Gaussian Process Based Model Predictive Controller for Imitation Learning

Vladimir Joukov, and Dana Kulić

Abstract— **Humans still outperform robots in most manipulation and locomotion tasks. Research suggests that humans minimize a task specific cost function when performing movements. In this paper we present a Gaussian Process based method to learn the underlying cost function, without making assumptions on its structure, and reproduce the demonstrated movement on a robot using a linear model predictive control framework. We show that the learned cost function can be used to prioritize between tracking and additional cost functions based on exemplar variance, and satisfy task and joint space constraints. Tuning the weighting between learned position and velocity costs produces trajectories of the desired shape even in the presence of constraints. The approach is validated in simulation with a simple 2dof manipulator showing joint and task space tracking and with a 4dof manipulator reproducing trajectories based on a human handwriting dataset.**

I. Introduction

Humans are capable of complex manipulation and locomotion tasks. They are able to achieve energy-efficient gait, reject disturbances, handle changing loads, and adapt to environmental constraints. Using inspiration from the human body, robotics researchers aim to develop systems with similar capabilities [10]. It is often infeasible to manually program a complex robot behavior by hand, imitation learning allows to use demonstrations of an expert (human) to learn and transfer their skill onto a robot.

Approaches to imitation learning can be separated into trajectory and controller based categories. The former focuses on learning a model of the trajectory in either joint or task space. Once a model is learned, human-like trajectories can be generated and tracked by manipulators utilizing classical control methods. The latter learns the controller used by the demonstrator. A manipulator employing the learned controller should then automatically produce human like motion [17].

A common trajectory based learning method is to segment the expert demonstration into a collection of movement primitives. Once learned, primitives can be used to produce trajectories for the robot to follow using feedback control. Temporally and spatially invariant movement primitives based on nonlinear dynamic equations have been proposed by Ijspeert et al. [7]. They learn a goal oriented movement or a limit cycle and can be updated on-line. Movement primitives can also be probabilistic, by modeling the exemplar trajectories using Gaussian Mixture Models in a lower dimensional latent space [3]. The optimal trajectory on the latent space, subject to body constraints, can be solved by Lagrangian optimization and projected back to joint space. Another probabilistic method represents the trajectories as a set of states and probabilities of transitioning between them using Hidden Markov Models (HMMs) [8]. Optimization can then be used to generate human-like trajectories based on the trained HMMs [11]. Recently, optimization techniques have also been proposed to identify a trajectory that optimally trades off between similarity to human demonstration, smoothness, and obstacle constraints [14].

When learning the controller directly it is often assumed that the expert control policy is minimizing some integral cost function. If the cost is a linear combination of known basis functions it is possible to estimate their coefficients using bi-level optimization which consists of coefficient estimation on the upper level and constrained optimal control problem on the lover level [4]. A key drawback of this approach is the assumption that the basis functions are known and their coefficients remain constant throughout the entire motion. Furthermore, the bi-level approach is computationally demanding, making it difficult to use with large exemplar datasets. Often the trajectories are re-sampled at large intervals to make the problem computationally tractable [4]. By assuming the exemplar trajectories are optimal and employing the Karush-Kuhn-Tucker (KKT) necessary optimality conditions it is possible to simplify finding the coefficients to a least squares problem [15].

Trajectory based methods may model the distribution of human movements, but how this information is used is left up to the controller. On the other hand, controller methods typically use a fixed cost function for the entire duration of the trajectory and do not consider the variance in human motion. To incorporate probabilistic modeling into a control scheme Englert *et al.*used Gaussian Processes (GPs) to model manipulator dynamics and expert exemplars [5]. The control policy is then learned as a radial basis function network such that the Kullback-Leibler divergence is minimized between the learned GPs. This approach trains a robot specific controller and does not incorporate task or joint space constraints. Calinon used Gaussian Mixture Models (GMM) to build task-parameterized models and showed how they can be utilized in a model predictive control framework to generate desired accelerations [2]. Building the GMM from multiple frames of reference leads to good generalization of the task, however the double integrator controller formulation does not explicitly take into account the manipulator's torque limits and dynamics.

In this paper we show that a GP model trained with expert demonstrations can be directly utilized as a time variant quadratic cost function for task or joint space manipulator control. The proposed GP based cost is quadratic, correctly incorporates variance of the expert exemplars, and does not assume any specific structure. The learned cost function is then incorporated into a linear model predictive control (MPC) framework along with any additional costs and system constraints. Utilizing a task space inverse dynamics controller within the framework allows the approach to learn and reproduce task space end effector trajectories while automatically handling any differences between expert and robot kinematics, dynamics, and constraints. We validate the proposed approach in simulation, the learned controller is shown to accurately track the trajectory while simultaneously minimizing control effort.

The rest of the paper is organised as follows. Section II provides background on Gaussian Process regression. Section III includes the derivation of the proposed approach, utilizing Gaussian Processes to model state variables and extract a quadratic cost function. Section IV validates the learned controller in simulation on a double pendulum system and on real human data. Finally, section V concludes the paper and discusses future research directions.

II. Background

A. Gaussian Processes

In machine learning Gaussian Processes are a popular tool to express distribution over functions [16]. A stochastic process $g(\mathbf{p})$ can be described by its mean and covariance functions:

$$m(\mathbf{p}) = \mathbb{E}[g(\mathbf{p})] \tag{1}$$

$$k(\mathbf{p}, \mathbf{p}^T) = \mathbb{E}[(g(\mathbf{p}) - m(\mathbf{p}))(g(\mathbf{p}^T) - m(\mathbf{p}^T))] \tag{2}$$

Assuming all random variables $g(\mathbf{p})$ are jointly Gaussian, one can say that $g(\mathbf{p}) \sim GP(m(\mathbf{p}), k(\mathbf{p}, \mathbf{p}^T))$ is a Gaussian Process.

1) Observation Prediction: Consider a simple Gaussian Process with zero mean and some covariance function $k(\mathbf{p}, \mathbf{p})$. Given c training input output pairs $\{\mathbf{P},\ \mathbf{G}\} = \{\mathbf{p}_i, g_i, i = 1 \cdots c\}$ the joint distribution of the training data and a previously unseen test point $\{\mathbf{p}_*, g_*\}$ can be written as:

$$\begin{bmatrix} \mathbf{G} \\ g_* \end{bmatrix} \sim \mathcal{N}(0,\ \begin{bmatrix} k(\mathbf{P}, \mathbf{P}) & k(\mathbf{P}, \mathbf{p}_*) \\ k(\mathbf{p}_*, \mathbf{P}) & k(\mathbf{p}_*, \mathbf{p}_*) \end{bmatrix}). \tag{3}$$

Typically, it is assumed that the exact outputs are not available and instead the training outputs $y = g(\mathbf{p}) + \epsilon$ are corrupted with by zero mean Gaussian noise with variance σ_n^2. In this case, the predictive equations for the Gaussian process are:

$$g_* | \, \mathbf{P}, \mathbf{Y}, \mathbf{p}_* \sim \mathcal{N}(\bar{g}_*, \Sigma_*), where \tag{4}$$

$$\bar{g}_* = k(\mathbf{p}_*, \mathbf{P})[k(\mathbf{P}, \mathbf{P})^{-1} + \sigma_n^2 I]\mathbf{Y} \tag{5}$$

$$\Sigma_* = k(\mathbf{p}_*, \mathbf{p}_*) - k(\mathbf{p}_*, \mathbf{P}) \underbrace{[k(\mathbf{P}, \mathbf{P}) + \sigma_n^2 I]^{-1}}_{K^{-1}} k(\mathbf{P}, \mathbf{p}_*). \tag{6}$$

Since differentiation is a linear operator, the derivative of a Gaussian process with respect to the inputs is also a Gaussian process [12].

2) Covariance Function: Note that the covariance function of the outputs, $k(\mathbf{p}, \mathbf{p})$, is defined in terms of the inputs. Typically the squared exponential covariance function is used in Gaussian Process regression,

$$k(\mathbf{p}_1, \mathbf{p}_2) = \sigma_f^2 exp(-\frac{1}{2l^2}(\mathbf{p}_1 - \mathbf{p}_2)^T(\mathbf{p}_1 - \mathbf{p}_2)) \tag{7}$$

where σ_f^2 and l are tuning parameters, also known as the hyperparameters.

3) Output Probability Density Function: After collecting training data points and tuning the covariance function parameters, the predictive equations for the Gaussian process can also be used to compute the probability density at a previously unseen point $\{\mathbf{p}_*, g_*\}$ using the Gaussian distribution probability density function.

$$pdf(g_* | \, \mathbf{p}_*) = \frac{1}{2\sqrt{\pi \Sigma_*}} exp(\frac{(g_* - \bar{g}_*)^2}{2\Sigma_*}) \tag{8}$$

Where $\bar{g}_*$ and Σ_* are functions of $\mathbf{p}_*$ computed using equations 6 and 7.

B. Model Predictive Control

Model predictive control uses the dynamical model of the plant to predict the state $\mathbf{x_k}$ over a horizon of length N and finds the optimal control law $\mathbf{u_k}$ to minimize a cost function. When dealing with linear systems of the form

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} + \mathbf{B}\mathbf{u_k} \tag{9}$$

and a quadratic cost function

$$J = \sum_{i=k+1}^{i=k+N} \mathbf{x_i}^T \mathbf{Q_i} \mathbf{x_i} + \mathbf{u_i}^T \mathbf{R_i} \mathbf{u_i} \tag{10}$$

where $\mathbf{Q_i}$ and $\mathbf{R_i}$ are positive definite matrices, the optimal control signal can be efficiently computed while satisfying linear constraints on the state and control signal.

$$U_{min} \le \mathbf{P_u}\mathbf{u_k} \le U_{max} \tag{11}$$

$$X_{min} \le \mathbf{P_x}\mathbf{x_k} \le X_{max} \tag{12}$$

By re-writing the state equations over a predictive horizon of length N only in terms of the control input $\mathbf{u_{1:N}}$:

$$\begin{bmatrix} \mathbf{x_{k+1}} \\ \mathbf{x_{k+2}} \\ \vdots \\ \mathbf{x_{k+N}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \\ \mathbf{AB} & \mathbf{B} & \mathbf{0}\cdots & \\ \vdots & & & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix}}_{\bar{\mathbf{S}}} \underbrace{\begin{bmatrix} \mathbf{u_k} \\ \mathbf{u_{k+1}} \\ \vdots \\ \mathbf{u_{k+N-1}} \end{bmatrix}}_{\mathbf{u}_{1:N}} + \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}}_{\bar{\mathbf{T}}} \mathbf{x_k} \tag{13}$$

the cost function appears in quadratic form

$$J = \mathbf{u}_{1:N}^T \underbrace{(\bar{\mathbf{R}} + \bar{\mathbf{S}}^T \bar{\mathbf{Q}} \bar{\mathbf{S}})}_{H} \mathbf{u}_{1:N} + \underbrace{\mathbf{x}_k 2 \bar{\mathbf{T}}^T \bar{\mathbf{Q}} \bar{\mathbf{S}}}_{f^T} \mathbf{u}_{1:N}. \tag{14}$$

where $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$ are block diagonal concatenations of $\mathbf{Q_i}$ and $\mathbf{R_i}$ over the horizon. Standard quadratic programming methods can then be used to solve for the optimal $\mathbf{u}_{1:N}$. The first optimal control signal is applied to the plant and the process is repeated [6].

III. Gaussian Process Model Predictive Control

We propose to utilize the probability density function of a Gaussian Process trained on exemplar data as the cost function in a linear model predictive control framework. Our formulation allows us to learn the cost function from exemplar trajectories as well as incorporate any additional desired quadratic cost functions. Consider multiple exemplar trajectories of an n dimensional state vector $\mathbf{x} = [x_1 x_2 \cdots x_n]^T$. We would like our controller to track $\mathbf{x}(t)|t \in [t_0\ t_f]$ provided from expert human demonstrations. For example these can represent complex fighter pilot maneuvers we would like to apply on an autonomous aerial vehicle or human reaching motions we would like to reproduce on a manipulator. From the exemplar trajectories we can compute the variance of each state variable at each sample of time. At times when the expert is particularly concerned with a state variable we expect the variance to be low while at a time when the state variable is not important for task completion the variance will be high [19]. Typically, during a reaching motion the human hand will have high variance throughout the middle of the trajectory and low variance towards then end as it approaches the target [13]. Our learned optimal controller should follow the mean trajectory very closely during the low variance regions and can focus on minimizing additional cost functions in the high variance regions.

Using the exemplar trajectories, assuming they are temporally aligned, we build a time based Gaussian Process model for each of the state variables and directly assign the variance at each time step using the output noise parameter σ_x.

$$\begin{aligned} x_1 &\sim GP_1(t) \sim \mathcal{N}(\bar{x}_1(t), \Sigma_1(t)) \\ x_2 &\sim GP_2(t) \sim \mathcal{N}(\bar{x}_2(t), \Sigma_2(t)) \\ &\vdots \\ x_n &\sim GP_n(t) \sim \mathcal{N}(\bar{x}_n(t), \Sigma_n(t)) \end{aligned}$$

Where $\bar{x}_i(t)$ and $\Sigma_i(t)$ are functions of the exemplar data and current time t.

Discretizing the system with a sampling time of Δt we use the learned GP and temporal horizon of N samples $\mathbf{t} = \{t, t+\Delta t \ldots t+N\Delta t\}$ to predict the mean $\bar{\mathbf{x}}_{it} = \{\bar{x}_i(t)\ \bar{x}_i(t+\Delta t) \cdots \bar{x}_i(t+N\Delta t)\}$ and an $N \times N$ covariance matrix $\mathbf{\Sigma}_{it}$ for each of the state variables. Making the assumption that the state variables are independent we can compute the probability density of the horizon for the entire state $\mathbf{x_t} = \{\mathbf{x}(t)\ \mathbf{x}(t+\Delta t) \cdots \mathbf{x}(t+N\Delta t)\}$ as

$$pdf(\mathbf{x_t}|\mathbf{t}) = \prod_{i=1}^{n} (2\pi)^{-1/2} |\mathbf{\Sigma}_{it}|^{-1/2} e^{-\frac{1}{2}(\mathbf{x}_{it}-\bar{\mathbf{x}}_{it})^T \mathbf{\Sigma}_{it}^{-1}(\mathbf{x}_{it}-\bar{\mathbf{x}}_{it})} \tag{15}$$

Notice that the maximum of the pdf over the interval $[t_0\ t_f]$ with respect to $\mathbf{x}$ is the mean of the Gaussian Process state representation which closely tracks the mean of the expert exemplars. We view the pdf as a time dependent cost function of the state for a horizon of N samples where the cost is proportional to the distance to the Gaussian Process mean, weighted by the variance. Furthermore since probability density is positive definite we can instead minimize its negated logarithm. Taking the logarithm and removing terms independent of the components of state vector horizon $\mathbf{x}_{it}$

$$log(-pdf(\mathbf{x_t}|\mathbf{t})) \sim \sum_{i=0}^{n} \frac{1}{2}\mathbf{x}_{it}^T \mathbf{\Sigma}_{it}^{-1}\mathbf{x}_{it} - \bar{\mathbf{x}}_{it}^T \mathbf{\Sigma}_{it}^{-1}\mathbf{x}_{it} \tag{16}$$

Using element manipulation of $\mathbf{x}_{it}$, $\bar{\mathbf{x}}_{it}$, and $\mathbf{\Sigma}_{it}^{-1}$ the above equation can be written in matrix form in terms of the state $\mathbf{x_t}$ over a horizon of length N and the GP predicted horizon mean $\bar{\mathbf{x}}_\mathbf{t}$ and covariance $\mathbf{\Sigma}_\mathbf{t}^{-1}$.

$$log(-pdf(\mathbf{x_t}|\mathbf{t})) \sim \frac{1}{2}\mathbf{x}_\mathbf{t}^T \mathbf{\Sigma}_\mathbf{t}^{-1}\mathbf{x_t} - \bar{\mathbf{x}}_\mathbf{t}^T \mathbf{\Sigma}_\mathbf{t}^{-1}\mathbf{x_t} \tag{17}$$

The quadratic form and positive definiteness of the covariance matrix allows us to directly utilize it in the standard constrained linear model predictive control formulation, incorporating any additional costs and placing constraints on the state as well as the control input. Assuming a sampling interval of Δt we add the Gaussian Processes cost to H and f^T in equation (14)

$$H' = H + \bar{\mathbf{S}}^T \mathbf{\Sigma}_\mathbf{t}^{-1} \bar{\mathbf{S}} \tag{18}$$

$$f'^T = f^T - \bar{\mathbf{S}}^T (\bar{\mathbf{x}}_\mathbf{t}^T \mathbf{\Sigma}_\mathbf{t}^{-1})^T \tag{19}$$

and use a modified cost function

$$J' = \mathbf{u}_{1:N}^T H' \mathbf{u}_{1:N} + f'^T \mathbf{u}_{1:N}. \tag{20}$$

The solution of the modified quadratic problem results in a model predictive controller that will closely track expert exemplar mean in regions of low variance and focus on minimizing additional costs in areas of high variance.

A. Application to Manipulators

Our goal is to apply the proposed approach to control a manipulator and reproduce human motion while satisfying the manipulator's joint limits and torque constraints, task space constraints, as wells as minimizing additional quadratic cost functions such as control effort. We define the state of the system as the joint ($\mathbf{q}$, $\dot{\mathbf{q}}$) and end effector ($\mathbf{x}$, $\dot{\mathbf{x}}$) positions and velocities and linearize the dynamics about the current operating point.

$$\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{1} & \mathbf{\Delta t} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{\Delta t^2 J}}{2} & \mathbf{1} & \mathbf{\Delta t} \\ \mathbf{0} & \mathbf{\Delta t \dot{J}} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}_{k} + \begin{bmatrix} \frac{\mathbf{\Delta t^2}}{2} \\ \mathbf{\Delta t} \\ \frac{\mathbf{\Delta t^2 J}}{2} \\ \mathbf{J} \end{bmatrix} \ddot{\mathbf{q}} \tag{21}$$

Where $\mathbf{J}$ and $\dot{\mathbf{J}}$ are the Jacobian and its derivative and $\ddot{\mathbf{q}}$ is the control signal calculated in the inverse dynamics sense [18] using the manipulator's joint torques τ, the inertia matrix $\mathbf{M}$, and the vector of Coriolis, centrifugal, and gravitational terms $\mathbf{V}$ at the current state.

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\tau - \mathbf{V}) \tag{22}$$

This formulation allows us to train the GP using either joint trajectories if the kinematics of the manipulator are the same as those of the expert, or end effector trajectories. Similarly, constraints and additional cost functions can be in terms of both joint and task space. The linearity of the inverse dynamics controller allows us to add torque constraints to the system

$$\tau_{\mathbf{min}} - \mathbf{V} \leq \mathbf{M}\ddot{\mathbf{q}} \leq \tau_{\mathbf{max}} - \mathbf{V} \tag{23}$$

and quadratic torque cost of $\tau^T \mathbf{R}_\tau \tau$ by incorporating it into equation 14

$$\bar{\mathbf{R}} = \mathbf{M}^{\mathbf{T}}\mathbf{R}_\tau \mathbf{M} \tag{24}$$

$$f^T = \mathbf{x}_k 2\bar{\mathbf{T}}^T\bar{\mathbf{Q}}\bar{\mathbf{S}} + \mathbf{M}^{\mathbf{T}}\mathbf{R}_\tau^{\mathbf{T}}\mathbf{V}. \tag{25}$$

A useful feature of the proposed approach is that only positional data is needed for training while the velocity GPs can be computed through GP differentiation, avoiding numerical differentiation, which is sensitive to measurement noise, and ensuring that the relative quadratic costs between position and velocity are correct. Also note that since GP estimates a continuous function, the approach does not require the same training and control sampling rates. Thus we can use low sample rate training data in a fast controller without additional data interpolation.

IV. Experimental Results

First we illustrate the proposed approach on a 2 dof planar manipulator acting in the y-z plane. The proposed approach is capable of prioritizing additional cost functions in regions of high exemplar variance and provides accurate tracking in low variance regions. The simulation uses GPs trained either in joint or task space. Next, in simulation, we use the Barret WAM manipulator and the Lasa hand writing dataset [9] to show how the proposed approach can be used with GPs learned from real human motion.

A. Planar Manipulator

We aim to track a joint and a task space trajectory with a 2 link planar manipulator acting in the y-z plane. Both links are 0.5 meters in length, the mass of the first and second links are 3 and 2 kg respectively and the inertia matrix is computed using the thin rod assumption. An error of 10% is added to the dynamic parameters when computing the inverse dynamics to simulate imperfect modeling. We generate a trajectory as a sum of sinusoids and use it in both joint and task space simulations.

$$\begin{aligned} y = q_1 &= sin(2t)0.5 + sin(t)0.2 + sin(10t)0.05 + 1 \\ z = q_2 &= sin(3t)0.2 + sin(1.5t)0.1 + sin(7t)0.035 + 1 \end{aligned} \tag{26}$$

The GP is trained by sampling the trajectory at 50Hz, using the squared exponential covariance function with σ_f and l set to 30 and 10 respectively. Our model predictive controller uses a horizon of 20 time steps and the sampling rate is 100Hz. In addition to the GP we add a torque cost of $R_\tau = 10^{-3}$.

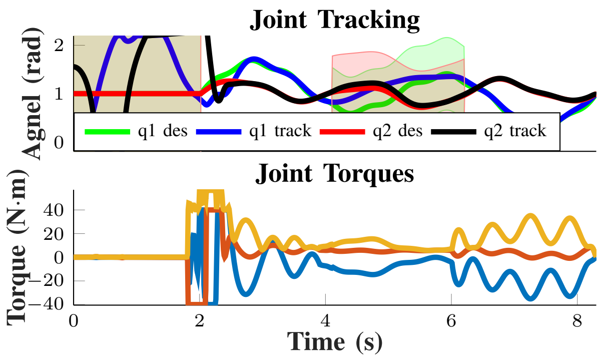

Fig. 1. Trajectory tracking (top), applied joint torques and torque norm (bottom).The trajectory begins with a static region of 2 seconds and transitions into the sum of sinusoids 26. In the static trajectory region the GP variance is extremely high and thus the model predictive controller prioritizes minimizing torque and allows the manipulator to swing freely. As soon as the low variance region enters the predictive horizon, maximum torque is applied to quickly converge on the desired trajectory. In the medium variance region there is a trade-off between accurate tracking and torque minimization. The torque quadratic cost acts as a tuning coefficient for the trade-off.

1) Joint Space Tracking: In this case the state only includes the joint angles and velocities. The trajectory begins with a 2 second static region and progresses to the sum of sinusoids 26. In the static trajectory region GP output variance σ_n is set to 10^3 to prioritize torque cost, allowing the manipulator to swing freely. In the middle region we set it to 0.75 to show how the proposed approach can automatically trade-off between tracking accuracy and minimization of additional cost functions. For the rest of the trajectory the GP σ_n is 0.05 for accurate tracking. Figure 1 shows the controller performance.

2) Task Space Tracking: Next we use the same planar manipulator to track a 6 second task space trajectory in the y-z plane. The y and z coordinates are generated as a sum of sinusoids (Equation 26) and their GP is trained with σ_n of 0.05. The state of the system includes both joint and task space components as defined in 21. The following constraints are added into the MPC formulation:

$$\begin{aligned} StateConstraints: &\; q_2 \leq 2,\; y_{ee} \leq 1.6,\; z_{ee} \geq 0.8 \\ ControlConstraints: &\; |\tau| \leq 40 \end{aligned}$$

By increasing the weight of the GP position or velocity quadratic costs it is possible to focus on tracking the mean trajectory or keeping the velocity profile and thus the general shape of the trajectory. When a constraint is within the MPC horizon and the controller is focused on maintaining the velocity profile it will actuate the joints to maintain end effector velocity even if this results in deviation from the mean exemplar trajectory. This allows the proposed approach to reproduce trajectories similar in shape to the exemplars even if the exemplar mean trajectory is not reachable due to task constraints. Figure 2 shows the end effector trajectory tracking with default GP position and velocity costs and velocity cost increased 100 times.

The proposed controller successfully tracks both joint space and task space trajectories while satisfying constraints. The predictive horizon of the controller allows it to have smooth transitions and maintain the trajectory shape while

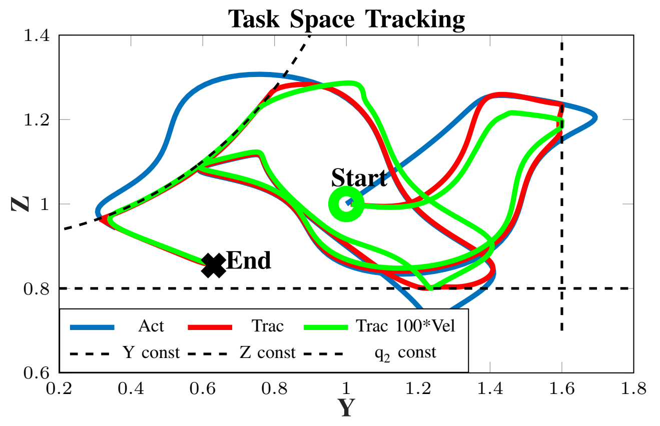

Fig. 2. Task space trajectory tracking. The proposed method accurately tracks the end effector trajectory while satisfying the task space, joint, and torque constraints. With the default GP costs, the manipulator begins at rest and applies maximum torque to quickly increase the end effector velocity and converge on the mean trajectory. It smoothly transitions at the constraint boundaries due to the default, small velocity cost. When the GP velocity cost is increased the manipulator attempts to maintain the velocity profile and thus the general shape of the mean trajectory. Thus as it approaches a task space constraint it deviates from the trajectory and executes the sharp corners before the constraint.

satisfying constraints. The accurate tracking performance shows MPC to be robust against errors in the dynamic parameters.

B. Lasa Writing Dataset

To show that the proposed approach can be used for imitation learning from human motion exemplars we utilize the Lasa hand writing dataset and track the task space trajectories with the 4dof Barret WAM manipulator [1]. Because the manipulator is direct drive, the dynamics are highly coupled and need to be handled by the controller to achieve good tracking. The Lasa dataset contains [9] 2D hand writing motions completed on a tablet, users start in initial positions close to each other and end each repetition at the same goal point. To compute the mean trajectory and the variance we assume each repetition is performed in the same amount of time. We train the x and y task space GP components on the mean trajectory samples at 50Hz, setting σ_n to the exemplar variance, figure 3 shows an exemplar of the trained GPs. The z component GP is trained with a constant zero trajectory and variance of 10^{-3} to ensure tracking on the $x-y$ plane. The MPC horizon and sampling rate are set to 45 and 100Hz respectively. To verify the controller's robustness to dynamic parameter errors, 10% parameter error is added to the inverse dynamic model used by the controller.

First we show the ability of the proposed approach to trade off between tracking and control effort based on exemplar variance. We compare the tracking performance with zero torque cost and with constant cost of 0.005. For each shape the GP velocity cost determines the trade-off between position error and shape similarity. Figure 4 shows the 2d tracking performance and the trajectory plotted against x and y GP components and table I provides the mean joint torques for different settings of the cost weights.

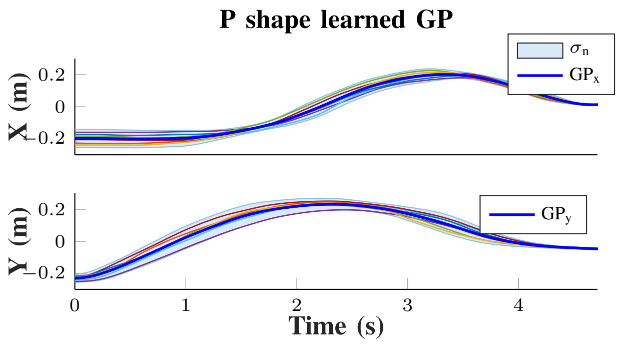

Fig. 3. Gaussian Process learned for 'P' hand written shape from the Lasa dataset and the exemplar trajectories, thin multicolored lines are the expert exemplars. The human trajectories illustrate that there is more variance at the start and middle of the trajectories, while variance decreases towards the end as the demonstrator approaches the target.

TABLE I
MEAN JOINT TORQUES.

	GShape	Angle	Trapezoid
Zero τ_{cost}	3.77	4.16	3.91
0.005 τ_{cost}	3.18	3.35	3.38
0.005 τ_{cost}, 10 Vel	3.33	3.53	3.50

Next we demonstrate that the proposed approach can satisfy task space constraints and by weighting the GP velocity cost reproduce the general shape of the trajectory. For this simulation the GP velocity cost is increased 100 times if the predicted horizon forward or backward in time is outside the task space limits. If the cost is primarily dependent on tracking error to the GP mean the optimal trajectory is right along the constraint boundary. As the manipulator reaches the boundary it needs to greatly change the end effector velocity, deviating form the GP velocity profile. By shifting the weight towards the GP velocity cost the manipulator deviates from the mean trajectory such that the velocity profile is maintained through the constrained region. This feature of the controller can be used to generate human like trajectories in workspaces with different constraints.

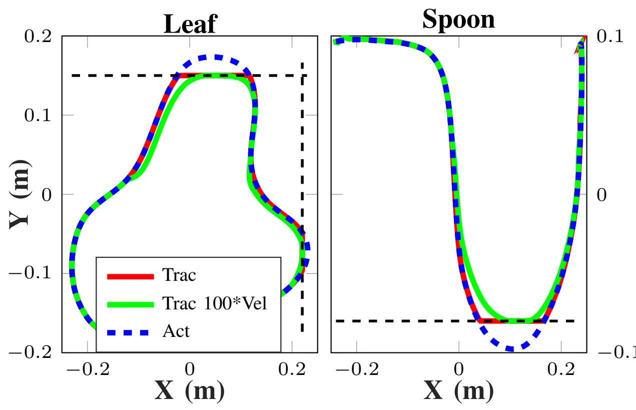

Fig. 5. Constrained task space trajectory tracking using the 4dof Barret Wam robot. Typical performance of the proposed controller (red) focuses on mean exemplar trajectory tracking while satisfying constraints. Increasing the weight of the GP velocity cost results in a trajectory (green) that further deviates from the exemplar mean but maintains the overall shape while satisfying constraints.

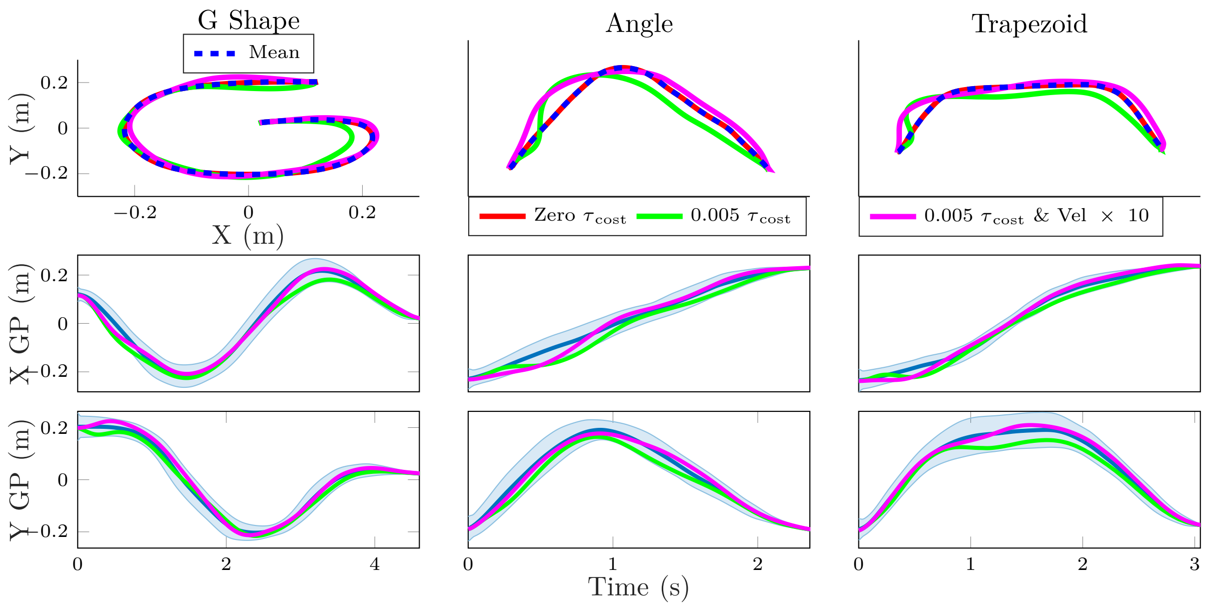

Fig. 4. Tracked hand written shapes without torque cost, with torque cost, and with velocity cost multiplier of 10. Top: tracked trajectory on the x-y plane. Bottom: Learned trajectory GPs and the tracked trajectories when torque cost is present. The tracking error is never greater than the standard deviation in the exemplars. In all cases the goal point has very low variance and thus is prioritized over torque minimization allowing the manipulator to reach it within 1mm accuracy. With GP velocity cost multiplier of 10 the shape characteristics of the trajectories and better preserved.

V. Conclusion and Future Work

We proposed a novel model predictive control method that learns a time dependent quadratic cost function based on expert exemplar data. By modeling the exemplar data using Gaussian Processes we can ensure that the controller will accurately track the desired trajectory in low variance exemplar regions and focus on minimizing additional quadratic cost functions in the high variance regions. Using the model predictive control framework allows the controller to satisfy constraints on task space, joint space, and joint torques. Furthermore, adjusting the relative weights of the learned position and velocity Gaussian Process costs allows the controller to maintain the desired general trajectory shape while satisfying constraints. The proposed controller was validated on a double pendulum system in simulation showing accurate tracking and constraint satisfaction in both task and joint space. We also show that the method is capable of reproducing trajectories based on real human exemplars using a complex 4dof manipulator. Currently the controller requires manual torque cost and velocity multiplier tuning, future work will include extracting these from exemplar trajectories. We will also extend the approach to incorporate torque derivative terms and adaptive dynamics. Finally, the approach will be validated on a real manipulator.

References

[1] Barret technology inc, barret 7-dof wam, 2015.
[2] S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.
[3] S. Calinon and A. Billard. Learning of gestures by imitation in a humanoid robot. Technical report, Cambridge University Press, 2007.
[4] D. Clever, R. M. Schemschat, M. L. Felis, and K. Mombaur. Inverse optimal control based identification of optimality criteria in whole-body human walking on level ground. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*, pages 1192–1199. IEEE, 2016.
[5] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21(5):388–403, 2013.
[6] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practicea survey. *Automatica*, 25(3):335–348, 1989.
[7] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.
[8] M. Karg and D. Kulić. Modeling movement primitives with hidden markov models for robotic and biomedical applications. *Hidden Markov Models: Methods and Protocols*, pages 199–213, 2017.
[9] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.
[10] D. Kulić, G. Venture, K. Yamane, E. Demircan, I. Mizuuchi, and K. Mombaur. Anthropomorphic movement analysis and synthesis: a survey of methods and applications. *IEEE Transactions on Robotics*, 32(4):776–795, 2016.
[11] J. Kwon and F. C. Park. Natural movement generation using hidden markov models and principal components. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1184–1194, 2008.
[12] A. McHutchon. Differentiating gaussian processes. 2013.
[13] J. Y. Nashed, F. Crevecoeur, and S. H. Scott. Rapid online selection between multiple motor plans. *Journal of Neuroscience*, 34(5):1769–1780, 2014.
[14] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann. Guiding trajectory optimization by demonstrated distributions. *IEEE Robotics and Automation Letters*, 2(2):819–826, 2017.
[15] A.-S. Puydupin-Jamin, M. Johnson, and T. Bretl. A convex approach to inverse optimal control and its application to modeling human locomotion. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 531–536. IEEE, 2012.
[16] C. E. Rasmussen. Gaussian processes for machine learning. 2006.
[17] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1431):537–547, 2003.
[18] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.
[19] E. Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907–915, 2004.