An Evaluation of Path Planners for Guidance With Vision Based Simultaneous Localization and Mapping

Holly P. Borowski*† and Eric W. Frew‡
University of Colorado, Boulder, CO 80302

This paper discusses implementation and compares results from three different path planning methods for guidance through a cluttered environment using vision-based simultaneous localization and mapping (SLAM). The planners are implemented using sequential quadratic programming (SQP), a genetic algorithm, and a rapidly-exploring random tree (RRT). Various planning horizon lengths are explored. The planners are compared for resulting path length from a start to a goal location and back, obstacle avoidance, and computation time. A short planning horizon SQP method resulted in shortest path lengths, a medium horizon genetic algorithm produced the best obstacle avoiding trajectories, and the short horizon SQP planner required the least planning time.

I. Introduction

Motion planning in an unknown environment has applications for a wide variety of unmanned vehicles. In particular, unmanned aircraft systems (UAS) operating in cluttered environments such as forests or urban areas must sense obstacles and plan paths which avoid collision to safely arrive at a goal location.[7] Sensing, estimation, and planning tasks are often constrained by limited computational resources when operating small unmanned aircraft (UA). Vision-based simultaneous localization and mapping (SLAM) may be used for unmanned systems constrained by limited payload capacity and computational capabilities.[9] A path planner which can use SLAM estimates of vehicle and obstacle states to quickly produce feasible planned trajectories is desired.

In this scenario, the aircraft discovers obstacles by taking bearing measurements as it moves toward the goal. Because new obstacles are frequently discovered, and because obstacle observability from bearings-only vision measurements depends on UA motion relative to obstacles, planned aircraft trajectories must account for uncertainty in relative obstacle positions and be frequently updated based on newly discovered obstacles or changing position estimates for known obstacles. Receding horizon control (RHC) methods are well-suited for such applications that exhibit dynamic observability.[3–5] In RHC, a partial plan toward the goal is developed and executed for a limited finite planning horizon. Upon completing a portion of that plan, a new partial plan is developed, and this process continues until the goal has been reached. Alternately, at each planning step a full trajectory to the goal may be determined. In either case, when newly discovered obstacles or refined aircraft and obstacle state estimates render the current path infeasible, a new safe trajectory should be generated.

The path planning problem for UA performing vision based SLAM has been addressed in Refs. 4, 5 with dynamically adapted planning horizon lengths based on information rate of change. In that work, sequential quadratic programming (SQP) was used to plan aircraft trajectories. Alternatively, an evolutionary algorithm for UA path planning which incorporates a replanning scheme based on updated information has been developed and evaluated.[13] Further, rapidly-exploring random trees (RRT) have been used to successfully develop plans online for autonomous land vehicle movement.[8]

In this study, the vision based SLAM and path planning framework developed in Refs. 4, 5 is extended to examine three different path planning methods for autonomous travel through a cluttered environment. The three path planning methods are compared for their performance. Performance is evaluated by obstacle

*This material is based upon work supported by the National Science Foundation under Grant No. 1116010
†Graduate Research Assistant, Aerospace Engineering Sciences, holly.borowski@colorado.edu
‡Associate Professor, Aerospace Engineering Sciences, eric.frew@colorado.edu

avoidance and total path length. Further, computation times for each method are discussed. The three trajectory generation methods analyzed seek optimal control sequences which minimize distance to the goal and avoid obstacles using SQP, a genetic algorithm, and an RRT-based scheme. The SQP planner is based on the framework of Refs. 4,5, and the RRT based method loosely follows the closed-loop planner presented in Ref. 8. Both the SQP and genetic algorithm planners generate trajectories in an RHC fashion, whereas the RRT based method determines a full trajectory to the goal at each planning step.

This paper is organized as follows: Section II introduces notation, describes system dynamics, and defines the objective of the planners, Section III discusses each implemented method in further detail, Section IV presents the results of the three methods, and Section V discusses those results.

II. Problem Statement

The preliminaries and problem statement presented here follow the setup described in Refs. 4, 5; this implementation and analysis builds from the methods presented there.

The UA must simultaneously sense obstacles, estimate their positions, and travel through a cluttered environment from a start to a goal location and back. Aircraft and obstacle positions are assumed to be two dimensional, so that $\mathbf{p}_k^T = \begin{bmatrix} x_k & y_k \end{bmatrix}$ is the aircraft position in the inertial frame at time k and ψ_k is its heading. Additional aircraft states include an accelerometer scale factor error $\mathbf{a}$ and accelerometer and rate gyro biases $\mathbf{b}$. The ith obstacle's position in the inertial frame is $\mathbf{x}_o^{(i)} = [x_o^{(i)} \quad y_o^{(i)}]^T$. Aircraft speed is held constant, this speed and the aircraft heading angle, ψ, fully describes its velocity vector. The full state vector is given by

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_o \end{bmatrix} \tag{1}$$

where

$$\mathbf{x}_a = \begin{bmatrix} x & y & \psi & \mathbf{a}^T & \mathbf{b}^T \end{bmatrix}^T, \quad \mathbf{x}_0 = \begin{bmatrix} x_o^{(1)} & y_o^{(1)} & x_o^{(2)} & y_o^{(2)} & \cdots & x_o^{(n)} & y_o^{(n)} \end{bmatrix}, \tag{2}$$

Aircraft speed, s, is assumed constant in this work, so that the UA is controlled solely by its turn rate, u, which is limited to $|u| \leq \omega_{\max}$. Aircraft position and heading evolve according the following (exact) discretized unicycle kinematic model:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \end{bmatrix} = \mathbf{f}(\mathbf{x}_{a,k}, u_k) = \begin{bmatrix} x_k + s \cdot T_s \operatorname{sinc}(\gamma) \cos(\psi_k + \gamma) \\ y_k + s \cdot T_s \operatorname{sinc}(\gamma) \sin(\psi_k + \gamma) \\ \psi_k + u_k \cdot T_s \end{bmatrix}, \tag{3}$$

where T_s is the sample time, $\gamma = 0.5 \cdot u \cdot T_s$, and $\operatorname{sinc}(\gamma)$ is the sine cardinal function.

The estimator described in Ref. 9 is used for this study to process IMU and bearings only camera measurements, which are functions of the system state and aircraft control input,

$$\mathbf{z}_k = \begin{bmatrix} z_{\text{imu},k} \\ z_{\text{cam},k} \end{bmatrix} = \begin{bmatrix} h_{\text{imu},\,k}(\mathbf{x}_k, u_k) + v_k \\ h_{\text{cam},\,k}(\mathbf{x}_k, u_k) + w_k \end{bmatrix} \tag{4}$$

where v_k and w_k are zero mean, spacially uncorrelated, white Gaussian measurement noise. The measurements are used to to obtain aircraft state estimates $\hat{\mathbf{x}}_{a,k}$, obstacle position estimates $\hat{\mathbf{x}}_{o,k}$, and their covariances $\mathbf{P}_{aa,k}$, $\mathbf{P}_{oo,k}$, and $\mathbf{P}_{ao,k}$. Estimator dynamics evolve according to

$$\hat{\mathbf{x}}_{k+1} = \begin{bmatrix} \hat{\mathbf{x}}_{a,k+1} \\ \hat{\mathbf{x}}_{o,k+1} \end{bmatrix} = g_{\text{state}}(\hat{\mathbf{x}}_k, u_k, \mathbf{z}_{k+1}) \tag{5}$$

$$\mathbf{P}_{k+1} = \begin{bmatrix} \mathbf{P}_{aa,k+1} & \mathbf{P}_{ao,k+1} \\ \mathbf{P}_{ao,k+1} & \mathbf{P}_{oo,k+1} \end{bmatrix} = g_{\text{cov}}(\mathbf{P}_k, \hat{\mathbf{x}}_k, u_k, \mathbf{z}_{k+1}). \tag{6}$$

The functions g_{state} and g_{cov} update the state and covariance estimates based on new measurements, and are covered in further detail in Ref. 9. Estimation errors are assumed to be zero mean and normally distributed.

A block diagram of the guidance, navigation, and control framework for this system is shown in Figure 1. This paper focuses on the trajectory generation element, which is highlighted in green. Three different path planing methods are compared while all other elements of this framework are kept fixed.

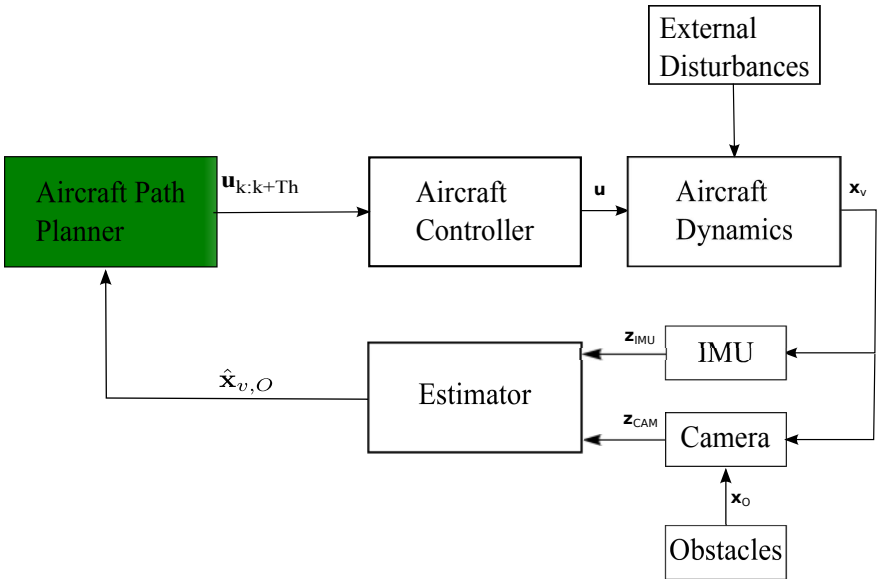


Figure 1: Guidance, Navigation, and Control System

A. Objective

The objective of this work is to find the sequence of control inputs which minimize the distance the UA must travel to move from a start to a goal location and back through a cluttered environment. The UA trajectory must be collision free. Since obstacle and aircraft positions are uncertain, collision avoidance is initially formulated as a chance-constraint on aircraft position. This stochastic constraint is reformulated as a set of deterministic constraints on the relative position between the aircraft and each obstacle.

Aircraft guidance is achieved using a receding horizon approach. At time k an open loop optimal control problem is solved over a finite time horizon of length $T_h > 0$. The resulting path is followed over the control horizon $T_c \leq T_h$ and the process is repeated at time $k + T_c$. Thus, the problem of interest here is:

Problem 1 (Chance-Constrained Receding Horizon Optimization)

$$\begin{aligned} \text{minimize} \quad & J_{rhc}(\mathbf{u}_{k:k+T_h-1}) && (7)\\ \text{subject to} \quad & \mathbf{x}_k \text{ given} && (8)\\ & \mathbf{x}_{j+1} = f(\mathbf{x}_j, u_j), \quad \forall j \in [k, \cdots, k+T_h-1] && (9)\\ & \text{Prob}(\|\mathbf{p}_j - \mathbf{x}_o^{(i)}\| \geq r_{safe}) \geq 1-\varepsilon, \quad \forall i, \forall j && (10) \end{aligned}$$

The full sequence of turn rate commands is given by $\mathbf{u}_{k:k+T_h-1} = [u_k, u_{k+1}, \cdots, u_{k+T_h-1}]^T$ and J_{rhc} denotes the objective function. The safety radius, r_{safe} is a predetermined obstacle standoff distance which accounts for both obstacle and aircraft sizes. The parameter $1 > \varepsilon > 0$ is the highest permitted collision probability at any given time. Although a collision probability of zero is clearly desired, uncertainty in obstacle positions dictates that a nonzero probabilistic constraint be set.

Because obstacle position estimate errors are assumed to be zero mean and normally distributed, an obstacle with position estimate $\hat{\mathbf{x}}_o^{(i)}$ and information matrix $\mathbf{I}_i^{-1} = \mathbf{P}_o^{(i)}$, has probability

$$\text{Prob}\left(\mathbf{x}_o^{(i)} \in A\right) = \sqrt{\frac{|\mathbf{I}_i|}{2\pi}} \int_A e^{-\frac{1}{2}(\mathbf{p}-\hat{\mathbf{x}}_o^{(i)})^T \mathbf{I}_o^{(i)} (\mathbf{p}-\hat{\mathbf{x}}_o^{(i)})} dA \tag{11}$$

of being in a given area A. By letting $A_{v,k}$ be the circle of radius r_{safe} surrounding the aircraft at time k, Eq. (10) is satisfied when $\text{Prob}\left(\mathbf{x}_o^{(i)} \in A_{v,k}\right) < \varepsilon \quad \forall i, \forall k$. However, because the integral of Eq. (11) does not have a general closed form solution, computing this probability for an arbitrary area $A \in \mathbb{R}^2$ is unrealistic for an online planning problem. Instead, a standoff region based on the probability ellipse defined by each obstacle's position and covariance estimates will be used to conservatively guarantee that Eq. (10) is met.

Consider the weighted distance metric between two points $\mathbf{p}_1$ and $\mathbf{p}_2$

$$\|\mathbf{p}_1 - \mathbf{p}_2\|_M^2 := (\mathbf{p}_1 - \mathbf{p}_2)^T M (\mathbf{p}_1 - \mathbf{p}_2) \tag{12}$$

where M is a weighting matrix and $\mathbf{x}$ is a random variable describing the true obstacle position. The set of points $\mathbf{p}$ that satisfy the equation $\|\mathbf{p} - \mathbf{p}_0\|_M^2 = r$ form an ellipse centered at position $\mathbf{p}_0$ with major and minor ellipses defined by M. When $M = I_i$ and $\mathbf{p}_0 = \hat{\mathbf{x}}_o^{(i)}$, Eq. 12 defines the Mahalanobis distance between the obstacle position estimate and position $\mathbf{p}$, and its level sets correspond to level sets of the probability density function of Eq. (11). Then, if $\mathbf{p} \sim \mathcal{N}_2\left(\hat{\mathbf{x}}_o^{(i)}, P_o^{(i)}\right)$,

$$\|\mathbf{p} - \hat{\mathbf{x}}_o^{(i)}\|_{I_i}^2 \sim \chi_2^2 \tag{13}$$

so that the probability that the obstacle falls within the ellipse defined by $r = \chi_{2,\varepsilon}^2$ is given by

$$\text{Prob}\left\{\|\mathbf{p} - \hat{\mathbf{x}}_o^{(i)}\|_{I_i}^2 \leq \chi_{2,\varepsilon}^2\right\} = 1 - \varepsilon \tag{14}$$

where $\chi_{2,\varepsilon}^2$ is inverse of the chi-square cumulative distribution function with two degrees of freedom and probability $1 - \varepsilon$.[15] Hence, requiring that

$$\|\mathbf{p}_k - \hat{\mathbf{x}}_{o,k}^{(i)}\|_{I_{i,k}}^2 \geq \chi_{2,\varepsilon}^2, \ \forall k, \ \forall i \tag{15}$$

or equivalently

$$c_\varepsilon(\mathbf{p}_k, \hat{\mathbf{x}}_{o,k}^{(i)}, I_{i,k}) := \chi_{2,\varepsilon}^2 - \|\mathbf{p}_k - \hat{\mathbf{x}}_{o,k}^{(i)}\|_{I_{i,k}}^2 \leq 0, \ \forall k, \ \forall i \tag{16}$$

where $\mathbf{p}_k$ is the aircraft position at time k, conservatively achieves the desired probability of $1 - \varepsilon$ that the aircraft will not collide at time k when $r_{\text{safe}} = 0$. Because the aircraft and obstacles have some physical size, the ellipse must then be expanded by r_{safe} to ensure that the desired constraint is met. In this case define

$$O_i^{-1} = U_i(\Sigma_i + r_{safe} \cdot I_2)V_i^T \tag{17}$$

where $P_o^{(i)} = U_i \Sigma_i V_i^T$ is the singular value decomposition of the covariance matrix and I_2 is the identity matrix.

Replacing the chance-constraint with the deterministic one leads to the optimization problem of interest:

Problem 2 (Deterministic Receding Horizon Optimization)

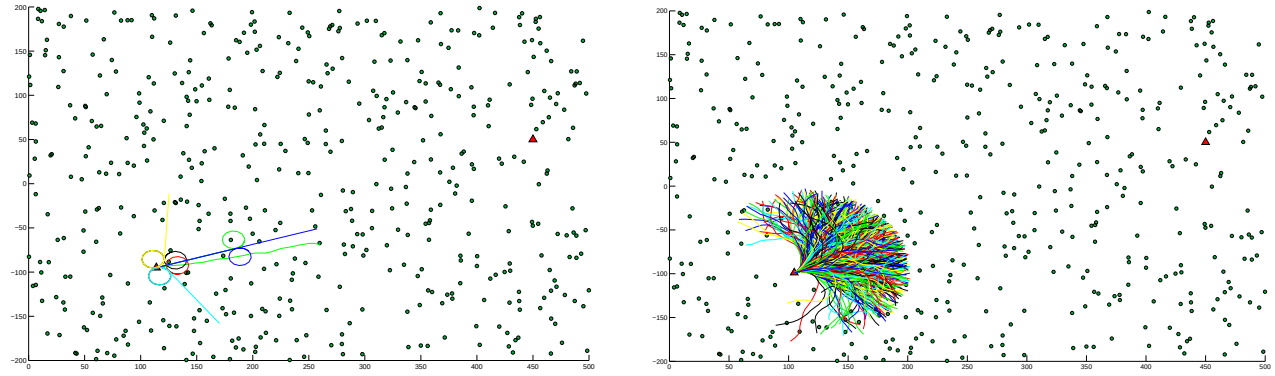
$$\begin{aligned}
\text{minimize} \quad & J_{rhc}(\mathbf{u}_{k:k+T_h-1}) && (18)\\
\text{subject to} \quad & \mathbf{x}_k \text{ given} && (19)\\
& \mathbf{x}_{j+1} = f(\mathbf{x}_j, u_j), \quad \forall j \in [k, \cdots, k + T_h - 1] && (20)\\
& c_\varepsilon(\mathbf{p}_j, \hat{\mathbf{x}}_{o,j}^{(i)}, O_{i,j}) \leq 0, \quad \forall i, \forall j && (21)
\end{aligned}$$

III. Methods

The three methods compared for trajectory generation in this study are a sequential quadratic program (SQP),[4] a genetic algorithm, and a closed-loop rapidly exploring random tree (RRT) method.[8] The first two methods use an RHC approach to generate trajectories for a fixed horizon, T_h, whereas the RRT based optimization scheme generates a full trajectory to the goal for each planning step based only on obstacles that can be reached over the time horizon T_h.

A. Planning Using Sequential Quadratic Programming

The first method uses an SQP optimizer to generate the control sequences $\mathbf{u}_{k:k+T_h-1}$. SQP is an iterative gradient based optimization method described in further detail by Nocedal and Wright,[11] and is executed here using the *fmincon* function in Matlab. The optimizer is initialized by first computing the value of the objective function for a collection of trajectories which contains 10 predetermined control sequences and 500 randomly generated allowable control sequences. The set of predetermined control sequences contains the trajectory of the previous plan, a straight trajectory, and various trajectories which combine straight segments with maximum turn rate segments. A characteristic collection of these predetermined and random trajectories used for optimizer initialization is shown in Figure 2. The lowest cost trajectory from this collection is provided to the optimizer as an initial guess.



(a) 10 Predetermined Trajectories

(b) 500 Randomly Chosen Trajectories

Figure 2: Representative collection of trajectories used to initialize the SQP planning method: the lowest cost of these will be given to the optimizer as an initial guess.

Optimization Objective

At each planning step k, SQP seeks the control sequence, $\mathbf{u}_{k:k+T_h-1}$ which minimizes the objective function

$$J_{\text{rhc}}^{(\text{SQP})}(\mathbf{u}_{k:k+T_h-1}) = J_{\text{nav}}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) + J_{\text{safe}}^{(SQP)}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) \tag{22}$$

where $\hat{\mathbf{x}}_k$ is the estimate of the aircraft state and obstacle positions at time step k[a]. Minimizing the navigation cost, J_{nav} encourages the aircraft to take a short path to the goal. The safety cost J_{safe} uses the barrier function method to transform the collision avoidance constraints into part of the objective function.

The navigation and safety cost functions follow those established in reference.[4]

Navigation Cost

The navigation cost consists of a summed quadratic cost function for the planned portion of the trajectory and an estimate of the cost to go for the remainder

$$J_{\text{nav}}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) = \sum_{j=k+1}^{k+T_h} \|\hat{\mathbf{p}}_j - \mathbf{p}_{\text{goal}}\|^2 + \tilde{J}_{\text{nav}}(\hat{\mathbf{x}}_{k+T_h}) \tag{23}$$

where $\hat{\mathbf{p}}_j$ is the position component of the aircraft state, and $\mathbf{p}_{\text{goal}}$ is the goal location. Predicted positions $\hat{\mathbf{p}}_j$ for $j = [k+1, \cdots, k+T_h]$ are computed using Eq. (3). The cost to go estimate is included to improve the stability of the RHC approach. It is given by $\tilde{J}_{\text{nav}}(\hat{\mathbf{x}}_{k+T_h})$, and assumes the aircraft moves in a straight line toward the goal from the final position in the trajectory over the distance

$$d = \left\lceil \frac{\|\hat{\mathbf{p}}_{k+T_h} - \mathbf{p}_{\text{goal}}\|}{\Delta d} \right\rceil \Delta d \tag{24}$$

which is the smallest distance greater than $\|\hat{\mathbf{p}}_{k+T_h} - \mathbf{p}_{\text{goal}}\|$ that can be traveled in an integer number of time steps, and $\Delta d = s \cdot T_s$ is the distance traveled in one time step, T_s, with aircraft speed s. The number

[a]We use the notation $J(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k)$ as reminder that the state estimate at time k is needed by the optimizer.

of time steps along this path is $N_{\text{nav}} = d/\Delta d$ so that the cost to go from time $k + T_h$ can be approximated:

$$\tilde{J}_{\text{nav}}(\hat{\mathbf{x}}_{k+T_h}) = \alpha \sum_{j=k+T_h}^{k+T_h+N_{\text{nav}}} \|\hat{\mathbf{p}}_j - \mathbf{p}_{\text{goal}}\|^2 \tag{25}$$

$$= \alpha \sum_{j=0}^{N_{\text{nav}}-1} ((N_{\text{nav}} - j)\Delta d)^2 \tag{26}$$

$$= \alpha \Delta d^2 \sum_{j=1}^{N_{\text{nav}}} j^2 \tag{27}$$

$$= \alpha \Delta d^2 \cdot \frac{N_{\text{nav}}(N_{\text{nav}} + 1)(2N_{\text{nav}} + 1)}{6} \tag{28}$$

where $\alpha \geq 1$ is a scaling factor which may be used to tune the cost to go estimate. In this work $\alpha = 1$.

Safety Cost

For the SQP method, the collision avoidance constraint discussed in Section A is accounted for by using a barrier function to apply a large penalty in the objective function to close obstacle approaches. This penalty is given by

$$J_{\text{safe}}^{(SQP)}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) = W_{\text{safe}} \sum_{j=k+1}^{k+T_h} \sum_{i=1}^{N_o} b\left(c_\varepsilon(\mathbf{p}_j, \hat{\mathbf{x}}_{o,j}^{(i)}, O_{i,j}), 0\right) \tag{29}$$

where $b(x, r)$ is the barrier function

$$b(x, r) = \begin{cases} 0 & x \leq r \\ (x - r) & x > r \end{cases}, \tag{30}$$

and N_o is the number of obstacles that have been detected at time step k. The weighting parameter, W_{safe}, sets the slope of the barrier function and is set to 10^{10} for this study.

Planning

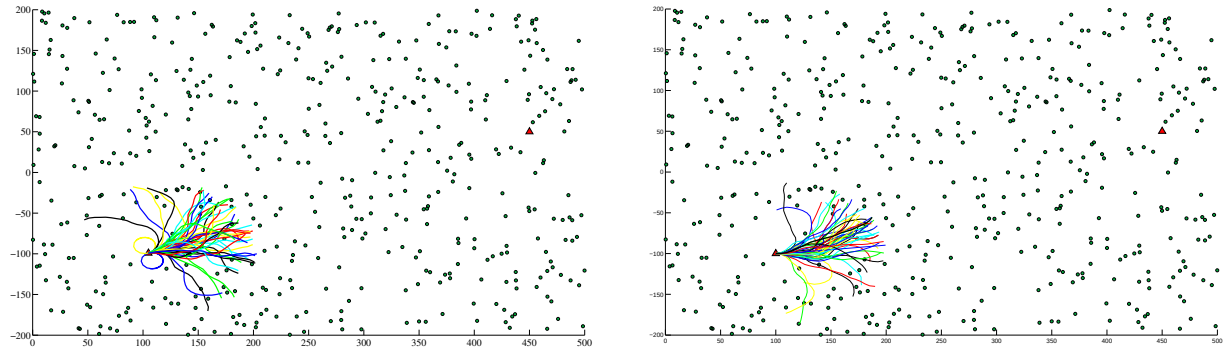
For each simulation with the SQP planner, the planning horizon is set at a fixed value, T_h. Three different length planning horizons are compared and described in Section IV. Each time a new obstacle comes into view, the current planned trajectory is checked for safety by computing J_{safe}. When either the current trajectory is found to be unsafe, i.e., $J_{\text{safe}} > 0$, or the control horizon $0 < T_c \leq T_h$ is reached, a replanning step is triggered and a new trajectory is generated. The control horizon is set as a fixed fraction of the planning horizon.

B. Genetic Algorithm Planning Method

Genetic algorithms are useful for approximating optimal solutions to problems similar to this one, in which local optima exist that may trap a gradient following optimization scheme. The theory behind genetic algorithms is further described by Sivanandam.[14]

The genetic algorithm implemented for this work begins with an initial population of size n, which is sorted according to cost. Each member of the population is a control sequence of length T_h. At each generation, selection is performed using a slotting scheme: the lowest cost control sequence receives n slots on the list, the next lowest cost control sequence receives $n - 1$ slots, and so on. In the combination and mutation phase, each new control sequence is formed by either combining two randomly chosen members from the slotted list, or combining a randomly chosen control sequence with a randomly generated one. The first type of combination is performed with 80% probability, and the second is performed with 20% probability; this is repeated until the next generation contains n members. Two control sequences are combined using a single point crossover with 50% probability, or by a linear combination of the two parent control sequences with 50% probability. Throughout all generations, the lowest cost solution produced so far is saved. When a stopping criterion is reached, this lowest cost control sequence is returned.

The initial population is seeded with one straight trajectory, i.e., $u_{k+i} = 0$ for $i = [0, \cdots, T_h - 1]$, a trajectory with all right turns at the maximum rate, i.e., $u_{k+i} = -\omega_{\max}$ for $i = [0, \cdots, T_h - 1]$, and a trajectory with all left turns at the maximum rate, i.e., $u_{k+i} = \omega_{\max}$ for $i = [0, \cdots, T_h - 1]$. The remainder of the initial population is filled by randomly generated allowable turn rate sequences. In this study, the population size is set at $n = 50$. A representative initial population and generation of trajectories is shown in Figure 3.



(a) Genetic Algorithm Initial Population

(b) Genetic Algorithm Generation

Figure 3: A representative initial population and generation generation from the genetic algorithm planning method. The dark black trajectory denotes the best solution found so far.

Optimization Objective

The implemented genetic algorithm seeks the control sequence $\mathbf{u}_{k:k+T_h-1}$ which minimizes the objective function

$$J_{\text{rhc}}^{(\text{GA})}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) = J_{\text{nav}}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) + J_{\text{safe}}^{(GA)}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k). \tag{31}$$

Here, the navigation cost function is the same as the one used for the SQP method, but the safety cost is formulated differently to take advantage of the structure of the genetic algorithm.

Safety Cost

The safety cost is given by

$$J_{\text{safe}}^{(GA)}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) = W_{\text{safe}} \sum_{j=k+1}^{k+T_h} \sum_{i=1}^{N_o} h\left(c_\varepsilon(\mathbf{p}_j, \hat{\mathbf{x}}_{o,j}^{(i)}, O_{i,j}), 0\right) \tag{32}$$

where $h(x, r)$ is the function

$$h(x,r) = \begin{cases} 0 & x \leq r \\ 1 + b(x,r) & x > r \end{cases}, \tag{33}$$

and $b(x, r)$ is the barrier function of Eq. (30). The safety weighting parameter is set such that $W_{\text{safe}} >> J_{\text{nav}}$ for paths within the bounds of the simulated environment. Here, $W_{\text{safe}} = 10^{10}$. By formulating the safety cost in this way, feasible and infeasible trajectories may be distinguished. A path violates the collision avoidance constraint will have a cost larger than W_{safe} and is marked infeasible. The barrier cost function, b, of Eq. (30) is added because the aircraft occasionally must plan from a location too close to an obstacle due to estimation errors; this is discussed in further detail below.

Stopping Criterion

For each planning cycle, the genetic algorithm continues until a stopping criterion is reached. Here, the stopping criterion is achieved when the lowest cost trajectory satisfies $J_{\text{rhc}}(\mathbf{u}_{k:k+T_h-1}|\hat{\mathbf{x}}_k) < W_{\text{safe}}$ and either

some predetermined number of generations have been generated, or some predetermined amount of time has elapsed. The first condition ensures the trajectory is feasible, according to estimated aircraft and obstacle positions. Occasionally, due to estimation errors, the aircraft must plan from a position that is too close to an obstacle; in this situation a trajectory with cost less than W_{safe} does not exist. For such situations, a second time threshold is established; if the planner reaches this point and still has not found a feasible trajectory, it returns the lowest cost trajectory found so far. The second term in the safety cost function of Eq. (32) encourages the aircraft to move away from the obstacle it is close to when this situation occurs.

Planning

As for the SQP planner, the genetic algorithm planning horizon T_h is set at a fixed value for each simulation, and two different length planning horizons are compared. Each time a new obstacle comes into view, J_{safe} is recomputed for the current trajectory. A replan is triggered if the current trajectory is found to be unsafe, or when the end of the control horizon T_c is reached.

C. RRT based Planning Method

The simulated RRT planning method loosely follows the Closed Loop RRT (CL-RRT) method described in Ref. 8. This planner was developed by modifying open source RRT code developed by Karaman and Frazzoli[6] which had been modified by Otte for speedup using multiple forests of random trees.[12] For consistency with the other two planning methods, this RRT-based method returns a plan in the form of an open loop sequence of turn rate commands. The aircraft executes this control sequence, checking a fixed portion of the planned trajectory for safety each time a previously undetected obstacle comes into camera range.

Expanding the tree

For this planner, a tree expansion method which simulates aircraft control replaces the standard RRT expansion process.[10] This controller simulating expansion algorithm is presented in Algorithm 1, and is a modified version of the closed loop tree expansion algorithm.[8] First, Dubins path length, not Euclidean distance as in standard RRT expansion, is used to determine the nearest node in the tree to a sampled state.[8] Then, a controlled trajectory is simulated from the nearest node using the sampled state as the reference input command. This path is checked for feasibility and if the simulated trajectory is obstacle free, the sampled state is added to the tree as a reference command.

Objective function

The cost of a path segment between two nodes of a tree is given by its distance. Letting $\mathbf{X}_{n_1,n_2} = \{\mathbf{x}_{v,0}, \mathbf{x}_{v,1}, \ldots, \mathbf{x}_{v,K}\}$ be the aircraft trajectory connecting nodes n_1 and n_2, the segment's cost is

$$J^{(\text{RRT})}_{\text{path segment}}(\mathbf{X}_{0:K}) = \sum_{j=1}^{K} \|\mathbf{p}_{v,j} - \mathbf{p}_{v,j-1}\|. \tag{34}$$

The cost of a full trajectory is the sum of the lengths of its segments. For trajectories which do not fully extend to the goal, the cost to go is estimated by the Dubins path distance metric described below. Otherwise, the cost to go is zero. In this implementation, the planner does not return a trajectory until a full feasible path to the goal has been determined.

Distance Metric

Because the distance traveled from a node to a sampled state depends on the node's heading and minimum turn radius, a metric that approximates the distance along the controlled trajectory between the two states is desired. Although the controller does not follow a Dubins path, the Dubins path length provides a estimate which may be used to find the nearest node to a given sample, or to approximate the cost to go from a leaf in the tree to the goal. Minimal length Dubins paths are discussed in detail in Ref. 2, and Ref. 8 provides a formula for computing Dubins path lengths.

Algorithm 1 RRT expansion

Sample a reference position, $\mathbf{x}_{\text{samp}} = \begin{bmatrix} x_{\text{samp}} & y_{\text{samp}} \end{bmatrix}$.
Sample a state inside the goal region with some probability, $q > 0$.
Sort the nodes in the tree by their Dubins path length, l, from the sampled state. Let $n_1, n_2, \ldots, n_k$ be the ordered list of nodes from smallest to largest associated length, l, to the state s
for $i = 1 \rightarrow k$ **do**
  Initialize $\mathbf{x}_{\text{next}} = \mathbf{x}_{n_i}$ and $\mathbf{p}_{\text{next}} = \begin{bmatrix} x_{n_i} & y_{n_i} \end{bmatrix}^T$, where $\mathbf{x}_{n_i} = \begin{bmatrix} x_{n_i} & y_{n_i} & \psi_{n_i} \end{bmatrix}^T$ is the aircraft state at node n_i.
  while $\|\mathbf{p}_{\text{next}} - \mathbf{x}_{\text{samp}}\| < tol$ **do**
    Update $x_{\text{curr}} = x_{\text{next}}$
    Using $\mathbf{x}_{\text{samp}}$ as a reference waypoint, simulate the waypoint following controller at rate $1/T_s$ to obtain the next aircraft state, $\mathbf{x}_{\text{next}} = \begin{bmatrix} \mathbf{p}_{\text{next}}^T & \psi_{\text{next}} \end{bmatrix}^T$ along the trajectory, $\mathbf{X}_{n,s}$.
    if the segment from $\mathbf{p}_{\text{curr}}$ to $\mathbf{p}_{\text{next}}$ is obstacle free **then**
      Add $\mathbf{x}_{\text{next}}$ to $\mathbf{X}_{n,s}$
    else
      break.
    end if
  end while
  if the trajectory is obstacle free **then**
    Add the final state of the simulated trajectory, $\mathbf{x}_{\text{next}}$ as a node in the tree, connected by an edge to n_i
    Store the reference command $\mathbf{p}_{\text{ref}} = s$ as the input to the controller which resulted in the simulated trajectory from n_i to $\mathbf{x}_{\text{next}}$.
    if n_i is not in the goal region **then**
      Estimate the cost to go by its Dubins path length to the goal.
    else
      The cost to go from n_i is 0.
    end if
    Break.
  end if
end for

Obstacle Avoidance

Obstacle avoidance is performed by requiring that Eq. (21) is satisfied for every position in the trajectory to a candidate node before adding it to the tree.

Aircraft Control

A simple waypoint following aircraft controller is implemented and described in Algorithm 2. This controller is used in both simulating trajectories for adding nodes to the tree, and for producing full trajectory plans in the form of turn rate command sequences. During flight toward a waypoint, if the angle ν between the aircraft's current heading and the heading to that waypoint is larger than a threshold angle, set here to $\pi/4$, then the aircraft is commanded at maximum turn rate toward the waypoint. Otherwise, commanded turn rate is proportional to the angle ν. This threshold angle was determined using heuristics; both larger and smaller values often resulted in paths which did not converge to the desired waypoint.

When attempting to connect a sampled state $\mathbf{x}_{\text{samp}}$ to the tree, the controller is simulated from the node in the tree with $W = \{\mathbf{x}_{\text{samp}}\}$. After the RRT algorithm is complete, and the lowest cost path has been found, the same aircraft controller is used to produce the desired planned turn rate sequence. In this case, W is a sequence of waypoints that lead from the aircraft's current location to the goal.

In this study, the distance tolerance *tol* before switching to the next waypoint is set to 10 m. At the goal location, this tolerance is reduced to 4 m. An example of the RRT expansion and the resulting waypoint following path from the start to the goal is show in Figure 4. For this example, a full set of exact obstacle positions were given to the planner; in aircraft planning simulations, the planner instead uses estimated positions of known obstacles to determine a trajectory. Nodes in the figure are waypoints which the aircraft

Algorithm 2 Aircraft Control

Given the current aircraft state $\mathbf{x}_{\text{curr}} = [x_{\text{curr}}, y_{\text{curr}}, \psi_{\text{curr}}]^T$ and a list of waypoints to follow $W = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$
Initialize the trajectory $\mathbf{X} = \emptyset$.
for $i = 1 \rightarrow n$ **do**
  while $\|\mathbf{p}_{\text{curr}} - \mathbf{w}_i\| < tol$ **do**
    Compute the heading angle, γ, from $\mathbf{x}_{\text{curr}}$ to $\mathbf{w}_i$: $\gamma = \text{atan2}(y_{w_i} - y_{\text{curr}}, x_{w_i} - x_{\text{curr}})$.
    Set $\nu = \text{fmod}(\gamma - \psi_{\text{curr}}, 2\pi)$
    if $7\pi/4 > |\nu| > \pi/4$ **then**
      Turn at $\omega_{\max}$ toward $\mathbf{w}_i$
    else
      Turn at rate $\frac{\omega_{\max}|\nu|}{\pi}$ toward $\mathbf{w}_i$.
    end if
  end while
end for

controller uses as reference commands to fly a trajectory toward the goal.

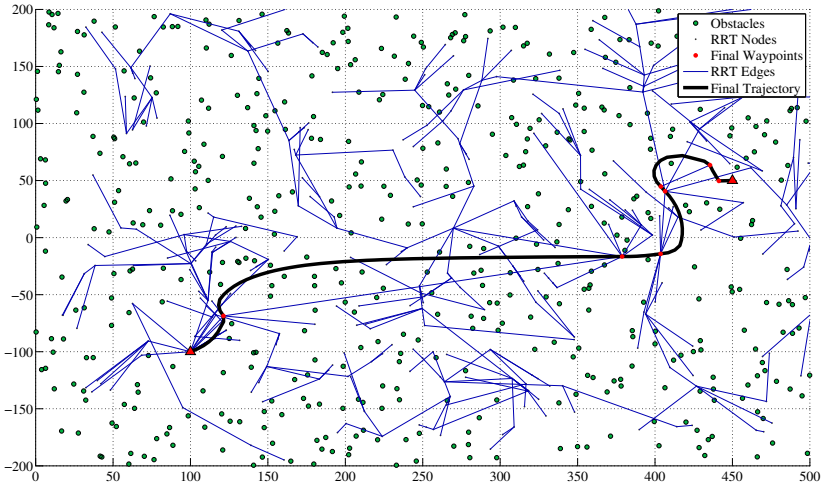


Figure 4: RRT Expansion and Waypoint Following

Execution

At each planning step this RRT-based planner produces a sequence of waypoints for the aircraft to follow from its current state to the goal. A sequence of turn rate commands is then generated according to Algorithm 2. With each new vision measurement, a fixed portion of the current path is checked for safety as in Section A. The entire trajectory is not checked, because as obstacles are discovered this would result in a high replanning rate. In particular, the trajectory is checked over the control horizon only and a new plan is initiated if $J_{\text{safe}}^{(SQP)}(\mathbf{u}_{k:k+T_c}|\hat{\mathbf{x}}_k) > 0$, where the safety cost is given by Eq. (29). Otherwise, a new trajectory is generated when a control horizon, T_c, is reached.

For comparison with other methods, a version of RHC is implemented using the RRT based method. Each generated plan is a control sequence extending to the goal from the current location. However, the set of obstacles $O^{(RRT)}$ considered by the planner are only those obstacles that could be reached over the planning horizon:

$$O^{(RRT)} = \{O_i \mid \|\mathbf{p}_k - \mathbf{x}_o^{(i)}\| \leq T_h * s\} \tag{35}$$

When replanning, the new trajectory is not determined entirely from scratch; before randomly sampling

states, an attempt is made to add waypoints from the previous best path to the tree. This takes advantage of the fact that a portion of the previous trajectory may still be feasible, even if a later portion results in collision with an obstacle.

IV. Simulation Results

A. Simulation Setup

One simulation consists of an out an back flight from a start to a goal location through a cluttered environment, as shown in Figures 5a - 5b. During the flight toward the goal, obstacles are discovered for the first time. On the return trip, the vehicle retains knowledge of discovered obstacle position estimates, and may further refine those estimates and use the prior knowledge for planning purposes. Simulation parameters are given in Table 1.

Table 1: Simulation Parameters

Initial aircraft state	$\mathbf{x} = \begin{bmatrix}100 & -100 & 0\end{bmatrix}^T$
Goal location	$\mathbf{p}_{\text{goal}} = \begin{bmatrix}450 & 50\end{bmatrix}^T$
Goal region	4 m circle surrounding the goal
Camera sensing range	$R_c = 100$ m
Camera field of view	FOV $= 100°$
Aircraft speed	$s = 10$ m/s
Planner sample rate	$T_s = 2\ Hz$
Vision measurement rate	$T_v = 2\ Hz$
IMU measurement rate	$T_{\text{IMU}} = 20\ Hz$
Planning horizon	$T_h = \frac{w_{\text{plan}} R_c}{s}$
Control horizon	$T_c = 0.2 \cdot T_h\ s$
Maximum turn rate	$\omega_{\max} = 1$ rad
Obstacle safety radius	$r_{\text{safe}} = 4$ m
Safety constraint parameter	$\varepsilon = 0.00001$
Bearing measurement noise error standard deviation	0.01 rad
Accelerometer scale factor error standard deviations	$\mathbf{a} = \begin{bmatrix}0.001\text{ m} & 0.001\text{ m}\end{bmatrix}^T$
Accelerometer and rate gyro bias standard deviations,	$\mathbf{b}\ \begin{bmatrix}0.001\text{ m} & 0.001\text{ m} & 0.0004\text{ rad}\end{bmatrix}^T$

Note that the planning sample time T_s is 2 Hz, but the simulation itself is integrated at the 20 Hz IMU measurement rate. Uncertainty is introduced into the simulation with bearing measurement error, accelerometer error, and gyro error. The aircraft must navigate through 500 obstacles at randomly generated positions within a 500 m by 400 m region. Obstacle positions are generated once and then held fixed over all simulations. For the SQP method and the genetic algorithm, short and medium RHC planning horizons, and a three-step lookahead over a long control horizon are implemented. The three-step lookahead method is further described in Ref 1. This method allows for a longer control horizon while taking advantage of the lower computation time required to create a plan that simulates only the next three control steps to determine each control rate command over a horizon of length T_c. The control horizon T_c for the SQP and genetic algorithm methods is 20% of the short and medium planning horizons; a replan is triggered when the end of the control horizon is reached, or when the current plan is found to be unsafe due to new or refined obstacle location estimates. In the three-step lookahead method, a plan of length T_c is generated, where T_c is a predetermined fixed control horizon length. A horizon length of 10 seconds corresponds a plan for the amount of time it would take the aircraft to travel in a straight line at its current heading to the edge of its camera range.

Fifty Monte Carlo simulations using each of the following methods for planning paths toward the goal were performed:

1. Minimization of the RHC objective function of Eq. (22) using SQP with a three-step lookahead, and control horizon $T_c = 3$ s. Abbreviated "Three step SQP"

2. Minimization of the RHC objective function of Eq. (22) using SQP with a long length planning horizon, $T_h = 10$ s, $T_c = 2$ s. Abbreviated "Long SQP"

3. Minimization of the RHC objective function of Eq. (22) using SQP with a short planning horizon, $T_h = 5$ s, $T_c = 1$ s. Abbreviated "Short SQP"

4. Minimization of the RHC objective function of Eq. (31) using a genetic algorithm with a three-step lookahead, and control horizon $T_c = 3$ s. Abbreviated "Three step GA"

5. Minimization of the RHC objective function of Eq. (31) using GA with a long length planning horizon, $T_h = 10$ s, $T_c = 2$ s. Abbreviated "Long GA"

6. Minimization of the RHC objective function of Eq. (31) using SQP with a short planning horizon, $T_h = 5$ s, $T_c = 1$ s. Abbreviated "Short GA"

7. Minimization of path distance of Eq. (34) to the goal using an RRT based method and including only obstacles which the simulated aircraft can reach within 5 seconds in each plan, $T_c = 1$ s. Abbreviated "Short RRT"

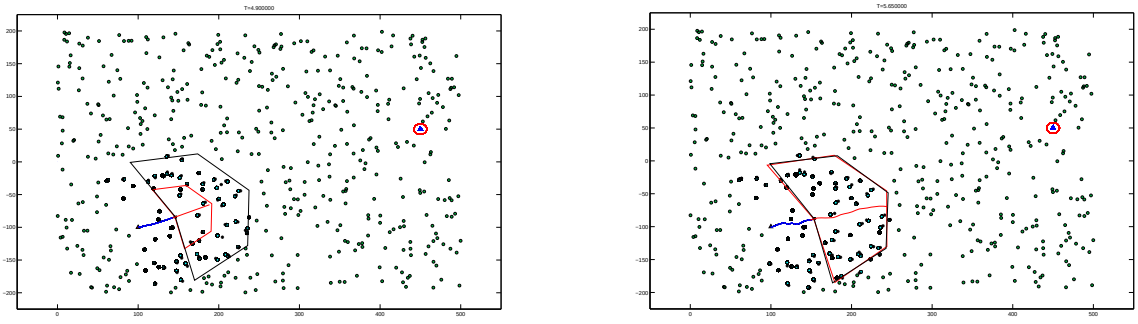
8. Minimization of path distance of Eq. (34) to the goal using an RRT based method and including only obstacles which the simulated aircraft can reach within 7.5 seconds in each plan, $T_c = 2$ s.. Abbreviated "Med RRT"

The short horizon genetic algorithm was given a maximum of five seconds per plan or 50 generations (whichever occurs first, provided the best discovered trajectory is feasible). The medium horizon genetic algorithm planner was given a maximum of 10 seconds or 100 generations.

The short RRT based planner was allotted 20 seconds per plan to determine a feasible path to the goal location, and the medium RRT based planner was allotted 30 seconds. A long horizon RRT based planner was also attempted. However, because the RRT based method treats obstacle probability ellipses as hard constraints, the large probability ellipses surrounding objects at the edge of its sensing horizon frequently made it difficult or impossible to find a feasible trajectory. The SQP planner was able to overcome this issue because it does not treat obstacles as hard constraints. Similarly, although the GA planner is designed to allow a path passing within the probability ellipse surrounding an obstacle if it has not discovered a feasible one in a reasonable amount of time.

Figure 5 depicts short and long planning horizons. In each figure, the blue line represents the traveled path, and the red line is the planned path. Undiscovered obstacles are green circles, blue circles are the actual positions of discovered obstacles, and black stars are their estimated positions. The planning horizon length is shown by the region bordered in red, and the camera sensing range is given by the region bordered in black. The goal center is a triangle with the surrounding goal region depicted by a red circle.

Figure 5a depicts a planned trajectory for a short horizon RHC planner, as used in the SQP and genetic algorithm methods. This horizon extends to half the camera range for a straight trajectory from the current aircraft location. Figure 5b depicts a long RHC planning horizon and planned trajectory characteristic of the SQP and genetic algorithm methods. This horizon extends to the camera range for a straight trajectory from the current aircraft location.



(a) Short Planning Horizon

(b) Long Planning Horizon

Figure 5: Short and long planning horizons

B. Performance Metrics

Performance is assessed with regards to final path length from the start to goal location and back, closest obstacle approaches, and percentage of path within r_{safe} of an obstacle.

The percentage of a path within r_{safe} of an obstacle depends both on estimator accuracy and aircraft movement. It provides a measure of how well the chance constrained optimization problem of Equation 10 is met via the deterministic optimization of Problem 2. With a safety constraint parameter of $\varepsilon = 0.00001$, the established safety constraint guarantees a probability of safety of

$$P_{\text{safe}} \geq (1-\varepsilon)^{k_{\text{final}} N_o}$$

where k_{final} is the final discretized time step and N_o is the total number of obstacles. Although ε is small, the magnitude of $k_{\text{final}} \cdot N_o$ is between 10^5 and 10^6, depending on total path length, so that the guaranteed value of P_{safe} is nearly 0. To see that the formulated constraint actually provides a reasonable guarantee of safety, it is assumed that the probability of colliding with an obstacle outside of the aircraft's sensing range is very close to zero. Because obstacles are uniformly distributed in the two dimensional world, the quantity $k_{\text{final}} \cdot N_o$ may be effectively replaced by $k_{\text{final}} \cdot N_o'$, where $N_o' = N_o \cdot \frac{A_{\text{sense}}}{A_{\text{total}}}$ is the number of obstacles expected to be within the aircraft's sensing radius at a given time. This raises the effective safety guarantee to a value near 0.65, again depending on trajectory length. The conservativeness built into the safety constraint formulation further raises the actual probability of safety; the metric reporting the percentage of a path within r_{safe} of an obstacle reflects this.

Next, computation performance is assessed with regards to number of replans per simulation second and planning time per unit path length. A planning time assessment is not included for the RRT based planner; this planner was written in the C programming language and called from Matlab; thus its planning time cannot be directly compared with the other two methods. However, because the RRT based planner was given a fixed amount of time to determine each required plan, this and the number of necessary replans provide some measure of its computational performance.

In the following results, the top performers for each metric are shaded in light grey.

C. Aircraft Performance Results

Average path lengths for aircraft guidance performed with each planner are shown in Table 2. The return trip segment includes the aircraft turn-around upon reaching the goal location so is expected to be longer than the outbound segment. On average, the short horizon SQP planner resulted in the shortest path lengths for both outbound and return legs. The long horizon genetic algorithm planner resulted in the longest average outbound and return path segments, and was also least consistent, reflected in its large path length standard deviation.

Table 2: Aircraft Path Lengths for Each Planner

Aircraft Path Length: Avg. (St. Dev.) in meters		
	Outbound Leg	**Return Leg**
Short GA	389.5 (34.8)	420.2 (37.6)
Long GA	719.3 (383.6)	780.6 (378.2)
3 Step GA	375.5 (5.1)	398.9 (9.8)
Short SQP	374.5 (4.3)	393.0 (10.2)
Long SQP	451.1 (51.4)	423.7 (34.9)
3 Step SQP	375.2 (4.4)	395.4 (6.7)
Short RRT	435.3 (38.1)	512.6 (69.8)
Med RRT	448.3 (65.5)	515.8 (91.9)

Next, Table 3 shows the closest obstacle approaches for each planner. For both outbound and return legs, the long horizon genetic algorithm paths keep the largest minimum standoff distance. In contrast both three step planners exhibited poor obstacle avoidance performance, with average closest obstacle approaches well within r_{safe}.

Table 3: Closest Obstacle Approaches for Each Planner

Closest Obstacle Approach: Avg. (St. Dev.) in meters		
	Outbound Leg	**Return Leg**
Short GA	4.2 (0.8)	4.7 (0.3)
Long GA	4.5 (1.2)	4.8 (1.0)
3 Step GA	2.4 (1.2)	3.6 (0.9)
Short SQP	3.9 (0.7)	4.5 (0.2)
Long SQP	4.2 (0.8)	4.2 (1.2)
3 Step SQP	2.7 (1.2)	3.4 (1.0)
Short RRT	4.3 (0.7)	4.4 (1.1)
Med RRT	4.4 (0.7)	4.2 (1.1)

Table 4 shows average percentages of traveled paths that come within r_{safe} of an obstacle. The three step SQP planner resulted in the most close approaches in the outbound leg, and the three step genetic algorithm resulted in the most close approaches in the return leg. The long horizon genetic algorithm and both RRT planners tied for the fewest close approaches on the outbound leg. The short horizon genetic algorithm and short horizon SQP planners produced paths which stayed outside of r_{safe} most on both outbound and return legs.

Table 4: Percent of Path Within r_{barrier} of an Obstacle for Each Planner

Percent of Path Within r_{barrier} Avg. (St. Dev.)		
	Outbound Leg	**Return Leg**
Short GA	0.2 (0.3) %	0.0 (0.0) %
Long GA	0.1 (0.2) %	0.03 (0.1) %
3 Step GA	1.5 (1.0) %	0.4 (0.4) %
Short SQP	0.3 (0.4) %	0.0 (0.0) %
Long SQP	0.2 (0.3) %	0.2 (0.6) %
3 Step SQP	1.1 (0.9) %	0.5 (0.5) %
Short RRT	0.1 (0.3) %	0.1 (0.4) %
Med RRT	0.1 (0.2) %	0.2 (0.5) %

D. Computational Performance Results

Average replanning frequencies for each planner are shown in Table 5. Replans are triggered when the end of the control horizon is reached, or when an updated obstacle position renders the current planned trajectory unsafe. Obstacle position estimates are updated when new obstacles are discovered, or when new measurements of known obstacles are taken. The "expected" column of Table 5 gives the replanning frequency if replans only occured at the end of each control horizon. The final column in Table 5 shows how much more often replanning occurs compared to the expected rate; this is expressed as a percentage of the expected rate.

Higher percentages in the final column reflect frequent safety replans, i.e., replans because the current planned trajectory is infeasible. In a sense, this indicates wasted planning effort, as it corresponds to plans through portions of the environment which the UA knows little about.

The SQP planners and the short horizon RRT planner have the lowest average replanning frequency relative to their expected rates. The long horizon genetic algorithm had the largest replanning frequency relative to its expected rate.

Table 5: Number of Replans for Each Planner

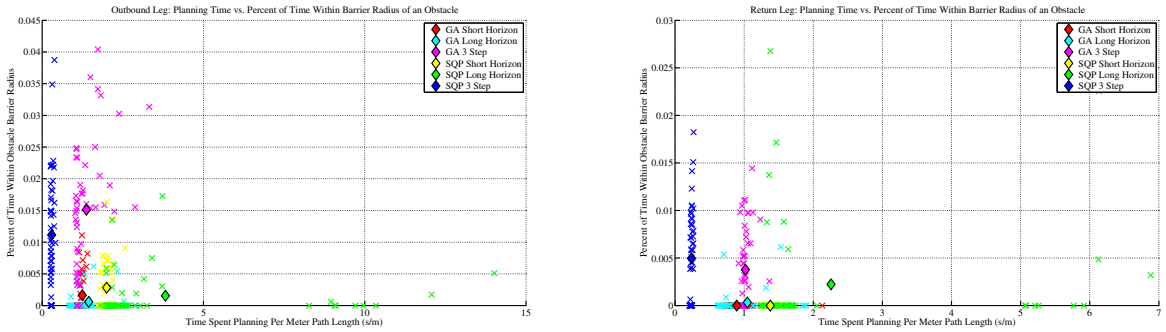
Number of Replans Per Simulation Second: Avg. (St. Dev.)			
	Expected	**Avg (St. Dev.)**	**% Above Expected**
Short GA	1.0	1.5 (0.07)	50%
Long GA	0.5	0.8 (0.12)	60%
3 Step GA	0.3	0.4 (0.02)	33%
Short SQP	1.0	1.0 (0.01)	0%
Long SQP	0.5	0.5 (0.02)	0%
3 Step SQP	0.3	0.4 (0.02)	33%
Short RRT	1.0	1.0 (0.01)	0%
Med RRT	0.67	0.7 (0.02)	5%

Next, Table 6 shows the time spent planning for each meter of path length for the various planners. For both outbound and return legs, the three step SQP planner spent the least amount of time planning per meter. In both directions, the long horizon SQP planner spent the most time planning per meter. Note that the genetic algorithm based planners are both allotted a certain amount of time for planning, whereas the SQP planners continue until minimizing the supplied cost function.

Table 6: Time Spent Planning Per Meter Path Length for Each Planner

Plan Time Per Meter Path Length in seconds per meter: Avg. (St. Dev.)		
	Outbound Leg	**Return Leg**
Short GA	1.3 (3.1)	0.9 (2.7)
Long GA	1.4 (1.3)	0.9 (0.7)
3 Step GA	1.4 (36.8)	1.0 (3.1)
Short SQP	2.0 (14.2)	1.4 (3.8)
Long SQP	3.8 (26.5)	2.3 (20.3)
3 Step SQP	0.3 (3.2)	0.2 (0.9)

Figures 6 - 7 consolidate these metrics and show their distribution for each planner. In Figure 6, computation time per meter path length is plotted against percent of time the aircraft spends within r_{safe} of an obstacle. The yellow, green, and blue diamonds represent averages for the SQP based planners. The red, blue, and magenta diamonds show averages for the genetic algorithm based planner. There does not appear to be correlation between computation time and obstacle avoidance performance in either case. These observations are further discussed in Section V.

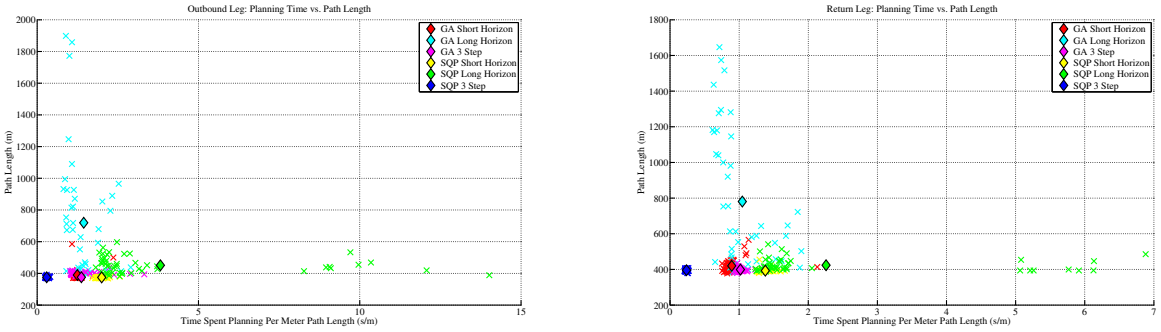


(a) Outbound Leg

(b) Return Leg

Figure 6: Computation Time per Meter Path Length vs. Percent of Time Within Barrier Radius of an Obstacle: diamonds represent averages over all simulations, and x's represent individual simulation results

Figure 7 depicts time spent planning per meter path length vs. total path length for each planner. The three step and short horizon planners resulted in the least variation in these two metrics, and also the best performance with regards to both. The long horizon SQP had relatively large variation in planning time, and the genetic algorithm planner had large variations in path length.



(a) Outbound Leg

(b) Return Leg

Figure 7: Computation Time per Meter Path Length vs. Total Path Length: diamonds represent averages over all simulations, and x's represent individual simulation results

V. Discussion

Obstacle avoidance performance

Obstacle avoidance performance is influenced by the coupling between three elements: the aircraft's path relative to obstacles, estimator performance, and the planner's ability to produce paths which avoid obstacles. Because obstacle positions are estimated using bearings only vision measurements, lateral movement relative to obstacles results in more accurate estimates of their positions. Note that in nearly all planners, obstacle avoidance improves on the return leg because obstacles are seen for a second time, so uncertainty in their position estimates is further reduced. In all cases, obstacle avoidance performance exceeded the guarantees of the established deterministic constraint discussed in section IV.B, reflecting the built in conservativeness of this formulation.

Although the short horizon SQP planner resulted in the shortest average path lengths, its obstacle avoidance performance was worse than other planners. On average, the short horizon SQP planner produced paths which passed within r_{safe} of an obstacle at least once during the flight. Its tendency to create direct paths toward the goal, and hence less lateral motion with respect to obstacles, may hurt its ability to produce obstacle avoiding paths. Planner performance also contributes to the aircraft's ability to avoid obstacles. For the SQP planners, paths within r_{safe} are assigned a cost which ramps up quickly as distance to the obstacle decreases. Paths within this barrier radius are not explicitly forbidden. These planners may allow the aircraft to travel slightly inside the estimated obstacle barrier radius if it results in a more direct path toward the goal.

Both three-step planners also performed poorly with respect to obstacle avoidance. The short-sightedness of these planners caused them to produce paths which passed within r_{safe} of obstacles frequently. Furthermore, paths created by the three step lookahead planners also resulted in a collision (pass well within r_{safe}) at least once during an average flight.

In contrast to the SQP planners, the genetic algorithm planner has a safety cost function designed to explicitly avoid travel inside the established probability ellipse surrounding each obstacle's estimated position unless a feasible trajectory cannot be found within a reasonable amount of time. The fact that the genetic algorithm planners resulted in paths which spent less time within r_{safe} of an obstacle than their SQP planner counterparts suggests that the genetic algorithm and its safety cost function formulation may be better suited for producing obstacle avoiding trajectories. The RRT planner is formulated to treat obstacles as constraints; on average RRT generated paths also spent less time inside obstacle barrier radii and had larger minimum obstacle standoff distances than SQP generated paths.

Long horizon planner path lengths and the impact of sampling strategy on performance

The long horizon SQP and the long horizon genetic algorithm planners both resulted in significantly higher average path lengths than their short horizon counterparts, indicating frequent indirect paths taken toward

the goal. Intuitively, performing a longer lookahead is expected to result in more direct paths and improved obstacle avoidance, yet the short horizon genetic algorithm performed better than long horizon planners in path length, and similarly in obstacle avoidance.

The long horizon genetic algorithm planner produced the largest average path lengths of all the planners. This may be a combination of the fact that the genetic algorithm treats obstacles and their surrounding probability ellipses as constraints, and the fact that the obstacles at the edge of its sensing radius have larger associated uncertainty. Thus, the long horizon genetic algorithm favors highly circuitous paths which avoid this uncertain region.

The long horizon SQP planner also produced relatively long paths toward the goal. Because the SQP method follows a gradient to minimize the objective function of Eq. (22), and because the objective function is highly non-convex due to randomly placed obstacles within a long planning horizon, optimizer initialization has a significant impact on planner performance. By initializing the optimizer with a value that is not close enough to the global minimum, the planner may instead return a control sequence which is a local optimum of the provided cost function. Intuitively, the tendency to get stuck in local minima corresponds to difficulty in "jumping over" obstacles, i.e., finding a lower cost trajectory on one side of an obstacle when initialized by a control sequence on the other side of that obstacle.

Recall that the SQP planner is initialized by providing the minimum cost control sequence from a collection of 10 predetermined control sequences and 500 randomly generated control sequences, shown in Figure 2. The randomly generated trajectories are taken by a uniform random sampling of the input space; however, this does not equate to a uniform random sampling of the output space, i.e., aircraft trajectories in two dimensional space. This bias is apparent in Figure 2b, and may result in inadequate SQP planner initialization for longer horizon planners in which the objective function contains more local minima than the shorter horizon counterparts.

Although the short horizon SQP initializer and genetic algorithm planner both use the same sampling method as their longer horizon counterparts, there are fewer obstacles to avoid within their short lookahead. Thus, the short horizon SQP planner is less likely to become stuck in a local minimum, as there are fewer local minima in the short horizon cost function to begin with. Moreover, a trajectory which heads straight for the goal, or mostly straight for the goal with relatively few turns to avoid obstacles, is more likely to be feasible in a short horizon plan. In contrast, a long horizon plan will often involve frequent turning to circumvent obstacles. This fact may allow the short horizon genetic algorithm planner to find more direct paths toward the goal than the longer horizon genetic algorithm planner. Recall that a straight trajectory is included in the genetic algorithm's initial population at each planning step. Other short horizon trajectories with only a few turns are not significantly different from this initial trajectory. If such a trajectory is safe, the genetic algorithm is likely to find it since it is similar to a member of its initial population.

Unlike the genetic algorithm and SQP planners, the RRT based planner draws samples directly from the output space, i.e., it samples waypoints in two dimensional space. It should then do a better job of exploring the space of positions where the aircraft may travel. However, this RRT based planner also produced longer paths toward the goal than the short horizon and three step planners. The waypoint follower used for this study may enhance the tendency to produce indirect paths. In this method, there is no restriction on the heading at which the aircraft arrives at each waypoint. Thus, the aircraft may arrive at a waypoint with a heading that is significantly different from the heading toward to goal or the next waypoint. By using a waypoint follower which enforces headings at waypoint arrival which are closer to the heading toward the goal, the RRT based planner performance may be improved.

Overall performance

Overall, the short horizon genetic algorithm average path lengths and average time within obstacle safety barriers was competitive with the best performing planners, while requiring the least computation time per unit path length of those planners with acceptable obstacle avoidance. Thus, among these planners tested, it is a good choice when computation time and resources are limited. The short horizon SQP planner also produced short paths to the goal but is less desirable because of its poor obstacle avoidance performance.

VI. Conclusion

This paper has presented results from implementation of three different path planners for guidance in a cluttered environment with vision based SLAM. The three planners analyzed and compared are an SQP based planner, a genetic algorithm planner, and an RRT based planner. Planning horizon lengths were varied and compared for the SQP and genetic algorithm planners. The short horizon SQP planner produced the shortest path lengths on average, but had poor obstacle avoidance performance. The long horizon genetic algorithm planner resulted in the best average obstacle avoidance, but produced the longest paths and required the second longest computation time. The short horizon genetic algorithm planner performed well in all categories, with shorter average path lengths than other planners with comparably good obstacle avoidance performance.

Extensions to this work will include further consideration of obstacle and aircraft position uncertainty for path planning. This may be accomplished by adding an information cost to the objective function which explicitly encourages the aircraft to move in a way that improves obstacle observability and thus obstacle position estimates.

References

[1]D. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334, 2005.

[2]L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497 – 516, 1957.

[3]Eric W Frew. Receding Horizon Control Using Random Search for UAV Navigation with Passive, Non-cooperative Sensing. *AIAA Guidance, Navigation and Control Conference and Exhibit*, (August), 2005.

[4]Eric W. Frew and Jack W. Langelaan. Adaptive planning horizon based on information velocity for vision-based navigation. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, number August, 2007.

[5]Eric W. Frew, Jack W. Langelaan, and Sungmoon Joo. Adaptive receding horizon control for vision-based navigation of small unmanned aircraft. In *American Control Conference*, 2006.

[6]Sertac Karaman and Emilio Frazzoli. http://sertac.scripts.mit.edu/rrtstar/category/software/.

[7]Sertac Karaman and Emilio Frazzoli. High-speed Flight in an Ergodic Forest. *Manuscript submitted to the IEEE Transactions on Robotics*, pages 1–17, 2012.

[8]Yoshiaki Kuwata, Sertac Karaman, Justin Teo, Emilio Frazzoli, JP How, and Gaston Fiore. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[9]Jack W. Langelaan. State Estimation for Autonomous Flight in Cluttered Environments. *Journal of Guidance, Control and Dynamics*, 30(5), 2007.

[10]Steven M. LaValle. Rapidly-Exploring Random Trees: A New Tool For Path Planning. Technical report, Computer Science Dept., Iowa State University, 1998.

[11]Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, volume 11. Springer-Verlag, New York, January 2011.

[12]Michael Otte and Nikolaus Correll. Path Planning with Forests of Random Trees: Parallelization with Super Linear Speedup. Technical Report April, University of Colorado, Boulder, 2011.

[13]David Rathbun, Sean Kragelund, Anawat Pongpunwattana, Brian Capozzi, and Metron Aviation. An Evolution Based Path Planning Algorithm For Autonomous Motion of a UAV Through Uncertain Environments. *Air Traffic Management for Commercial and Military Systems*, pages 8–21, 2002.

[14]S.N. Sivanandam and S.N. Deepa. *Introduction to Genetic Algorithms*. Springer, 2008.

[15]M.S. Srivastava. *Methods of Multivariate Statistics*. John Wiley & Sons, New York, 2002.