2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Parameterized Maneuver Learning for Autonomous Helicopter Flight

Jie Tang, Arjun Singh, Nimbus Goehausen, and Pieter Abbeel

Abstract— **Many robotic control tasks involve complex dynamics that are hard to model. Hand-specifying trajectories that satisfy a system's dynamics can be very time-consuming and often exceedingly difficult. We present an algorithm for automatically generating large classes of trajectories for difficult control tasks by learning parameterized versions of desired maneuvers from multiple expert demonstrations. Our algorithm has enabled the successful execution of several parameterized aerobatic maneuvers by our autonomous helicopter.**

I. INTRODUCTION

Trajectory following is a fundamental building block for many robotics tasks. By reducing the control problem to trajectory following, one can often suffer less from the curse of dimensionality as it becomes sufficient to consider a relatively small part of the state space during control policy design. Unfortunately, specifying the desired trajectory and building an appropriate model for the robot dynamics along that trajectory are often highly non-trivial, tightly coupled tasks. For the control design to benefit from being reduced to a trajectory following task, it typically requires that the target trajectory is at least approximately physically feasible. Specifying such a target trajectory can be highly challenging.

In the apprenticeship learning setting, where we have access to an expert who can provide demonstrations, it is natural to request a demonstration of the desired trajectory as the specification of the target trajectory. However, rarely will an expert be able to demonstrate exactly the trajectory we desire to execute autonomously. Repeated expert demonstrations *together* can often capture a desired maneuver, as different demonstrations deviate from the intent in different ways. Abbeel et al. [1] and Coates et al. [8] describe a generative probabilistic model that enabled them to extract an expert helicopter pilot's intended trajectory from multiple suboptimal demonstrations. They also show how multiple demonstrations can be leveraged to obtain a high accuracy dynamics model, which is specifically tuned to the particular maneuver in consideration.

Unfortunately, most robotics tasks require us to adapt our learned maneuvers to account for a changing environment: consider flying aerobatic helicopter maneuvers while avoiding trees and other obstacles. We may need to perform stall turns[1] of any altitude between 10 and 50 meters. An approach based on the work presented in [1] and [8] would require us to anticipate every possible stall turn altitude and gather expert demonstrations for each one in advance. This seems wasteful, as the different stall turn trajectories will share many properties.

In this paper, we present a probabilistic model-based algorithm (building upon [1], [8]), which makes efficient use of expert demonstrations by learning *parameterized maneuvers* rather than a discrete set of maneuvers. We first collect a wide range of executions of the maneuver of interest. When asked for a particular execution of the maneuver, such as a stall turn of a particular altitude, our algorithm generates the appropriate target trajectory.

We tested our algorithm on three aggressive helicopter maneuvers: stall turns, loops, and tic-tocs. Our algorithm successfully generates flyable parameterized maneuvers from a relatively small number of demonstrations. The generated trajectories closely match held-out trajectories. Our helicopter can perform these interpolated trajectories with an accuracy comparable to that of a human expert.

Fig. 1. Our Synergy N9 autonomous helicopter.

Videos of our autonomous helicopter flight results are available at the following page:

`http://rll.eecs.berkeley.edu/heli/icra10`

II. OVERVIEW

In many trajectory-following problems, the trajectories can be categorized into distinct maneuver classes. Furthermore, for many maneuver classes, a particular execution of a

The authors are with the Department of Electrical Engineering and Computer Sciences, UC Berkeley, CA 94720, U.S.A. Email: `jietang@eecs.berkeley.edu`, `arjun@hkn.eecs.berkeley.edu`, `nimbus@berkeley.edu`, `pabbeel@cs.berkeley.edu`.

[1]During a stall turn a forward flying helicopter pitches 90 degrees backwards and flies upwards until it reaches zero velocity. At that point it spins 180 degrees around its vertical axis, and descends nose down and levels out into forward flight. This is a standard aerobatic maneuver that can be useful for changing directions.

maneuver within the class can be specified using a small number of parameters. Throughout the paper, we use stall turns as a clarifying example of such a maneuver class.

We are given a number of demonstration trajectories (likely of unequal durations) from a single maneuver class. Our goal is to generate a new trajectory from the same maneuver class that satisfies a given parameterization. In order to specify parameterizations for different maneuver classes, we first define the concepts of *parameters* and *waypoints*.

A. Parameters

We define the parameters of a maneuver to be the defining attributes that one must specify when generating a trajectory from a maneuver class. In the case of stall turns, we use the trajectory's maximal altitude as the only parameter.

B. Waypoints

All trajectories within a single maneuver class share a similar shape and structure. This structure can be captured by a set of characteristic points that define the major variations between different trajectories within the class. For example, all stall turns generally share the same shape; however, two different stall turns are easily distinguished by their altitude at their peaks.

For each maneuver class, we define a set of characteristic points, which we refer to as *waypoints*. For stall turns, we define a single waypoint at the top of the maneuver, when the helicopter is at its maximal altitude.

Our trajectory learning algorithm specifies what values certain state variables should take at the waypoints. These values are derived from the input parameters. Rather than specifying all of the state variables at each waypoint, we only specify the subset of the state variables required to capture the key characteristics of the maneuver at that waypoint. We refer to this subset of state variables and the specified values as the *waypoint constraints*.

At the waypoint of a stall turn, the waypoint constraints specify an altitude equal to the input parameter and a vertical velocity of zero. This enforces that the waypoint corresponds to the maximal altitude. Our trajectory learning algorithm attempts to find a trajectory of the specified maneuver class that satisfies these generated waypoint constraints.

In addition to the defined waypoints for each maneuver class, we always include waypoints at the beginning and end of each trajectory. The constraints for these waypoints specify the position and orientation of the helicopter.

III. TRAJECTORY LEARNING ALGORITHM

Our trajectory learning algorithm estimates a target trajectory of a particular maneuver class from a set of demonstrations from the same maneuver class and parameters describing the desired trajectory.

The algorithm consists of two core steps. After initialization, the first step is time alignment; we align each of our demonstration trajectories to the current estimate of the target trajectory. The second step consists of a Kalman smoother used in conjunction with the EM algorithm to infer the target trajectory from the aligned demonstrations. These steps are summarized in Algorithm 1.

Input: Demonstrations D from desired maneuver class c, parameters p
Output: Estimate of target trajectory $\mathbf{z}$
Initialization:
  Find closest trajectory r;
  $w \leftarrow GenerateWaypointConstraints(c, p, r)$;
  Initialize $\mathbf{z} \leftarrow r$;
for $i = 1$ **to** $numIters$ **do**
  foreach *demonstration* $d \in D$ **do**
    $Align(\mathbf{z}, d)$;
  end
  $\mathbf{z} \leftarrow KalmanSmoother(D, w)$;
end

Algorithm 1: Outline of trajectory learning algorithm.

A. Initialization

Our algorithm requires three initialization steps. First, we find the closest trajectory from the same maneuver class, where distance is calculated in the parameter space. For stall turns, we find the trajectory of the closest altitude to the specified input parameter. We then use this trajectory along with the input parameter to generate the waypoint constraints, as described in Section III-D. Lastly, we set the initial estimate of the target trajectory to be this same closest trajectory.

B. Dynamic Time Warping

The demonstration trajectories collected from our expert vary considerably in size and duration. Furthermore, the important points in each trajectory may not occur at the same time; two stall turns of the same altitude may still reach their peaks at different times. To account for this, we use a dynamic programming algorithm known as dynamic time warping (DTW) in the speech-recognition literature [19] and the Needleman-Wunsch algorithm in the biological sequence alignment literature [18]. Dynamic time warping is often used to align multiple sequences to a single reference sequence.

The standard DTW algorithm performs poorly when the two sequences vary greatly in magnitude, shape, or duration [10]. For sequences with differences in magnitude, e.g. the top of a stall turn, DTW will change the time scale to match the exact magnitudes of the trajectories for as long as possible, rather than matching the overall shape of the trajectories. Our largest stall turns are up to two or three times larger and longer than our smallest stall turns. Dynamic time warping leads to unnatural alignments with abrupt changes in the time alignment rate (the difference in time from one time step to the next).

Because our trajectories correspond to physically feasible flights, we expect alignment rate to remain fairly constant

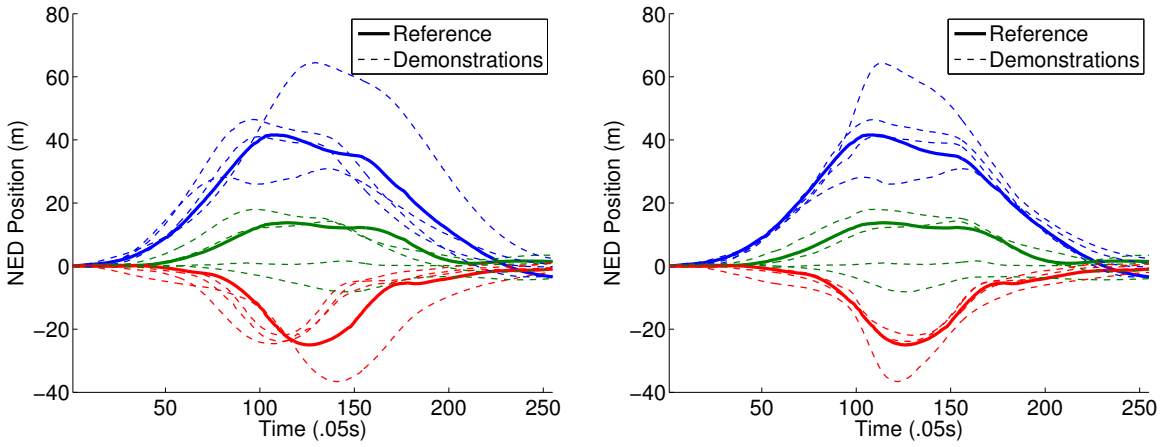


Fig. 2. Helicopter position during a stall turn: before (left) and after (right) alignment.

between time steps. Hence we modify DTW by adding a penalty[2] for changes in the time alignment rate. This algorithm produces alignments with fewer kinks and smoother rate changes over the course of the trajectory. Visual inspection indicates that important features of the underlying trajectories were better matched using the rate penalty penalty. Figure 2 shows an example alignment produced by DTW, both with and without rate penalty.

C. Kalman Smoother

We closely follow the generative model for helicopter trajectory learning given in [8], which we summarize here.

The demonstration trajectories, of possibly unequal lengths N^k, are sequences of helicopter states s_j^k (position, velocity, orientation in quaternion form, and angular rate), control inputs u_j^k, and model biases β_j^k (discussed in Section III-F), composed into a single state vector:

$$y_j^k = \begin{bmatrix} s_j^k \\ u_j^k \\ \beta_j^k \end{bmatrix} \text{ for } j = 0..N^k - 1, k = 0..M - 1.$$

The output trajectory is of length T:

$$z_t = \begin{bmatrix} s_t^\star \\ u_t^\star \\ \beta_t^\star \end{bmatrix} \text{ for } t = 0..T - 1.$$

We use the following notation:

$$\mathbf{y} = \{y_j^k \mid j = 0..N^k - 1, k = 0..M - 1\}$$

$$\mathbf{z} = \{z_t \mid t = 0..T - 1\}$$

The generative model used for the desired trajectory is given by an initial state distribution $z_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and an approximate model of the dynamics

$$z_{t+1} = f(z_t) + \omega_t^{(z)}, \qquad \omega_t^{(z)} \sim \mathcal{N}(0, \Sigma^{(z)}). \tag{1}$$

[2]The exact penalty term we use is the squared error between the last rate and the current rate.

The model represents each demonstration as a set of independent "observations" of the hidden trajectory $\mathbf{z}$. Specifically, our model assumes

$$y_j^t = z_t + \omega_j^{(y)}, \qquad \omega_j^{(y)} \sim \mathcal{N}(0, \Sigma^{(y)}). \tag{2}$$

D. Waypoint Constraints

From the input parameters, we generate waypoint constraints for each waypoint for the desired maneuver class. For example, when generating a stall turn, the input parameter specifies the desired altitude. We find the demonstration of a stall turn with the most similar altitude and extract the position and velocity from its waypoint. Rather than using the altitude of the demonstration in the waypoint constraints, we replace it with the desired altitude and set the velocity in the vertical axis to be zero.

In addition to the observations given by the demonstration trajectories, we use the generated constraints as direct observations of the hidden state:

$$q_j = I(z_t) + \omega_j^{(q)}, \qquad \omega_j^{(q)} \sim \mathcal{N}(0, \Sigma^{(q_j)}).$$

Here, q_j, the constraints at the jth waypoint, occur at time t and I is a function that selects the proper subset of the state z_t.

We set these observations to have very small variances, essentially forcing the Kalman smoother to pass the trajectory through the waypoint constraints while still respecting the dynamics model.

E. Expectation Maximization (EM)

It is well known that manual tuning of the variances of a Kalman filter can be difficult and time consuming. The EM algorithm provides a natural alternative by finding the covariances that maximize the log-likelihood of the data [12].

In our setting, however, we have a substantial amount of prior knowledge about the physical meaning of each variable. It is undesirable to ignore this knowledge by simply maximizing the likelihood. We fix the order of magnitude of the variances for each of the demonstration trajectories

to be consistent with our prior knowledge of the accuracy of our measurements. We use the EM step to weight the different trajectories relative to one another by increasing the observation variances on trajectories that are dissimilar to the target trajectory and keeping the observation variances on the other demonstrations low.

To achieve both these objectives, we constrain the variances to be of the form cV where V is the initial vector of observation variances and c is a scalar. We take the least-squares solution matching c against the variance vector E, found by taking the diagonal of the covariance matrix given by EM. This is equivalent to the projection $\frac{V^T E}{||V||}$ of E onto V. Intuitively, this allows the EM algorithm to determine the relative weight given to each observed trajectory without destroying dynamical consistency.

F. Model Biases

To improve our modeling accuracy, we use a time-varying model $f_t(\cdot)$ that is specific to the vicinity of the intended trajectory at each time t.

We express f_t as our "crude" model, f, augmented with a bias term $\beta_t^\star$:

$$z_{t+1} = f_t(z_t) + \omega_t^{(z)} \equiv f(z_t) + \beta_t^\star + \omega_t^{(z)}.$$

We have $\beta_{t+1}^\star \sim \mathcal{N}(\beta_t^\star, \Sigma^{(\beta)})$.

We incorporate the bias into our observation model by computing the observed bias $\beta_j^k = y_j^k - f(y_{j-1}^k)$ for each of the observed state transitions, and modeling this as a direct observation of the hidden model bias corrupted by Gaussian noise.

G. Drift

Different variations of the same maneuver often differ in ways that cannot be modeled by independent noise at each time step. For example, a shallow stall turn will consistently be lower than a very high stall turn. The differences between the trajectories are highly correlated over time and they are not explained well by simply having a Gaussian noise term in the observation model.

To capture such drift in the demonstration trajectories, we augment the latent trajectory's state with a "drift" vector δ_t^k for each demonstrated trajectory k, consisting of the drift in position and heading. The state observations are now noisy measurements of $z_t + \delta_t^k$ rather than only z_t.

IV. EXPERIMENTS

We evaluate our algorithm's ability to generate trajectories of the specified maneuver class that satisfy the generated waypoint constraints (Section IV-C). We also examine whether the generated trajectories are dynamically reasonable target trajectories by testing how reliably our controller can fly them (Section IV-D).

A. Autonomous Helicopter Platform

Our helicopter platform is a Synergy N9 with an on-board Microstrain 3DM-GX1 inertial measurements unit (IMU) and off-board ground-based cameras which provide stereo-based position estimates. We obtain the helicopter's position, orientation, velocity, and angular rate by fusing the sensor data with an extended Kalman filter. To collect the demonstrations we had our expert helicopter pilot fly stall turns, loops, and tic-tocs of varying sizes and time scales. Our feedback controller is a receding-horizon differential dynamic programming (DDP) controller (see [1], [8] for details).

B. Setup

We gathered 7-8 demonstrations of stall turns, loops, and tic-tocs that span a wide range of executions for each maneuver. All maneuvers were normalized to start at the origin facing north. We used the following parameterizations:

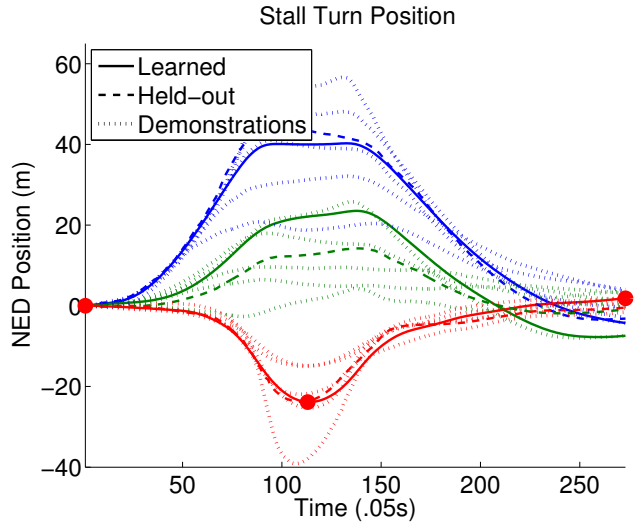
- Stall turns: the maximum altitude (z-coordinate) at the top of the turn.
- Loops: the maximum distance from the starting point on the x-axis (for perfectly circular loops, this would be the radius).
- Tic-tocs: the maximum distance from the starting point on the x-axis.

Before the time alignment step, each demonstration is oversampled to a length of three times the length of the target trajectory. This gives the algorithm more freedom in choosing alignment points. At each step of the algorithm we realign each of our demonstrations to the current estimate of the target trajectory.

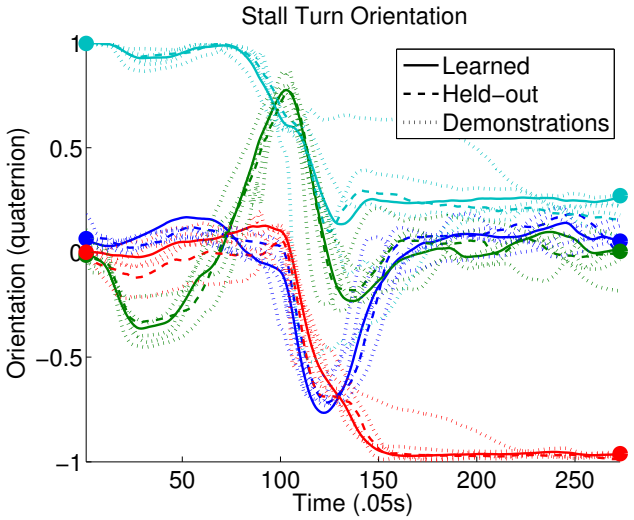
Our cost function for dynamic time warping is a weighted squared error function, where the weight for each axis is determined by the maneuver class. Certain axes are more relevant for some maneuvers than for others, while other axes are ignored because they tend to be very different across flights of the same maneuver class. For example, we weight the vertical axis for stall turns much more heavily than the horizontal axes. Furthermore, some axes are on different scales than others; for example, the quaternion entries are always between 0 and 1 while the position entries can get to be as large as 50 or even higher. We use these weights to account for these differences. The alignments are not very sensitive to the choice of weights as long as axes which are consistently different across multiple examples from the same maneuver class have significantly smaller weights.

C. Trajectory Generation Experiments

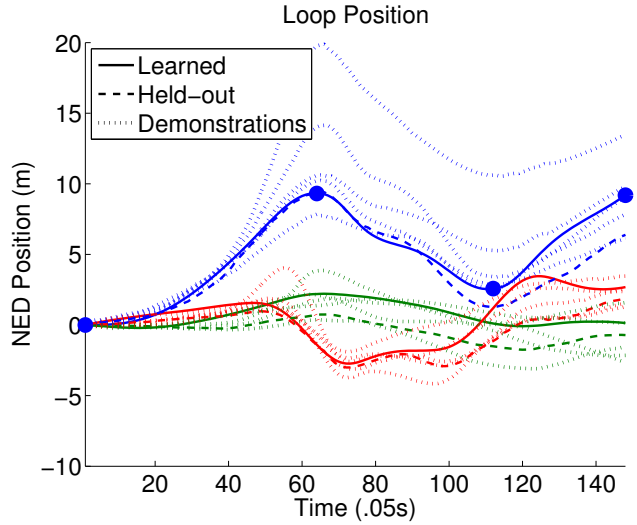
Our experiments investigate the ability of our algorithm to reproduce held-out demonstration trajectories. We gather a set of demonstrations for each maneuver class. For each run, we choose one demonstration and evaluate how well our algorithm can reproduce it based upon the other demonstrations from its maneuver class. We first extract the input parameters (e.g., the altitude of the target stall turn) from the



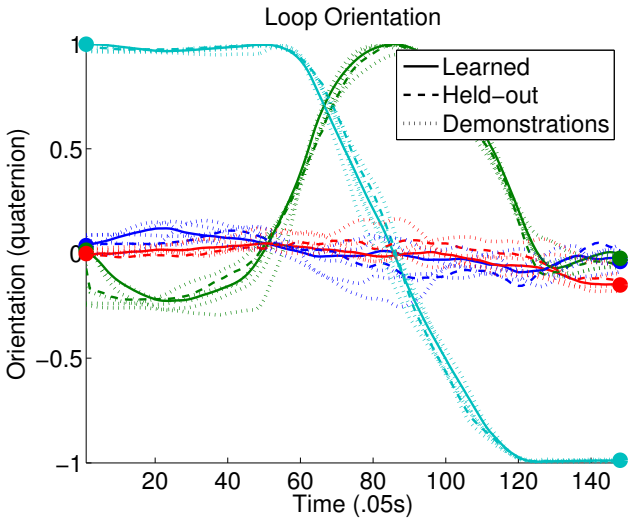
(a)



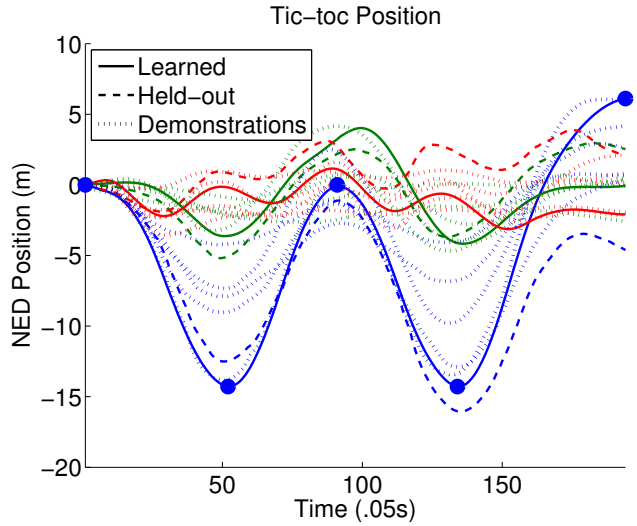
(d)



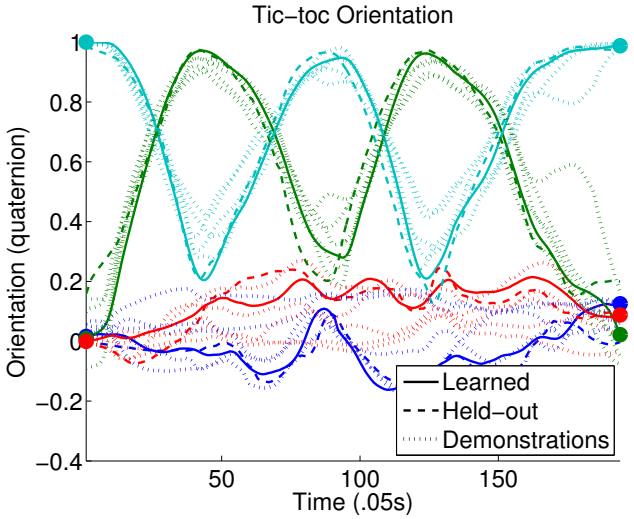
(b)



(e)



(c)



(f)

Fig. 3. Trajectory learning on a stall turn (top), loop (middle), and tic-toc (bottom). Each plot shows the demonstration trajectories (dotted), the held-out test trajectory (dashed), and the learned trajectory (solid). For the position plots, blue, green, and red represent north, east, and down, respectively. For the orientation plots, blue, green, red, and cyan represent q_x, q_y, q_z, and q_w, respectively.

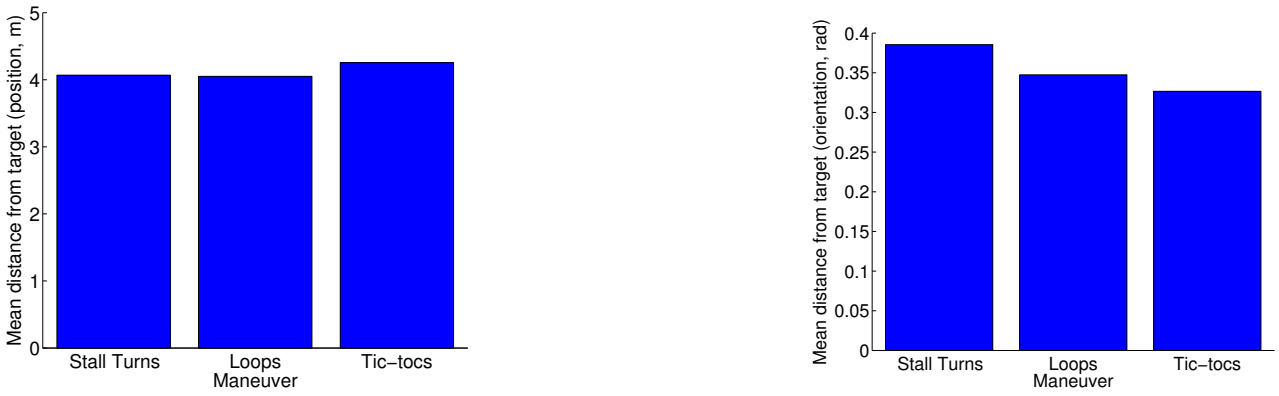


Fig. 4. Distance from target in position (left) and orientation (right).

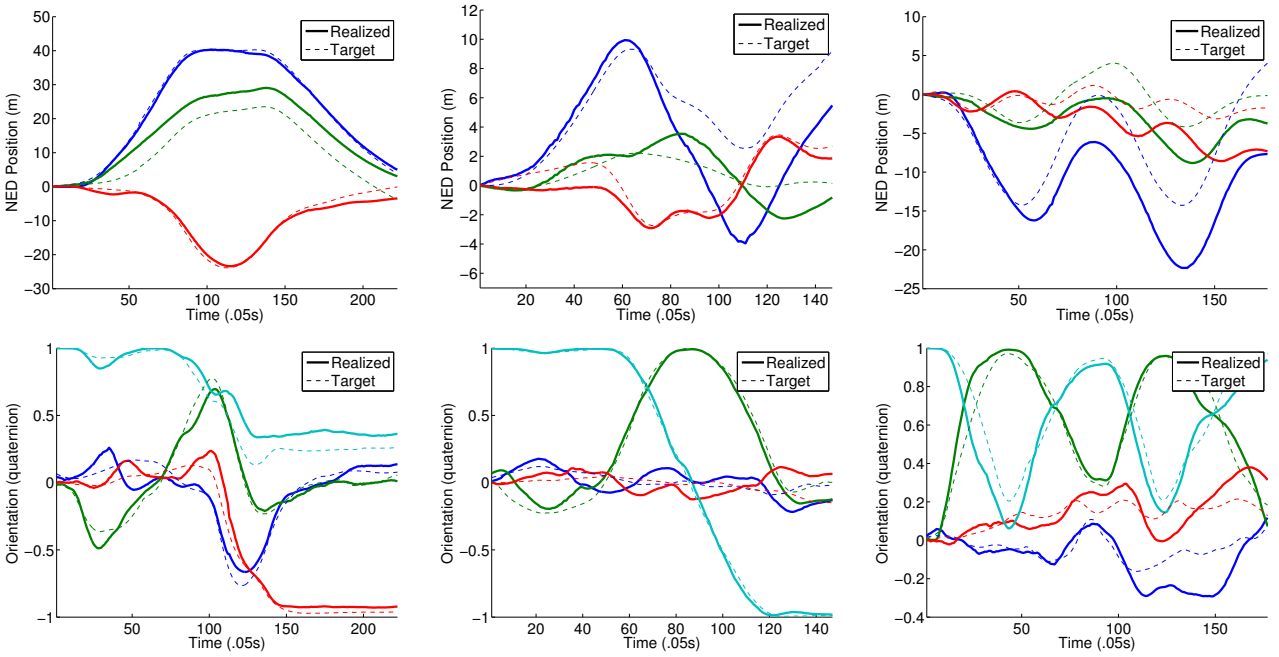


Fig. 5. Representative flight performance of our approach. Left to right: stall turn, loop, tic-toc.

held-out demonstration.[3] We then run our algorithm while excluding the held-out demonstration from being used by our algorithm and compare the generated trajectory to this held-out demonstration.

Figure 3 shows some representative generated target trajectories of stall turns, loops, and tic-tocs obtained using our method. The plots include the held-out trajectory aligned to the learned trajectory as well as the waypoint constraints. The generated trajectories closely match the held-out trajectories at the waypoints.

D. Flight Experiments

We investigate the realized flight performance of our generated target trajectories.

We flew three trajectories each for stall turns, loops, and tic-tocs. For each type of maneuver we chose a small, medium, and large demonstration to use as held-out trajectories, and picked a disjoint set of demonstration trajectories to learn from. We had our helicopter fly each trajectory twice. Figure 4 shows the average distance from the target in position and orientation. Figure 5 shows representative flights for each maneuver for our method.

Videos of our flight results are available at the following URL:

`http://rll.eecs.berkeley.edu/heli/icra10`

[3]For tic-tocs, we take the average of the two local minima on the x-axis and use this value as a single parameter.

V. RELATED WORK

The work by Coates *et al.* [8] is the most closely related. They consider the setting of learning a single intended trajectory and a high-precision dynamics model along that trajectory from several demonstrations. However, our probabilistic approach can generate many similar, parameterized maneuvers instead of a single desired trajectory. We also leverage their observation that, for a specific maneuver, it is

possible to obtain a very high fidelity dynamics model by combining a crude low-order dynamics model with corrections specific to that trajectory. A minor difference is the handling of time warping: Coates *et al.* use dynamic time warping with a cost function based on the log-likelihood of the sequence in question. This worked well when the demonstrations were all very similar, but we found that our weighted squared-error cost function with rate-change penalty yielded better alignments in our setting, in which the demonstrations were far less similar in size and time scale.

In [6], multiple demonstrations are used to learn a model for a robot arm and find an optimal controller in their simulator, initializing their optimal control algorithm with one of the demonstrations.

The work of [7] considered learning trajectories and constraints from demonstrations for robotic tasks. They do not consider the system's dynamics or provide a clear mechanism for the inclusion of prior knowledge.

Among others, [4] and, more recently, [3], [8] have exploited the idea of trajectory-indexed model learning for control.

Our work also has some similarities with recent work on inverse reinforcement learning, which extracts a reward function (rather than a trajectory) from the expert demonstrations. See, e.g., [2], [13]–[16], [20].

VI. CONCLUSIONS

We have proposed a generally applicable method for learning parameterized maneuvers that makes efficient use of expert demonstrations to learn a maneuver-specific local control model. We applied our algorithm to autonomous helicopter flight. Our algorithm has enabled our autonomous helicopter to fly challenging aerobatic maneuvers of different sizes from the same set of expert demonstrations.

VII. ACKNOWLEDGMENTS

We thank Garett Oku for piloting and building our helicopter platforms. Jie Tang is supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

REFERENCES

[1] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *Accepted for publication in the International Journal of Robotics Research*, 2010.

[2] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, 2004.

[3] P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *Proc. ICML*, 2006.

[4] C. H. An, C. G. Atkeson, and J. M. Hollerbach. *Model-Based Control of a Robot Manipulator*. MIT Press, 1988.

[5] O. Arikan and D. Forsyth. Interactive motion generation from examples, 2002.

[6] C. Atkeson and S. Schaal. Robot learning from demonstration. In *Proc. ICML*, 1997.

[7] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. In *IEEE Trans. on Systems, Man and Cybernetics, Part B*, volume 37, 2007.

[8] A. Coates, P. Abbeel, and A. Y. Ng. Learning for control from multiple demonstrations (Full version). *ICML*, pages 144–151, 2008. http://heli.stanford.edu/icml2008.

[9] A. C. Fang and N. S. Pollard. Efficient synthesis of physically valid human motion, 2003.

[10] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *First SIAM International Conference on Data Mining (SDM)*, 2001.

[11] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. In *Proc. SIGGRAPH*, 2002.

[12] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. MIT Press, 1999.

[13] G. Neu and C. Szepesvari. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proc. UAI*, 2007.

[14] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. ICML*, 2000.

[15] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proc. IJCAI*, 2007.

[16] N. Ratliff, J. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proc. ICML*, 2006.

[17] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. *Computer Graphics*, 1996.

[18] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 1970.

[19] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1978.

[20] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. In *NIPS 20*, 2008.

[21] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. *Computer Graphics*, 1995.

[22] D. J. Wiley and J. K. Hahn. Interpolation synthesis for articulated figure motion. *Virtual Reality Annual International Symposium*, 1997.