Probabilistic Pointing Target Prediction via Inverse Optimal Control

Brian D. Ziebart
Carnegie Mellon University
Pittsburgh PA USA
bziebart@cs.cmu.edu

Anind K. Dey
Carnegie Mellon University
Pittsburgh PA USA
anind@cs.cmu.edu

J. Andrew Bagnell
Carnegie Mellon University
Pittsburgh PA USA
dbagnell@ri.cmu.edu

ABSTRACT

Numerous interaction techniques have been developed that make "virtual" pointing at targets in graphical user interfaces easier than analogous physical pointing tasks by invoking target-based interface modifications. These pointing facilitation techniques crucially depend on methods for estimating the relevance of potential targets. Unfortunately, many of the simple methods employed to date are inaccurate in common settings with many selectable targets in close proximity. In this paper, we bring recent advances in statistical machine learning to bear on this underlying target relevance estimation problem. By framing past target-driven pointing trajectories as approximate solutions to well-studied control problems, we learn the probabilistic dynamics of pointing trajectories that enable more accurate predictions of intended targets.

Author Keywords

Cursor prediction, probabilistic inference, continuous control

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Miscellaneous;

INTRODUCTION

Pointing at targets is a fundamental task in graphical user interfaces for selecting icons, menus, buttons, and other graphical elements. An understanding of pointing task performance has been of key importance for designing user interfaces for efficient use. Card *et al.* [9] showed that target pointing tasks via computer mouse and joystick follow Fitts' law [12], which was previously proposed for physical pointing task completion times. Under this law, pointing at a target of width W at distance D, has an average task completion time T of:

$$T = \alpha + \beta \log_2 \left(1 + \frac{D}{W}\right),$$

where α and β are empirically estimated parameters. From it, the contributing factors to pointing task difficulty can be more abstractly attributed to the pointing task distance (D) and the target precision requirement $\left(\frac{1}{W}\right)$.

IUI'12, February 14–17, 2012, Lisbon, Portugal.

Over the past 15 years, human-computer interaction research has investigated how the virtual nature of target pointing in graphical user interfaces can be leveraged to outperform the performance implied by Fitts' law. Since interactions through graphical user interfaces are quite prevalent, small improvements in task efficiency, even at the millisecond scale, can have substantial aggregate benefits. Further, for the population of users with physical motor impairments, reducing pointing task difficulty can make otherwise inaccessible applications accessible.

A number of pointing facilitation approaches have been developed and analyzed that decrease the effective target distance and/or alter the required target precision of the pointing task. Specifically, approaches have been introduced that:

- Restrict pointing to selectable targets, skipping over non-interactive portions of the user interface [14];
- Adjust the dynamics (control-display gains) for pointing motions to more quickly gravitate to and then linger within intended targets [19, 29, 7, 2, 15];
- Dynamically move candidate pointing targets closer to the current pointing location [5];
- Enlarge candidate pointing targets to make selecting them easier [23];
- Employ larger cursor activation regions to require less than single-pixel precision to select targets [16, 13, 20, 10]; or
- Generate haptic responses to cursor control that serve as an additional indicator of target precision thresholds [28, 11].

Unfortunately, the successes of these approaches have not extended to settings where multiple targets are in close proximity to one another [4], which is common in many graphical user interfaces. McGuffin and Balakrishnan identify the key limitation as the inability to reliably predict users' intended targets [24]. Without this ability, too many potential targets must be considered by the interaction technique for it to be effectively employed. They posit that improvements in desired target prediction would increase the effectiveness of any of the previously developed pointing facilitation techniques [24]. This makes target prediction a key machine learning problem for improving human-computer interactions in graphical user interfaces.

We introduce a new approach for predicting the desired target of a partial pointing motion using predictive inverse optimal control [31, 30]. The approach assumes that previously observed pointing motions are approximate solutions to a control task. The state-action costs of this control task that best

explain those observed pointing motions are learned in terms of velocities, accelerations, and jerks (changes in acceleration). Using Bayesian reasoning for the target prediction task, larger probabilities are assigned to potential targets for which a partial trajectory is an efficient solution to the learned control task. We evaluate the approach to show its applicability for supporting pointing facilitation techniques.

REQUIREMENTS AND PREVIOUS APPROACHES

We begin by identifying the features required for a target prediction system and reviewing the previously developed techniques for target prediction.

Prediction Requirements for Pointing Facilitation

Four key characteristics for a pointing prediction system to be useful in support of pointing facilitation are:

- **Belief-based reasoning:** Pointing facilitation techniques have different target prediction certainty requirements for their benefits when employed on the intended pointing target to outweigh the detriment when applied to the incorrect target. For example, introducing large control-display gain for movement towards the most likely target may only be net beneficial when target certainty is high, while smaller gain modifications may provide net benefits at much lower levels of target certainty. Accurate probabilistic target predictions are needed to support this risk sensitivity.
- **Real-time efficiency:** Pointing tasks in graphical user interfaces are typically completed in fractions of a second. A pointing target prediction system must therefore be able to provide quick predictions on the order of tens of milliseconds to be useful in practice. This requirement eliminates many sophisticated machine learning techniques that are less computationally efficient.
- **Personalization and context awareness:** Users' pointing trajectories can differ greatly based on differences in capabilities (including motor-skill impairments), input devices, and contextual situation. A pointing target prediction system should take these individual and contextual differences into account, customizing to the particular user's patterns of usage in different situations.
- **Application independence:** The layout of selectable target locations and the sequences of targets selected differ substantially between the user interfaces of different applications. A target pointing prediction system should be able to generalize across these application differences.

Many different approaches have been developed for this prediction task [25, 18, 21, 3, 22]. However, we will argue throughout this paper that those approaches have lacked important properties required for supporting pointing facilitation systems, or have been too simplistic to provide accurate predictions in the general cursor target prediction setting.

Classification Prediction Approaches

Interaction techniques have been the main focus for many of the previously published papers on pointing facilitation. Relatively simple methods based on target proximity or current pointing motion are often employed by these contributions to select the target or targets upon which to apply those interaction techniques. Unfortunately, these simple target relevance estimation methods perform poorly in user interfaces with many possible targets located in close proximity, such as those in Figure 1.

Figure 1. A set of 25 potential pointing targets (tool icons) in close proximity from the GNU Image Manipulation Program. The nearest tool icon to the cursor position, particularly early in the pointing motion, is often not the intended target.

Early work specifically devoted to pointing target prediction treats the prediction task as a classification problem using properties of the motion trajectory state sequence as input. For example, Lane *et al.* [21] provides short-cuts for selecting predicted targets and base target predictions primarily on the distance to each potential target. Though probabilistic techniques have been employed, *e.g.*, using a multinomial distribution for targets conditioned on velocity, acceleration, and motion curvature characteristics [25], they have been in support of target classification rather than belief-based prediction. In contrast to our inverse optimal control approach, aggressive discretization of motion characteristics is needed limit the amount of data required to estimate model parameters and for predictions to remain computationally efficient.

Regression-Based Extrapolation Approaches

Many techniques for predicting the intended target of a partial pointing motion formulate the problem as a continuous-valued regression task. Asano *et al.* [3] employ a simple linear regression from the peak velocity of the trajectory to the total distance[1] to the endpoint:

$$\left|\left|\mathbf{s}_T - \mathbf{s}_1\right|\right|_2 = \alpha\, v_{\max} + \beta + \epsilon_i, \qquad (1)$$

where $\mathbf{s}_t = \begin{pmatrix} x_t \\ y_t \end{pmatrix}$ and $v_{\max} = \max_\tau \left|\left|\mathbf{s}_\tau - \mathbf{s}_{\tau-1}\right|\right|_2$.

Predictions are made by extrapolating according to the learned regression coefficients, α and β. The parameters are fit to the demonstrated trajectory data using maximum likelihood estimation. The standard linear regression model assumption that any errors in the model's predictions ϵ_i are Gaussian distributed (Equation 1) is employed. The

[1]We denote the Euclidean distance with the norm $||\mathbf{v}||_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$.

extrapolated prediction of the final cursor position under the trained model is then:

$$\mathbf{s}_T = \mathbf{s}_t + \frac{\mathbf{s}_t - \mathbf{s}_1}{\left|\left|\mathbf{s}_t - \mathbf{s}_1\right|\right|_2} \left(\alpha\ v_{\max} + \beta\right).$$

The maximum velocity has also been previously employed as an indicator of the halfway point of a cursor trajectory [18].

A more sophisticated approach [22] is based on the observation that in idealized motion, the jerk (*i.e.*, the third derivative of position) is minimized. Under this model, the velocity is assumed to be a quadratic function of the distance traversed to the intended target location. Predictions are performed by fitting the quadratic function to the partial trajectory, and then extrapolating the point at which the velocity returns to zero, as shown in Algorithm 1.

Algorithm 1 Quadratic extrapolation for predicting pointing targets from a partial pointing trajectory.

Input: A partial trajectory, $\mathbf{s}_1 \ldots \mathbf{s}_t$, up to time step t.
Output: Trajectory endpoint prediction $\mathbf{s}_T = (x_T, y_T)$.

1: Fit a quadratic function with parameters $\alpha_2, \alpha_1, \alpha_0$ to the partial trajectory's total distance.
2: Obtain the second root of the quadratic equation: $d_T = \frac{-\alpha_1 - \sqrt{\alpha_1^2 - 4\alpha_0\alpha_2}}{2\alpha_2}$.
3: Apply the correction multiplier γ based on the fraction of predicted trajectory distance completed: $d'_T = \gamma\ d_T$ (see [22] for details).
4: Predict $\mathbf{s}_T = \mathbf{s}_1 + d'_T\ \frac{\mathbf{s}_t - \mathbf{s}_1}{||\mathbf{s}_t - \mathbf{s}_1||_2}$

However, apart from the correction multiplier (Step 3 of Algorithm 1), this prediction model is incapable of using previously observed trajectories to improve future target predictions. This makes it particularly ill-suited for personalizing to the different capabilities and usage patterns of individuals.

The Gaussian assumption of target prediction errors in these regression-based approaches imposes symmetry constraints on prediction beliefs. Namely, points that are reflections across each of the predicted Gaussian's principal axes have the same probability density. When the regression models' errors differ from Gaussian distributions, the Gaussian assumption leads to models with much higher uncertainty than warranted. The approach we present in this paper provides a less restrictive posterior target distribution by employing Bayesian inference methods for target inference.

INVERSE OPTIMAL CONTROL POINTER FORECASTING

We employ computationally efficient inverse optimal control techniques to learn from and predict target-based pointing motions. At a high-level, our approach has three main elements:

1. It re-casts the discrete cursor pointing task from a control perspective with continuous states and actions;
2. It employs a probabilistic inverse optimal control technique [30] to model observed pointing trajectories given the intended pointing target; and
3. It leverages Bayes' rule to obtain posterior probabilities of a partial cursor trajectory's intended target.

Crucially, the combination of these three elements provides a powerful parametric learning approach that is computationally efficient for real-time use. We elaborate upon these three elements of our approach in the remainder of this section.

Continuous Control Representation of Pointing

We take a control perspective to modeling pointing motions in which the objective is to invoke control actions that direct the pointing motion to the origin. To facilitate this perspective, we project the original trajectory into a new axis system where the x-axis is aligned with the origin-to-target direction and the y-axis is orthogonal, as illustrated in Figure 2a. Under this projection, discrete cursor locations (in pixels) are treated as continuous-valued. As we shall see, this provides computational advantages. Additionally, to later obtain model parameters at reasonable scales, we equate one unit of distance in this new projection to 100 pixels of screen distance.

Using this coordinate system, we view the instantaneous state of motion at timestep t as:

$$\mathbf{s}_t = \left[x_t\ y_t\ \dot{x}_t\ \dot{y}_t\ \ddot{x}_t\ \ddot{y}_t\ \dddot{x}_t\ \dddot{y}_t\right]^{\mathrm{T}}.$$

The higher-order changes related to position correspond to common physical dynamics. They are the velocity $(\dot{x}_t, \dot{y}_t)$, the acceleration $(\ddot{x}_t, \ddot{y}_t)$, and the jerk (*i.e.*, change in acceleration, $\dddot{x}_t, \dddot{y}_t$):

$$\begin{pmatrix} \dot{x}_t \\ \dot{y}_t \end{pmatrix} = \begin{pmatrix} x_t - x_{t-1} \\ y_t - y_{t-1} \end{pmatrix} \qquad \begin{pmatrix} \ddot{x}_t \\ \ddot{y}_t \end{pmatrix} = \begin{pmatrix} \dot{x}_t - \dot{x}_{t-1} \\ \dot{y}_t - \dot{y}_{t-1} \end{pmatrix}$$
$$\text{and } \begin{pmatrix} \dddot{x}_t \\ \dddot{y}_t \end{pmatrix} = \begin{pmatrix} \ddot{x}_t - \ddot{x}_{t-1} \\ \ddot{y}_t - \ddot{y}_{t-1} \end{pmatrix}. \tag{2}$$

Importantly, when actions, $\mathbf{a}_t$, are viewed as the velocities specifying changes in position between consecutive timesteps, the state dynamics follow a linear relationship:

$$\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t + \epsilon_t, \tag{3}$$

where the optional noise term, ϵ_t is assumed to be drawn from a zero-mean Gaussian distribution with variance Σ. We denote the distribution of next state under this model of dynamics as $\tau(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.

Inverse Optimal Control for Prediction

The next element of our approach is the construction of a probabilistic model of complete cursor motion sequences given each motion's target location. This perspective allows us to employ inverse optimal control techniques from machine learning, which attempt to find the costs of a control process that makes demonstrated behavior (close to) optimal [17, 8, 26, 1, 27]. We leverage the maximum entropy variant of inverse optimal control [31, 30], which provides a state-conditioned probability distribution $\hat{\pi}$ over control actions (*i.e.*, cursor movements) $\mathbf{a}$ that is recursively defined as:

$$\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t) \propto e^{-Q(\mathbf{s}_t, \mathbf{a}_t)}. \tag{4}$$
$$V(\mathbf{s}_t) = \operatorname*{softmin}_{\mathbf{a}_t} \left\{\text{cost}(\mathbf{s}_t) + Q(\mathbf{s}_t, \mathbf{a}_t)\right\} \tag{5}$$
$$Q(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\tau(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}\left[V(\mathbf{s}_{t+1})|\mathbf{s}_t, \mathbf{a}_t\right], \tag{6}$$

where $\text{cost}(\mathbf{s}_t) \triangleq \mathbf{s}_t^{\mathrm{T}}\mathbf{M}\mathbf{s}_t$ is a quadratic cost function, and the value terms, $Q(\mathbf{s}, \mathbf{a})$ and $V(\mathbf{s}_t)$, are future cumulative expected costs. The *softmin* is a smoothed interpolation of the *minimum* function: $\text{softmin}_x f(x) = -\log \int_x e^{-f(x)} dx$. From this perspective, the recurrence of Equations 5 and 6 can be viewed as a probabilistic relaxation of the Bellman criteria for optimal control [6]; it (probabilistically) selects actions with smaller expected future costs in Equation 5 and recursively computes those future costs using the expectation over the decision process's dynamics in Equation 6. The $\mathbf{M}$ matrix contains the model's cost parameters. Parameter values are chosen so that the predictive control distribution $\hat{\pi}$ matches quadratic state properties of demonstrated behavior $\tilde{\pi}$ in expectation ($\mathbb{E}_f[g(x)] \triangleq \int_{x \in \mathcal{X}} f(x)\ g(x)\ dx$):

$$\mathbb{E}_{\hat{\pi},\tau}\left[\sum_t \mathbf{s}_t\ \mathbf{s}_t^{\mathrm{T}}\right] = \mathbb{E}_{\tilde{\pi},\tau}\left[\sum_t \mathbf{s}_t\ \mathbf{s}_t^{\mathrm{T}}\right]. \tag{7}$$

This set of constraints ensures that the trajectory's dynamic properties are maintained by the control policy estimate, $\hat{\pi}$. For example, the relationship between target-relative position (x) and target-directed velocity ($\dot{x}$) is important for characterizing target-directed motion.

When the state-transition dynamics are linear (Equation 3), the recursive relation of Equations 5 and 6 have closed-form solutions that are quadratic functions and the action distribution (Equation 4) is a conditional Gaussian distribution. This enables efficient $\mathcal{O}(T\ \dim(s)^2)$ time computation for time horizons T and much larger continuous state/action spaces than could be computed as discrete state/action spaces.

We make use of the probability distribution over each partial trajectory, $\mathbf{s}_2, \ldots, \mathbf{s}_t$ $(t \leq T)$, given the initial state $\mathbf{s}_1$ and target G:

$$\prod_{\tau=1}^{t} \pi(\mathbf{a}_t|\mathbf{s}_t, G) = e^{-\left(\sum_{\tau=2}^{t} \text{cost}(\mathbf{s}_\tau^G)\right) - V(\mathbf{s}_t^G) + V(\mathbf{s}_1^G)}, \tag{8}$$

where super-scripting the states with the target G makes the target-dependent projection explicit. Equation 8 prescribes larger probabilities to trajectories that efficiently reduce the future expected cost. A derivation and detailed algorithms for efficiently computing the policy and learning the model parameters appear in the Appendix.

Bayesian Target Prediction

We employ the learned trajectory model to estimate target probabilities. Using the probability distribution for a partial trajectory given a target location (Equation 8), we employ Bayes' rule to find the probability distribution of a target location G from a set of target locations $\mathcal{G}$ given a partial cursor trajectory. The posterior distribution,

$$P(\text{target } G|\text{partial trajectory } \mathbf{s}_1, \ldots, \mathbf{s}_t) = \frac{\prod_{\tau=1}^{t} \pi(\mathbf{a}_t|\mathbf{s}_t, G) P(G)}{\sum_{G' \in \mathcal{G}} \prod_{\tau=1}^{t} \pi(\mathbf{a}_t|\mathbf{s}_t, G') P(G')}, \tag{9}$$

assigns higher probabilities to targets towards which the partial trajectory efficiently makes progress.

This Bayesian formulation leverages a flexible **prior distribution** over targets $P(G)$ that reflect an initial belief for the next target without current cursor trajectory information. This distribution can be based on relevant contextual information, such as current application, target selection history, time of day, *etc*. However, as our focus in this work is on leveraging available cursor trajectory data to predict pointing targets, we simply assume a uniform prior distribution over possible targets, $P(G) = \frac{1}{|\mathcal{G}|}$.

The linear inverse optimal control approach we employ provides additional computational benefits for calculating these posterior probability distributions. Naïvely computing the posterior target distribution for a pointing motion of length t would entail calculating the quadratic value functions for each possible target, requiring $\mathcal{O}(|\mathcal{G}|t\ \dim(s)^2)$ time total for a set of potential targets $\mathcal{G}$. Instead, using simple projections of the state into different target-based reference frames, a single value function computation is needed and computing the posterior for each target is independent of the the time horizon t. We detail these efficient projections in the Appendix.

EXPERIMENTS

We employ a dataset of collected pointing trajectories to analyze and evaluate pointing target prediction approaches.

Collected Cursor Trajectory Data

Our dataset consists of cursor trajectories from 20 non-motor-impaired individuals that are frequent computer users captured at 100Hz. Each individual performed 300 different **target selection tasks** using computer mouse input devices. Each task requires moving a cursor from an origin point to a target click location—a circle—that the user selects via a button click to complete the task. Target locations were selected at random from the computer screen space subject to being 200 pixels away from the pointing task's starting point.

Figure 2a shows the x and y positions of trajectories from the 20 individuals on one task. The trajectories originate in the southwest corner of the plot and terminate in a small target click circle at the origin. Figure 2b and Figure 2c show how the trajectories vary over time in the x and y positions. As a pre-processing step, we discard any timesteps with no movement. These predominantly appear at the beginning of trajectories and correspond to the time that users seem to spend visually processing and responding to the pointing task.

As shown in Figure 2, a great deal of variability exists between trajectories, illustrating the sub-optimality of human control on pointing tasks. One particular trajectory in Figure 2 takes significantly more timesteps of motion to reach the target location and complete the task than other trajectories. Even more anomalous deviations are shown in Figure 3, where users slide their cursor along the outer borders of the window while completing the task (presumably due to losing track of the cursor and moving it to try to locate it). More subtle anomalies are: missing the boundary of the click target when the mouse button is pressed, responding slowly to the targeting task, and having the cursor in motion at the beginning of the targeting task.

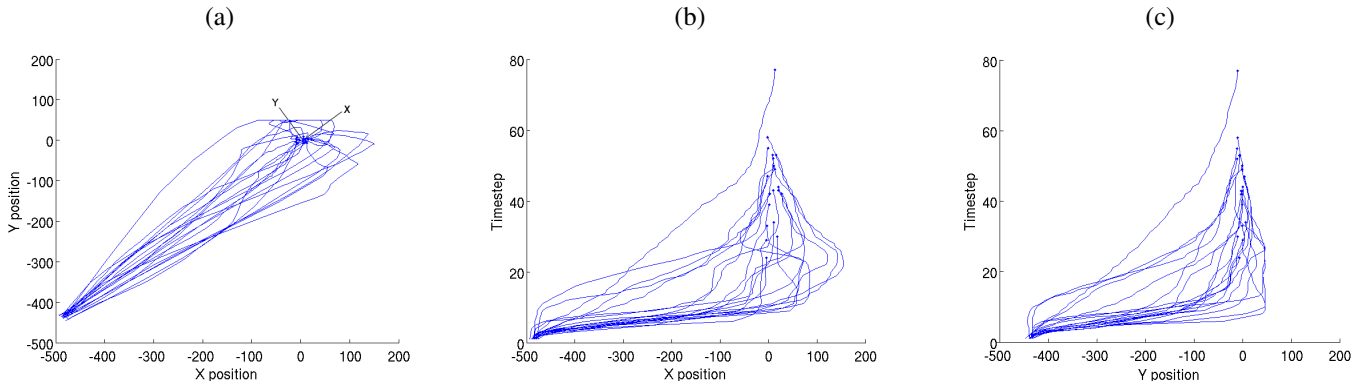

Figure 2. Cursor trajectories from 20 different users between the same pair of endpoints shown in space (a) and as a function of time (with each timestep representing 10 milliseconds) in each dimension (b, c). The target-based rotated axis frame is also shown in (a).

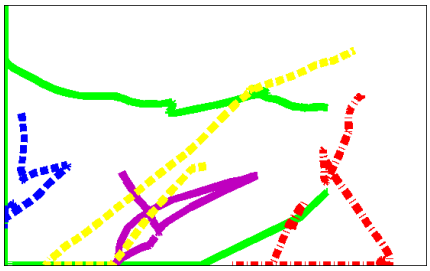

Figure 3. Five highly anomalous cursor trajectories that are very inefficient in reaching the ultimate target location.

To conservatively address the many sources of errors present in collected cursor pointing data, we discard the least task-efficient half of the cursor trajectories from each pointing task. We believe that many of the discarded trajectories can be useful sources of information for modeling cursor movements with additional pre-processing to appropriately remove properties of different errors. However, our point of emphasis in this paper is on the techniques for modeling and predicting pointing movements, so we instead use the more conservative dataset that does not require more sophisticated pre-processing.

Experimental Setup and Compared Approaches

We employ the cursor trajectories from 50 pointing tasks as training data to fit each model's parameters and employ the cursor trajectories from the remaining 250 cursor trajectories to evaluate the predictive performance of each model.

The target prediction models that we compare are:

- **Current position:** A simple baseline that predicts that the current position will be the pointing target.
- **Velocity extrapolation [3]:** The maximum velocity regression approach (Equation 1) with parameters fit from the training data set.
- **Quadratic extrapolation [22]:** The quadratic extrapolation algorithm (Algorithm 1) fit to each partial trajectory.
- **Inverse optimal control** The inverse optimal control approach using maximum causal entropy.

We obtain probabilistic predictions from the current position and regression-based approaches by assuming that prediction errors are distributed according to a 2-dimension Gaussian:

$$\mathbf{s}_T \sim N(f(\mathbf{s}_1, \ldots, \mathbf{s}_t), \Sigma(\mathbf{s}_1, \ldots, \mathbf{s}_t)),$$

where f is the regression mean estimation function and Σ is one of 11 covariance matrices estimated from the training set of cursor motions. The covariance matrix employed is selected based on an estimate of the motion trajectory's progress using: its duration for the *current position* model; the location where the maximum velocity occurs in the motion sequence for the *velocity extrapolation* model, or the estimated quadratic for the *quadratic extrapolation* model.

Model Fitting

We analyze each model's fit to the training data in this section. We begin with the assumptions made by the extrapolation approaches: (1) correlation between maximum velocity and target distance; and (2) a quadratic fit to the trajectory's velocity as a function of distance.

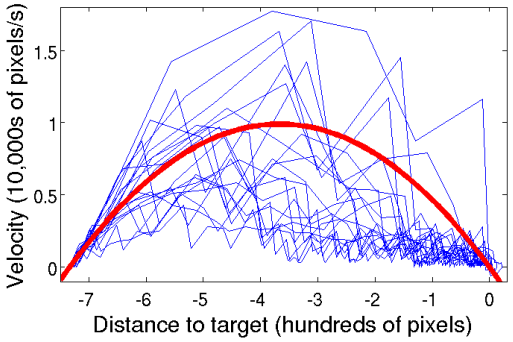

Figure 4. Velocity and position of movement in the target-oriented axis (with target at 0) for 20 participants cursor motions (thin blue lines). The quadratic function (smooth red line) fits the average of these motions very nicely, even though the individual motions deviate greatly from the idealize quadratic.

The variability in pointing movement velocities in Figure 4 is typical across tasks; trajectories reach the target by following a wide range of velocity patterns with differing maximum

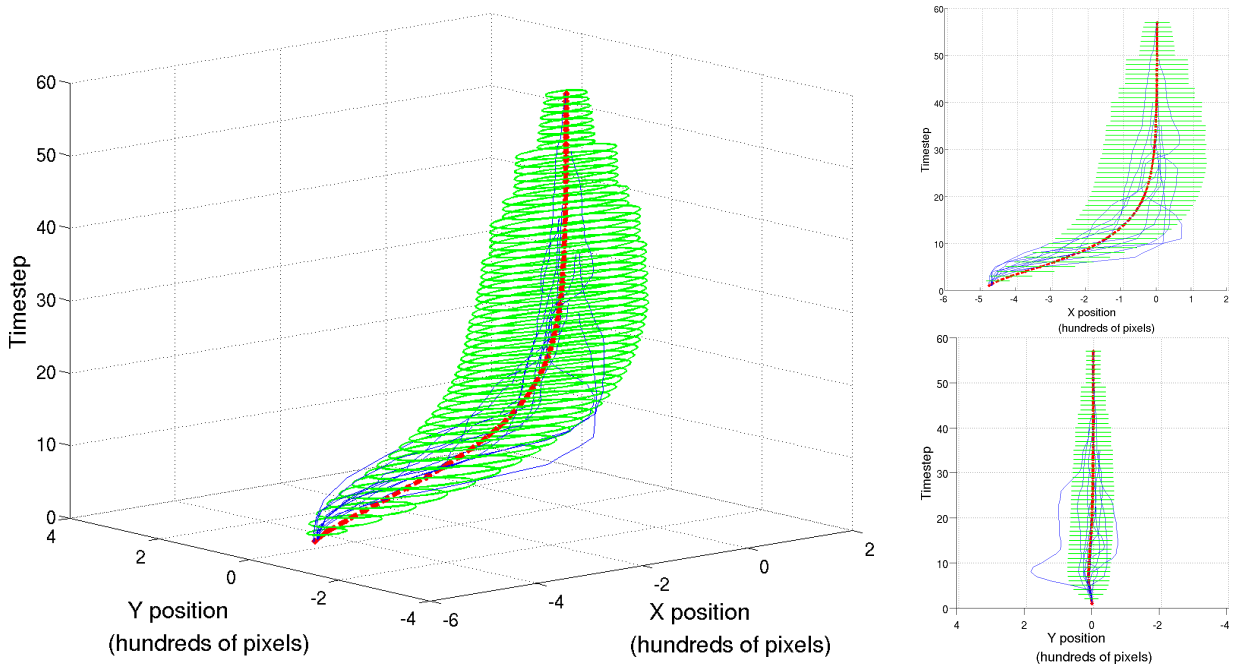

Figure 6. Three perspectives of demonstrated trajectories (thin blue lines) in x and y position over time along with the mean (thicker red line) and 95% confidence region (green ovals) of the predictive maximum causal entropy model at each point in time.

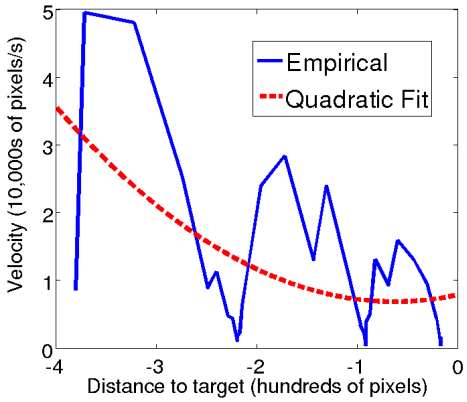

Figure 5. Velocity and position of movement in the target-oriented axis. The trajectory appears to have been decomposed into three sequences of small movements towards the target position. This composition confounds the quadratic model, which no longer predicts any endpoint (*i.e.*, the quadratic has no real roots).

velocity. Unfortunately, this introduces a large degree of uncertainty in the velocity extrapolation model obtained by regression from maximum velocity to target distance.

The average of our collected trajectories very closely fits the idealized motion quadratic (Algorithm 1), as shown in Figure 4. However, individual trajectories deviate greatly from the idealized quadratic. Indeed, for 3.1% of trajectories, no prediction is provided by the quadratic fit because the quadratic is convex (as shown in Figure 5) rather than concave. Many other quadratic fits, though concave, do not provide accurate target distance predictions. This becomes increasingly prevalent when fitting the quadratic function using only a small portion of the motion sequence.

The inverse optimal control model's fit to a set of demonstrated pointing trajectories in Figure 6. The prediction is conditioned on the starting point and the target location. The 95% confidence intervals of the prediction are adjusted based on the varying trajectory durations; completed trajectories are treated as being known at the target center. Though the confidence intervals generally do not contain 95% of the demonstrated trajectories due to the model being a simplified abstraction of a user's control-feedback loop, the model provides a reasonably good fit to the demonstrated trajectories.

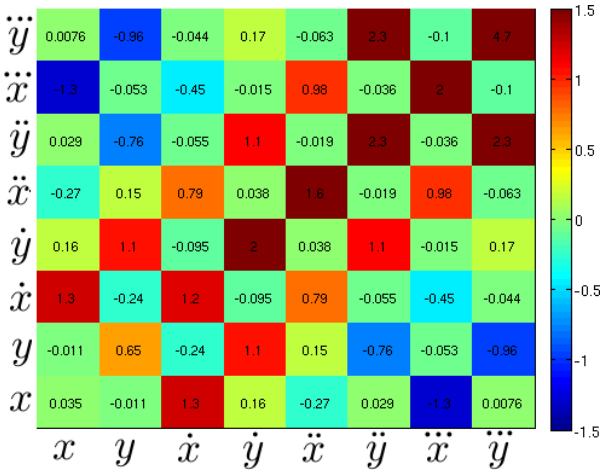

Figure 7. Heat map for the quadratic cost coefficients of matrix M learned by the inverse optimal control approach. Positive values correspond to costs while negative values can be interpreted as rewards.

A heat map of the learned quadratic parameter matrix M of the inverse optimal control model is shown in Figure 7. Each entry of the heat map is a cost coefficient corresponding to a

combined quadratic term. For example, the $x\dot{x}$ term penalizes velocities that are away from the target location ($x < 0$ and $\dot{x} < 0$ or $x > 0$ and $\dot{x} > 0$) with coefficient 1.3. Along the diagonal (bottom left to top right), the cost coefficients are all positive, penalizing deviations away from zero. In particular, large higher-order dynamics (accelerations and jerks) are heavily penalized. The diagonal ($x\dot{x}$, $y\dot{y}$, $\dot{x}\ddot{x}$, $\dot{y}\ddot{y}$, $\ddot{x}\dddot{x}$, and $\ddot{y}\dddot{y}$) is similarly positive. This primarily supports the stability of the trajectory by penalizing the reinforcement of growing dynamics. Note that the heat map has a checkerboard pattern; high magnitude coefficient weights are primarily associated with same-axis combinations.

Pointing Target Prediction

We evaluate each trained model's performance in predicting each pointing motion's target given only a partial pointing trajectory. Though our collected dataset only involves a single target location, we introduce a set of closely grouped targets around the true target and ask: *How well could we predict the actual target?* Our assumption is that additional irrelevant targets do not significantly alter pointing trajectories compared to single-target pointing tasks.

We employ a challenging set of closely clustered targets to assess predictive performance: a 9-by-9 grid of potential targets spaced apart by 50 pixels. This type of configuration has previously been identified as difficult for target prediction [4]. The specific setting we employ is similar to, but larger than, the array of tool icons shown in Figure 1.

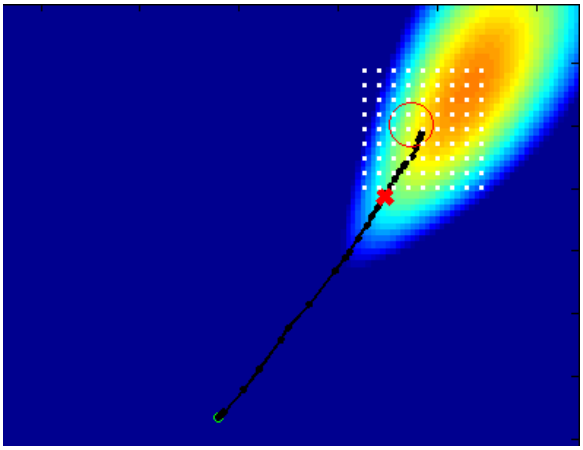

Figure 8. The likelihood function of all target locations (each pixel on the screen) given a partial trajectory (the black line up to the "X"). The initial point is denoted as a small green circle and the final target region is denoted as a larger circle. Each pixel of the figure represents 10 pixels of monitor space.

Our evaluation procedure is best described using Figure 8 for reference. Note that the task target in Figure 8 is contains six prediction task targets, making this a particularly difficult prediction task. Also in the figure, the likelihood function of the inverse optimal control approach, $P(\mathbf{s}_2, \ldots, \mathbf{s}_t | \mathbf{s}_1, G)$, is shown for each potential target point throughout the screen given one particular partial cursor trajectory (the black line up to the red "X"). Unlike regression-based approaches, that this likelihood function is not constrained to be Gaussian, which allows a great deal more flexibility than linear regression-based techniques. Target predictions for a specific set of targets—the 81 white points in the figure—are made by evaluating the likelihood function at each potential target position and normalizing after weighting by the prior probability (uniform for our experiments). For the regression-based models, we evaluate the target probability only at each target location and appropriately re-normalize the distribution.

We measure predictive performance using the classification error (*i.e.*, the frequency with which the model's most probable target is not the true target), and the log-loss of the pointing motion's true target G^*,

$$-\frac{1}{N}\sum_{i=1}^{N} \log_2 \hat{P}(\text{target } G^{i,*} | \mathbf{s}_1^i, \ldots \mathbf{s}_t^i).$$

The latter measures the amount of uncertainty of the intended target under each model's predictive probability distribution. This can be interpreted as the amount of information (in bits) needed to distinguish the true target from the others under a target prediction system's predictive belief. When the target is known with certainty, 0 bits are required, and when the distribution is most uncertain (a uniform distribution over 81 targets), $log_2\ 81 = 6.65$ bits are required. Though improvement of a pointing facilitation technique's utility to a user is the ultimate evaluation metric for a pointing prediction method, that evaluation measure is highly dependent on the pointing facilitation technique employed. Log-loss is the natural measure for belief-based predictive performance that should be useful in improving pointing facilitation techniques.

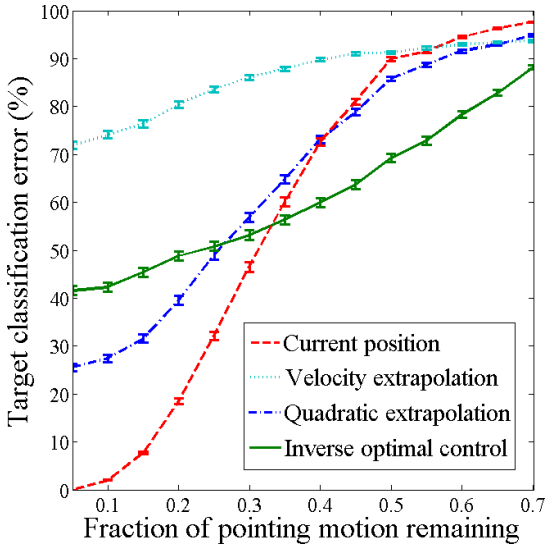

Figure 9. Mean target prediction classification error as a function of the amount of pointing trajectory remaining (in terms of time) with 95% confidence intervals.

Our evaluations are shown in Figure 9 and Figure 10 as functions of the fraction of the cursor trajectory remaining at the point in time that the prediction is made, ranging from 5% to 70% remaining—beyond which, no model performs particularly well for this task. It is important to note that the predictive model does not know what percentage of the trajectory has been completed.

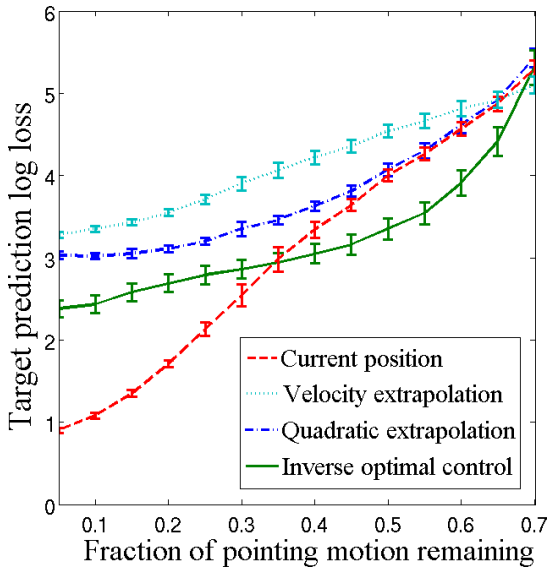

Figure 10. Target prediction log loss as a function of the amount of pointing trajectory remaining (in terms of time) with 95% confidence intervals.

For both measures, the *current position* model provides high-quality predictions when the fraction of pointing motion remaining is small. Indeed, when 95% of the motion is completed, essentially only selecting the target remains, and the *current position* assumption provides 100% accuracy. However, while that assumption biases the model to have good performance in the almost-completed-task region, significantly fewer classification errors are obtained with the *quadratic extrapolation* model when at least 50% of the pointing motion remains and by the *velocity extrapolation* model when at least 60% of the pointing motion remains. However, neither extrapolation method significantly outperforms the *current position* model under log-loss, presumably because log-loss penalizes these models for predictions based on the highly variable maximum velocities and quadratic fits that may be anomalous, as illustrated in Figure 4 and Figure 5.

The *inverse optimal control* model provides significant performance improvements over the three other models when large portions of pointing motion remain. Specifically, with between 40% and 60% of the motion trajectory remaining, it is more than twice as accurate for target classification than all other models and is only significantly outperformed in log-loss by the *current position* model when 30% or less of the pointing motion remains to be completed. We believe that the predictive performance characteristics of the *inverse optimal control* model will prove to be more beneficial for selecting appropriate pointing facilitation techniques because accurate predictions earlier in the pointing motion have the potential to provide greater benefits than later, when the motion has essentially reached the target. Understanding to what degree the *current position* model's superior performance when small amounts of pointing motion remain results from inherent bias of the model's assumption or whether the *inverse optimal control* approach can be improved to provide similar performance is an important question for future research.

DISCUSSION

We began this paper by discussing key requirements needed by a pointing target prediction system to support pointing facilitation techniques. We introduced an inverse optimal control approach that outperform previously developed techniques in critical early portions of pointing motions when predicting intended pointing targets and satisfies the requirements we discussed. Importantly, it learns parameters closely associated with motion dynamics, making it applicable to a wide range of settings. While training the model is relatively slow and computationally expensive, making predictions is fast and can be accomplished in real-time for moderate numbers of targets (*e.g.*, less than 200). Though we focused on aggregate modeling of cursor trajectories from a group of users, the approach we presented is well-suited for learning quadratic cost coefficients for each user or populations of users with similar patterns of usage—an important focus for improving computer accessibility. Additionally, contextual information can be easily incorporated into the prior target distribution in this approach.

We conclude by emphasizing that the performance of our approach in this paper should serve as a starting point. We employed a relatively simply set of statistics to base our model on in this paper—quadratic statistics of positions, velocities, accelerations, and jerks. We expect improved performance can be realized by augmenting these with linear and/or time-dependent statistics of the pointing motions. Many other opportunities exist for further improving the approach. For example, viewing pointing trajectories as mixtures of behaviors (such as some steady movements and some overshooting/correcting movements) may be beneficial for leveraging the less efficient trajectories that were discarded in the analyses of this paper while remaining computationally efficient for real-time prediction tasks. We plan to investigate these extensions and the incorporation of this predictive approach with specific facilitation techniques in future work.

ACKNOWLEDGMENTS

We thank Jin-Hyuk Hong for providing cursor pointing data, source code, and useful discussions, as well as the National Science Foundation's Quality of Life Technology Center and ONR MURI grant N00014-09-1-1052 for support of this research.

REFERENCES

1. Abbeel, P., and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proc. International Conference on Machine Learning* (2004), 1–8.

2. Ahlström, D., Hitz, M., and Leitner, G. An evaluation of sticky and force enhanced targets in multi target situations. In *NordiCHI '06: Proceedings of the 4th Nordic Conference on Human-Computer Interaction*, ACM (New York, NY, USA, 2006), 58–67.

3. Asano, T., Sharlin, E., Kitamura, Y., Takashima, K., and Kishino, F. Predictive interaction using the Delphian desktop. In *Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, ACM (2005), 133–141.

4. Balakrishnan, R. "Beating" Fitts' law: virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies 61*, 6 (2004), 857–874.

5. Baudisch, P., Cutrell, E., Robbins, D., and Czerwinski, M. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch-and pen-operated systems. In *Proceedings of the 13th IFIP TCI3 Conference on Human-Computer Interaction (INTERACT)* (2003), 57–64.

6. Bellman, R. A Markovian decision process. *Journal of Mathematics and Mechanics 6* (1957), 679–684.

7. Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2004), 519–526.

8. Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V. Linear matrix inequalities in system and control theory. *Society of Industrial and Applied Mathematics 15* (1994).

9. Card, S., English, W., and Burr, B. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics 21*, 8 (1978), 601–613.

10. Chapuis, O., Labrune, J., and Pietriga, E. Dynaspot: Speed-dependent area cursor. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, ACM (2009), 1391–1400.

11. Dennerlein, J., and Yang, M. Haptic force-feedback devices for the office computer: Performance and musculoskeletal loading issues. *Human Factors 43*, 2 (2001), 278–286.

12. Fitts, P. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 6 (1954), 381–391.

13. Grossman, T., and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2005), 281–290.

14. Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Object pointing: a complement to bitmap pointing in guis. In *Proceedings of Graphics Interface 2004*, Canadian Human-Computer Communications Society (2004), 9–16.

15. Hurst, A., Mankoff, J., Dey, A., and Hudson, S. Dirty desktops: using a patina of magnetic mouse dust to make common interactor targets easier to select. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ACM (2007), 183–186.

16. Kabbash, P., and Buxton, W. The "prince" technique: Fitts' law and selection using area cursors. In *Proc. of the ACM CHI Conference on Human Factors in Computing Systems* (1995), 273–279.

17. Kalman, R. When is a linear control system optimal? *Trans. ASME, J. Basic Engrg. 86* (1964), 51–60.

18. Keuning-Van Oirschot, H., and Houtsma, A. Cursor displacement and velocity profiles for targets in various locations. In *Proceedings of Eurohaptics* (2001), 108–112.

19. Keyson, D. Dynamic cursor gain and tactual feedback in the capture of cursor movements. *Ergonomics 40*, 12 (1997), 1287–1298.

20. Kobayashi, M., and Igarashi, T. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 949–958.

21. Lane, D., Peres, S., Sándor, A., and Napier, H. A process for anticipating and executing icon selection in graphical user interfaces. *International Journal of Human-Computer Interaction 19*, 2 (2005), 241–252.

22. Lank, E., Cheng, Y., and Ruiz, J. Endpoint prediction using motion kinematics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2007), 637–646.

23. McGuffin, M., and Balakrishnan, R. Acquisition of expanding targets. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, ACM (2002), 57–64.

24. McGuffin, M., and Balakrishnan, R. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI) 12*, 4 (2005), 388–422.

25. Münch, S., and Dillmann, R. Haptic output in multimodal user interfaces. In *Proceedings of the 2nd International Conference on Intelligent User Interfaces*, ACM (1997), 105–112.

26. Ng, A. Y., and Russell, S. Algorithms for inverse reinforcement learning. In *Proc. International Conference on Machine Learning* (2000), 663–670.

27. Ratliff, N., Bagnell, J. A., and Zinkevich, M. Maximum margin planning. In *Proc. International Conference on Machine Learning* (2006), 729–736.

28. Wall, S., and Harwin, W. Quantification of the effects of haptic feedback during a motor skills task in a simulated environment. In *Proc. Second PHANToM Users Research Symposium* (2000).

29. Worden, A., Walker, N., Bharat, K., and Hudson, S. Making computers easier for older adults to use: area cursors and sticky icons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1997), 266–271.

30. Ziebart, B. D., Bagnell, J. A., and Dey, A. K. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning* (2010), 1247–1254.

31. Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proc. AAAI Conference on Artificial Intelligence* (2008), 1433–1438.

APPENDIX: MAXIMUM CAUSAL ENTROPY LINEAR QUADRATIC REGULATION

We describe the maximum causal entropy approach to inverse optimal control and its efficient algorithms for the setting with linear dynamics and quadratic cost functions.

Formulation of the probability distribution

The distribution over actions we employ in this work (Equation 4) and the recursive relation defining its variables (Equation 5) are obtained by maximizing the causal entropy,

$$H(\mathbf{a}||\mathbf{s}) = \mathbb{E}_{\hat{\pi},\tau}\left[-\sum_{t=1}^{T}\log\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)\right], \qquad (10)$$

subject to the constraints of Equation 7 [30]. We refer the reader to the previous paper introducing the general maximum causal entropy approach for a better understanding of the robust predictive guarantees that it provides.

Linear dynamics case

In the case of linear dynamics, the recurrence values (Equation 5) are quadratic functions of the form:

$$\begin{aligned} Q(\mathbf{a}_t,\mathbf{s}_t) &= \begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{C}_{a,a} & \mathbf{C}_{a,s}\\ \mathbf{C}_{s,a} & \mathbf{C}_{s,s}\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix} \\ &\quad + \begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{F}_a\\ \mathbf{F}_s\end{bmatrix}, \text{ and} \\ V(\mathbf{s}_t) &= \mathbf{s}_t^{\mathrm{T}}\mathbf{D}\mathbf{s}_t + \mathbf{s}_t^{\mathrm{T}}\mathbf{G}. \end{aligned}$$

Expanding the definitions of Equation 5 and 6, we have:

$$\begin{aligned} Q(\mathbf{a}_t,\mathbf{s}_t) &= \mathbb{E}_{\tau(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}[\mathbf{s}_{t+1}^{\mathrm{T}}\mathbf{D}\mathbf{s}_{t+1} + \mathbf{s}_{t+1}^{\mathrm{T}}\mathbf{G}|\mathbf{s}_t,\mathbf{a}_t] \\ &= (\mathbf{A}\mathbf{s}_t+\mathbf{B}\mathbf{a}_t)^{\mathrm{T}}\mathbf{D}(\mathbf{A}\mathbf{s}_t+\mathbf{B}\mathbf{a}_t) + \mathrm{tr}(\mathbf{D}\Sigma) \\ &\quad + \mathbf{a}_t^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{G} + \mathbf{s}_t^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{G} \\ &= \begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B} & \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{A}\\ \mathbf{A}^{\mathrm{T}}\mathbf{D}\mathbf{B} & \mathbf{A}^{\mathrm{T}}\mathbf{D}\mathbf{A}\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix} \\ &\quad + \begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{B}^{\mathrm{T}}\mathbf{G}\\ \mathbf{A}^{\mathrm{T}}\mathbf{G}\end{bmatrix} + \text{const, and} \\ V(\mathbf{s}_t) &= -\log\int_{\mathbf{a}_t}\exp\{-(\mathbf{a}_t^{\mathrm{T}}\mathbf{C}_{a,a}\mathbf{a}_t + 2\mathbf{a}_t^{\mathrm{T}}\mathbf{C}_{a,s}\mathbf{s}_t \\ &\quad + \mathbf{s}_t^{\mathrm{T}}\mathbf{C}_{s,s}\mathbf{s}_t + \mathbf{s}_t^{\mathrm{T}}\mathbf{F}_s + \mathbf{a}_t^{\mathrm{T}}\mathbf{F}_a)\}\, d\mathbf{a}_t + \mathbf{s}_t^{\mathrm{T}}\mathbf{M}\mathbf{s}_t \\ &= \mathbf{s}_t^{\mathrm{T}}(\mathbf{C}_{s,s}+\mathbf{M}-\mathbf{C}_{a,s}^{\mathrm{T}}\mathbf{C}_{a,a}^{-1}\mathbf{C}_{a,s})\mathbf{s}_t \\ &\quad + \mathbf{s}_t^{\mathrm{T}}(\mathbf{F}_s - \mathbf{C}_{a,s}^{\mathrm{T}}\mathbf{C}_{a,a}^{-1}\mathbf{F}_a) + \text{const}, \end{aligned}$$

Thus the set of update rules for the quadratic functions are: $\mathbf{C}_{a,a} = \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}$; $\mathbf{C}_{s,a} = \mathbf{C}_{a,s}^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{A}$; $\mathbf{C}_{s,s} = \mathbf{A}^{\mathrm{T}}\mathbf{D}\mathbf{A}$; $\mathbf{F}_a = \mathbf{B}^{\mathrm{T}}\mathbf{G}$; $\mathbf{F}_s = \mathbf{A}^{\mathrm{T}}\mathbf{G}$; $\mathbf{D} = \mathbf{C}_{s,s} + \mathbf{M} - \mathbf{C}_{a,s}^{\mathrm{T}}\mathbf{C}_{a,a}^{-1}\mathbf{C}_{a,s}$; and $\mathbf{G} = \mathbf{F}_s - \mathbf{C}_{a,s}^{\mathrm{T}}\mathbf{C}_{a,a}^{-1}\mathbf{F}_a$.

Parameter Estimation

The model parameters in matrix $\mathbf{M}$ are fit from demonstrated state sequences. The likelihood of the demonstrated trajectories under the action distribution, Equation 4, is a convex function of those free parameters. This guarantees that standard gradient-based techniques will converge to the choice of parameters that best explains the demonstrated trajectories. The gradients have an intuitive interpretation: they are the differences of the optimization constraints (Equation 7),

$$\nabla_{\mathbf{M}}\, L = \mathbb{E}_{\hat{\pi},\tau}\left[\sum_t \mathbf{s}_t\,\mathbf{s}_t^{\mathrm{T}}\right] - \mathbb{E}_{\tilde{\pi},\tau}\left[\sum_t \mathbf{s}_t\,\mathbf{s}_t^{\mathrm{T}}\right]. \qquad (11)$$

These values can be directly obtained from the Gaussian distribution of state over time. Namely, if $\mathbf{x}$ is normally distributed with mean μ_x and covariance matrix Σ_x, $\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \mu_x\mu_x^{\mathrm{T}} + \Sigma_x$.

The parameters can then be optimized by iteratively applying the gradients of Equation 11 with a decaying learning rate, η, *e.g.*, $\mathbf{M} \leftarrow \mathbf{M} + \eta\nabla_{\mathbf{M}}$. However, in practice, an exponentiated update, $\mathbf{M} \leftarrow e^{\log\mathbf{M}+\eta\nabla_{\mathbf{M}}}$, preserves a useful property (positive semi-definiteness) of the $\mathbf{M}$ matrix, aiding optimization.

Efficient State Projections

Computing the target posterior requires repeated calculation of state values, $V(\mathbf{s}^G)$, and cumulative trajectory costs, $\sum_{\tau=2}^{T}\mathrm{cost}(\mathbf{s}_\tau^G)$ for different goals (Equation 9). Naïvely, computing the cumulative trajectory costs is computationally expensive, requiring $\mathcal{O}(t|\mathbf{s}|^2)$ time. However, due to the linearity of the cost function, previously computed cumulative costs can be re-used.

Consider the transformation of a cost function from an initial goal G to a new goal G' using rotation matrix $\mathbf{R}_G^{G'}$ and translation vector $\Delta_G^{G'}$:

$$\begin{aligned} \sum_{\tau=1}^{t}\mathbf{s}_\tau^{G'\mathrm{T}}\mathbf{M}\,\mathbf{s}_\tau^{G'} &= \mathbf{M}\cdot\sum_{\tau=1}^{t}\mathbf{s}_\tau^{G'}\mathbf{s}_\tau^{G'\mathrm{T}} \\ &= \mathbf{M}\cdot\sum_{\tau=1}^{t}(\mathbf{R}_G^{G'}\mathbf{s}_\tau^G + \Delta_G^{G'})(\mathbf{R}_G^{G'}\mathbf{s}_\tau^G + \Delta_G^{G'})^{\mathrm{T}} \\ &= \mathbf{M}\cdot\mathbf{R}_G^{G'}\left(\sum_{\tau=1}^{t}\mathbf{s}_\tau^G\mathbf{s}_\tau^{G\mathrm{T}}\right)\mathbf{R}_G^{G'\mathrm{T}} + t\mathbf{M}\cdot\Delta_G^{G'}\Delta_G^{G'\mathrm{T}} \\ &\quad + 2\mathbf{M}\cdot\mathbf{R}_G^{G'}\left(\sum_{\tau=1}^{t}\mathbf{s}_\tau^G\right)\Delta_G^{G'\mathrm{T}}. \end{aligned}$$

Thus, with the previously computed quadratic and linear sums for projection G, only matrix operations independent of the time horizon t are needed. This reduces the $\mathcal{O}(t\,\dim(s)^2)$ time computation to $\mathcal{O}(\dim(s)^2)$ time.