The Planner Ensemble: Motion Planning by Executing Diverse Algorithms

Sanjiban Choudhury[1] and Sankalp Arora[1] and Sebastian Scherer[1]

***Abstract*— Autonomous systems that navigate in unknown environments encounter a variety of planning problems. The success of any one particular planning strategy depends on the validity of *assumptions* it leverages about the structure of the problem, e.g., Is the cost map locally convex? Does the feasible state space have good connectivity? We address the problem of determining suitable motion planning strategies that can work on a diverse set of applications. We have developed a planning system that does this by running competing planners in parallel.**

In this paper, we present an approach that constructs a *planner ensemble* - a set of complementary planners that leverage a diverse set of assumptions. Our approach optimizes the submodular selection criteria with a greedy approach and lazy evaluation. We seed our selection with learnt priors on planner performance, thus allowing us to solve new applications without evaluating every planner on that application. We present results in simulation where the selected ensemble outperforms the best single planner and does almost as well as an off-line planner. We also present results from an autonomous helicopter that has flown missions several kilometers long at speeds of up to $56^{\mathrm{m}}/_{\mathrm{s}}$ which involved avoiding unmapped mountains, no-fly zones and landing in cluttered areas with trees and buildings. This work opens the door on the more general problem of adaptive motion planning.

I. Introduction

The problem of designing a real-time motion planning system for an application is an open problem. It is currently solved by human intuition on what a good approach might be, picking one's favourite planner and augmenting it with layers of heuristics for speedup. Suprising still, different groups come up with different solutions - e.g compare the urban challenge approaches [1], [2]. Thus, there is a demand of a systematic approach to designing a planning solution that works best for the given application.

In this paper, our application was to design a motion planning system for an autonomous helicopter that has to fly *smoothly* in *unknown* environments that may span mountains, canyons and come into land avoiding buildings and trees. Fig. 1 shows two types of situations that can occur in the application. Given the requirement is for smooth paths, a trajectory optimization (such as CHOMP [3]) would be a natural choice. However, local minima (Fig. 1d) demand global reasoning - but one has to decide on the search space and search strategy of this global reasoning. One solution is to hand engineer a complicated solution that intelligently (using heursitics) decides what and when to optimize. Our approach was to run in parallel multiple complementary approaches and select the best one.

[1]The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA sanjiban,basti@cmu.edu, asankalp@andrew.cmu.edu

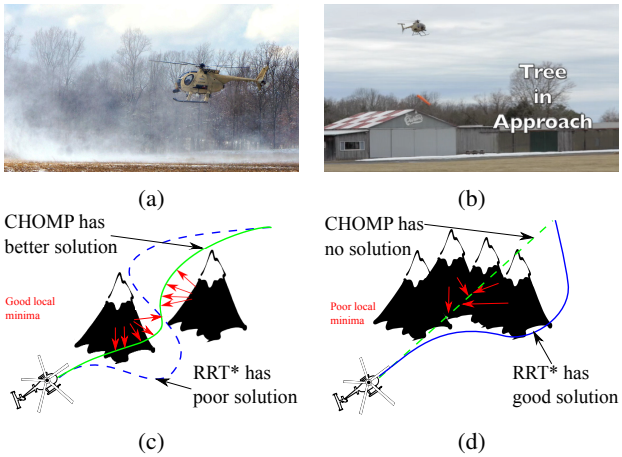


Fig. 1: (a) Boeing's Unmanned Little Bird comes in to land autonomously in the snow. This was the platform used for our experiments. (b) The application involved avoiding many unknown terrain features such as a tree on the approach profile (c) Flying between mountains. The cost map is locally convex and the gradients push the output of the optimizer into the minimum. Other search algorithms cannot find the valley as effectively. (d) A poor local minimum where the gradients are conflicting. A global planner can easily circumvent the situation.

In this paper we present the *planner ensemble*. Each element in the ensemble is a planning algorithm with different parameters. The total size of this space is exponential in the number of algorithms, different modes of graph creation and different types of heuristics. We show that picking the best ensemble for an application is equivalent to optimizing a submodular criterium [4]. Since evaluating all possible planners on an application is expensive (even in offline cases), we resort to machine learning to learn priors on planning performance. These priors are used to determine the lazy evaluation of a planner.

Machine learning techniques have been used in [5] to learn the effectiveness of a global optimization, to predict trajectories in a new environment by learning from a database, and to predict the initial effectiveness of a trajectory for optimization [6]. The work closest to us is [7] in which machine learning is used to predict a sequence of options. Different learners are used for different slots - the marginal loss of selecting an action for one slot is handed as training data for the next slot. We differ by learning priors for each planner and computing the marginal gains explicitly - allowing us to add new planners and try new applications without having to retrain.

There is also a considerable amount of work on running planners in parallel to solve a motion planning task [8], [9],

[10]. Other works [11] use heuristics to adapt the search to the structure of the environment. A recent work [12] uses multiple inadmissible heuristics to guide a discrete graph search. We believe the approach introduced in this paper can be used to select effective heuristics given features about the environment.

We have already presented in [13] the details of our motion planning system for the autonomous helicopter. The reason behind the high performance of the system was because of the powerful combination of CHOMP [3] and RRT*-AR [14]. In this paper, we present a novel approach to automatically arrive at such a decision by characterizing performance of planners.

Our main contributions are:

- An ensemble of specialized planners that leverage diverse assumptions to maximize the performance of the system in an application.
- An automated ensemble selection that uses learnt priors on planner perfomance to create a custom set of planners for a given application.
- Results for the ensemble on an autonomous helicopter performing missions at high speeds.

II. Problem Statement

Let $X \subset \mathbb{R}^n$ be the configuration space of the system. Let $X_{obs} \subset X$ be invalid configurations that result in collision with obstacle. The dynamics of the system are specified as a dynamics constraint $g(x, \frac{dx}{dt}, \ldots, \frac{d^r x}{dt^r}) \leq 0, x \in X$, where r is the relative degree. Let the trajectory $\xi : [0,1] \to X$ be a smooth mapping from time to configuration. The planning problem is to find the shortest dynamically feasible trajectory from start x_0 to the goal x_f that is collision free. This is expressed as follows:

$$\begin{aligned} \underset{x(t)}{\text{minimize}} \quad & \int_0^1 \|\dot{x}(t)\| \mathrm{dt} \\ \text{subject to} \quad & x(0) = x_0 \\ & x(1) = x_f \\ & g(x, \frac{dx}{dt}, \ldots, \frac{d^r x}{dt^r}) \leq 0 \\ & x(t) \in X \setminus X_{obs}, t \in [0,1] \end{aligned} \tag{1}$$

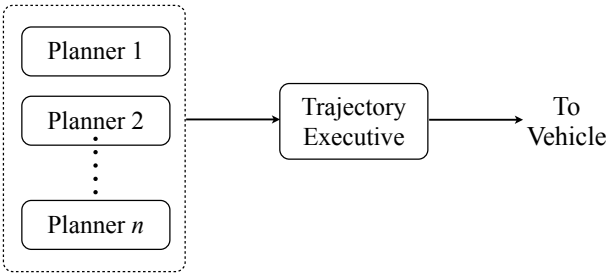
We define an *application* as a distribution over different parameters of (1).

The success of a planner depends on the distribution of X_{obs}. For example, there are configurations of X_{obs} which results in (1) being locally convex around an initial guess. In such situation, a trajectory optimization algorithm has a very high performance. There are also configurations of X_{obs} which result in a cluttered environment with low visibility, i.e., the straight line joining points far apart in the configuration space intersect X_{obs}. In such situations, sampling based algorithms have degraded performance due to most of the connections failing during the collision check.

III. Planner Ensemble

A. Overview

The central idea is to have an ensemble of planners running in parallel as shown in Fig 2. These planners work in parallel, often in a complimentary fashion, and bid their plans. These plans are then fed to a trajectory executive. The executive is a low latency verifiable unit whose job is to receive plans, ensure safety and send the best plan to the vehicle.



(a)

Fig. 2: The Planner Ensemble

B. Ensemble Generation

We first formalize the idea of an ensemble with the help of some supporting definitions. Let the database of all planners be $\mathcal{P}$. A planner is an element $\mathcal{P}_i \in \mathcal{P}$. We use the definitions provided in [15] to define a planner. It can be thought of as a process operating on an *implicit graph* to make an *explicit tree*. For a discrete graph search, the implicit graph is a lattice with local connectivity. For sampling based algorithms, the implicit graph consists of points sampled from continuous space with a nearest neighbour ball to define connectivity. For trajectory optimization, the implicit graph is the continuous manifold. The explicit tree creation process varies between dynamic programming and some approximate versions. Heuristics are employed both in the implicit graph and explicit tree creation process. A survey of heuristics in sampling based motion planning literature is provided in [16]. We will delve into more details about $\mathcal{P}_i$ in subsection IV-A, but note that all such possible combinations imply $\mid \mathcal{P} \mid$ is very large.

Let $f \in \mathcal{F}$ denote a particular planning problem, i.e., an instance of (1). We define an application $\mathcal{A}$ as a distribution over planning problems $\mathbf{Pr}_{\mathcal{A}}(f)$.

Let $s_i = \{0, 1\}$ be the event of planner $\mathcal{P}_i$ finding a *feasible* solution, i.e., only satisfying the constraints in (1). Let $\mathbf{Pr}(s_i \mid f)$ be the probability of this event for a planning problem f. For deterministic planners, this reduces to an indicator function.

Let $V_i(s_i, f) \in [0, 1]$ be the expected solution quality of $\mathcal{P}_i$ on f given the event s_i. This is inversely proportional to the cost. In this work, we use $V_i(1, f) = e^{-\beta \varepsilon_i}$, where $\varepsilon_i \geq 0$ is the expected suboptimality fraction of the solution and β is a scaling factor. Lack of a solution implies 0 score, i.e., $V_i(0, f) = 0$.

Let $\Gamma \subset \mathcal{P}$ be an ensemble of planners. $\Gamma(i)$ refers to the i^{th} planner in the ensemble.[1] The optimization objective can be stated formally as

$$\begin{aligned} \underset{\Gamma \subset \mathcal{P}}{\text{maximize}} \quad & \int \hat{V}_\Gamma(f)\, \mathbf{Pr}_\mathcal{A}(f)\, \mathrm{df} \\ \text{subject to} \quad & \mid \Gamma \mid \leq B \end{aligned} \tag{2}$$

where B is the budget of the ensemble and $\hat{V}_\Gamma(f)$ is the expected score of the ensemble of f. This can be expressed as

$$\hat{V}_\Gamma(f) = \sum_{s_\Gamma} \max_{i \in \Gamma} \{V_i(s_i, f)\}\, \mathbf{Pr}(s_\Gamma \mid f) \tag{3}$$

where $s_\Gamma = \{s_i : i \in \Gamma\}$ is a particular configuration of success/failures of the elements in the ensemble Γ, where the number of configurations is $2^{|\Gamma|}$.

There are two major reasons why the optimization is intractable. Firstly, the problem in (2) is NP-hard and the combinatorial approach has a worst case complexity $O(\mid \mathcal{P} \mid^B)$. Secondly, the evaluation of (3) is expensive since we dont have analytical expressions for any of the terms. The evaluation involves sampling from $\mathbf{Pr}_\mathcal{A}(f)$ and for each f, running the planners in the ensemble. In the next subsection, we present an approach to alleviate both of these problems.

C. Greedy Selection with Lazy Evaluation

As the problem is NP-hard, we present a greedy selection with lazy evaluation algorithm to optimize the criteria in (2). A greedy approach has a bounded sub-optimality guarantee as the criterium is *sub-modular* [2].

The objective is to select the best set Γ^* on an application $\mathbf{Pr}_\mathcal{A}(f)$ (2). Even though the setting is offline, since the space of planners $|\mathcal{P}|$ is large, not every element can be evaluated. If every element were to be evaluated we would get a table $s_i(f_t)$ and $V_i(s_i, f_t)$ for every planner $\mathcal{P}_i$ and sampled features f_t from $\mathbf{Pr}_\mathcal{A}(f)$. Greedy selection on this table would give a set with bounded sub-optimality.

Since the evaluation of (3) is expensive, we approach this problem by learning priors on the performance of each planner in $\mathcal{P}$. Let $\hat{f} \approx f$ be an approximation of the planning problem f in a feature space. The priors learnt are $\mathbf{Pr}(s_i \mid \hat{f})$ and $V_i(s_i, \hat{f})$. These priors allow us to analytically evaluate (3) without actually running the ensemble on the application. We will delve into the details of how these are learned in section IV.

Before we move on to the algorithm, we briefly comment on the assumptions made by the usage of priors. Firstly, $\mathbf{Pr}(s_i \mid \hat{f})$ and $V_i(s_i, \hat{f})$ incorporate the performance fluctuation due to approximation error of $\hat{f}$, which we assume to be small given sufficiently expressive features. Secondly, we assume conditional independence of the success probabilities of the ensemble given the features, i.e, $\mathbf{Pr}(s_\Gamma \mid \hat{f}) = \prod_{i \in \Gamma} \mathbf{Pr}(s_i \mid \hat{f})$. This assumption holds if the feature is expressive enough to represent the success of a planner without any other additional knowledge.

[1] For readability, we will refer to $\mathcal{P}_i$ as i.

[2] This is similar to the set coverage function

TABLE I: Planning strategies and their learnt prior

Planners	**Optimizes**	**Constraints**	**Prediction Accuracy** [%]	**Regression Error** (RMSE)
A*	Y	Dubins	80.61	0.071
CHOMP	Y	Smoothness	88.09	0.015
RRT*	Y	Dubins	84.69	0.253
RRT*-Tunnel	Y	Dubins	84.01	0.099
RRT*-Obs	Y	Dubins	71.43	0.214
RRT	N	Dubins	91.49	0.306
Lazy-RRT	N	Dubins	89.46	0.323
RRT-Connect	N	Dubins	94.56	0.349
LBTRRT	N	Dubins	90.47	0.210
T-RRT	N	Dubins	92.17	0.325
EST	N	Dubins	91.16	0.348
PDST	N	Dubins	87.07	0.274
KPIECE	N	Dubins	87.42	0.360
BKPIECE	N	Dubins	72.45	0.413
LBKPIECE	N	Dubins	71.08	0.409
STRIDE	N	Dubins	85.71	0.374
SPARS	N	None	76.53	0.295
SPARS2	N	None	75.51	0.293
SBL	N	None	78.91	0.298
PSBL	N	None	78.91	0.301

The algorithm we present greedily selects terms based on the marginal gains. Given an ensemble Γ, we define the analytical marginal gain of adding an element e to this ensemble as

$$\Delta(e \mid \Gamma) = \int \left(\hat{V}_{\Gamma \cup \{e\}}(f) - \hat{V}_\Gamma(f) \right) \mathbf{Pr}_\mathcal{A}(f)\, \mathrm{df} \tag{4}$$

Algorithm 1 describes the lazy greedy approach. The application is initially sampled for a set of feature vectors $\hat{\mathcal{F}}_\mathcal{A}$. At every iteration, an approximation of (4) is estimated for every planner using the learnt priors and applying conditional independence. This step is orders of magnitude faster than running the planner on all the problems in $\hat{\mathcal{F}}_\mathcal{A}$. The planner with the largest marginal gain is selected and then the true marginal gain is computed by executing the planner on the application problems. The results of the trial are also used to update the priors for the planner. If the true marginal gain is still higher than the estimate of the next best planner, the planner is added to the ensemble.

IV. Learning Planning Performance Priors

A. Planning Strategies

In this work, we use the term planner $\mathcal{P}_i$ to imply a planning strategy. The core part of the planning strategy consists of applying a planning algorithm to solve the problem in (1). In addition, the strategy includes reductions and simplifications of (1) as well as leveraging assumptions about the structure of the problem.

Table I enlists the database of planners $\mathcal{P}$. Each planner was given 1.0s computation time. The first column indicates if the algorithms were optimizing the criteria or simply searching for a feasible option. The second column indicates the extent to which the planners attempted to satisfy the

Algorithm 1 Planner Ensemble Selection

Input
  $\mathbf{Pr}_{\mathcal{A}}(f)$: Application feature distribution
  $\mathbf{Pr}(s_i \mid \hat{f})$: Learnt prior of feasibility
  $V_i(s_i, \hat{f})$: Learnt prior of performance
  B : Ensemble budget
Output
  Γ : The selected ensemble
Variables
  Φ : Set of planners that have been evaluated
  $\rho(e)$: Marginal gain of planner e
procedure LAZYGREEDY
  $\Gamma \leftarrow \emptyset$ ▷ Initialize ensemble to empty set
  $\Phi \leftarrow \emptyset$ ▷ Initialize evaluated set to empty set
  $(\mathcal{F}_{\mathcal{A}}, \hat{\mathcal{F}}_{\mathcal{A}}) \leftarrow$ SAMPLE($\mathbf{Pr}_{\mathcal{A}}(f)$) ▷ Sample from application to get set of planning problems and feature vectors
  while $\mid \Gamma \mid < B$ **do**
    $\rho(e) \leftarrow \hat{\Delta}(e \mid \Gamma), \forall e \in \mathcal{P} \setminus \{\Gamma \cup \Phi\}$ ▷ Compute marginal gain estimates using learnt prior applied on $\hat{\mathcal{F}}_{\mathcal{A}}$
    $e \leftarrow \arg\max_{e' \in \mathcal{P} \setminus \Gamma} \rho(e')$ ▷ Select planner with maximum marginal gain
    $(\rho(e)) \leftarrow$ EVALUATE($e, \Gamma, \mathcal{F}_{\mathcal{A}}$)) ▷ Execute planner e on application to get actual marginal gain
    $\Phi \leftarrow \Phi \cup \{e\}$ ▷ Add planner to evaluated set. This no longer needs to use priors.
    if $\rho(e) < \rho(e') \; \forall e' \in \mathcal{P} \setminus \{\Gamma \cup e\}$ **then** ▷ Check if e is still the best planner
      $\Gamma \leftarrow \Gamma \cup \{e\}$ ▷ Add e to ensemble
end procedure

constraints - we note the fact that none of these planners were designed to fully satisfy the constraint $g(\dots)$ in (1) in their original form. Instead they either attempted to find a smooth solution, or a solution that satisfied Dubin's dynamics or no constraints at all. We then use a fast non-linear projection operator approach, Dynamics Projection Filter [13], to project the solution on the constraint manifold.

To allow CHOMP to solve the problem, we created a distance field extending beyond X_{obs} to allow gradients to guide it out of collision. The smoothness term was used to allow solutions to satisfy $g(\dots)$ as much as possible before projection. RRT*-Tunnel is a variant of RRT* [17] that samples in the $\mathbb{R}^3$ workspace in a tunnel around the analytic solution. The tunnel is an assumption about how far from the initial guess the optimal solution may lie. RRT*-Obs is another variant of RRT* that samples on the obstacle manifold. The planners that did not attempt optimization were post-processed to get smooth outputs. We used the open source OMPL [18] for most of the planners other than CHOMP and A*.

B. Features

The feature vector $\hat{f}$ consists of two components - global connectivity information and local distance field information.

1) Global Connectivity: A coarse lattice of fixed resolution spanning the whole workspace is created and centred on the line joining the start and goal. The feature vector is then $\hat{f} = \{1(e \notin X_{obs})\}$ where $1(.)$ is the indicator function, e is the edge. This feature vector approximates the global connectivity of the workspace.

2) Local Distance Field: A dense lattice of fixed resolution of bounded volume around the straight line joining start and goal is created. For every vertex on this lattice, the distance field is sampled for the distance and direction to the nearest obstacle. This feature vector contains information about the local convexity of the objective and the obstacle density.

C. Training and Testing

Given the lack of standard training datasets in this area of work, we synthesized a dataset of 10000 planning problems by permutations and combinations of various scenarios that emphasize one or multiple major assumptions leveraged by an algorithm. Fig. 3 shows some samples from the dataset.

We used Random Forests [19] for learning success probabilities and performance of the planners because of their robustness to noise. To model the success probabilities, we used a discriminative classifer rather than learning a generative model for performance reasons. To get a model of planner performance, we did regression on the sub-optimality ratios. These ratios were obtained by running RRT* with a longer time budget and then taking the minimum cost incurred on a planning problem by any planner. The accuracy on a test dataset is shown in Table I.

V. ENSEMBLE SELECTION IN SIMULATION

We used parameters of the model of Boeing Unmanned Little Bird (ULB) [13] with mission speeds of $30^{\mathrm{m}}/_{\mathrm{s}}$. We selected two contrary applications - *Forest* and *City* - and ran simulations with fully known and unknown (sensor limited) situation.

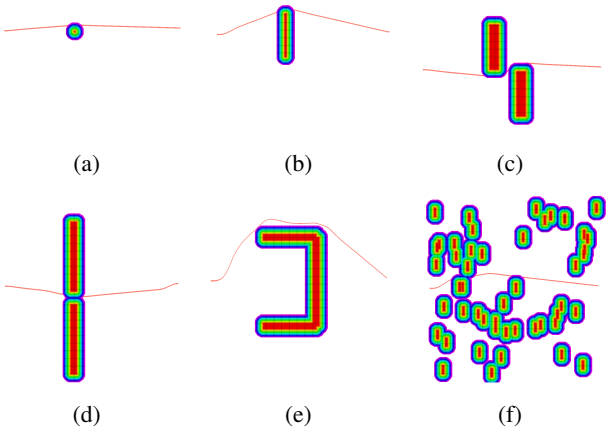


Fig. 3: Some samples of building blocks used to create the training dataset. Each scenario focusses on a particular strenth / weakness that planners have. Permutations of these make a planning problem. (a) Pole - Locally convex (b) Wall - Non-convex, requires large excursion (c) Baffle - Local minima on either side, narrow entry passage. (d) Corridor - Start and goal halves of state space have only one small link (e) Bugtrap - Requires planners to search backwards (f) Various combinations of random environments

A. *Known Environment*

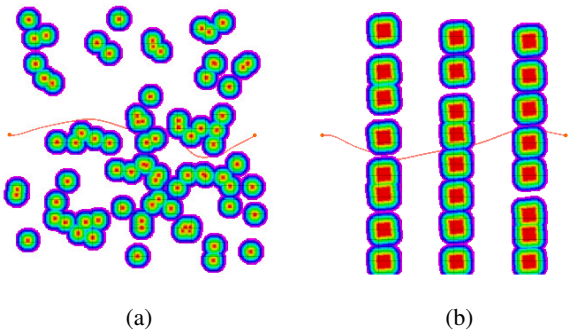


Fig. 4: Samples of the distance field from *known* applications. The red regions are the obstacles and upto the green expansion (50m) is considered to be in collision. (a) Forest application: poisson forest field [20] (b) City application: randomly perturbed city blocks

Fig. 4 shows snapshots from both applications. As the entire environment is known, the sensor plays no role. To clearly illustrate the ensemble selection process, Table II shows every step. Along with the priors, the actual value on evaluation is shown in brackets. In this case, RRT*-Tunnel and CHOMP were selected as the two planners. The lazy evaluation corrects for error in the estimation of the performance of CHOMP. The reason these two planners performed well is because the optimal solution was always within a tunnel of the initial guess and the cost function was usually convex around the initial guess. Fig. 4a is the output of CHOMP.

For the city application, we see in Table III that A* and RRT*-Tunnel were chosen. This is because this environment required reasoning in a global sense, discretization effects were acceptable and on occasion when the solution happened to lie near the initial guess, the RRT*-Tunnel was the only algorithm able to sample densely enough to get a solution as shown in Fig. 4b.

TABLE II: Ensemble selection for known forest application

Planners	**Iteration 1** Prior (Eval)	**Iteration 2** Prior (Eval)	**Iteration 3** Prior (Eval)
A*	0.694	0.694	0.003
CHOMP	**0.989** (0.681)	0.681	**0.021** (0.069)
RRT*	0.700	0.700	0
RRT*Tunnel	0.983	**0.983** (0.848)	-
RRT*-Obs	0.483	0.483	0
RRT	0.540	0.540	0
Lazy-RRT	0.563	0.563	0
RRTConnect	0.510	0.510	0
LBTRRT	0.638	0.638	0.006
T-RRT	0.533	0.533	0
EST	0.529	0.529	0
PDST	0.588	0.588	0
KPIECE	0.491	0.491	0
BKPIECE	0.772	0.772	0
LBKPIECE	0.491	0.491	0
STRIDE	0.482	0.482	0
SPARS	0.342	0.342	0
SPARS2	0.330	0.330	0
SBL	0.456	0.456	0.002
PSBL	0.458	0.458	0.004
Ensemble			
Slot 1		**RRT*-Tunnel**	**RRT*-Tunnel**
Slot 2			**CHOMP**

B. *Unknown Environment*

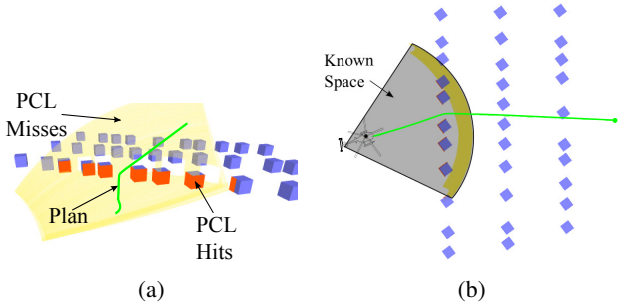


Fig. 5: Absence of a prior map makes the planning problem seem much easier. We illustrate the view of the robot as it explores the city. (a) The point cloud hits and misses correspond to seen obstacles and maximum range returns respecitvely. As far as the robot can see there is only a front row of obstacles (b) The RRT*-tunnel is easily able to plan through the first row and is unaware of the other obstacles - thus producing a high quality path.

In these environments, the environment was discovered on the fly by the sensor. Evaluation of a planner implies simulating with the sensor [3]. According to Table III, the forest results remained unchanged but the city results were completely changed.

We examine this problem in Fig. 5. Since the city is *incrementally* revealed to the robot, it is for the most part easier, even convex around the intial guess on many occasions. This

[3]Making dataset dependent on prediction creates a data set distribution problem similar to [21]. We circumvent this for now by gathering datasets by running the offline planner.

TABLE III: Ensemble v/s individual performance for different applications

Criteria	**Ensemble Slot 1** Planner(Score)	**Ensemble Slot 2** Planner(Score)	**Total** Score
Known Forest	RRT*Tunnel(0.85)	CHOMP(0.68)	0.92
Unknown Forest	RRT*Tunnel(0.72)	CHOMP(0.51)	0.85
Known City	A*(0.68)	RRT*Tunnel(0.59)	0.78
Unknown City	RRT*Tunnel(0.80)	CHOMP(0.54)	0.88

makes the use of A* wasteful when much higher quality solutions free of discretization effects can be obtained. The ensemble selection system is automatically able to leverage this assumption. We also see the effect of combining two planners in the forest environment leads to 18% improvement and within 85% of offline performance.

VI. Autonomous Helicopter Flight

The planning system has been stress tested on an autonomous helicopter over a period of 4 months. Missions have ranged from avoiding no-fly zones, terrain obstacles and landing straight to ground while avoiding enroute trees, facing the wind direction and dealing with clutter at the landing zone. A detailed case-by-case analysis of these results can be found in [13].

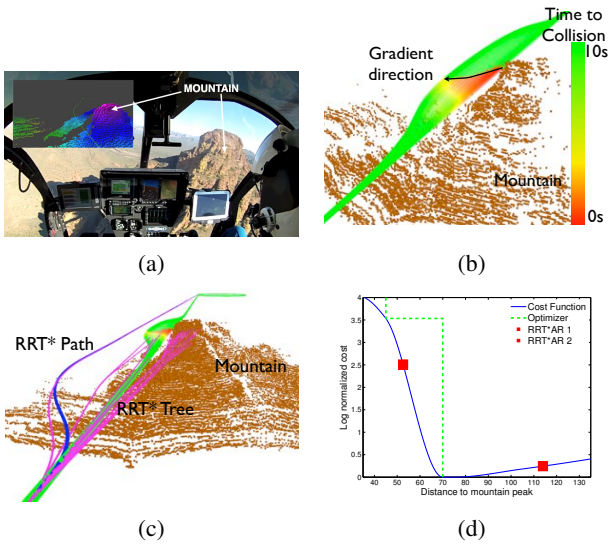
For flight on an actual helicopter, we appended a collision avoidance objective, time to collision [13], which essentially penalizes high speed flight near obstacles and can scale naturally with speed. From the ensemble selection analysis, the planners selected were **RRT*-Tunnel** and **CHOMP**. We modified the RRT*-Tunnel to produce alternate routes (RRT*-AR [14]).

Fig. 6 shows an example of avoiding a mountain in one of the missions. This was a typical situation where CHOMP produces a high quality solution while the RRT*-Tunnel produces paths in the vicinity of the optimal solution. On another very similar mission in the same environment, a completely opposite situation was faced as shown in Fig 7. Since the system was told to pass between a mountain and no fly zone, it got stuck in a poor local minima and the RRT*-Tunnel produced a path.

We also ran full system simulations on environments such as the grandcanyon (Fig. 8) and urban environment (Fig. 9). In the grandcanyon environment, CHOMP was selected 65.82% and RRT*-AR 34.18%. In the urban environment, CHOMP was selected 29.17% while RRT*-AR was selected 70.83%. The ensemble was able to keep the vehicle safe and successfully complete 100% of missions in these scenarios. The percentages indicate how the effectiveness of a planner changes throughout an application.

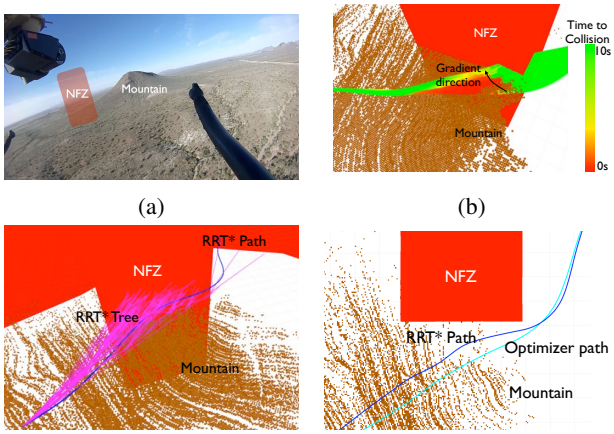
VII. Discussion

We have presented an approach to solve diverse planning problems with an ensemble of complementary planners. We designed an ensemble selection system based on a greedy approach with lazy evaluation using learnt priors of planning performance. We presented results in simulation and for autonomous helicopters where the ensemble had



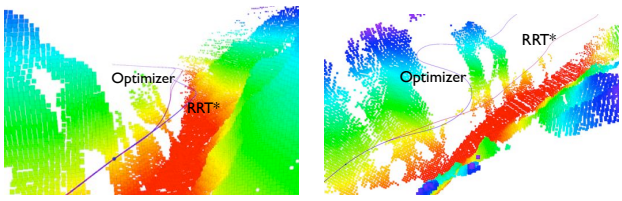
(a) (b) (c) (d)

Fig. 6: Flying around an unmapped mountain in Mesa, AZ. (a) Safety pilot's view of the mountain (b) The gradient due to time to collision objective takes the trajectory away from the obstacle (c) The RRT*-AR optimal path avoids the mountain by a much larger distance than required. The RRT*-AR tree contains branches on either side of the cost valley (d) The log normalized cost function valley with distance from the mountain peak. The optimizer descends into the valley within 8 iterations. The RRT*-AR computes alternate routes lying on either side of the valley.



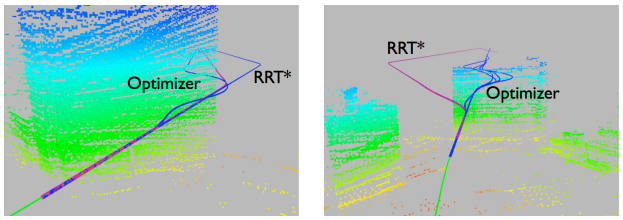
(a) (b) (c) (d)

Fig. 7: Flying between a no-fly zone (NFZ) and an unmapped mountain in Mesa, AZ (a) The skid camera view of the scenario (b) The gradient due to the time to collision pushes the trajectory into the forbidden NFZ. (c) The RRT*-AR tree is very diverse and contorts to find a near optimal trajectory (f) Comparison of the RRT*-AR trajectory to optimizer shows that RRT*-AR is safer



(a) (b)

Fig. 8: Both RRT*-AR and Optimizer dominate interchangeably in a mission in the Grand Canyon (a) Optimizer dominates here because the gradients from the edges push it into a global minima (b) Optimizer gets trapped in a bad local minima in a space between two obstacles. RRT*-AR plans around the obstacle.



(a) (b)

Fig. 9: Planners selected interchangeably in an urban mission scenario. (a) The optimizer follows a nice gradient around the building while the RRT*-AR computes paths that miss the optimum valley (b) The optimizer gets trapped in a local minimum inside a building. The RRT*-AR circumvents the obstacle.

better performance on the mission than a single planner. Even though our approach shows promising results, there are several issues that make this a hard problem to solve.

The first issue concerns the noise in the learnt priors. The priors are learnt over a training distribution which is different from the application distribution. This makes it difficult to bound the performance of the selected ensemble from the best ensemble Γ^*. The approach used in [7], where a cascade of regressors are used may reduce the noise level. However the difference in distribution is still a key issue.

The second issue is that this approach does not address an online setting. In the online setting, the algorithm is required to produce a different ensemble Γ at every time step to bound the regret with respect to the best ensemble Γ^* in hindsight. Since each element cannot be evaluated (as in weighted experts [22]), this must be framed as a multi-armed bandit problem [23]. In addition, the utilization of the features f would allow for one to have a lower regret in comparison to not using the priors.

The third issue is with regards to a more complex matter of distributions. Not only is $\mathbf{Pr}_{\mathcal{A}}(f)$ unknown, it should also be a function of the selected ensemble Γ. This is because the planners dictate where the robot navigates and the kind of problems it is likely to encounter. Work done in [21] provides some potential direction for further research.

Another key source of difficulty is that the features must be able to capture all aspects of the environment in order to effectively predict planner performance. For practical purposes, this might be a very difficult function to learn. A more predictable property is to learn the interdependence among planners. When one planner is evaluated, this model can reliably predict the likelihood of success of a dependent planner. Work done on multi-armed bandits with dependence [24] shed some light on this matter.

VIII. Acknowledgement

This work would not have been possible without the dedicated efforts of the entire AACUS TALOS team and was supported by ONR under contract N00014-12-C-0671.

References

[1] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[2] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[3] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 489–494.

[4] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, p. 19, 2012.

[5] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, and M. Sciandrone, "Machine learning for global optimization," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 279–303, 2012.

[6] J. Pan, Z. Chen, and P. Abbeel, "Predicting initialization effectiveness for trajectory optimization," in *Robotics and Automation, 2014. ICRA'14. IEEE International Conference on*. IEEE.

[7] D. Dey, T. Y. Liu, M. Hebert, and J. A. Bagnell, "Contextual sequence prediction with application to control library optimization," *Robotics*, p. 49, 2013.

[8] S. Caselli and M. Reggiani, "Randomized motion planning on parallel and distributed architectures," in *Parallel and Distributed Processing, 1999. PDP'99. Proceedings of the Seventh Euromicro Workshop on*. IEEE, 1999, pp. 297–304.

[9] D. J. Challou, M. Gini, and V. Kumar, "Parallel search algorithms for robot motion planning," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 46–51.

[10] M. Otte and N. Correll, "C-forest: parallel shortest path planning with superlinear speedup," *Robotics, IEEE Transactions on*, vol. 29, no. 3, pp. 798–806, 2013.

[11] J. Denny, M. Morales, S. Rodriguez, and N. M. Amato, "Adapting rrt growth for heterogeneous environments," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1772–1778.

[12] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a*," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

[13] S. Choudhury, S. Arora, and S. Scherer, "The planner ensemble and trajectory executive: A high performance motion planning system with guaranteed safety," in *AHS 70th Annual Forum, Montreal, Quebec, Canada*, May 2014.

[14] S. Choudhury, S. Scherer, and S. Singh, "Rrt*-ar: sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3947–3952.

[15] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Bit*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs," *arXiv preprint arXiv:1405.5848*, 2014.

[16] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Access, IEEE*, vol. 2, pp. 56–77, 2014.

[17] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robotics: Science and Systems*, 2010.

[18] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] S. Karaman and E. Frazzoli, "High-speed flight in an ergodic forest," *IEEE Transactions on Robotics (submitted)*, 2012.

[21] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *arXiv preprint arXiv:1011.0686*, 2010.

[22] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and computation*, vol. 108, no. 2, pp. 212–261, 1994.

[23] A. Mahajan and D. Teneketzis, "Multi-armed bandit problems," in *Foundations and Applications of Sensor Management*. Springer, 2008, pp. 121–151.

[24] S. Pandey, D. Chakrabarti, and D. Agarwal, "Multi-armed bandit problems with dependent arms," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 721–728.