Mitigating Network Latency in Fog Robotic based Tele-Operation by Motion Segmentation and Synthesis

Nan Tian,[1,2] Ajay Kummar Tanwani[1], Ken Goldberg[1] and Somayeh Sojoudi[1,2,3]

***Abstract*— Network latency is a major problem in Cloud and Fog Robotics for human robot interaction in 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 present a set of communication protocols for mitigating network latency when remotely tele-operating a robot arm. We develop a recognition, segmentation and motion synthesis approach from tele-operator demonstrations using Hidden Semi-Markov Models (HSMMs) and linear quadratic tracking controller. We evaluate the approach by first learning a motion segmentation and synthesis model for each letter in a human hand-written letter dataset, and share the model locally on both the tele-operator and the robot site. We recognize the corresponding letter from partial demonstration of the tele-operator online and synthesize the trajectory on the robot arm based on the learned models. Performance evaluation of recognition and synthesis errors in providing assistance to tele-operator while performing letter motions in the presence of communication delays suggest the feasibility of latency mitigation for Fog Robotics.**

I. 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 where a human operator remotely controls a robot arm in performing everyday life tasks as diverse as minimally invasive surgeries, security/surveillance, telepresence, warehouse management, remote patient monitoring, inspection/exploration in deep underwater or space missions [1]. tele-operation provides a low cost solution to offload tedious work from humans and 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, which are 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.

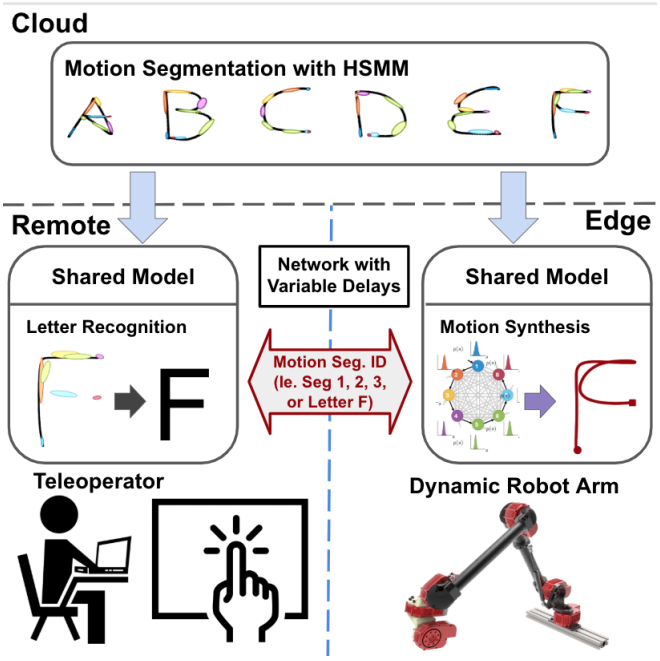

Fig. 1. **Intelligent Motion Segmentation and Synthesis System for Latency Mitigating: (Top) The Cloud** encodes GMM/HSMM models for handwritten letters. **(Left) The Remote** tele-operator interface recognizes letters and motion segments based on user's partial demonstration, and send compact information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

Fog Robotics distributes compute and storage of ML models between the Cloud and the Edge resources to meet lower latency requirements [2]. One way to mitigate latency with Fog Robotics is to hide network latency inside robot motion executions. To do that, we need an ML based closed-loop controller that can recognize and predict what motion the tele-operator intends to perform, and can synthesize and execute similar motion segments on the robot. Such supervised autonomy can help the robot move to intermediate targets on time and eliminate network delays.

In this paper, we demonstrate a prototype of a remote human that tele-operates a dynamic robot to draw hand-written letters. We follow a four-stage methodology to this end: 1) learn a dictionary of Hidden Semi-Markov Models (HSMMs) [3], [4] 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

The AUTOLab at Berkeley (automation.berkeley.edu)
[1]Department of Electrical Engineering and Computer Science (EECS), University of California, Berkeley, USA; {neubotech, ajay.tanwani, goldberg, sojoudi}@berkeley.edu
[2]Department of Mechanical Engineering, University of California, Berkeley, USA;
[3]Tsinghua-Berkeley Shenzhen Institute.

to human demonstration based on the learned models; (4) synthesize motion segments with linear quadratic tracker (LQT) [5], [6] at the Edge, so that the robot controller can catch up to the remote human demonstrations (Fig. 1 and Fig. 3). We propose two network latency mitigation protocols based on our motion segment recognition and synthesis approach, and further provide outlook to future closed-loop, interactive Fog Robotic tele-operation systems for general human motions.

A. Contributions

This paper makes three contributions:

1) Probabilistic learning-based motion segmentation using HSMM to encode hand-written letters in the Cloud.
2) Share the models locally on the tele-operator site for recognizing which model to invoke from partial demonstration, and on the remote robot site for generating corresponding motion at the Edge dynamically with a robot arm.
3) Propose network latency mitigation protocols that can anticipate and generate motions interactively based on partial demonstrations from the tele-operator interface.

II. RELATED WORK

Cloud, Edge, and Fog Robotics Cloud Robotics, introduced by James Kuffner in 2010 [7], refers to any robot or automation system that relies on either data or code from a network to support its operation [8]. 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 [2], [9]. A closed-loop Cloud-Edge hybrid controller was built to control a dynamic balancing robot in our earlier work [10].

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 [11] [12] [13], 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 [1]. Further, network latency can hide within robot motion execution in Cloud Robotics [14]. Motion synthesis using a generative model [4] is needed to achieve latency mitigation for interactive tele-operations.

Motion Segmentation and Synthesis for robotics has been explored with dynamic motion primitives (DMP) [15] [16], recurrent neural netorks (RNNs) [17], stochastic optimal control [5], transition state clustering [18], Gaussian mixture models [19], HSMM and LQT [4], [20] for trajectories, and human skeleton movements. Learning from Demonstration (LfD) is a promising way to learn a model from examples demonstrated by a teacher [21]. In this work, we focus on the latency mitigation protocols for tele-operation with supervised autonomy using existing motion segmentation and synthesis algorithms.

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 [22]. We leverage upon generative models such as HMM and HSMM to build assisted tele-operation system between the human and robot [1]. Our system for motion segmentation and synthesis fall into the supervisory control of the tele-operation spectrum.

The rest of the paper is organized as follows: we introduce the problem statement in Sec. III. We then present the motion segmentation and synthesis approach with HSMMs and LQT in Sec. IV, followed by latency mitigation protocols in Sec. V. In Sec. VI, we describe the experiments and results, which is followed by an outlook to our future work in VII.

III. 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 $\boldsymbol{\xi}_t = [\vec{\boldsymbol{x}}_t, \dot{\vec{\boldsymbol{x}}}_t, \ddot{\vec{\boldsymbol{x}}}_t]^\top$.

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

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

With the above definitions, we frame the motion segmentation and synthesis for the tele-operation task over the network:

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.

IV. MOTION SEGMENTATION AND SYNTHESIS

In this section, we first present a commonly used stationary point heuristic for motion segmentation as a baseline, and then present our probabilistic motion segmentation and synthesis approach with HSMMs and LQT.

A. Motion Segmentation with Stationary Point Heuristic

To establish a motion segmentation base-line with hand-written letter demonstrations, we first use well known minimum velocity and acceleration heuristics H [16] to automatically identify stationary points x_s,

$$x_s \in \{{}^{l}\boldsymbol{\xi}_t \mid H \approxeq 0\} \quad \text{where} \quad H = ||\dot{\vec{\boldsymbol{x}}}_t||^2 + ||\ddot{\vec{\boldsymbol{x}}}_t||^2. \quad (5)$$

We perform K-means to group these stationary points into clusters $i \in K$ with centroid-means of $\boldsymbol{\mu}^i$. We reassign cluster centroid IDs so that these IDs represent the sequential order in the demonstrations. Motion segments can then be defined as trajectories between adjacent clusters of stationary points,

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

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, upon receiving the closest stationary point cluster recognized by the Remote based on below equations

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

B. Probabilistic Segmentation with HSMMs

With probabilistic generative models, we are able to synthesize robot motion along with performing motion segmentation. We encode and decode hand-written letter demonstrations with a HSMM in a probabilistic manner, and use LQT to synthesize motions [20]. This technique generalizes well from a limited number of demonstrations than the motion segmentation re-play base-line technique described in the last section.

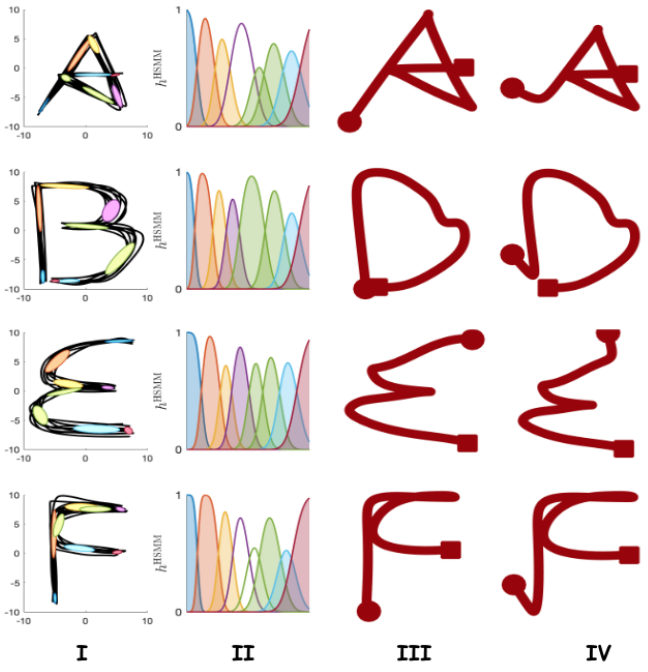

Fig. 2. **Trajectory generation with HSMM: (I) HSMM mixtures overlay on data** We learn a HSMM for each letter from eight trajectory samples per letter. **(II) HSMM State Probabilities** of a given trajectory inferred through forward-backward Viberbi algorithm **Generated Trajectories: (III)** from the same start positions (circle) as the original demon, and **(IV)** from different start positions (circle) to show autonomy and robustness

1) Temporal Encoding/Decoding for Letter Recognition: In order to select which mixture component is required for motion generation, we need to recognize which letter the tele-operator is performing based on partial demonstration sequence, and take into account encode and decode both the temporal and spatial information. We use hidden semi-Markov model (HSMM) [4] to encode and decode temporal state sequences.

A multi-variate Gaussian is used as observation distribution for a latent state $z_t \in \{1, 2, \ldots, K\}$ in HSMM at time t. The model is parameterized by $\theta = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \mu_i^D, \Sigma_i^D\}_{i=1}^{K}$, where $a_{i,j}$ refer to the transition probabilities across segments, Π_i are the initial state probabilities, mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ correspond to the observation distribution for a given latent state, and $\mathcal{N}(\mu_i^D, \Sigma_i^D)$ give the probability of staying D consecutive steps in a given segment. Model parameters are learned using an Expectation-Maximization algorithm.

We use the forward-backward Viterbi algorithm to decode the 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 $\{\boldsymbol{\xi}_1 ... \boldsymbol{\xi}_t\}$ in a Hidden Markov Model can be calculated as [23],

$$h_{t,i} = P(z_t \mid \boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_t) = \frac{\alpha_{t,i}}{\sum_{k=1}^{K} \alpha_{t,k}}. \quad (8)$$

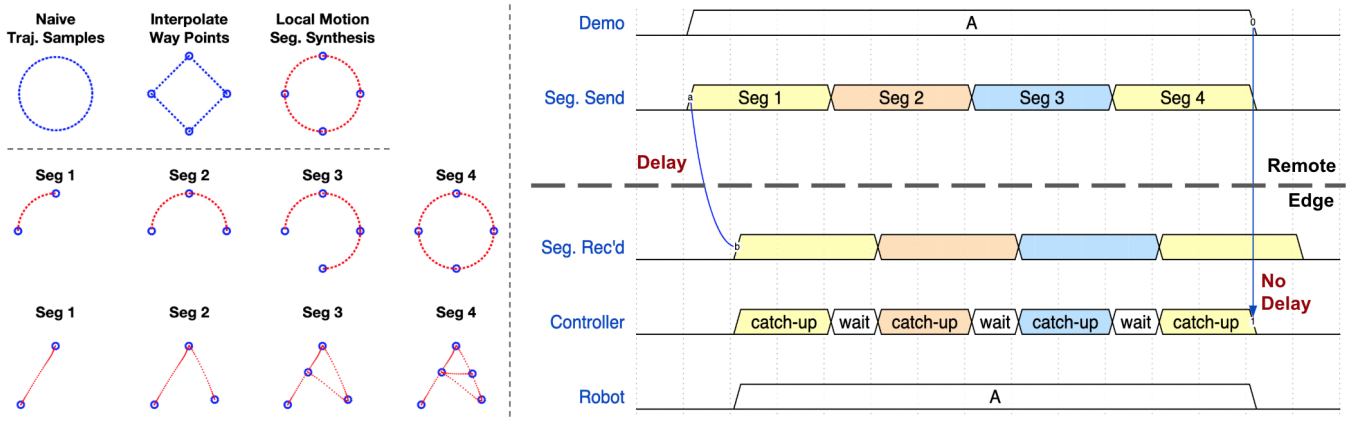

Fig. 3. **Motion Segmentation based Latency Mitigation Protocol**. **(I) Circle vs. Square** This toy example shows the naive, undesired, and desired trajectories that can be generated for our system. **(II) Stationary Point Motion Segmentation** Here we show that we can perform motion segmentation by automatically detecting and grouping stationary way-points from data. We can then execute these motion segments in order to perform tele-operation in segments. **(III) Latency Mitigation Protocol One:** This illustrate our first latency mitigation protocol where segments of motion are transmitted to the Edge. The robot controller can in turn execute them in an elevated speed to eliminate network delays.

where the forward variable α is,

$$\alpha_{t,i} = \Big(\sum_{j=1}^{K} \alpha_{t-1,i} a_{j,i}\Big) N(\boldsymbol{\xi}_t \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i). \tag{9}$$

The additional state duration probability in a HSMM models the demonstration such that the state transition depends on not only current state, but also on the elapsed duration in the current state. The modified forward variable in a HSMM is,

$$\alpha_{t,i} = \sum_{d=1}^{min(D^{\max}, t-1)} \sum_{j=1}^{K} \alpha_{t-d,i} a_{j,i} N(d \mid \mu_i^D, \Sigma_i^D), \tag{10}$$

where D represent state duration steps in HSMM. For more details, see [20] and [6].

Given eight handwritten sample trajectories per letter represented by position and velocity $\boldsymbol{\xi}_t = [\vec{x}_t; \dot{\vec{x}}_t]$, we train a separate HSMM model to encode each letter in the alphabet. The resulting models for each letter is shown in Fig. 2I and Fig. 7I.

To recognize the letter ID based on the available partial trajectory $\{\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_t\}$, we apply eq. (8) to all 26 HSMMs with the parameters ${}^l\theta$ where $l \in \{A, B, ...Z\}$. The HSMM model with the highest probability is selected as the letter that is being recognized based on partial trajectory:

$$l := \underset{l \in \{A,B,...Z\}}{\operatorname{argmax}} \; P(z_t \mid \boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_t; {}^l\theta) \tag{11}$$

2) 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,

$$\boldsymbol{z_t} = \{z_t, ..., z_{T_D}\} = \underset{i}{\operatorname{argmax}} \quad \alpha_{t,i}. \tag{12}$$

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:

$$\hat{\boldsymbol{\xi}}_t \sim N(\hat{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_{z_t}, \hat{\boldsymbol{\Sigma}}_t = \boldsymbol{\Sigma}_{z_t}) \quad \text{where} \quad t \in \{t...T_D\} \tag{13}$$

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 $\boldsymbol{R}$.

$$c_t(\boldsymbol{\xi}_t, \boldsymbol{u}_t) = \sum_{t=1}^{T} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t)^\top \boldsymbol{Q}_t (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t \tag{14}$$
$$s.t. \quad \dot{\boldsymbol{\xi}}_t = \boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{B}\boldsymbol{u}_t$$

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 [6] and [4].

V. LATENCY MITIGATION PROTOCOLS

A. Protocol One: Catch-up and Wait

Based on previous findings that robot motion executions can hide network latency [14], we propose latency mitigation protocols for tele-operation using an intelligent motion segmentation and synthesis. Fig 3III 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.

We use a circle drawing example to illustrate why both motion segmentation and synthesis are needed for tele-operation. (Fig 3I) 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 intermediate target points, the Edge controller would interpret the arcs as linear paths via interpolation, so that the robot would move in a square instead of a circle. Further, if both the Remote and the Edge share the shapes of these motion segments, the Edge can then fill in the gaps between waypoints to reproduce motions similar to the tele-operator's. The shape information can either be raw motion segments pre-stored at the Edge, or learned generative models that can help the Edge synthesize motions.

Control sequence re-played with Protocol One are shown in Fig. 3II for letter "A" and all letters in Fig. 7I.

B. Protocol Two: Autonomous Network Latency Mitigation

Benchmarks of letter recognition and motion synthesis suggest that the motion recognition at the Remote controller accuracy is low initially, and can lead to large synthesis errors (see section VIII and Fig. 5I & II). Consequently, we wait for the tele-operator to perform partial demonstration sequence such that we can recognize the corresponding letter to synthesize on the remote site and mitigate the network latency. We modify Protocol One into Protocol Two to make it more practical for supervised tele-operation.

In this new protocol (Fig. 5III), 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 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.

A further modification of this protocol, we denote as Protocol Three, is also possible. Illustrated in Fig. 6 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 Three can be more general, but is not implemented in this work.

VI. EXPERIMENTS AND RESULTS

We use a handwritten letter dataset to train the HSMM generative model. The dataset contains eight sample trajectories per letter in the alphabet. Each sample trajectory contains 200 sample points, each include 2D position in the range $[-10, 10]$ cm. By differentiation of the position data, we extract additional velocity and acceleration features that are needed to encode motion segmentation and synthesis models. This pre-processing is also followed on partial trajectories in decoding during human demonstrations. 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. 2).

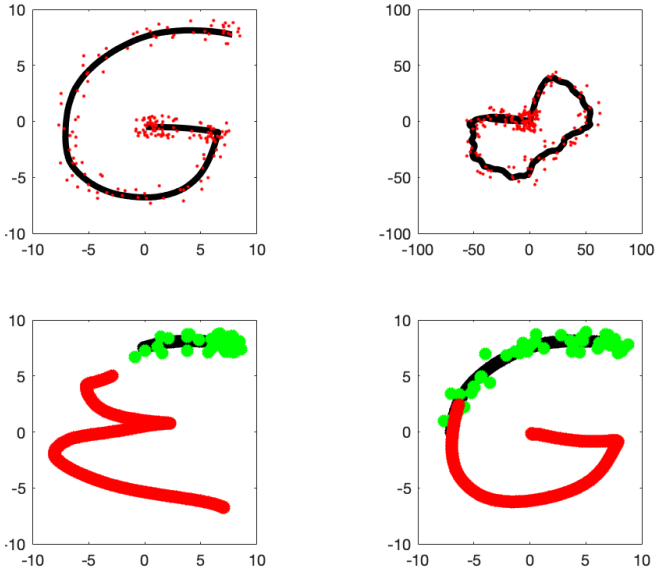

Fig. 4. **Interactive Recognition and Synthesis Trials: (Top) Uniformly Distributed Noise Injected** to position (left, $\mathbf{\Sigma} = 2$ cm) and velocity (left, $\mathbf{\Sigma} = 20$ cm/sample) of trajectory "G" for benchmark trials. **(Bottom) Synthesized Trajectory (red)** based on letter recognitions of partial noisy demonstrations with different length **(black with green noise)**. **Shorter demonstration (left)** cause more recognition failures, in this case, recognize the trajectory as "E". False recognition cause higher synthesis error. **Longer partial demonstrations (right)** help reducing both recognition and synthesis errors during tele-operation. **See demo video https://youtu.be/fjlx5kXiMhc**

A. 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. 4). 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.

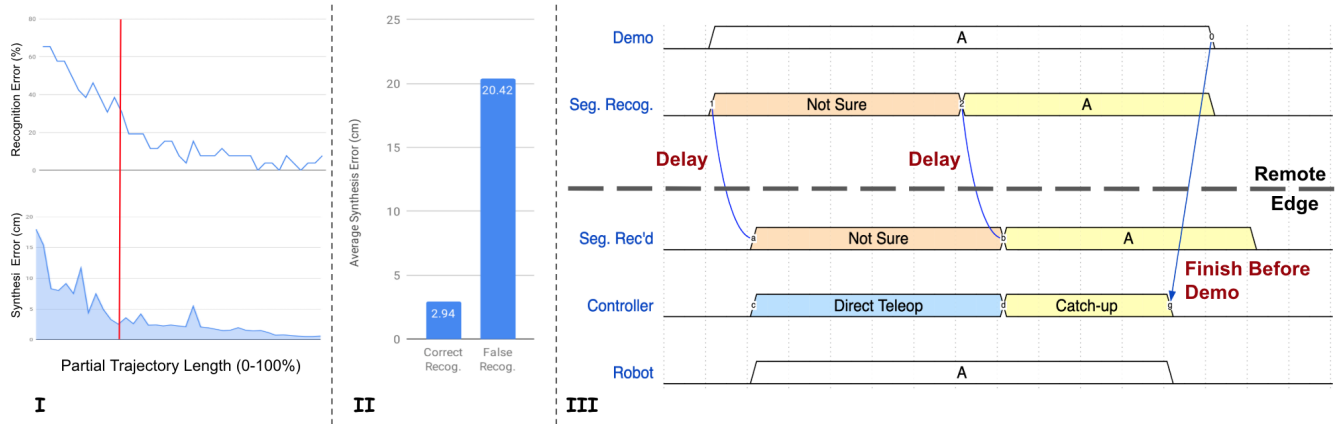

Fig. 5. **(I) Recognition (top) and Synthesis (bottom) Errors vs. Length of Trajectory** shows that both errors reduce dramatically as demonstration progresses passing the 30% (red line) **(II) Synthesis Error** is much lower when recognition is correct, suggesting that recognition error is the main contributor to synthesis error. **(III) Modified Latency Mitigation Protocol Two** for handwritten letter segmentation and regeneration, based on findings in (I) and (II). **See video for a demonstration when the Edge controller finish executing the synthetic motion before the Remote tele-operator https://youtu.be/fjlx5kXiMhc**

$$\text{Recognition Error} = 1 - \frac{N_{Success}}{N_{Total}}, \tag{15}$$

$$\text{Synthesis Error} = \frac{||\hat{\boldsymbol{\xi}}_{t,T} - \boldsymbol{\xi}_{t,T}||_2}{T - t}. \tag{16}$$

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 $\boldsymbol{\Sigma}_{pos} = 2$ cm, $\boldsymbol{\Sigma}_{vel} = 2$ cm/sample, Fig. 4 top). Fig. 4 (bottom) shows examples of generated trajectory based on correct and wrong recognition results. In Fig. 5I, 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. 4 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. 5II where the synthesis errors of all the trials with correct and wrong recognitions are compared against each other.

B. Latency Mitigation Effects

We want to observe how much latency the system can tolerate for the two latency protocols. Protocol One with stationary point segmentation is used as base-line. Intuitively, Protocol One 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 Two 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 Two (https://youtu.be/fjlx5kXiMhc), 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 Two naturally has more tolerance against unpredictable latency than Protocol One 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.

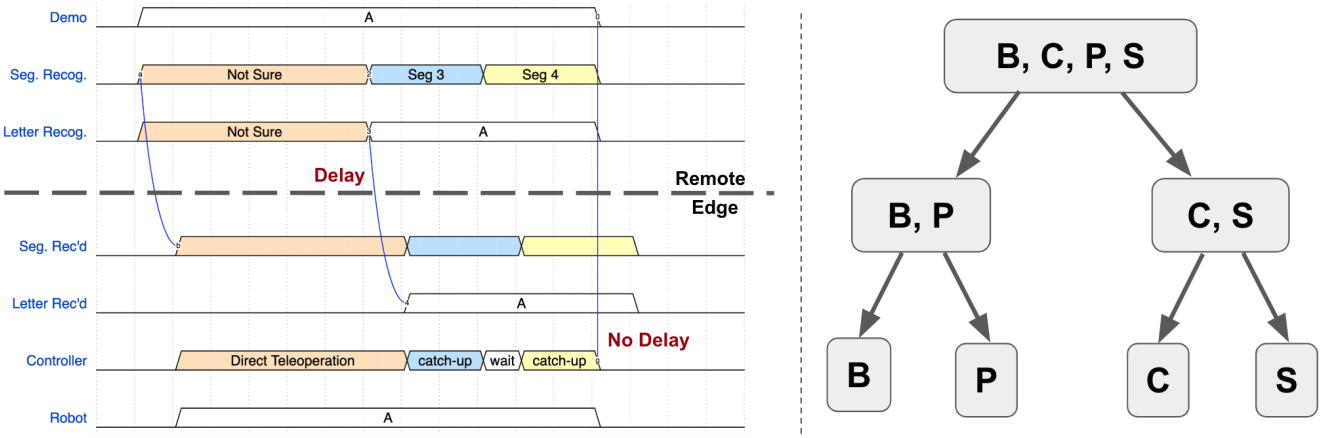

Fig. 6. **(Left) Protocol Three** that recognize both letters and motion segments **(Right) Proposing a future hierarchical GMM/HSMM** that can recognize and generate longer motion segments for the entire alphabet.

VII. CONCLUSIONS 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.

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. 6 Right). For example, in the letter set $\{B, C, P, S\}$, super-nodes $\{B, P\}$ and $\{C, S\}$ contain letters that are 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, we plan to evaluate our approach to encode whole body motions for interactive human-robot imitation, following strong results for 2D handwritten letter drawing dataset.

VIII. 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 support 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] A. K. Tanwani and S. Calinon, "A generative model for intention recognition and manipulation assistance in teleoperation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2017, pp. 43–50.

[2] A. K. Tanwani, N. Mor, J. Kubiatowicz, J. Gonzalez, and K. Goldberg, "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)*, 2019.

[3] S.-Z. Yu, "Hidden semi-Markov models," *Artificial Intelligence*, vol. 174, pp. 215–243, 2010.

[4] A. K. Tanwani, J. Lee, B. Thananjeyan, M. Laskey, S. Krishnan, R. Fox, K. Goldberg, and S. Calinon, "Generalizing robot imitation learning with invariant hidden semi-markov models," 2018.

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

[6] A. K. Tanwani, "Generative Models for Learning Robot Manipulation Skills from Humans," Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, Switzerland, 2018.

[7] J. J. Kuffner *et al.*, "Cloud-enabled robots," in *IEEE-RAS international conference on humanoid robotics, Nashville, TN*, 2010.

[8] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[9] S. Gudi, S. Ojha, B. Johnston, J. Clark, and M.-A. Williams, "Fog robotics: An introduction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[10] N. Tian, J. Chen, M. Ma, R. Zhang, B. Huang, K. Goldberg, and S. Sojoudi, "A fog robotic system for dynamic visual servoing," *arXiv preprint arXiv:1809.06716*, 2018.

[11] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems*, vol. 21, no. 1, pp. 84–99, 2001.

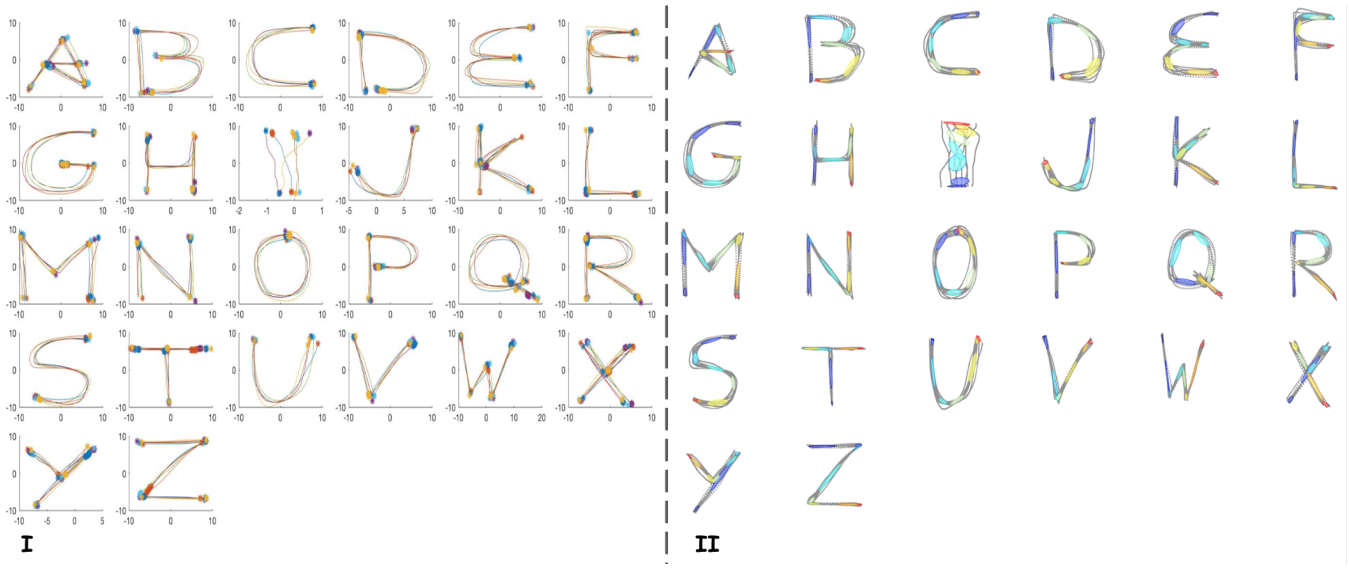

Fig. 7. **All Handwritten Letter Motion Segmentation** using **(I) Stationary Point Heuristics** and **(II) GMM/HSMM Clusters**

[12] H. Wu, L. Lou, C.-C. Chen, S. Hirche, and K. Kuhnlenz, "Cloud-based networked visual servo control," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 2, pp. 554–566, 2013.

[13] H. Wang, Y. Tian, and N. Christov, "Event-triggered observer based control of networked visual servoing control systems," *Journal of Control Engineering and Applied Informatics*, vol. 16, no. 1, pp. 22–30, 2014.

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

[15] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, "Movement segmentation using a primitive library," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3407–3412.

[16] F. Meier, E. Theodorou, and S. Schaal, "Movement segmentation and recognition for imitation learning," in *Artificial Intelligence and Statistics*, 2012, pp. 761–769.

[17] D. Berio, M. Akten, F. F. Leymarie, M. Grierson, and R. Plamondon, "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, 2017, p. 25.

[18] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, *Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning*. Cham: Springer International Publishing, 2018, pp. 91–110.

[19] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[20] A. Tanwani and S. Calinon, "Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model," *Robotics and Automation Letters, IEEE*, vol. 1, no. 1, pp. 235–242, 2016.

[21] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[22] D. Song, A. K. Tanwani, and K. Goldberg, "Networked-, cloud- and fog-robotics," in *Robotics Goes MOOC*, B. Siciliano, Ed. Springer, 2019.

[23] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77:2, pp. 257–285, 1989.