Motion Segmentation and Synthesis for Latency Mitigation in a Cloud Robotic Tele-Operation System

Nan Tian, Ajay Kummar Tanwani, Ken Goldberg and Somayeh Sojoudi

Abstract Network latency is a major problem in Cloud Robotics for human robot interaction such as tele-operation. Routing delays can be highly variable in a heterogeneous computing environment, imposing challenges to reliably tele-operate a robot with a closed-loop feedback controller.

In this work, we attempt to solve the latency problem with machine learning (ML) and supervised shared autonomy. By sharing Gaussian Mixture Model (GMM), Hidden Semi-Markov Models (HSMMs), and linear quadratic tracking (LQT) controllers between the Cloud and the robot, we build a motion recognition, segmentation, and synthesis framework for Cloud Robotic tele-operation. We further introduce a set of latency mitigation network protocols under this framework. As a proof-of-concept, we program a dynamic robot arm using this framework to perform learned hand-written letter motions. With this task, we benchmark the motion recognition errors, motion synthesis errors, and the latency mitigation performances of this Cloud Robotic tele-operation system.

1 Introduction

Cloud Robotics enables robots with limited computation power to offload compute and storage to the Cloud through the Internet. It allows robots to access computation intense machine learning (ML) models in the Cloud. One important application of Cloud Robotics is tele-operation when a human operator remotely controls a robot arm to perform everyday life tasks such as minimally invasive surgeries, secu-

Nan Tian
Department of EECS, UC Berkeley, e-mail: neubotech@berkeley.edu

Ajay Kummar Tanwani
Department of EECS, UC Berkeley, e-mail: ajay.tanwani@berkeley.edu

Ken Goldberg
Department of EECS, UC Berkeley, e-mail: goldberg@berkeley.edu

Somayeh Sojoudi
Department of EECS, UC Berkeley, e-mail: sojoudi@berkeley.edu

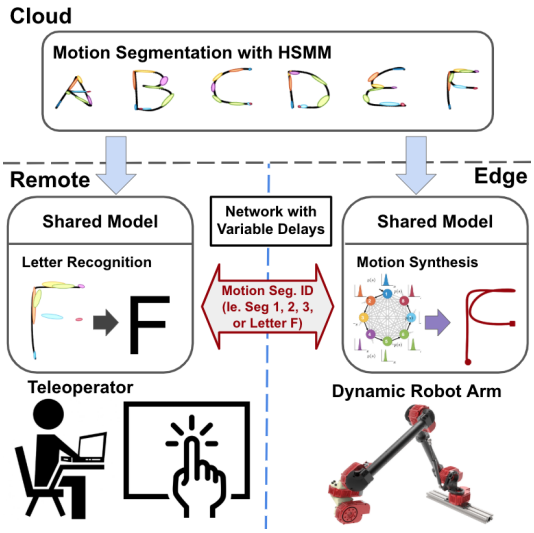

Fig. 1 Intelligent Motion Segmentation and Synthesis System for Latency Mitigating: (Top) The Cloud encodes GMM/HSMM models for handwritten letters. **(Left) The Remote** tele-operator interface recognizes letters and motion segments based on user's partial demonstration, and send compact information to **(Right) the Edge** robotic controller where segments of motion are executed in a way that reduces effects of network latency.

rity/surveillance, telepresence, warehouse management, remote patient monitoring, inspection/exploration in deep underwater or space missions [14]. Tele-operation provides a low cost solution to offload tedious works from people, so that they can reach distant and/or hazardous environment. The task entails a closed-loop controller for the robot to react to human operator input interactively in real-time.

Network latency is a major problem in Cloud Robotics and long-range tele-operation. It is caused by propagation delays and network routing delays which can be highly variable and unpredictable. This imposes challenges to build a reliable cloud-based real-time closed-loop controller to tele-operate dynamic robots, because variable delays in the feedback communication can lead to uncontrollable oscillations. These can be unsafe for human rich environments. Further, an unpredictable lag in response to a human action can cause counter-intuitive human robot interactions, which would lead to sub-optimal user experience.

One way to mitigate latency in Cloud Robotics is to hide network latency inside segments of robot motion executions. To do this, we need an ML based closed-loop controller that can recognize and predict which motion the tele-operator intends to perform, and can synthesize and execute similar motion segments on the robot. Such supervised and shared autonomy can help the robot move to intermediate targets on time while eliminating network delays.

In this paper, we demonstrate a prototype ML system that assists a remote tele-operator to interactively control a dynamic robotic arm for drawing handwritten letters. This is done in four-stages: (1) learn a dictionary of Hidden Semi-Markov Models (HSMMs) [15, 21] for each letter by segmenting the motion into K clusters; (2) share these models with both remote tele-operator interface and the robot edge controller; (3) remotely command the robot to execute these segments in response to partial human demonstration using inferencings from the learned models; (4) synthesize motion segments with linear quadratic tracker (LQT) [3, 13] at the Edge, so that the robot controller can catch up to the remote human demonstrations during execution (Fig. 1 and Fig. 3).

We further propose another network latency mitigation protocol (Fig. 6) based on our motion segment recognition and synthesis approach, and provide outlooks to future closed-loop, interactive Cloud Robotic tele-operation systems for general human motions.

This work makes below contributions:

1. Probabilistic learning-based motion segmentation using HSMM models to encode hand-written letters in the Cloud.
2. Generate synthetic motions with LQT controller at the Edge for stable, dynamic robot control.
3. Propose network latency mitigation protocols that predict and generate motions interactively based on partial demonstrations from the tele-operator.

2 Related Work

2.1 Cloud, Edge, and Fog Robotics

Cloud Robotics, introduced by James Kuffner in 2010 [8], refers to any robot or automation system that relies on either data or code from a network to support its operation [6]. It can be used to provide powerful machine learning systems for distributed robots. Network costs in the form of privacy, security latency, bandwidth, and reliability present a challenge in Cloud Robotics. Fog Robotics, a variant of Cloud Robotics, has been introduced recently to bring cloud computing resources closer to the robot to balance storage, compute and networking resources between the Cloud and the Edge [5, 16]. A closed-loop Cloud-Edge hybrid controller was built to control a dynamic balancing robot in our earlier work [17].

2.2 Latency Mitigation

Latency Mitigation is important as unpredictable network latency presents primary challenge in building a closed-loop interactive robotic controller over the network. Network controlled system (NCS) often encounters similar problems [22][20] [19], and the delays can be dealt with predictive control and a delay compensator. Previous work on intention recognition showed that intent prediction can assist tele-operator to perform robotic manipulation task under various network conditions [14]. Further, network latency can hide within robot motion execution in Cloud Robotics [18]. Motion synthesis using a generative model [15] is needed to achieve latency mitigation for interactive tele-operations.

2.3 Motion Segmentation and Synthesis

Motion Segmentation and Synthesis for robotics has been explored with dynamic motion primitives (DMP) [9] [10], recurrent neural netorks (RNNs) [2], stochastic optimal control [3], transition state clustering [7], Gaussian mixture models [4], HSMM and LQT [12, 15] for trajectories, and human skeleton movements. Learning from Demonstration (LfD) is a promising way to learn a model from examples demonstrated by a teacher [1]. In this work, we focus on the latency mitigation protocols for tele-operation with supervised autonomy using existing motion segmentation and synthesis algorithms.

2.4 Tele-operation with Shared Autonomy

Tele-operation controllers range from direct control to supervisory control with shared control in the middle. The more autonomous the tele-operation system is, the more tolerant it is against network delays [11]. We leverage generative models such as GMM, HMM, and HSMM to build assisted tele-operation system between the human and robot [14]. Our system for motion segmentation and synthesis fall into the supervisory control of the tele-operation spectrum.

3 Problem Statement

Consider a tele-operator that controls a robot arm in a remote site. The tele-operator performs a partial demonstration of a trajectory $\boldsymbol{\xi}$ comprising of datapoints $\boldsymbol{\xi}_t \in \mathbb{R}^N$ at time t,

$$\boldsymbol{\xi} = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, ..., \boldsymbol{\xi}_t ..., \boldsymbol{\xi}_T\} \qquad t \in 1, 2...t \tag{1}$$

where $\boldsymbol{\xi}_t$ is a column vector of position, velocity, and acceleration, respectively, in 2D space, so $\boldsymbol{\xi}_t = [\boldsymbol{x_t}, \dot{\boldsymbol{x}}_\mathbf{t}, \ddot{\boldsymbol{x}}_\mathbf{t}]^\top$.

We assume that the demonstration $\boldsymbol{\xi}$ comprises of the segments $\{z_i\}_{i=1}^D \in \mathbb{Z}$ that constitute the latent space of the demonstrated trajectory

$$\boldsymbol{\xi}_t \in \{z_{T_1}^1, z_{T_2}^2, ... z_{T_D}^D\} \tag{2}$$

where $z_{T_D}^D$ is the D^{th} segment index with the duration of T_D. More precisely, each motion segment is

$$\boldsymbol{\xi}_{T_D}^D = \boldsymbol{\xi}_{t_D, t_D + T_D}^D \tag{3}$$

where t_D is the starting time of the segment, and $\boldsymbol{\xi}_{t_D}^D$ and $\boldsymbol{\xi}_{t_D+T_D}^D$ are the starting and ending point of the D^{th} segment. We define starting points $\boldsymbol{\xi}_{t_D}^D$ as the way-points of trajectory.

Without loss of generality, we assume that the trajectory demonstration corresponds to a handwritten letter l denoted as ${}^l\boldsymbol{\xi}$ where $l \in \{A, B, \ldots, Z\}$. In the

first stage, the objective is to learn models of motion segments from tele-operator demonstrations for each letter. This is the encoding step. Subsequently during the decoding step, the learned segments are used for recognizing the intention of the tele-operator as writing a particular letter l from the partial demonstration sequence, and synthesize the motion for letter l on the remote robot. We denote the generated motion sequence on the robot with a hat as

$$ {}^{l}\hat{\boldsymbol{\xi}} = {}^{l}\,\hat{\boldsymbol{\xi}}^{1}_{T_1}, {}^{l}\,\hat{\boldsymbol{\xi}}^{2}_{T_2}, \ldots {}^{l}\hat{\boldsymbol{\xi}}^{D}_{T_D} \qquad l \in \{A, B, C, ..., Z\} \tag{4} $$

With the above definitions, we frame the motion segmentation and synthesis for the tele-operation task over the network (see Fig. 1):

1. **The Cloud**: learn models from data ${}^{l}\boldsymbol{\xi}$ to represent motion segments ${}^{l}\boldsymbol{\xi}^{D}_{T_D}$, and share the learned models on both the Remote and Edge controllers.
2. **The Remote**: Recognize current motion segment ID D and letter ID l from partial tele-operator demonstration $\boldsymbol{\xi}_t$, and send these high level commands to the Edge on the remote site.
3. **The Edge**: Given learned models, upon receiving D (segment ID), l (letter ID), synthesize motion segments ${}^{l}\hat{\boldsymbol{\xi}}^{D}$ or trajectory ${}^{l}\hat{\boldsymbol{\xi}}$ so that the robot can finish motion execution before the designated duration T.

4 MOTHION SEGMENTATION FOR LATENCY MITIGATION

4.1 Circle vs. Square, A Toy Example

We use a circle drawing example to illustrate that both motion segmentation and synthesis are necessary for tele-operation. (Fig 2) In the naive case, drawing a circle requires the robot's end-effector to follow a densely sampled circle trajectory. If the samples were sent through network one-by-one, unpredictable variable delays could affect the circle drawing significantly.

If we break the circle into four segments and only send out the way-points, the Edge controller would interpret the arcs as linear paths via interpolation. The robot would draw a square instead of a circle. However, if both the Remote and the Edge share the shapes of these motion segments, the Edge can fill in the trajectories between way-points to reproduce motions similar to the tele-operator's. The shape information shared can either be pre-stored raw motion segments or learned generative models that are capable of motion recognition and synthesis.

4.2 Motion Segmentation with Stationary Point Criteria

To establish a motion segmentation base-line with handwritten letter demonstrations, we first use well known minimum velocity and acceleration heuristics H [10] to automatically identify stationary points x_s.

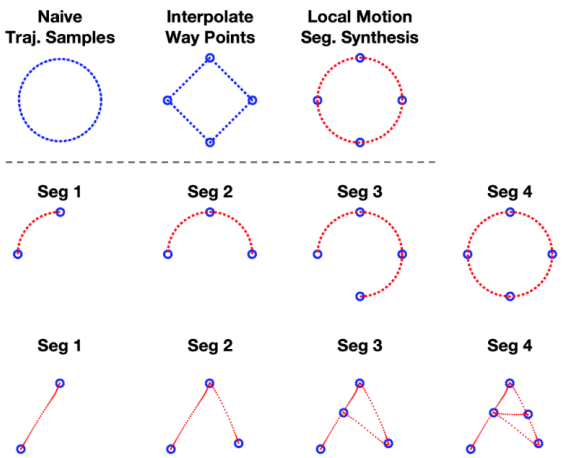

Fig. 2 Motion Segmentation. (Row I) Circle vs. Square This toy example shows the naive, undesired, and desired trajectories that can be generated for our system. **(Row II & III) Motion Segmentation with Stationary Points** Here we show that we can perform motion segmentation using stationary way-points from data, both for a circle and handwritten letter A. After segmentation, we execute these motions one-by-one to perform tele-operation.

$$x_s \in \{{}^l\boldsymbol{\xi}_t \mid H \approxeq 0\} \quad \text{where} \quad H = ||\dot{\boldsymbol{x}}_\mathbf{t}||^2 + ||\ddot{\boldsymbol{x}}_\mathbf{t}||^2 \tag{5}$$

We perform K-means to group these stationary points into clusters $i \in K$ with centroid-means of μ^i. Cluster centroid IDs are re-ordered so that they are in the sequential order of the demonstrations. Motion segments can then be defined as trajectories between adjacent clusters of stationary points.

$$\boldsymbol{\xi} = \{\boldsymbol{\xi}^1_{T_1}, \boldsymbol{\xi}^2_{T_2}, ...\boldsymbol{\xi}^K_{T_K}\} \text{ where } \mu^i \in \{\mu^1, \mu^2, ...\mu^K\} \tag{6}$$

We then share the re-ordered k-mean clusters and example trajectories to both the Remote and the Edge controllers, so that the Edge can replay pre-stored motion segments based on the closest clusters:

$$i^{\text{ID}} := \underset{i \in \{1,...,K\}}{\text{argmin}} \; ||x_s - \mu^i||^2 \tag{7}$$

Control sequence re-played with Protocol I are shown in Fig. 2 for letter “A".

4.3 Latency Mitigation Protocol I: Catch-up and Wait

Based on previous findings that robot motion executions can hide network latency [18], we propose latency mitigation protocols for tele-operation using an intelligent motion segmentation and synthesis. Fig 3 presents the simplest form of this protocol. The Remote controller recognizes which segment the tele-operator is performing, and predicts where the intermediate target, or way-point of this segment is. The Remote controller sends motion segments to the Edge robot controller to execute, with a delay that includes both network latency and recognition delay. The Edge controller speeds up the motion execution so that the robot can catch up to the

human demonstration segment-by-segment. In the end, the robot finishes the entire trajectory as if there were no delays in the network transmission.

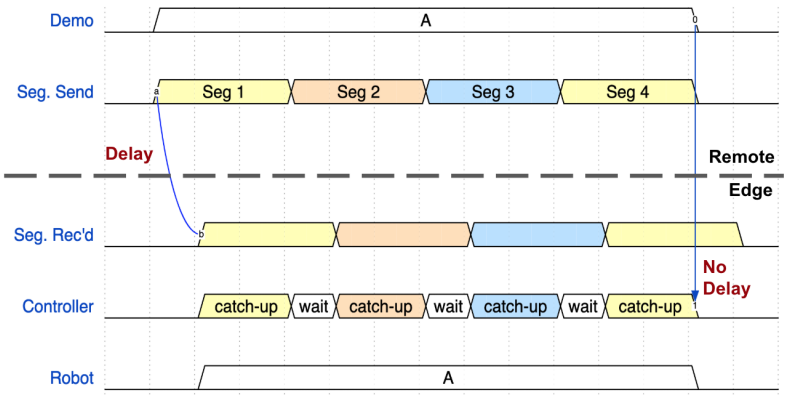

Fig. 3 Latency Mitigation Protocol I: Tele-operation commands are sent in segments, and the robot controller executes these motions in a catch-up and wait fashion to mitigate network latencies.

5 PROBABILISTIC MOTION SEGMENTATION AND SYNTHESIS

With probabilistic generative models, we need to recognize which letter the tele-operator is performing based on partial trajectory demonstrations. Therefore, we need to encode and decode both temporal and spatial information. We use GMM (spatial) and HSMM (temporal) to encode and decode hand-written letter demonstrations. We then use LQT to synthesize motions [12]. This technique generalizes well from a limited number of demonstrations than the motion segmentation re-play base-line technique described in the last section.

5.1 Spatial Encoding/Decoding

Given eight handwritten sample trajectories per letter represented by position and velocity $\boldsymbol{\xi}_t = [\boldsymbol{x_t}; \dot{\boldsymbol{x}}_\mathbf{t}]$, we train a separate GMM model to encode each letter in the alphabet.

$$P(\boldsymbol{\xi}_t \mid \theta) = \sum_{i=1}^{K} \pi_i N(\boldsymbol{\xi}_t, \mid \mu_i, \Sigma_i) \tag{8}$$

where $P(\boldsymbol{\xi}_t \mid \theta)$ is the probability density function of sample point $\boldsymbol{\xi}_t$ conditioned on parameters $\theta = \{\pi_i, \mu_i, \Sigma_i\}_{i=1}^{K}$, a set of prior π_i, mean μ_i, and covariance matrix Σ_i for each of the K mixtures. The GMM are learned using Expectation-Maximization

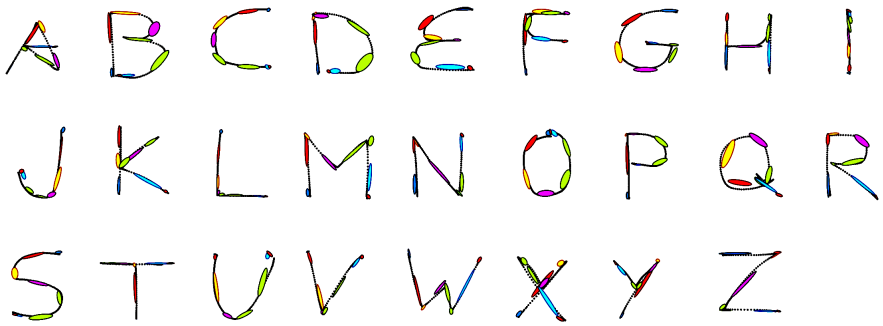

Fig. 4 Motion Segmentation of All Handwritten Letters using GMM: colored 2D Gaussian clusters overlaying on demonstration trajectory. GMM clusters are used to represent motion segments, and are used to generate synthetic motions.

(EM) algorithms. The resulting GMM mixture models for each letter is shown in Fig. 4 and Fig. 5I.

During decoding, given a sample $\boldsymbol{\xi}_i$ and the GMM for a single letter, we decide the sample belong to which mixture $z_t = i$ using maximum log likelihood

$$i^{z_t} := \underset{i \in \{1,...K\}}{\text{argmax}} \quad log\big(\pi_i N(\boldsymbol{\xi}_t \mid \mu_i, \Sigma_i)\big) \tag{9}$$

5.2 Temporal Encoding/Decoding for Letter Recognition

We use both hidden Markove model(HMM) or its generalization hidden semi-Markov model(HSMM) [13] to encode and decode temporal state sequences. GMMs obtained from above are used as latent states $z_t = i$ in HMM at time t. During encoding, the GMM-based HMM model parameterized by $\theta = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \mu_i, \Sigma_i\}_i$ learns: (a) transition probabilities $a_{i,j}$, (b) emission probabilities Π_i, (c) mean μ_i and covariance Σ_i via EM algorithm. Here, $a_{i,j}$ represents transition probabilities between the K Gaussians in GMM, and $i, j \in \{1, ...K\}$ are indexes of Gaussian mixtures.

We use the forward-backward Viterbi algorithm to decode latent states from z_t from forward variable $\alpha = P(z_t = i, \boldsymbol{\xi}_1 ... \boldsymbol{\xi}_t \mid \theta)$. The probability of a data point $\boldsymbol{\xi}_t$ to be in state i at time t given the partial observation $\{\boldsymbol{\xi}_1 ... \boldsymbol{\xi}_t\}$ can be calculated as:

$$h_{t,i} = P(z_t \mid \boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_t) = \frac{\alpha_{t,i}}{\sum_{k=1}^{K} \alpha_{t,k}} \tag{10}$$

where the forward variable α is

$$\alpha_{t,i} = \Big(\sum_{j=1}^{K} \alpha_{t-1,i} a_{j,i}\Big) N(\boldsymbol{\xi}_t \mid \mu_i, \Sigma_i) \tag{11}$$

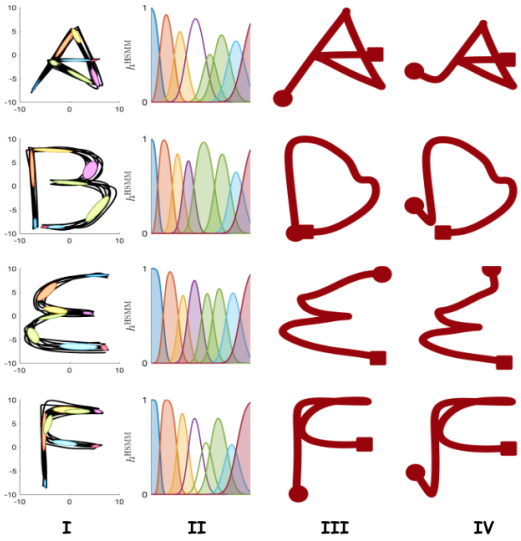

Fig. 5 Trajectory generation with HSMM: (I) HSMM mixtures overlay on data We learn a HSMM for each letter from eight trajectory samples per letter. **(II) HSMM State Probabilities** of a given trajectory inferred through forward-backward Viberbi algorithm **Generated Trajectories: (III)** from the same start positions (circle) as the original demon, and **(IV)** from different start positions (circle) to show autonomy and robustness

HSMM generalized HMM by explicitly modeling an additional state duration probability so that state transition depends on not only current state, but also on the elapsed time in the current state. In HSMM, forward variable can be calculated:

$$\alpha_{t,i} = \sum_{s=1}^{min(s^{\max},t-1)} \sum_{j=1}^{K} \alpha_{t-s,i} a_{j,i} N(s \mid \mu_i^s, \Sigma_i^s) \tag{12}$$

where s represent state duration steps in HSMM. For more details, please refer to [13] and [14].

To recognize the letter ID based on the available partial trajectory $\{\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_t\}$, we apply eq. (10) to all 26 HMMs with the parameters ${}^l\theta$ where $l \in \{A, B, ...Z\}$. The HMM model with the highest probabilities is selected as the letter that is being recognized based on partial trajectory:

$$l := \underset{l \in \{A,B,...Z\}}{\operatorname{argmax}} \; P(z_t \mid \boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_t; {}^l\theta) \tag{13}$$

5.3 Motion Synthesis based on Predicted State Sequence

We compute the desired state sequence $\boldsymbol{z_t}$ in future using the forward variable at time t using the forward variable for the most likely decoded letter,

$$\boldsymbol{z_t} = \{z_t, ..., z_{T_D}\} = \underset{i}{\operatorname{argmax}} \quad \alpha_{t,i}. \tag{14}$$

The desired state sequence is used for a step-wise reference trajectory distribution $N(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ by assigning the predicted parameters $\hat{\boldsymbol{\mu}}_t$ and $\hat{\boldsymbol{\Sigma}}_t$ at time t as the parameters

$\boldsymbol{\mu}_{z_t}$ and $\boldsymbol{\Sigma}_{z_t}$ for the predicted future states $\boldsymbol{z_t}$. Samples at time t can be generated from this reference trajectory distribution:

$$\hat{\boldsymbol{\xi}}_t \sim N(\hat{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_{z_t}, \hat{\boldsymbol{\Sigma}}_t = \boldsymbol{\Sigma}_{z_t}) \quad \text{where} \quad t \in \{t...T_D\} \tag{15}$$

The Edge robot controller uses a linear quadratic tracking (LQT) to synthesize trajectory in order to follow the demonstrated observation sequence in a smooth manner weighted by $\boldsymbol{Q}_t = \hat{\boldsymbol{\Sigma}}_t^{-1}$ while minimizing the control cost $\boldsymbol{u}$ weighted by $\boldsymbol{R}$.

$$\begin{aligned} c_t(\boldsymbol{\xi}_t, \boldsymbol{u}_t) &= \sum_{t=1}^{T} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t)^\top \boldsymbol{Q}_t (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t \\ s.t. \quad \dot{\boldsymbol{\xi}}_t &= \boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{B}\boldsymbol{u}_t \end{aligned} \tag{16}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ represent the double integrator system as a simplified analogue of robot dynamical system. For more details on LQT, refer to [13] and [15].

5.4 Latency Mitigation Protocol II: Recognize and Finish

Benchmarks of letter recognition and motion synthesis suggest that the motion recognition is poor at the initial part of the tele-operator demonstration, which can lead to large synthesis errors (see section 6 and Fig. 7). Therefore, we modify Protocol I into Protocol II to make it more practical for supervised tele-operation.

In this new protocol (Fig. 6III), during the initial period when the Remote controller is not sure about which letter the tele-operator is demonstrating, the Edge controller follows the exact trajectory of the tele-operator, while tolerating the

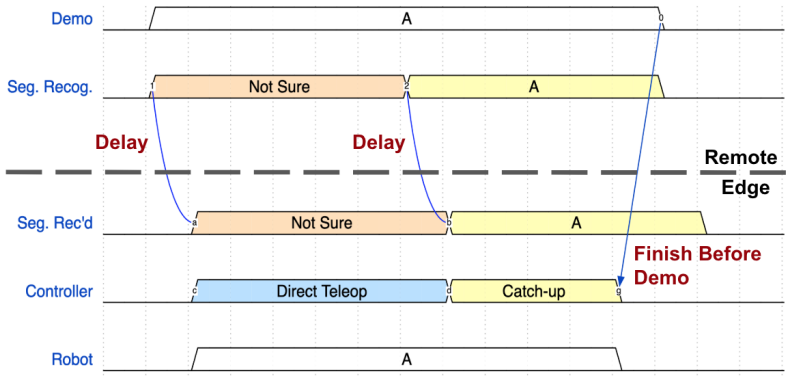

Fig. 6 Latency Mitigation Protocol II for tele-operating handwritten letters: Phase I: the Robot perform direct tele-operation when the Cloud is not sure about which letter the tele-operator is demonstrating. There is network delays associated with this phase. **Phase II:** the Robot finishes the letter motions when the Cloud recognizes the letter from partial demonstration. In this phase, the robot motion is generated locally at the robot with GMM/HSMM. The motion is also executed at an accelerated speed automatically so that it can counter network latencies. **See video for demo.**

network delay. As soon as the Remote controller recognizes and decides which letter is being drawn, the Edge controller receives the letter ID, and commits to drawing the recognized letter through motion synthesis. This way, during the latency mitigation second phase, the Edge can catch up or surpass human demonstration, so that it can reduce or eliminate network latencies.

6 EXPERIMENTS AND RESULTS

We use a handwritten letter dataset to train the GMM based HSMM model. The dataset contains eight sample trajectories per letter in the alphabet. Each sample trajectory contains 200 sample points with 2D position in the range of $[-10, 10]$ cm. By differentiation of the position data, we extract additional velocity and acceleration features that are needed to encode motion segmentation and synthesis models. We learn 26 HSMM models (one for each letter) in the encoding stage, and use the models to extract the state sequences, regenerate trajectories that have either the same or different starting points as the original trajectory in the decoding stage (Fig. 5 III and IV).

6.1 Recognition vs. Synthesis Error

There are two stages in the decoding phase: 1) recognition of letter given partial trajectory for HSMM model selection; 2) prediction of future state sequences so that a trajectory can be generated using LQT. The two decoding stages are associated with separate phases in the latency mitigation protocol–recognition and synthesis phase.

To quantitatively evaluate our system, we benchmark recognition and synthesis performance on variable length trajectories with injected noise (Fig. 7). Given a partial trajectory $\boldsymbol{\xi}_t$ ending at time t, recognition error is defined as the number of wrong letter recognition trials over total trials, whereas synthesis error is the average position L_2 error between the generated trajectory $\hat{\boldsymbol{\xi}}_{t,T}$ and the respective demonstration segment $\boldsymbol{\xi}_{t,T}$ from time t to finish time T.

$$\text{Recognition Error} = 1 - \frac{N_{Success}}{N_{Total}}, \tag{17}$$

$$\text{Synthesis Error} = \frac{||\hat{\boldsymbol{\xi}}_{t,T} - \boldsymbol{\xi}_{t,T}||_2}{T - t}. \tag{18}$$

Note that the synthesis error is normalized by the number of time samples generated, which accounts only part of the entire trajectory and has $T - t$ sample points. This way, L_2 distance of each sample contribute equally to synthesis error, so that we can compare synthesis error across partial trajectory generations with different lengths.

We conducted 10 trials per letter on partial trajectories with variable length $(0-100\%)$ injected with uniformly distributed random noise (variance $\sigma_{pos} = 2$ cm,

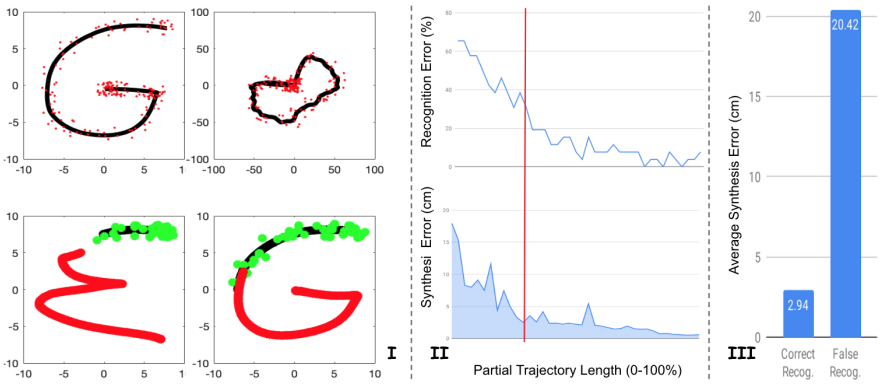

Fig. 7 **See video for demo. (I) Interactive Recognition and Synthesis Trials: (Top) Uniformly Distributed Noise Injected** to position (left, σ = 2 cm) and velocity (left, σ = 20 cm/sample) of trajectory "G" for benchmarks. **(Bottom) Synthesized Trajectory** (red trajectory) based partial noisy demonstrations (black trajectory with green noise). **(bottom left) Shorter partial demonstration** causes the model to falsely recognize the trajectory as "E". **(bottom right) Longer partial demonstration** provides correct recognition and generates intended trajectory "G". **(II) Recognition (top) and Synthesis (bottom) Errors vs. Length of Trajectory** shows that both errors reduce dramatically as demonstration progresses passing the 30% (red line) **(III) Synthesis Error** is much lower when recognition is correct, therefore, recognition error contributes to the majority of the synthesis error.

σ_{vel} = 2 cm/sample, Fig. 7I (top)). Fig. 7I (bottom) shows examples of generated trajectory based on correct and wrong recognition results. In Fig. 7II, we plot both recognition and synthesis error of all trials against the length of the partial trajectory shown to the system.

We observe that both recognition and synthesis drop dramatically around 30% trajectory demonstration length. This suggests that recognition is not reliable for short trajectories with 30% length, and it becomes more reliable as the demonstration progresses (Fig. 7II bottom).

It also suggests a strong correlation between recognition and synthesis error, as, naturally, synthesis error would grow dramatically if the letter recognition is wrong. We show that recognition error contributes to the majority of the synthesis error in Fig. 7III where the synthesis errors of all the trials with correct and wrong recognitions are compared against each other.

6.2 Latency Mitigation Effects

We want to observe how much latency the system can tolerate for the two latency protocols. Protocol I with stationary point segmentation is used as base-line. Intuitively, Protocol I can tolerate delays at most to a fraction of the length of segments. The duration of the four segments of the letter "A" are 63, 43, 81, 12 steps. Assuming that the robot can move twice as fast as the demonstrator, then the system can tolerate up to 31, 21, 40, and 6 sample points. Consequently, it can mitigate up to 0.5, 0.3, 0.7, and 0.1 seconds of delays respectively when a 60 Hz sample rate is

assumed. Any delay that is lower than the estimated duration, unpredictable it might be, is going to be eliminated.

Protocol II can tolerate more delays as motion synthesis allow it to be autonomous over the entire second phase of the protocol, after successful letter recognition. In the demo video for Protocol II, we show the interaction between the Remote demonstrator and the Edge controller when drawing letter “G", “H", “B", “P", and “K". The synthesized trajectory is red during first phase, and it changes when the system is not sure which letter it is early on in the demonstration. After recognize the letter with high confidence, the Remote controller execute the motion at 2x speed, so that the robot can finish the motion even before the tele-operator.

The second phase lasts for 153, 107, 83, 61, and 127 steps for letters “G", “H", “B", “P", and “K", which lasts 2.6, 2.8, 1.4, 1.0, and 2.1 seconds. Half of that period, or 1.3, 1.4, 0.7, 0.5, and 1.0 seconds, is used to mitigate latency. Protocol II naturally has more tolerance against unpredictable latency than Protocol I because of the autonomous motion generation phase.

To gain high confidence in letter recognition, we use a 40 sample window (0.6 seconds) during which the recognition result needs to be the same letter in order to enter the second phase. This recognition delay is introduced to trade for the price to eliminate network delays during the second phase. We believe that the benefit of eliminating not only unpredictable network delays, but also potential instabilities in a dynamical system is justified at the cost of recognition delay.

7 DISCUSSION AND FUTURE WORK

We presented an intelligent latency mitigation tele-operation system for handwritten letter drawing. Motion segmentation and synthesis are used to reduce the effects of network latency by hiding network delays inside generated synthetic motion segments. We used two different algorithms to perform motion segmentation based on either: 1) stationary points heuristics with K-means, or 2) HSMM state sequences. The HSMM method is particularly desirable for motion synthesis. We introduced and evaluated latency mitigation communication protocols based on recognition and synthesis error for writing hand-written letters.

There are trade-offs, however, in the latency mitigation system. Although we reduce the effect of unpredictable network latency on a dynamical system, we introduce recognition delays into the system that reduce the time period for robot controller to catch up to the tele-operator. Consequently, longer motion segments are more desirable, leaving more room for the Remote site to recognize the segment and for the Edge robot controller to autonomously generate the movement.

A further modification of this protocol, we denote as Protocol III, is also possible. Illustrated in Fig. 8 left, the Edge controller in Protocol Three would synthesize and execute motion in segments instead of finishing the entire motion all at once during the second phase when the correct letter is recognized. Protocol III can be more general, but is not implemented in this work.

In future work, we plan to hierarchically group Gaussian mixtures to represent mega-segments of the entire dataset, so that a hierarchical HSMM can be recognized and regenerated mega-segments based on a execution tree (Fig. 8 Right). For exam-

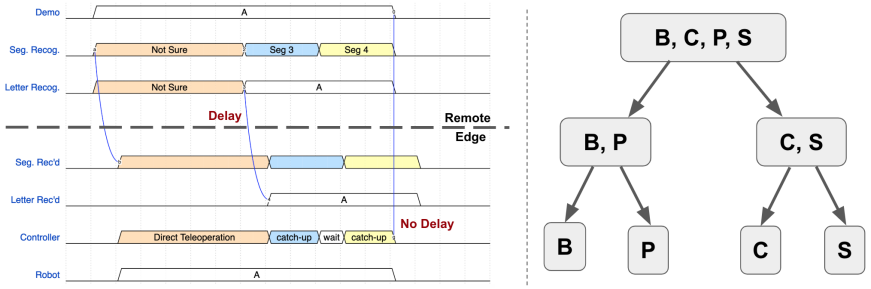

Fig. 8 **(Left) Future Protocol III** that recognize both letters and motion segments for intelligent latency mitigation **(Right) Proposing a future hierarchical GMM/HSMM** that can recognize and generate longer motion segments for the entire alphabet. These **videos of "B,P" and "C, S"** illustrate the concept. Notice that the model should decide to continue drawing "B" or to stop and finish "P" at super-node $\{B, P\}$. Ideally, a single hierarchical GMM/HSMM model should represent all 26 letters in the alphabet.

ple, in the letter set $\{B, C, P, S\}$, super-nodes $\{B, P\}$ and $\{C, S\}$ contain letters that are partially similar in the drawing process. In such a hierarchical HSMM model, when drawing the letter B, the model should traverse top-to-bottom in the tree to command the Edge controller to execute motion segment P first, the meta-segment shared by B and P. The controller would then decide whether to finish drawing letter B with an additional motion segment or not at super-node $\{B, P\}$, upon additional demonstration given by the tele-operator.

Moreover, following strong results for 2D handwritten letter drawing dataset, we plan to use our system to imitate 3D-human-skeleton based HRI interactions leveraging human-human interaction datasets.

ACKNOWLEDGMENTS

We thank Prof. Joseph Gonzalez for discussions on hybrid synchronous and asynchronous Systems. We also thank Matthew Tesch, David Rollinson, Curtis Layton, and Prof. Howie Choset from HEBI robotics for supports on the 5DoF robot arm.

Fundings from Office of Naval Research, NSF EPCN, and Cloudminds Inc. are acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Sponsors.

References

[1] Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robotics and autonomous systems 57(5):469–483

[2] Berio D, Akten M, Leymarie FF, Grierson M, Plamondon R (2017) Calligraphic stylisation learning with a physiologically plausible model of movement and recurrent neural networks. In: Proceedings of the 4th International Conference on Movement Computing, ACM, p 25

[3] Borrelli F, Bemporad A, Morari M (2011) Predictive control for linear and hybrid systems. Cambridge University Press

[4] Calinon S, D'halluin F, Sauser EL, Caldwell DG, Billard AG (2010) Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression. IEEE Robotics and Automation Magazine 17(2):44–54

[5] Gudi S, Ojha S, Johnston B, Clark J, Williams MA (2017) Fog robotics: An introduction. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

[6] Kehoe B, Patil S, Abbeel P, Goldberg K (2015) A survey of research on cloud robotics and automation. IEEE Transactions on Automation Science and Engineering 12(2):398–409

[7] Krishnan S, Garg A, Patil S, Lea C, Hager G, Abbeel P, Goldberg K (2018) Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning, Springer International Publishing, Cham, pp 91–110. DOI 10.1007/978-3-319-60916-4_6

[8] Kuffner JJ, et al. (2010) Cloud-enabled robots. In: IEEE-RAS international conference on humanoid robotics, Nashville, TN

[9] Meier F, Theodorou E, Stulp F, Schaal S (2011) Movement segmentation using a primitive library. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 3407–3412

[10] Meier F, Theodorou E, Schaal S (2012) Movement segmentation and recognition for imitation learning. In: Artificial Intelligence and Statistics, pp 761–769

[11] Song D, Tanwani AK, Goldberg K (2019) Networked-, cloud- and fog-robotics. In: Siciliano B (ed) Robotics Goes MOOC, Springer

[12] Tanwani A, Calinon S (2016) Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model. Robotics and Automation Letters, IEEE 1(1):235–242, DOI 10.1109/LRA.2016.2517825

[13] Tanwani AK (2018) Generative Models for Learning Robot Manipulation Skills from Humans. PhD thesis, Ecole Polytechnique Federale de Lausanne, Switzerland

[14] Tanwani AK, Calinon S (2017) A generative model for intention recognition and manipulation assistance in teleoperation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pp 43–50, DOI 10.1109/IROS.2017.8202136

[15] Tanwani AK, Lee J, Thananjeyan B, Laskey M, Krishnan S, Fox R, Goldberg K, Calinon S (2018) Generalizing robot imitation learning with invariant hidden semi-markov models. 1811.07489

[16] Tanwani AK, Mor N, Kubiatowicz J, Gonzalez J, Goldberg K (2019) A fog robotics approach to deep robot learning: Application to object recognition and grasp planning in surface decluttering. In: IEEE International Conference on Robotics and Automation (ICRA)

[17] Tian N, Chen J, Ma M, Zhang R, Huang B, Goldberg K, Sojoudi S (2018) A fog robotic system for dynamic visual servoing. arXiv preprint arXiv:180906716

[18] Tian N, Kuo B, Ren X, Yu M, Zhang R, Huang B, Goldberg K, Sojoudi S (2018) A cloud-based robust semaphore mirroring system for social robots. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), IEEE, pp 1351–1358

[19] Wang H, Tian Y, Christov N (2014) Event-triggered observer based control of networked visual servoing control systems. Journal of Control Engineering and Applied Informatics 16(1):22–30

[20] Wu H, Lou L, Chen CC, Hirche S, Kuhnlenz K (2013) Cloud-based networked visual servo control. IEEE Transactions on Industrial Electronics 60(2):554–566

[21] Yu SZ (2010) Hidden semi-Markov models. Artificial Intelligence 174:215–243

[22] Zhang W, Branicky MS, Phillips SM (2001) Stability of networked control systems. IEEE Control Systems 21(1):84–99