RRT, or SST for short.

Ultimately, the heavy computation required to solve for a

trajectory may mean that RRTs cannot yet be used in real time for this application. However, they may still hold potential for discovering aggressive slung-load maneuvers offline, which could be kept in a maneuver library for another online motion planning technique.

CONCLUSIONS

A Rapidly-exploring Random Tree was applied to the motion planning problem for a helicopter with slung-load and tested for potential use and further development. Four planning models were examined for their performance in four representative longitudinal tasks. The results show the relative qualities of each planning model, and its ability to fully explore the state space. In particular, results were very sensitive to the size of the sampled area and the amount of computation available. Further directions of research were suggested to improve the quality of solutions provided by the RRT while limiting the computational complexity of the planner-planning model combination.

ACKNOWLEDGMENTS

This study is funded by the U. S. Army under the Vertical Lift Research Center of Excellence (VLRCOE) program managed by the National Rotorcraft Technology Center, Aviation and Missile Research, Development and Engineering Center under Cooperative Agreement W911 W61120010 between the Georgia Institute of Technology and the U. S. Army Aviation Applied Technology Directorate. The authors would like to acknowledge that this research and development was accomplished with the support and guidance of the NRTC. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Aviation and Missile Research, Development and Engineering Center or the U.S. Government.

REFERENCES

[1]Johnson, E. N. and Mooney, J. G., "Preliminary Evaluation of Rapidly-Exploring Random Trees for Sling-Load Flight Guidance," Proceedings of the 2nd Asian/Australian Rotorcraft Forum and the 4th International Basic Research Conference on Rotorcraft Technology, 2013.

[2]Asseo, S. J. and Whitbeck, R. F., "Control Requirements for Sling-Load Stabilization in Heavy Lift Helicopters," *Journal of the American Helicopter Society*, Vol. 18, (3), 1973, pp. 23–31.

[3]Micale, E. C. and Poli, C., "Dynamics of Slung Bodies Utilizing a Rotating Wheel for Stability," *Journal of Aircraft*, Vol. 10, (12), 1973, pp. 760–763.

[4]Raz, R., Rosen, A., and Ronen, T., "Active Aerodynamic Stabilization of a Helicopter/Sling-Load System," *AIAA Journal of Aircraft*, Vol. 26, (9), 1989, pp. 822–828.

[5]Key, D. L., "Airworthiness Qualification Criteria for Rotorcraft With External Sling Loads," Technical Report January, National Aeronautics and Space Administration, Moffett Field, California, 2002.

[6]Hoh, R. H., Heffley, R. K., and Mitchell, D. G., "Development of Handling Qualities Criteria for Rotorcraft with Externally Slung Loads," Technical Report October, National Aeronautics and Space Administration, Moffett Field, California, 2006.

[7]Bisgaard, M., Cour-harbo, A., and Bendtsen, J. D., "Swing Damping for Helicopter Slung Load Systems using Delayed Feedback," Proceedings of AIAA Conference on Guidance, Navigation, and Control, 2009.

[8]Faust, A., Cruz, P., Fierro, R., and Tapia, L., "Aerial Suspended Cargo Delivery through Reinforcement Learning: Adaptive Motion Planning Research Group Technical Report TR13-001," Technical report, University of New Mexico, 2013.

[9]LaValle, S. M., "Rapidly-exploring random trees: A new tool for path planning," Technical report, Computer Science Department, Iowa State University, 1998.

[10]LaValle, S. M. and Kuffner, J. J. J., "Rapidly-exploring random trees: Progress and prospects," Workshop on the Algorithmic Foundations of Robotics, 2000.

[11]Kuffner, J. J. J. and LaValle, S. M., "RRT-connect: An efficient approach to single-query path planning," IEEE International Conference on Robotics and Automation, 2000.

[12]Bruce, J. R. and Veloso, M., "Real-time randomized path planning for robot navigation," *Robots and Systems, 2002. IEEE/RSJ*, 2002.

[13]Lindemann, S. R. and LaValle, S. M., "Incrementally reducing dispersion by increasing Voronoi bias in RRTs," IEEE International Conference on Robotics and Automation, 2004.

[14]Atramentov, A. and LaValle, S. M., "Efficient nearest neighbor searching for motion planning," *IEEE International Conference on Robotics and Automation*, 2002, pp. 632–637. doi: 10.1109/ROBOT.2002.1013429

[15]Karaman, S. and Frazzoli, E., "Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, Vol. 30, (7), 2011, pp. 846–894.

[16]Li, Y., Littlefield, Z., and Bekris, K. E., "Asymptotically Optimal Sampling-based Kinodynamic Planning," Workshop on Algorithm Foundations of Robotics, 2014.

[17]Mettler, B., Kong, Z., Goerzen, C., and Whalley, M., "Benchmarking of Obstacle Field Navigation Algorithms for Autonomous Helicopters," American Helicopter Society Annual Forum, 2010.

[18]Karaman, S. and Frazzoli, E., "Optimal kinodynamic motion planning using incremental sampling-based methods," Proceedings of the IEEE Conference on Decision and Control, 2010.