Clustered Trajectory Motion Prediction for Human-Robot Collaboration

Alex Cuellar
Interactive Robotics Group.
MIT
Cambridge, MA
alexcuel@mit.edu

***Abstract*—As automation becomes more integral to workplaces, homes, and other spaces occupied by humans, robots require the ability to predict and avoid humans. In this project, I propose an algorithm able to predict the trajectory of a robot's human collaborator through space given observed clustered trajectories. Given that each human and environment will present unique sets of trajectories, this paper focuses on developing an algorithm able to learn online and make predictions with limited data. Our algorithm builds off the trajectory clustering and segmentation method presented in the HABITS algorithm and makes short-term predictions with more accuracy than mean trajectory methods such as BEST-PTA. Specifically, our algorithm generates a sequence of temporally iterative Gaussian Process Regressions (GPRs) based on near-by points in a dataset to predict a human's trajectory in small but discrete time steps. We tested our algorithm with two datasets of walked human trajectories, and outperformed the vanilla BEST-PTA mean trajectory prediction by 33% and 17.9% with respect to the average displacement error.**

I. Introduction

While robot motion planning as a field has seen significant research in past years, the presence of humans requires any planning system to react in a changing environment. Therefore, a robot must have some ability to predict the human's position over time to effectively plan. Online clustering algorithms such as HABITS [1] tackle part of this problem by grouping similar segments of a human's path. Algorithms such as BEST-PTA [2] then make spatial predictions by aligning a partial trajectory to the cluster's mean trajectory, resulting in a relatively accurate spatial prediction. In this paper, we introduce an algorithm which leverages more information from a partial trajectory to refine spatial predictions beyond the aligned mean trajectory used by the vanilla BEST-PTA algorithm.

To more formally define the problem, we represent the partial trajectory with l observed points as $X = \{\vec{x}_1 \dots \vec{x}_l\}$ where $\vec{x}_i \in \mathbb{R}^3$ is the i-th point in the observed portion of the partial trajectory. Additionally, let $Y_j \in \mathcal{Y}$ be the j-th trajectory in our dataset of observed trajectories. As with our partial trajectory, we represent the i-th point of trajectory Y_j as $\vec{y}_{j,i}$. Our goal is to predict $\hat{X} = \{\hat{\vec{x}}_{l+1}, \dots \hat{\vec{x}}_T\}$ using the dataset $\mathcal{Y}$.

II. Related Works

Many current methods of human trajectory prediction rely on LSTMs to predict paths forward based on images, including [3, 4, 5]. A. Alahi recognizes other humans in an open space and learns to predict trajectories based on these inter-personal social dynamics [3]. Varshneya takes a similar approach, but focuses on learning motion dynamics based on the static objects in the scene for use in more complex spaces [4]. D. M Huynh has a very similar approach to Varshneya, but trains the LSTM based on changes in position rather than sequences of absolute position, giving improved results [5]. In contrast to all three approaches, we wish to make predictions based on limited data. Therefore, we focus on making predictions based on less hungry methods and predict directly off the spatial information of trajectories rather than images.

Other methods rely less on deep learning and make predictions more based on structure in the environment, such as finite defined endpoints (i.e. sinks, workbenches, etc). J Elfring trains a GHMM to discretize environments based on a set of key points though which trajectories diverge and change direction [6]. The algorithm combines this spatial graph with analysis of social forces to make predictions on a human's path through space. Therefore, while the algorithm uses past trajectories to determine desired end-points and break the space into discretized sections, the final smooth trajectory focuses more on the dynamics of social forces rather than prediction from trajectories.

Other prediction algorithms unreliant on deep learning make predictions based directly on observed trajectories. Wiest and Dietmayer propose a prediction algorithm for vehicles, focusing especially on turns [7]. The method uses GMMs to determine the optimal coefficients for a Chebyshev polynomial used to complete a partial trajectory. Therefore, the complexity of any predicted trajectory is limited by the order polynomial used. This constraint is unimportant when predicting a specific task (such as vehicles turning) where the order polynomial necessary can be pre-determined. However, since we want to learn on-line with arbitrary tasks, we cannot estimate the order polynomial necessary to appropriately encode a trajectory. Mainprice and Berenson take another approach, determining the likelihood that voxels in space will be occupied given a partial trajectory [8]. Similar to BEST-PTA trajectory alignment, the algorithm learns to label a partial trajectory as a type of gesture, and determines a representative motion for each gesture with Gaussian Mixture Regression (GMR). However, the algorithm does not further refine the prediction beyond a

representative trajectory of the chosen label. Conversely, our algorithm uses HABITS as a way to label a partial trajectory, then further refine a prediction beyond the representative trajectory.

III. Method

To predict human motion using groups of clustered trajectories, we use iterative instances of Gaussian Process Regression (GPR) to predict the change in a partial trajectory's position from one time step to another. Section $3A$ explains the high-level framework of our iterative prediction. Sections $3B - E$ then describe each constituent piece in more detail.

A. Iterative Framework

Our algorithm is initially given a partial trajectory whose last observed point is $\vec{x}_l$. First, we use a kdtree to determine the KNN points of $\vec{x}_l$ in $\mathcal{Y}$ with respect to the 3 spatial dimensions and time, where time is defined as the number of time steps since the start of the trajectory (i.e. the four dimensional vector $(\vec{x}_l, l)$). These neighbors are used as input to a GPR which predicts the spatial change between $\vec{x}_l$ and the first unknown time step. Denote this change as $\Delta\hat{\vec{x}}_{l+1} = (\Delta\hat{x}_{l+1,x}, \Delta\hat{x}_{l+1,y}, \Delta\hat{x}_{l+1,z})$. Therefore, the first point of our prediction will be:

$$\hat{\vec{x}}_{l+1} = \vec{x}_l + \Delta\hat{\vec{x}}_{l+1} \tag{1}$$

For the next iteration, we find the KNN of $\hat{\vec{x}}_{l+1}$ in $\mathcal{Y}$ to inform another instance of GPR and predict $\hat{\vec{x}}_{l+2}$. We continue this iterative process until the algorithm has predicted enough points such that our full predicted trajectory $\hat{X}$ extends the temporal length of X to the average length of the trajectories in $\mathcal{Y}$.

B. GPR Formulation

When we find the nearest neighbors to a point of interest $\hat{\vec{x}}_i$, we use them as the data for a GPR with 7 features. 3 features are the neighbor's (x, y, z) coordinates, one is the number of time steps after the start of its respective trajectory, and the final 3 features are defined as the direction in which the trajectory is moving. These 3 direction variables are included because trajectories which are moving in a similar direction tend to continue moving in tandem. Therefore, it is advantageous to weigh neighboring points higher if their direction of movement is similar to the point $\hat{\vec{x}}_i$ in question. This direction for a point $\vec{y}_{n,t}$ is defined as:

$$\Delta\vec{y}_{n,t} = \vec{y}_{n,t} - \vec{y}_{n,t-1} \tag{2}$$

Denote the 7-dimensional feature vector of the j-th nearest neighbor as $\vec{a}_j = (\vec{y}_{n,t}, t, \Delta\vec{y}_{n,t})$. We extract the same features from $\hat{\vec{x}}_i$. Here, the direction features are based on:

$$\Delta\hat{\vec{x}}_i = \begin{cases} \hat{\vec{x}}_i - \hat{\vec{x}}_{i-1} & i > l+1 \\ \hat{\vec{x}}_{l+1} - \vec{x}_l & i = l+1 \\ \vec{x}_l - \vec{x}_{l-1} & i = l \end{cases} \tag{3}$$

and call this feature vector $\vec{a} = (\hat{\vec{x}}_i, i, \Delta\hat{\vec{x}}_i)$. With these features, we define covariance matrices to compare the similarity between pairs of points in the neighbors:

$$C_{OO}(\vec{a}_j, \vec{a}_k) = f(\vec{a}_j, \vec{a}_k) \tag{4}$$

where f is a kernel function. The choice of kernel function is explained in section $3C$. For this matrix, we use the subscript OO because we are comparing pairs of observed points in $\mathcal{Y}$. Likewise, we calculate the similarity between the neighbors and the feature vector $\vec{a}$:

$$C_{OP}(\vec{a}_j) = f(\vec{a}_j, \vec{a}) \tag{5}$$

where the subscript OP indicates that we are comparing an observed point to the point in our partial trajectory. We use these covariance matrices to infer the spatial change of our predicted trajectory along one dimension. Let $\vec{d}_{i,x}$, $\vec{d}_{i,y}$, and $\vec{d}_{i,z}$ be the vectors containing the x, y, and z portions of $\Delta\vec{y}_{n,t+1}$ across every point in our nearest neighbors.

Therefore, GPR will predict each dimension of $\Delta\hat{\vec{x}}_{i+1}$ individually. For example, along the y dimension, the change is:

$$\Delta\hat{x}_{i+1,y} = C_{OP}(C_{OO} + \sigma_n^2 I)^{-1}\vec{d}_{i,y} \tag{6}$$

where σ_n^2 is a parameter which accounts for both noise and numeric stability by ensuring our covariance is always invertible.

Algorithm 1: Close Sample Prediction

input: ($\mathcal{Y}$,X,K);
H $\leftarrow$ Hyperparameters from CG or RProp
T $\leftarrow$ Average trajectory length in $\mathcal{Y}$
$Q_l \leftarrow$ K Nearest Neighbors of $\vec{x}_l$ in $\mathcal{Y}$
$(\Delta\hat{x}_{l+1,x}, \Delta\hat{x}_{l+1,y}, \Delta\hat{x}_{l+1,z}) \leftarrow$ GPR$(\vec{x}_l|Q_l, H)$
$\hat{\vec{x}}_{l+1} = \vec{x}_l + (\Delta\hat{x}_{l+1,x}, \Delta\hat{x}_{l+1,y}, \Delta\hat{x}_{l+1,y})$
for *t = l + 2:T* **do**
  $Q_t \leftarrow$ K Nearest Neighbors of $\hat{\vec{x}}_{t-1}$ in $\mathcal{Y}$
  $(\Delta\hat{x}_{x,t}, \Delta\hat{x}_{y,t}, \Delta\hat{x}_{z,t}) \leftarrow$ GPR$(\hat{\vec{x}}_{t-1}|Q_t, H)$
  $\hat{\vec{x}}_t = \hat{\vec{x}}_{t-1} + (\Delta\hat{x}_{t,x}, \Delta\hat{x}_{t,y}, \Delta\hat{x}_{t,z})$
end
Return: $(\hat{\vec{x}}_{l+1} ... \hat{\vec{x}}_T)$

C. Choice of Kernel Function

There are many standard choices of kernel functions for various types of regression problems. For this algorithm, we test with 3 common kernel functions: Squared exponential, Matern 3/2, and Rational Quadratic. Each function is depicted along 1 dimension in Fig. 1. We chose these functions as they will always map feature vectors with smaller euclidean distances to higher likelihoods. In other words, $||\vec{a} - \vec{b}|| < ||\vec{c} - \vec{d}|| \iff f(\vec{a}, \vec{b}) > f(\vec{c}, \vec{d})$.

Specifically, the squared exponential kernel function is:

$$f(\vec{a}, \vec{b}) = \sigma^2 \exp\left(-\frac{||\vec{a} - \vec{b}||^2}{2l^2}\right) \tag{7}$$

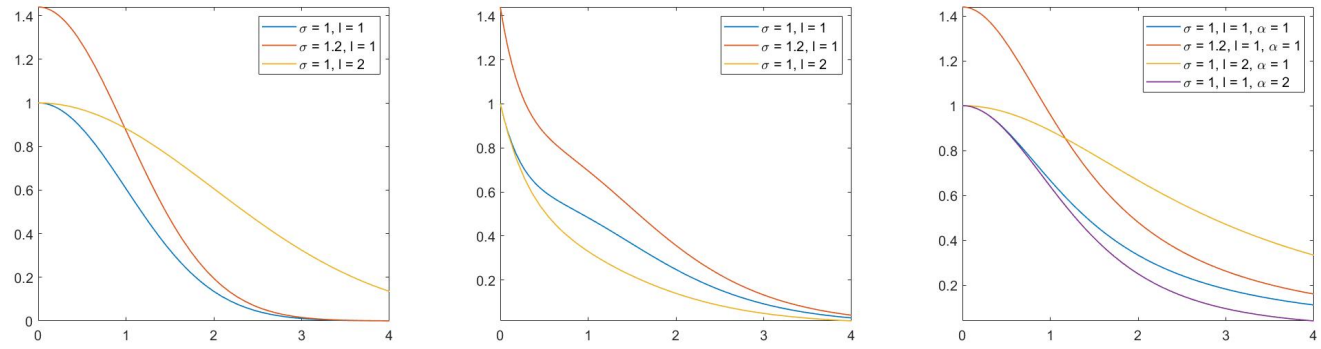

Fig. 1. A 1-Dimensional depiction of the three kernel functions tested for our algorithm. When we are using these kernel functions, x is the euclidean distance between two feature vectors and $f(x)$ is the calculated similarity based on x. (Left) Squared Exponential. (Center) Matern 3/2. (Right) Rational Quadratic.

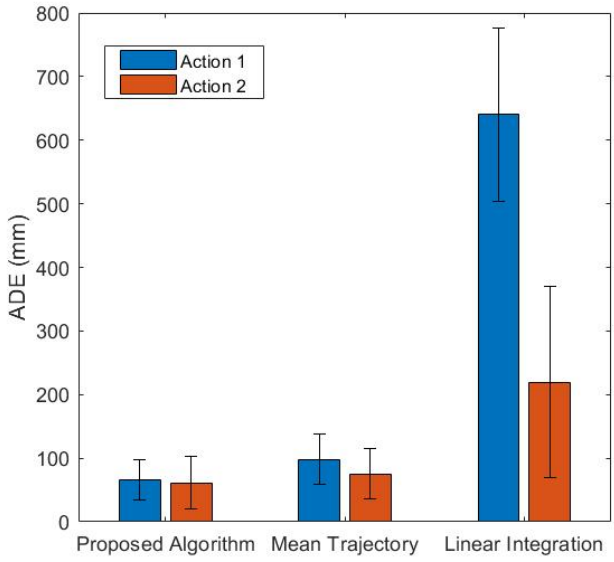

Fig. 2. The mean and standard deviations of ADE for our algorithm and 2 baselines – the aligned mean trajectory determined by the vanilla BEST-PTA and the single step linear integration approximation. These methods are compared across both actions we tested.

where σ and l are hyperparameters. The Matern 3/2 kernel is:

$$f(\vec{a},\vec{b}) = \sigma^2\left(1+\sqrt{3}\frac{||\vec{a}-\vec{b}||}{2l}\right)\exp\left(\sqrt{3}\frac{||\vec{a}-\vec{b}||}{2l}\right) \quad (8)$$

Where σ and l are again hyperparameters. Lastly, the rational quadratic kernel function is:

$$f(\vec{a},\vec{b}) = \sigma^2\left(1+\frac{||\vec{a}-\vec{b}||^2}{2\alpha l^2}\right)^{-\alpha} \quad (9)$$

where σ, l, and α are all hyperparameters.

D. Hyperparameter Optimization

For any choice of kernel function f, we need to determine a set of hyperparameters to inform the similarity function represented by the kernel. In order to optimize these hyperparameters, we attempt two different methods for gradient-based optimization: Nonlinear Conjugate Gradients (CG) [9] and Resilient Back-propagation (RProp) [10].

Nonlinear Conjugate Gradient methods minimize an arbitrary function $f(x)$ iteratively through:

$$x_{k+1} = x_k + a_k d_k \quad (10)$$

Where the stepsize a_k is defined by a line-search and d_k is defined as:

$$d_k = \begin{cases} -\nabla f(x_k) & k = 1 \\ -\nabla f(x_k) + \beta_k d_{k-1} & k \geq 2 \end{cases} \quad (11)$$

We use the Ribiere-Polyak definition of β_k as:

$$\beta_k = \frac{\nabla f(x_k)^T(\nabla f(x_k) - \nabla f(x_{k-1})}{||\nabla f(x_{k-1})||^2} \quad (12)$$

Note that we estimate gradients via comparison between chosen points in x_k since no symbolic gradient is available.

The RProp algorithm on the other hand only utilizes the sign of a function's gradient, and the update is simply:

$$x_{k+1} = x_k + \eta_k \text{sgn}(\nabla f(x_k)) \quad (13)$$

where

$$\eta_k = \begin{cases} \eta_{k-1}\cdot c_p & \nabla f(x_k)\cdot\nabla f(x_{k-1}) > 0 \\ \eta_{k-1}\cdot c_n & \nabla f(x_k)\cdot\nabla f(x_{k-1}) < 0 \\ \eta_{k-1} & otherwise \end{cases} \quad (14)$$

Where $c_p > c_n$, ensuring we move less along the gradient when the gradient's sign is changing between time steps (i.e. when two time steps ask us to go in opposite directions). In our tests, we use the most common values $c_p = 1.2$ and $c_n = .5$. If we wish to optimize the hyperparameters for each temporal

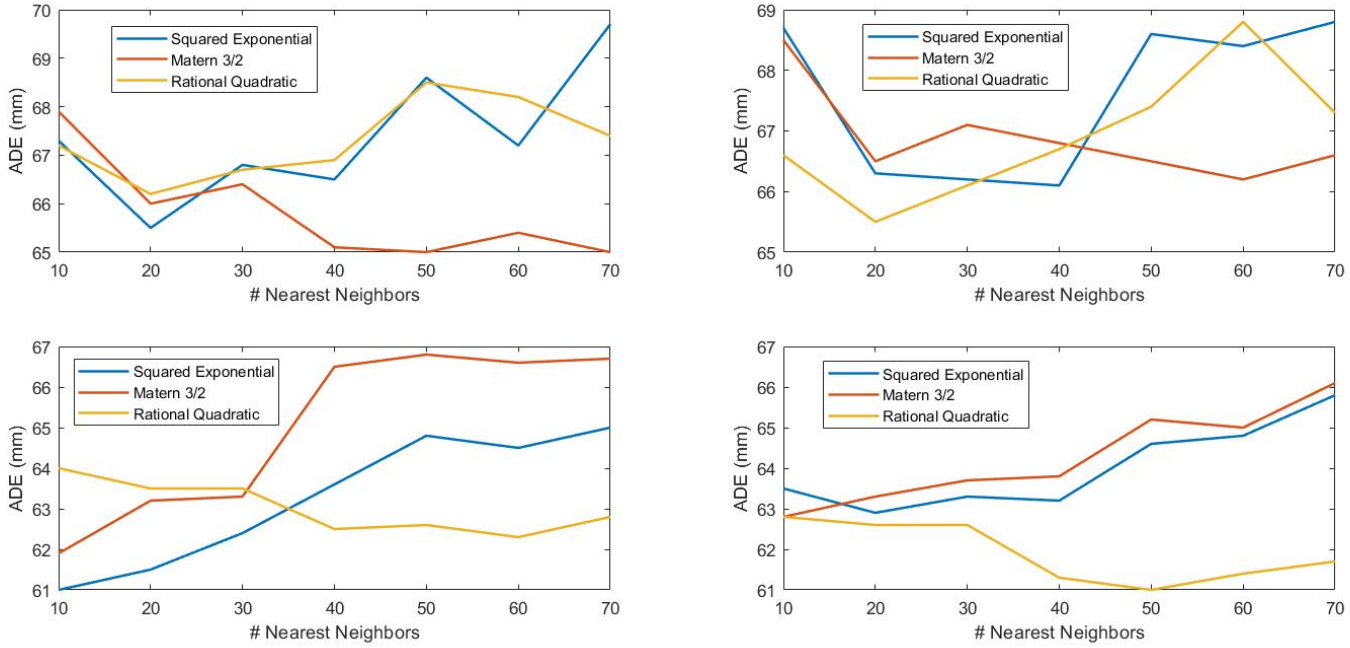

Fig. 3. A comparison of the ADE for different actions and hyperparameters. Each graph shows the ADE vs number of nearest neighbors used in GPR for each kernel function. Each of the four graphs show these data for a unique combination of action and hyperparameter optimization method. (Top Left) ADE for action 1 using CG optimization. (Top Right) ADE for action 1 using Rprop optimization. (Bottom Left) ADE for action 2 using CG optimization. (Bottom Right) ADE for action 2 using Rprop optimization

iteration of GPR in our prediction, we found the algorithm's run time becomes prohibitively long. Instead, we optimized the kernel function's hyperparameters over a random subset of points in the trajectories $\mathcal{Y}$ and use these hyperparameters for each iterative instance of GPR. When using this sample-based method, we did not see substantial loss in prediction accuracy.

E. Complementary Filter with Mean Trajectory

The prediction method described in sections $3A - D$ works to decrease error of predictions in the short term (see section 4 for details on analysis). However, as the algorithm predicts further into the future, the errors incurred by our method and the BEST-PTA mean trajectory approximately converge. Additionally, the standard deviation of error incurred in our prediction can increase faster with time than the mean trajectory (see Fig 4). To combat this increase in error standard deviation, we introduce a complementary filter favoring our prediction method early on and the mean trajectory later. Let the predicted trajectory be $\hat{X}$, the mean trajectory be $\bar{X}$, and a function that maps a time step $\bar{X}$ to the DTW aligned time step in X be $a(t)$. Our complementary filter is:

$$\hat{X}_{t,filtered} = \sigma(t,\theta)\bar{\vec{x}}_{a(l)+t} + (1-\sigma(t,\theta))\hat{\vec{x}}_t \qquad (15)$$

where θ are parameters defining the center and width of our sigmoid:

$$\sigma(t,\theta) = \frac{1}{1+e^{-\theta_2(x-\theta_1)}} \qquad (16)$$

Currently, we are simply using $\theta_1 = T/2$ and $\theta_2 = 4/T$ where T is the total number of points in our prediction. In future work, we will optimize over these parameters to minimize error.

IV. Evaluation and Results

To evaluate the algorithm, we compare our performance against BEST-PTA's aligned mean trajectory and single step velocity integration. For our analysis, we focus primarily on standard metrics for analyzing human motion prediction across 2 optimization parameters - kernel function and hyperparameter optimization method.

A. Dataset and Methodology

For our analysis, we focus on walked paths during a task where the participant was asked to retrieve several objects from various locations in a room and drop them off one at a time in a predetermined region. We use HABITS to cluster each full trial into segmented paths of particular actions (i.e. exit from one region and entrance into another). To evaluate our algorithm, we use leave-one-out-cross-validation on the action datasets, beginning our prediction 20 samples (.4 seconds) into the trajectory (i.e. the observed portion of X has a length of 20).

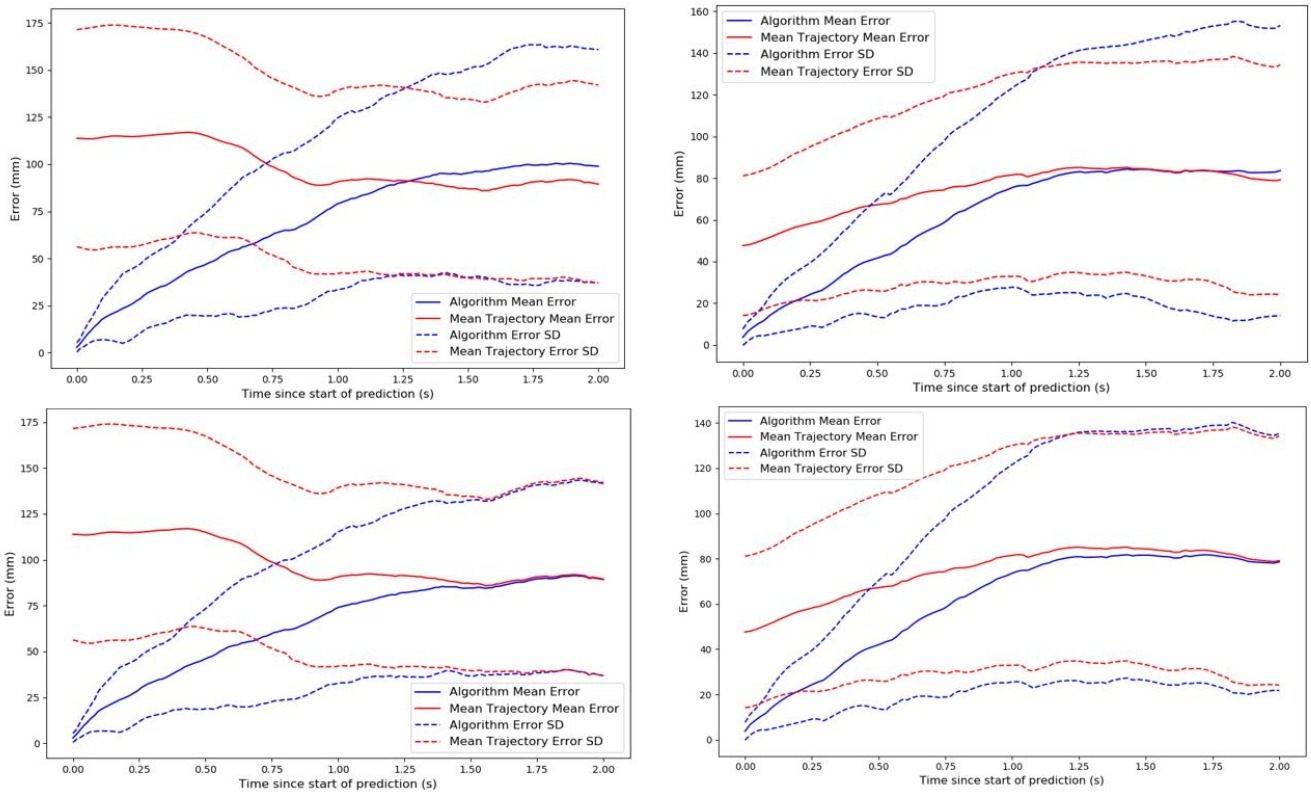

Fig. 4. Graphs comparing the euclidean error incurred by BEST-PTA's mean trajectory and our method with respect to time. Each graph shows the errors for a combination of action and usage of complementary filter. (Top Left) Action 1 without complementary filter. (Top Right) Action 2 without complementary filter. (Bottom Left) Action 1 with complementary filter. (Bottom Right) Action 2 with complementary filter.

In this dataset, a PhaseSpace motion capture system is used to track a participant's head at a rate of 50 Hz. While HABITS segments each trial into many actions, we focus on two actions and one participant. This participant completed action 1 45 times and action 2 22 times, each taking $\approx 2-3$ seconds to complete.

For our analysis, we use 2 baselines: the aligned mean trajectory generated by BEST-PTA and single step velocity integration (i.e. a linear prediction starting at the last observed point). The metric we use to analyze our algorithm is the Average Displacement Error (ADE) defined as the average euclidean distance between the prediction and the ground truth trajectory segment over all time steps. Note that we only consider a number of time steps equal to the minimum of the true trajectory and the predicted trajectory. In other words, we only consider the time steps for which there is a one-to-one mapping between the ground truth trajectory and the predicted trajectory.

B. Results

The results of our evaluation give evidence that our algorithm provides an improvement against both the mean trajectory prediction and single step velocity integration with respect to ADE. While the two actions have slightly different minimum ADEs across optimization method and kernel function, using a squared exponential kernel with 20 nearest neighbors and optimization via conjugate gradients minimizes overall error across both actions. As seen in Fig 2, these optimal parameters result in a 33% lower ADE with respect to the mean trajectory and an 89.8% lower ADE with respect to single step velocity integration for action 1. For action 2, we see a 17.9% lower ADE with respect to the mean trajectory prediction and a 72.0% lower ADE with respect to single step velocity integration. Furthermore, across both actions, our algorithm shows statistical significance over each baseline using a t-test with a p-value of .05.

While we can define optimal parameters for the algorithm, the relatively low disparity in ADE across all parameters (as shown in Fig 3) is worthy of note. I believe this to be a result of relatively consistent signals in the data resulting from HABITS clustering. Since all trajectories are already segmented and clustered by position, directionality, start, and end points, the data give a very clear signal for the prediction. Therefore, changes in parameters always read near-identical signals from the data and make consistently well-informed predictions.

C. Impact of the Complementary filter

Fig 4 provides insight into how our algorithm performs over time with respect to the mean trajectory with and without the complementary filter. In both action 1 and action 2 without the complementary filter, our algorithm gives significantly lower error than the mean trajectory for approximately the

first 1.25 seconds of the prediction, at which point the errors approximately converge on the mean trajectory's prediction. Additionally, for both actions we see the standard deviation of the prediction expand wider than the mean trajectory as the errors of the two converge.

In contrast, when applying the complementary filter, neither action shows the standard deviation of error expanding. More interestingly, with the complementary filter applied, the average ADE decreases in action 1 by 8.0% and in action 2 by 2.8%, using the optimal parameters described in section $4B$. This improved error seems to largely result from the complementary filter allowing our algorithm to increase the time during which we can make predictions better than the mean trajectory. However, more testing and analysis on more actions will be necessary to make any defensible claims on the ADE benefits of a complementary filter.

REFERENCES

[1] J. Shah C. Fourie, P. Lasota. Human anticipation based on integrating temporal, spatial, and semantic components (habits).

[2] J. Shah P. Lasota. Bayesian estimator for partial trajectory alignment. 2019.

[3] V. Ramanathan A. Robicquet L. Fei-Fei S Savarese A. Alahi, K. Gel. Social lstm: Human trajectory prediction in crowded spaces. pages 961–971, 2016.

[4] G. Srinivasaraghavan Varshneya. Human trajectory prediction using spatially aware deep attention models.

[5] G. Alaghband D. M Huynh. Scene-lstm: A model for human trajectory prediction.

[6] M. Steinbuch J Elfring, R. Monengraft. Learning intentions for improved human motion prediction. pages 591–602, 2014.

[7] U. Kreßel Wiest, M. Höffken and K. Dietmayer. Probabilistic trajectory prediction with gaussian mixture models. pages 141–146, 2012.

[8] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. 2013.

[9] P. L. Bartlett A. J. Smola. Sparse greedy gaussian process regression. pages 619–629, 2001.

[10] Blum M. M Blum, R. A. Martin and Martin A. Riedmiller. Optimization of gaussian process hyperparameters using rprop. 2013.