Planning Preference-balancing Motions with Stochastic Disturbances

Aleksandra Faust[1], Nick Malone[1], and Lydia Tapia[1]

***Abstract*— Robots that operate in real-world conditions often perform complex tasks in the presence of stochastic disturbances. The source of the disturbances can be widely varied, including but not limited to, hardware imperfections, atmospheric changes, and measurement inaccuracies. These disturbances pose a great control challenge because stochastic drift induces changes in the robot's speed and direction. This paper presents an online trajectory generation method for robots to complete *preference-balancing* tasks under stochastic disturbances. Task learning is done off-line assuming no disturbances, and then trajectories are planned online in the presence of disturbances using the current observed information. We model the robot as a stochastic control-affine system with unknown dynamics impacted by a Gaussian process. This paper introduces a supervised machine learning method in lieu of a traditional greedy policy. We verify the method in simulation for an aerial vehicle cargo delivery and a flying inverted pendulum task. Results show the presented method works on a range of problems and outperforms the deterministic method in the presence of non-zero mean disturbances.**

I. INTRODUCTION

Real-world conditions pose many challenges to physical robots. One such challenge is the introduction of stochastic disturbances that can cause positional drift. For example, atmospheric changes, hardware wear-and-tear or measurement inaccuracies are possible sources of stochastic disturbances [3]. Disturbances, along with complex nonlinear system dynamics, make traditional solutions (e.g., adaptive and robust control modeling), which solve this problem using full system dynamics knowledge, difficult or intractable [16].

We are concerned with a specific class of robotic motion plannings tasks described with set of preferences, *preference-balancing tasks* (PBT) [9]. PBTs have a single goal (destination) state, but the trajectory that completes the task needs to meet opposing preferences, such as speed and quality. As a motivational example, we consider a quadrotor with a suspended load task (Fig. 1a). The goal of this task is to fly a quadrotor to a goal state while minimizing the residual oscillations of the freely suspended load. With its complicated dynamics, this problem is difficult for a human to demonstrate, which renders impractical methods that rely on expert demonstration [1]. This task is a PBT because it requires balancing opposite preferences, e.g., moving the quadrotor and not agitating the cargo.

This paper presents a novel online trajectory generation for PBTs under stochastic disturbances. The method uses a state-value function approximation learned deterministically off-line with reinforcement learning while assuming *no* stochastic disturbances. The learned state-value function approximation is then used to generate trajectories that compensate for non-zero mean stochastic disturbances. The method bridges the gap between off-line motion planning [17] and controls-based trajectory tracking [3]. Traditionally, off-line planning produces either a reference path or trajectory assuming a stationary environment [17]. Trajectory tracking methods stabilize the system around the reference trajectory by adjusting for experienced disturbances [3]. For the standard planning-tracking pipeline to work for PBTs, the reference trajectory must complete the task, and the trajectory tracking must minimize the same task preferences used in the planning phase. The method we propose combines trajectory planning and tracking into one step, *online trajectory planning*. Online trajectory planning has two advantages over the traditional planning-control pipeline. First, *online trajectory planning* does not require replanning if the goal changes. Second, the method is model-free, meaning that it does not rely on the analytical knowledge of the system dynamics, and it generates trajectories without learning the system model, in contrast to learning-based MPC methods [21].

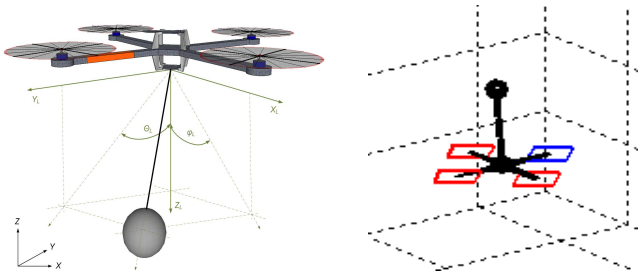
Robotic systems are high-dimensional systems, most naturally represented with continuous states and actions. Here, we model the system as an unknown control-affine system [16] with continuous states and actions. The system is controlled through acceleration. We assume that we are able to interact with the unknown system dynamics through a simulator or available samples. A Gaussian process, defined with its mean and variance, is used to model the disturbance of the input [3]. We also assume that the probability distribution can be measured outside of the planning algorithm [25]. This can be done, for example, by measuring true acceleration of the system with an accelerometer and estimating the error, the difference between the observed and the input acceleration. The system should perform a given task for a range of initial conditions. The goal of this work is to be able to take a task learned with deterministic assumptions, and adapt it to perform in real-time in the presence of changing disturbances. The deterministic learning decides the priorities between the preferences in a deterministic environment. These learned priorities are then used in an online planning phase in the presence of disturbances. The priorities remain unchanged when disturbance are added to the system. The online planning phase uses another layer of learning to find inputs that compensates for the disturbances in order to produce a motion that adheres to the preferences' priorities.

Our previous work developed an *Axial Sum* planner for deterministic control-affine systems [11]. The planner works in a batch reinforcement leaning (RL) setting, using an con-

*Faust was supported in part by NM Space Grant; Malone and Tapia were supported in part by the National Institutes of Health (NIH) Grant P20RR018754 to the Center for Evolutionary and Theoretical Immunology.

[1]The authors are with Computer Science Department, University of New Mexico, USA {afaust,nmalone,tapia}@cs.unm.edu

The authors would like to thank Peter Ruymgaart for helpful discussions on modeling the external disturbances.



(a) Quadrotor with suspended load (b) Inverted pendulum

Fig. 1. preference-balancing task examples.

tinuous action fitted value iteration (CAFVI) [11], an approximate value iteration method adapted for continuous action Markov Decision Processes (MDP). Typically, approximate value iteration-based methods learn an approximation of the state-value function, $\boldsymbol{V}$, off-line. Then in a separate phase, they plan trajectories using a greedy policy with respect to the learned state-value function approximation [6]. In our case the state-value function is approximated with a linear map of features that are selected to be squared preferences [9]. For instance, the features for the balancing inverted pendulum task (Fig. 1b) are squares of the pendulum's displacement from the upright position, pendulum's velocity, and vehicle's velocity. These features are then weighted by CAFVI. In the planning phase, we used the *Axial Sum* policy to approximate a greedy policy by finding the best actions on each axis and then combining them together. We showed that the planner based on the *Axial Sum* policy leads the system to the goal when CAFVI, applied to a control-affine system with a bounded drift and quadratic features, results in all negative weights [11]. Although we showed that the *Axial Sum* can compensate for some levels of zero-mean noise [11], [12], the method stops working in the presence of external disturbances. This is because, the external disturbance produces larger than the allowed drift onto the system. For the *Axial Sum* planner, Lagrangian interpolation finds the best action on each axis. Here, we propose using supervised machine learning to find the best action with respect to the current disturbance. The key extension from [11] is the use of least squares linear regression in lieu of interpolation to estimate near-optimal action on each axis. This extension allows us to apply the method to non-zero mean disturbances with the only limits being the system's physical limits.

II. RELATED WORK

Our method uses ideas from reinforcement learning, motion planning, and control theory. However, it differs from the classical control methods such as LQR, perturbation techniques, and adaptive controls which all require knowledge (analytical or learned) of the system dynamics and state-value function [3]. For example, piecewise linearization has been used for quadrotor trajectory tracking under wind-gust disturbances [2]. Another prior approach requires knowledge of the system dynamics and uses harmonic potential fields for UAV motion planning in environments with a drift field [20]. Similarly, [23] solves the system's dynamics. In contrast, our method is model-free.

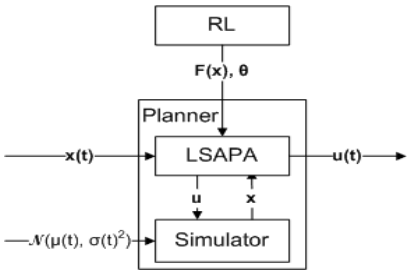


Fig. 2. Flow diagram for learning and planning preference-balancing tasks.

Path planning and obstacle avoidance in the presence of stochastic wind for a blimp was solved using dynamic programming and augmented MDPs [15]. Other methods to handle motion planning and trajectory generation under uncertainties use low-level controllers for stabilization of trajectories within reach tubes [8], or trajectory libraries [18]. The flying inverted pendulum with zero-mean disturbances was solved using RL [12] and first principles [4], [14].

A new class of sampling based planning methods optimistically narrows the search space [5], [7], [19], [26]. Gradient descent methods for policy approximation work well in some convex cases. However, they require an estimate of the gradient, can be stuck in local minima, and can be slow to converge [13]. In this paper, we show a fast policy approximation that works for near-linear objective functions.

III. PRELIMINARIES

Our goal is to plan a preference-balancing task on a control-affine system in the presence of an external stochastic disturbance. Figure 2 describes the planner's flow. We assume that RL provides a feature vector, $\boldsymbol{F}$, and weights, $\boldsymbol{\theta}$, learned with a method with no disturbances, such as CAFVI with Axial Sum policy [11]. In the planning phase, we assume that we have a black-box simulator of the system and know the current probability distribution of the disturbance. For instance, assuming the presence of an accelerometer, let $M = [\hat{\ddot{\boldsymbol{x}}}_{t-k}\ \hat{\ddot{\boldsymbol{x}}}_{t-k} \dots \hat{\ddot{\boldsymbol{x}}}_{t-1}]^T$ be a sequence of acceleration observations, and $T = [\ddot{\boldsymbol{x}}_{t-k}\ \ddot{\boldsymbol{x}}_{t-k} \dots \ddot{\boldsymbol{x}}_{t-1}]^T$ be input applied to the system by the planner at time steps $t-k, \dots, t-1$. Then the mean of the disturbance at the time step t can be estimated as $\mu(t) = \mu_M - \mu_T$ with variance as $\sigma^2(t) = \text{Var}(M - T)$ (the difference between the observed and applied input). Another more precise method to estimate the input error is using a Kalman filter [16].

The planner generates trajectories for a physical system. At every time step, t, the proposed method, Least Squares Axial Policy Approximation (LSAPA), observes a state, $\boldsymbol{x}(t)$, while the simulator receives current disturbance levels, $\mathcal{N}(\mu(t), \sigma(t)^2)$. Sampling the simulator, LSAPA finds a near-optimal input, $\boldsymbol{u}(t)$, to apply to the system.

We model a robot as a discrete time, control-affine system with stochastic disturbance, $\boldsymbol{D} : X \times U \to X$,

$$\boldsymbol{D}: \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)(\boldsymbol{u}_k + \boldsymbol{\eta}_k). \tag{1}$$

States $\boldsymbol{x}_k \in X \subseteq \mathbb{R}^{d_x}$ belong to the position-velocity space and the control input is acceleration, $\boldsymbol{u}_k \in U \subseteq \mathbb{R}^{d_u}$. The input space is a compact set containing origin, $\mathbf{0} \in U$. The Lipschitz continuous function $\boldsymbol{g} : X \to \mathbb{R}^{d_x} \times \mathbb{R}^{d_u}$ is regular

outside the origin, $\boldsymbol{x}_k \in X \setminus \{\mathbf{0}\}$. The drift $\boldsymbol{f} : X \to \mathbb{R}^{d_x}$, is bounded and Lipschitz continuous. The non-deterministic term, $\boldsymbol{\eta}_k$, is a Gaussian process with with a known mean and distribution $\mathcal{N}(\mu_{\boldsymbol{\eta}_k}, \sigma^2_{\boldsymbol{\eta}_k})$; it acts as an additional and unpredictable external force on the system. Time step k is omitted from the notation when possible.

As in [11], our goal is to learn a preference-balancing task that takes the system to the origin in a timely-manner while reducing along the trajectory preferences given by matrix $\boldsymbol{A}\boldsymbol{x} = [\boldsymbol{a}_1 \dots \boldsymbol{a}_{d_g}]$. Each of the vectors $\boldsymbol{a}_i$ defines a task preference. For instance, vector $\boldsymbol{a}_i$ that corresponds to preference to reduce the displacement of the inverted pendulum on the quadrotor, will have components that correspond to the position of the pendulum be set to one, while the rest of the components will be equal to zero.

The state-value function approximation is

$$V(\boldsymbol{x}) = \sum_{i=1}^{d_g} \theta_i F_i(\boldsymbol{x}). \tag{2}$$

Vector $\boldsymbol{F}(\boldsymbol{x}) = [F_1(\boldsymbol{x}), ..., F_{d_g}(\boldsymbol{x})]^T$ is a feature vector, and $\boldsymbol{\theta} = [\theta_1, ..., \theta_{d_g}]^T$ is the parametrization that we learn. The feature vector is selected with the task in mind,

$$F_i(\boldsymbol{x}) = \|\boldsymbol{a}_i^T \boldsymbol{x}\|^2, \quad i = 1, ..., d_g. \tag{3}$$

Greedy policy, $\boldsymbol{h}^*(\boldsymbol{x}) = \operatorname{argmax}_{\boldsymbol{u}\in U} V(D(\boldsymbol{x}, \boldsymbol{u}))$ is optimal with respect to the state-value function V. The problem is that in continuous action spaces greedy policy calculation becomes an optimization problem over an unknown objective function $V \circ D$.

RL literature often works with *action-value function*, $Q : X \times U \to \mathbb{R}$, a measure of the discounted accumulated reward collected when action $\boldsymbol{u}$ is taken at state $\boldsymbol{x}$ [24]. In relation to state-value function, V (2), action-value Q can be represented as

$$Q(\boldsymbol{x}, \boldsymbol{u}) = V(D(\boldsymbol{x}, \boldsymbol{u})) = . \sum_{i=1}^{d_g} \theta_i \boldsymbol{F_i}(D(\boldsymbol{x}, \boldsymbol{u})) \tag{4}$$

Thus, we learn the approximation for the greedy policy

$$\boldsymbol{h}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{u}\in U} Q(\boldsymbol{x}, \boldsymbol{u}). \tag{5}$$

IV. METHODS

The *Least squares axial policy approximation* (LSAPA) policy extends the method of [11] to handle non-zero mean disturbances. This is done by first learning feature weights off-line *without* disturbances and then using those learned weights for online trajectory planning *with* disturbances. LSAPA bridges the gap between learning without disturbances and planning with them. The method in [11] is applicable to zero-mean disturbances due to the use of Lagrangian interpolation to find an approximation to the maximal Q value. The Lagrangian interpolation uses only three points to interpolate the underlying quadratic function and this compounds the error from the disturbances. In contrast, our new method, LSAPA, uses least squares regression with many sample points to compensate for the induced error.

Specifically the method in [11] describes *an axial sum* policy. Consider a fixed arbitrary state $\boldsymbol{x}$, in a control-affine system (1) with state-value approximation (2), action-value function, Q, is a quadratic function of the input $\boldsymbol{u}$ [11]. Axial sum policy approximation [11] finds an approximation for the maximum local Q function for a fixed state x. It works in two steps, first finding maxima on each axis independently and then combining them together. To find a maximum on an axis, the method uses Lagrangian interpolation to find the coefficients of the quadratic polynomial representing the Q function. Then, an action that maximizes the Q function on each axis is found by zeroing the derivative. The final policy is a piecewise maximum of a convex and simple vector sums of the action maxima found on the axes. The method is computationally-efficient, scaling linearly with the action space dimensionality $O(d_u)$. It is also consistent, as the maximum selections do not depend on the selected samples.

Because deterministic axial policies are sample independent, they do not adapt to changing conditions or external forces. We extend the deterministic axial policies to the presence of disturbances via LSAPA. LSAPA uses least squares regression, rather than Lagrangian interpolation, to select the maximum on a single axis. This change allows the LSAPA method to compensate for the error induced by non-zero mean disturbances. We now present finding the maximum on ith axis using the least squares linear regression with polynomial features.

Definition *Q-axial restriction* on i^{th} axis is a univariate function $Q_{\boldsymbol{x},i}(u) = Q(\boldsymbol{x}, u\boldsymbol{e_i})$.

Q-axial restriction on i^{th} axis is a quadratic function,

$$Q_{\boldsymbol{x},i}(u) = \boldsymbol{p}_i^T [u^2 \; u \; 1]^T,$$

for some vector $\boldsymbol{p}_i = [p_{2,i} \; p_{1,i} \; p_{0,i}]^T \in \mathbb{R}^3$ based on results in [11]. Our goal is to find $\boldsymbol{p}_i$ by sampling the input space U at fixed state.

Suppose, we collect d_n input samples in the i^{th} axis, $U_i = [u_{1,i} \; ... \; u_{d_n,i}]^T$. The simulator returns state outcomes when the input samples are applied to the fixed state $\boldsymbol{x}$, $X_i = [\boldsymbol{x}'_{1,i} \; ... \; \boldsymbol{x}'_{d_n,i}]^T$, where $\boldsymbol{x}'_{j,i} \leftarrow D(\boldsymbol{x}, u_{j,i}), \quad j = 1, ..., d_n$. Next, Q-estimates are calculated with (4),

$$\boldsymbol{Q_i} = [Q_{\boldsymbol{x},1}(u_{1,i}) \; ... \; Q_{\boldsymbol{x},d_n}(u_{d_n,i})]^T,$$

where $Q_{\boldsymbol{x},j}(u_{j,i}) = \theta^T F(\boldsymbol{x}'_{j,i}), \quad j = 1, ..., d_n$. Using the supervised learning terminology the Q estimates, $\boldsymbol{Q_i}$, are the labels that match the training samples U_i. Matrix,

$$C_i = \begin{bmatrix} (u_{1,i})^2 & u_{1,i} & 1 \\ (u_{2,i})^2 & u_{2,i} & 1 \\ & \dots & \\ (u_{d_n,i})^2 & u_{d_n,i} & 1 \end{bmatrix},$$

contains the training data projected onto the quadratic polynomial space. The solution to the supervised machine learning problem,

$$C_i \boldsymbol{p}_i = \boldsymbol{Q_i} \tag{6}$$

fits $\boldsymbol{p}_i$ into the training data C_i and labels $\boldsymbol{Q_i}$. The solution to (6),

$$\hat{\boldsymbol{p}}_i = \operatorname*{argmin}_{\boldsymbol{p}_i} \sum_{j=1}^{d_n} (C_{j,i}\boldsymbol{p}_i - Q_{\boldsymbol{x},j}(u_{j,i}))^2 \tag{7}$$

is the coefficient estimate of the Q-axial restriction. Because Q is quadratic, we obtain its critical point by zeroing the

first derivative,

$$\hat{u}_i^* = -\frac{\hat{p}_{1,i}}{2\hat{p}_{2,i}}.$$

Lastly, we ensure that the action selection falls within the allowed action limits,

$$\hat{u}_i = \min(\max(\hat{u}_i^*, u_i^l), u_i^u), \tag{8}$$

where u^l and u^u are lower and upper acceleration bound on the i^{th} axis, respectively.

Repeating the process of estimating the maxima on all axes and obtaining $\hat{\boldsymbol{u}}_{\boldsymbol{i}} = [\hat{u}_1, ..., \hat{u}_{d_u}]$, we calculate the final policy with

$$\hat{\boldsymbol{h}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{h}_{\boldsymbol{c}}^Q(\boldsymbol{x}), & Q(\boldsymbol{x}, \boldsymbol{h}_{\boldsymbol{c}}^Q(\boldsymbol{x})) \geq Q(\boldsymbol{x}, \boldsymbol{h}_{\boldsymbol{n}}^Q(\boldsymbol{x})) \\ \boldsymbol{h}_{\boldsymbol{n}}^Q(\boldsymbol{x}), & \text{otherwise} \end{cases} \tag{9}$$

where

$$\boldsymbol{h}_{\boldsymbol{n}}^Q(\boldsymbol{x}) = \sum_{i=1}^{d_u} \hat{u}_i \boldsymbol{e_i}, \qquad \text{(non-convex policy)}$$

$$\boldsymbol{h}_{\boldsymbol{c}}^Q(\boldsymbol{x}) = d_u^{-1} \boldsymbol{h}_{\boldsymbol{n}}^Q(\boldsymbol{x}) \qquad \text{(convex policy)}$$

The policy approximation (9) combines the simple vector sum of the non-convex policies (8) with the convex sum policy. The convex sum guarantees the system's monotonic progression towards the goal, but the simple vector sum (non-convex policy) does not [11]. If, however, the vector sum performs better than the convex sum policy, then (9) allows us to use the better result.

V. RESULTS

To evaluate online trajectory planning using LSAPA, we use two tasks, aerial cargo delivery and balancing a flying inverted pendulum. Both tasks are learned with a deterministic CAFVI [11], and here we evaluate planning under varying non-zero mean disturbances and compare it to the baseline deterministic axial policy [11]. Due to space limitations, full problem definitions are omitted and are the same as in [11] for aerial cargo delivery and as in [12] for the flying inverted pendulum. All learning and planning occurs at 50 Hz. All trajectory planning was performed on a single core of Intel Core i7 system with 8GB of RAM, running the Linux operating system using Matlab 2011.

A. Swing-free aerial cargo delivery

We first consider a swing-free aerial cargo delivery task. The task requires a quadrotor carrying a load on a suspended rigid cable, to deliver the cargo to a given location with the minimal residual load oscillations [10]. The task has applications in delivery supply and aerial transportation in urban environments. The task is easily described. Yet, it is difficult for human demonstration as it requires a careful approach to avoid destabilizing the load. Although we evaluate in simulation only, the fidelity of the simulator was confirmed experimentally in [11], [10], [22]. The results of the experimental studies show that the simulator's predictions of the load are within 5° of the experimental observations, while the simulator's predictions of the quadrotor's center of mass are within $1cm$ [11], [10], [22].

We use the same features as in [11]. Feature vector $\boldsymbol{F}$, consists of the position of the quadrotor relative to the goal $\|p\|^2$, the quadrotor's speed $\|v\|^2$, the position of the load relative to the quadrotor $\|\eta\|^2$, and the load's speed $\|\dot{\eta}\|^2$ [10]. The state space is a 10 dimensional vector of the UAV's and load's positions and velocities. The action space is the three dimensional acceleration of the quadrotor's center of the mass with a maximum acceleration of 3 meters per second squared. We learn the task using a deterministic CAFVI, which results in the weights $\boldsymbol{\theta} = [-86290 \; -350350 \; -1430 \; -1160]^T$.

To test the quality of planning under external disturbances, we plan 100 trajectories starting 5m from the goal using the learned weights, $\boldsymbol{\theta}$, and varying disturbance parameters. The trajectories are 10 seconds long and start at (-2 , 2, 1) meters from the origin. We compare the proposed LSAPA planner to a deterministic Axial Sum [11]. Table I shows the characteristics of the resulting trajectories. Due to the constant presence of the disturbance, we consider the average position of the quadrotor and the load over the last second, rather than simply expecting to reach the goal region. Note the accumulated squared error, typically used to measure quality of tracking methods, is not appropriate for LSAPA because of the lack of a reference trajectory. Thus, we measure if the system arrives and stays near the goal. The results in Table I show that planning time with LSAPA is an order of magnitude smaller than the 10 second trajectory duration, allowing ample time to plan the trajectory in a real-time closed feedback loop. This is because we do not need the entire trajectory preplanned in real-time, only the next input. The planning time for the deterministic Axial Sum policy is faster than planning with LSAPA. This is expected because the deterministic policy uses 3 samples per input dimension, while the stochastic policy in this case uses 300 samples. Next in Table I, we see that the stochastic policy produces consistent residual distance to the goal. The larger the variance of the disturbance, the larger the error. When the mean is 2 m/s^2 and the standard deviation is 1, the stochastic policy results start degrading. This is because the upper limit on the action space is 3 m/s^2, and the drift starts overwhelming the system. The deterministic policy, learning and acting on the same data, fails to bring the system near the goal. As expected, the two policies show similar behavior only for the zero-mean noise with small (0.5) variance.

Figure 3 shows the trajectories planned with LSAPA and a deterministic Axial Sum in environment with $\mathcal{N}(2, 0.5^2)$ disturbance. Although both the quadrotor's and the load's speeds are noisy (Figures 3a and 3b), the position changes are smooth, and the quadrotor arrives near the goal position where it remains. This is in contrast to trajectories planned with a deterministic axial sum that never reach the origin.

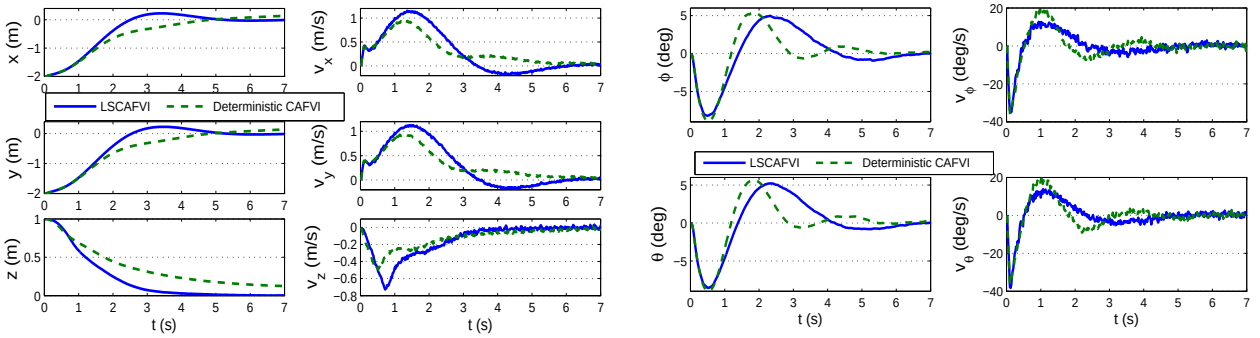
B. Flying Inverted Pendulum

Another task we consider is a flying inverted pendulum. It consists of a quadrotor-inverted pendulum system in a plane. The goal is to stabilize the pendulum and keep it balanced as the quadrotor hovers [12]. We split the task in two: the pole stabilization and quadrotor slowdown. The features for the first task are squares of the pendulum's position and velocity relative to the goal upright position. The second task has an

TABLE I

SUMMARY OF PLANNING RESULTS FOR SWING-FREE DELIVERY TASK AVERAGED OVER 100 TRIALS. POLICY, INJECTED DISTURBANCE PROCESS DISTRIBUTION, TIME TO PLAN A TRAJECTORY, AVERAGE DISTANCE FROM THE GOAL AND LOAD DISPLACEMENT DURING THE LAST 1 SECOND OF THE FLIGHT, AND MAXIMUM LOAD DISPLACEMENT.

Policy	Dist.		Planning Time (s)		Distance (cm)		Swing (°)		Max. Swing (°)	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
LSAPA	0.00	0.50	0.79	0.20	1.36	0.47	0.35	0.17	12.63	0.07
	1.00	0.50	0.88	0.22	1.38	0.28	0.17	0.06	12.49	0.06
	2.00	0.50	0.84	0.09	1.22	0.21	0.17	0.07	12.86	0.16
	0.00	1.00	1.05	0.28	1.67	0.72	0.26	0.13	12.59	0.16
	1.00	1.00	1.06	0.07	3.71	1.24	0.27	0.09	12.54	0.15
	2.00	1.00	1.07	0.17	10.81	2.53	0.47	0.16	13.29	0.38
Deterministic Axial Sum	0.00	0.50	0.37	0.04	1.49	0.55	0.44	0.20	12.69	0.05
	1.00	0.50	0.53	0.03	14.78	0.45	0.11	0.03	13.12	0.06
	2.00	0.50	0.53	0.01	29.36	0.45	0.11	0.04	13.84	0.07
	0.00	1.00	0.46	0.07	1.25	0.49	0.28	0.17	12.69	0.11
	1.00	1.00	0.52	0.00	14.91	0.81	0.20	0.06	13.16	0.11
	2.00	1.00	0.53	0.01	31.44	1.23	0.27	0.09	13.90	0.14



(a) Quadrotor trajectory

(b) Load trajectory

Fig. 3. Cargo delivery task - trajectory created with LSAPA compared to a trajectory created with deterministic axial sum with disturbance of $\mathcal{N}(2, 0.5^2)$

additional feature of a square of the quadrotor's velocity. The state space is a vector of the quadrotor's velocity, and the pendulum's position and velocity. The action space is a two dimensional vector of quadrotor's acceleration in a plane horizontal to the ground. The maximum acceleration is 5 m/s^2. The reward is one when the target zone is reached, and zero otherwise. The simulator used is a linearized model of the full dynamics of a planar flying inverted pendulum. With the exception of the maximum acceleration, the set up above is the same as in [12], and the policy used for learning is the deterministic axial sum.

In the planning phase, we use a disturbance probability density function $\mathcal{N}(1, 1^2)$ and a pole initial displacement of 23°. While the deterministic sum solves this problem and balances the inverted pendulum in the absence of disturbances and small zero-mean disturbances ($\mathcal{N}(0, 0.5^2)$), it fails to balance the inverted pendulum for non-zero mean disturbances. In contrast, LSAPA policy solves the task (Fig. 4). Fig. 4a shows the quadrotor's trajectory, and Fig. 4b displays pendulum position in Cartesian coordinates relative to the target position above the quadrotor. The first subtask brings the pole upright (0 to 5 seconds). Then the second subtask slows down the quadrotor (after 5 seconds). The pole is slightly disturbed during the initial moments of the second subtask but returns to an upright position.

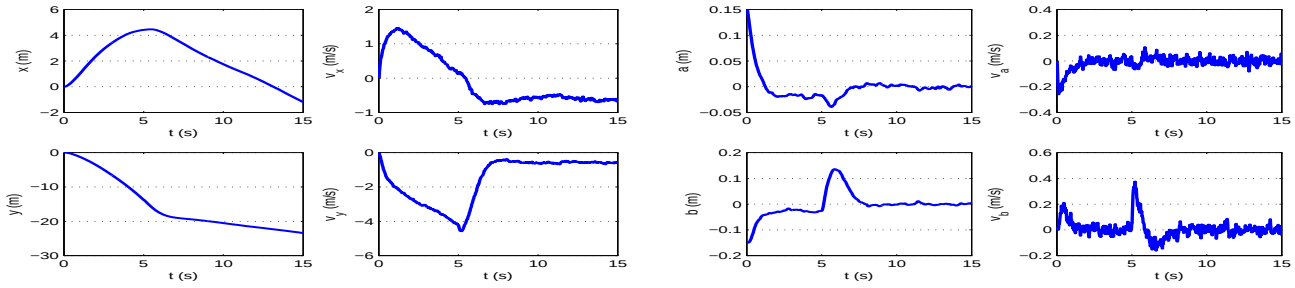
Figure 5 depicts the results of the trajectory characteristics for increasing number of samples in LSAPA. The smallest number of samples is three. The accumulated reward (Fig. 5a) increases exponentially below 10 samples. The gain decreases between 10 and 20 samples. Thus, the peak performance is reached after 20 samples. Sampling beyond that point brings no gain. We see the same trend with the pole displacement (Fig. 5b) and speed magnitude (Fig. 5c).

VI. CONCLUSIONS

We presented a novel method for policy approximation for robots performing preference-balancing tasks in environments with external stochastic disturbances. This policy allows the system to adapt to changing external disturbances due to atmospheric changes or deteriorating hardware. Feature weights are learned off-line (without stochastic disturbances) and then the method uses least squares linear regression to find an optimal action on each axis (with stochastic disturbances). The resulting action is a combination of the axial maxima. This paper takes an empirical approach to assess the safety of the policy. In the preliminary results, we showed that the method is applicable for a non-trivial practical problems.

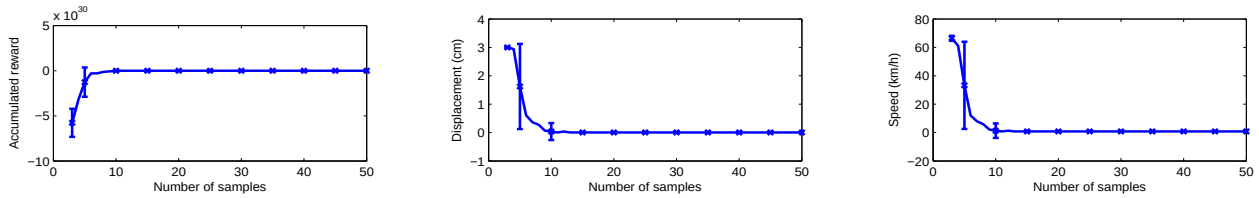
REFERENCES

[1] P. Abbeel. *Apprenticeship learning and reinforcement learning with application to robotic control*. PhD thesis, Stanford University, Stanford, CA, USA, 2008.



(a) Quadrotor trajectory (b) Inverted trajectory

Fig. 4. Flying inverted pendulum trajectory created with stochastic axial sum policy with distrubance of $\mathcal{N}(1, 1^2)$.



(a) Accumulated reward (b) Displacement (c) Speed

Fig. 5. Trajectory characteristic per number of samples in flying pendulum with disturbance $\mathcal{N}(1, 1^2)$, calculated LSAPA; mean and standard deviation over 100 trials shown.

[2] K. Alexis, G. Nikolakopoulos, and A. Tzes. Constrained-control of a quadrotor helicopter for trajectory tracking under wind-gust disturbances. In *MELECON 2010-2010 15th IEEE Mediterranean Electrotechnical Conference*, pages 1411–1416. IEEE, 2010.

[3] K. J. Astrom. *Introduction to Stochastic Control Theory*. Technology & Engineering. Academic Press, 1970.

[4] D. Brescianini, M. Hehn, and R. D'Andrea. Quadrocopter pole acrobatics. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3472–3479. IEEE, 2013.

[5] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-armed bandits. *J. Mach. Learn. Res.*, 12:1655–1695, July 2011.

[6] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[7] L. Busoniu, A. Daniels, R. Munos, and R. Babuska. Optimistic planning for continuous-action deterministic systems. In *2013 Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, in press 2013.

[8] J. A. DeCastro and H. Kress-Gazit. Guaranteeing reactive high-level behaviors for robots with complex dynamics. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 749–756. IEEE, 2013.

[9] A. Faust. *Reinforcement Learning and Planning for Preference Balancing Tasks*. PhD thesis, University of New Mexico, Albuquerque, NM, July 2014.

[10] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia. Learning swing-free trajectories for uavs with a suspended load. In *IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, pages 4887–4894, May 2013.

[11] A. Faust, P. Ruymgaart, M. Salman, R. Fierro, and L. Tapia. Continuous action reinforcement learning for control-affine systems with unknown dynamics. *Acta Automatica Sinica, in press*, 2014.

[12] R. Figueroa, A. Faust, P. Cruz, L. Tapia, and R. Fierro. Reinforcement learning for balancing a flying inverted pendulum. In *Proc. The 11th World Congress on Intelligent Control and Automation*, July 2014.

[13] H. Hasselt. Reinforcement learning in continuous state and action spaces. In M. Wiering and M. Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 207–251. Springer Berlin Heidelberg, 2012.

[14] M. Hehn and R. D'Andrea. A flying inverted pendulum. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 763–770. IEEE, 2011.

[15] H. Kawano. Study of path planning method for under-actuated blimp-type uav in stochastic wind disturbance via augmented-mdp. In *Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on*, pages 180–185, July 2011.

[16] H. Khalil. *Nonlinear Systems*. Prentice Hall, 1996.

[17] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[18] A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, pages 543–558. Springer, 2013.

[19] C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *Proc. of Int. Conference on Automated Planning and Scheduling*, 2011.

[20] A. A. Masoud. A harmonic potential field approach for planning motion of a uav in a cluttered environment with a drift field, Orlando, FL, USA. In *50th IEEE Conference on Decision and Control and European Control Conference*, pages 7665–7671, dec 2011.

[21] F. Mueller, A. Schoellig, and R. D'Andrea. Iterative learning of feed-forward corrections for high-performance tracking. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3276–3281, Oct 2012.

[22] I. Palunko, P. Cruz, and R. Fierro. Agile load transportation : Safe and efficient load manipulation with aerial robots. *IEEE Robotics Automation Magazine*, 19(3):69 –79, sept. 2012.

[23] S. Shen, N. Michael, and V. Kumar. Stochastic differential equation-based exploration algorithm for autonomous indoor 3d exploration with a micro-aerial vehicle. *I. J. Robotic Res.*, 31(12):1431–1444, 2012.

[24] R. Sutton and A. Barto. *A Reinforcement Learning: an Introduction*. MIT Press, MIT, 1998.

[25] E. Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Comput.*, 17(5):1084–1108, May 2005.

[26] T. J. Walsh, S. Goschin, and M. L. Littman. Integrating sample-based planning and model-based reinforcement learning. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pages 612–617. AAAI Press, 2010.