Mitigating Network Latency in Cloud-Based Teleoperation using Motion Segmentation and Synthesis

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

University of California, Berkeley; Berkeley, CA 94720, USA.
{neubotech,ajay.tanwani,goldberg,sojoudi}@berkeley.edu

Abstract. Network latency is a major problem in Cloud Robotics for human robot interactions such as teleoperation. Routing delays can be highly variable in a heterogeneous computing environment, imposing challenges to reliably teleoperate a robot with a closed-loop feedback controller. By sharing Gaussian Mixture Models (GMMs), 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 teleoperation; and we introduce a set of latency mitigation network protocols under this framework. We use this framework in experiments with a dynamic robot arm to perform learned hand-written letter motions. We then study the motion recognition errors, motion synthesis errors, and the latency mitigation performance.

Keywords: Motion Segmentation, Teleoperation, Latency Mitigation, Cloud and Fog Robotics

1 Introduction

Recent breakthroughs in artificial intelligence (AI), augmented reality (AR), and intelligent robots (IR) are changing the face of "Globalization". Richard Baldwin offers a vision of Global Robotics ("Globotics") [2] where "telecommuting" goes global, performed by "telemigrants: highly skilled, low-cost foreign workers working remotely from their home countries via the Internet. These workers remotely perform not only service and professional jobs such as clerical work and technical support, but also physical jobs such as security and manufacturing, the latter achieved by remotely teleoperating robots.

Cloud Robotics can enhance such robot teleoperation by distributing computation and memory to remote servers. Cloud-based Teleoperation allows both teleoperators and robots to access computation intense machine learning (ML) modules in the Cloud. This way, teleoperators can control robots in distant or hazardous environment with human level intelligence; and robots can learn physical skills and sub-skills from teleoperators via the Cloud. It can provide a low cost solution to offload tedious tasks from people to robots on a global scale. This

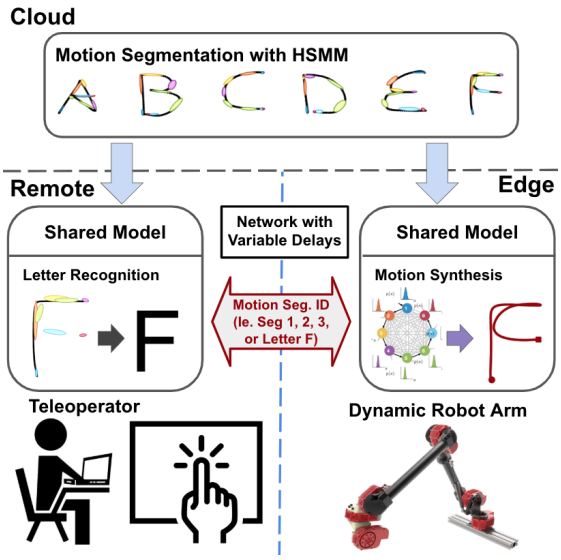

Fig. 1. Intelligent Motion Segmentation and Synthesis System for Latency Mitigating: (Top) The Cloud encodes GMM/HSMM models for handwritten letters. **(Left) The Remote** teleoperator interface recognizes letters and motion segments based on user's partial demonstration. **(Right) The Edge** robotic controller synthesize motion segments based on compact prediction information sent by the Remote. The Edge then executes the motion in a way that reduces effects of network latency.

can enable many specialized real-life tasks with high demands: minimally invasive surgeries, intelligent manufacturing, warehouse management, remote patient cares, inspection/exploration in deep underwater, space missions, and rescues in hazardous environments [15].

Many of these applications require the teleoperator to control the robot to interact with a human in a dynamic environment. Therefore, a low latency robot controller is expected in teleoperation. However, network latency prevents us from designing a global low latency controller for Cloud-based Teleoperations. Caused by imperfect network protocols and long communication distances, network delays are impossible to eliminate and can be unpredictable as well. A long and unpredictable network delay in response to teleoperator command can cause counter-intuitive human robot interactions. This would lead to sub-optimal user experience, or even cause unsafe robot operations under human rich environments.

Previous network researches have been focusing on improving network protocols to minimize network delays. In this work, we are proposing a more general way to mitigate network latency in teleoperation by leveraging machine learning (ML) and shared-autonomy.

The idea is: since many robotic tasks can be divided into motion segments (or sub-skills), we can hide network latency inside these motion segments during executions. To do this, we need an ML based controller that is shared between the teleoperator and the robot. It should (1) recognize and predict the teleoperator's intended motion, and then (2) synthesize and execute similar motion segments on the robot based these predicted motion. Such supervised shared-autonomy system can control the robot to move to intermediate targets on time with

learned sub-skills while hiding network delays inside the robot motion executions at the same time.

In this paper, we demonstrate a ML system prototype that assists a remote teleoperator 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) [23, 16] for each letter by segmenting the motion into K clusters, (2) share these models with both remote teleoperator interface and the robot edge controller, (3) remotely command the robot to execute these segments in response to partial human demonstration by inference from the learned models, (4) synthesize motion segments with linear quadratic tracker (LQT) [4, 14] at the Edge, so that the robot controller can catch up to the remote human demonstrations during execution (Fig. 1 and Fig. 3).

We propose three network latency mitigation protocols (Fig. 3, Fig. 6, and Fig. 8) based on this motion segment recognition and synthesis approach.

This work makes the following contributions:

1. A probabilistic learning-based motion segmentation algorithm using HSMMs to encode hand-written letters in the Cloud.
2. An LQT controller at the Edge for stable, dynamic robot control.
3. Three network latency mitigation protocols that predict and generate motions interactively based on partial demonstrations from the teleoperator.

2 Related Work

2.1 Cloud, Edge, and Fog Robotics

Cloud Robotics, introduced by James Kuffner in 2010 [10], 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 [6, 17]. A closed-loop Cloud-Edge hybrid controller was built to control a dynamic balancing robot in our earlier work [20].

2.2 Latency Mitigation

Latency Mitigation is important as unpredictable network latency presents a primary challenge in building an interactive robotic controller over the network. Network controlled system (NCS) often encounters similar problems [24][22] [21], and the delays can be dealt with predictive control and a delay compensator. Previous work on intention recognition showed that intent prediction can assist the teleoperator to perform robotic manipulation task under various network

conditions [15]. Furthermore, network latency can hide within robot motion execution in Cloud Robotics [19]. Motion synthesis using a generative model [16] is needed to achieve latency mitigation for interactive teleoperations.

2.3 Motion Segmentation and Synthesis

Motion Segmentation and Synthesis for robotics has been explored with dynamic motion primitives (DMPs) [12] [11], recurrent neural networks (RNNs) [3][7], stochastic optimal control [4], transition state clustering [9], Gaussian mixture models (GMMs) [5], HSMM and LQT [18, 16] 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 teleoperation with supervised autonomy using existing motion segmentation and synthesis algorithms.

2.4 Teleoperation with Shared Autonomy

teleoperation controllers range from direct control to supervisory control with shared control in the middle. The more autonomous the teleoperation system is, the more tolerant it is against network delays [13]. We leverage generative models such as GMM, HMM, and HSMM to build an assisted teleoperation system between the human and robot [15]. Our system for motion segmentation and synthesis falls into the supervisory control of the teleoperation spectrum.

3 System Design

There are three components of our system (Fig. 1): (1) the Cloud hosts machine learning (ML) modules including GMM/HSMM learning modules; (2) the Remote predicts teleoperator's intended motions using ML models shared from the Cloud; (3) the Edge, upon receiving intended motion ID from the Remote, generates trajectories using the shared models from the Cloud to control a robot.

We use a circle drawing example to illustrate that both motion segmentation and synthesis are necessary for our teleoperation system (Fig 2 top). Typically, 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. Instead, if we break the circle into four segments and send out only 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.

On the other hand, 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 teleoperator's. These shared shapes are either in the form of pre-stored raw motion segments in section 5 or in the form of learned generative models that are capable of motion recognition and synthesis in section 6.

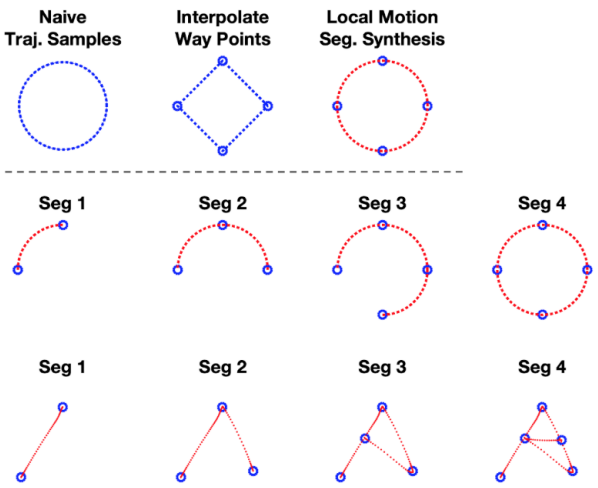

Fig. 2. Motion Segmentation. (Row I) Circle vs. Square This toy example shows the naive, undesired, and desired trajectories that can be generated for our system. **(Row II & III) Motion Segmentation with Stationary Points** Here we show that we can perform motion segmentation using stationary way-points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

4 Problem Statement

Consider a teleoperator that controls a robot arm in a remote site. The teleoperator 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

$$\boldsymbol{\xi}_t \in \{z^1_{T_1}, z^2_{T_2}, ...z^D_{T_D}\} \tag{2}$$

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

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

where t_D is the starting time of the segment, and $\boldsymbol{\xi}^D_{t_D}$ and $\boldsymbol{\xi}^D_{t_D+T_D}$ are the starting and ending point of the D^{th} segment. We define starting points $\boldsymbol{\xi}^D_{t_D}$ 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 teleoperator 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 teleoperator as writing a particular letter l from the partial demonstration

sequence and synthesizing the motion for letter l on the remote robot. We denote the generated motion sequence on the robot with a hat as

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

With the above definitions, we frame the motion segmentation and synthesis for the teleoperation task over the network into these three parts (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 teleoperator 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.

5 MOTION SEGMENTATION FOR LATENCY MITIGATION

5.1 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 [11] to automatically identify stationary points x_s.

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

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 replayed with Protocol I are shown in Fig. 2 for letter “A”.

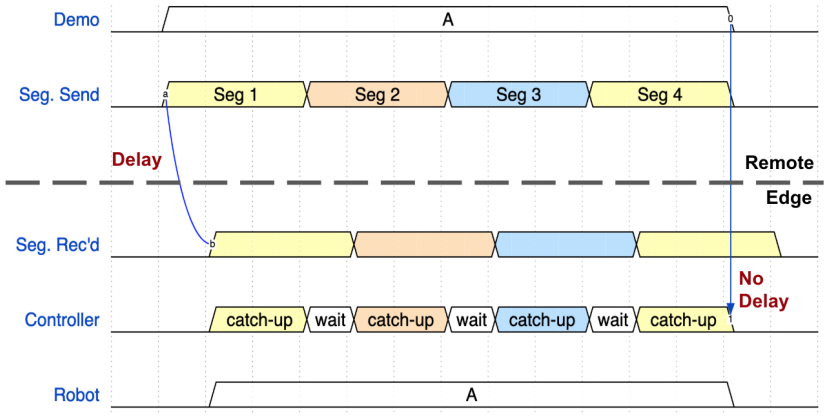

Fig. 3. Latency Mitigation Protocol I: teleoperation commands are sent in segments, and the robot controller executes these motions in a catch-up and wait fashion to mitigate network latency. **See video demo of Protocol I using a dynamic robot arm**

5.2 Latency Mitigation Protocol I: Catch-up and Wait

Based on previous findings that robot motion executions can hide network latency [19], we propose latency mitigation protocols for teleoperation using an intelligent motion segmentation and synthesis. Fig 3 demonstrates the simplest form of this protocol.

The Remote controller first recognizes which segment the teleoperator is performing. It then predicts where the intermediate target, or way-point, of this segment is. It finally 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.

6 PROBABILISTIC MOTION SEGMENTATION AND SYNTHESIS

With probabilistic generative models, we need to recognize which letter the teleoperator 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 [18]. This technique generalizes well from a limited number of demonstrations than the motion segmentation re-play base-line technique described in the last section.

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

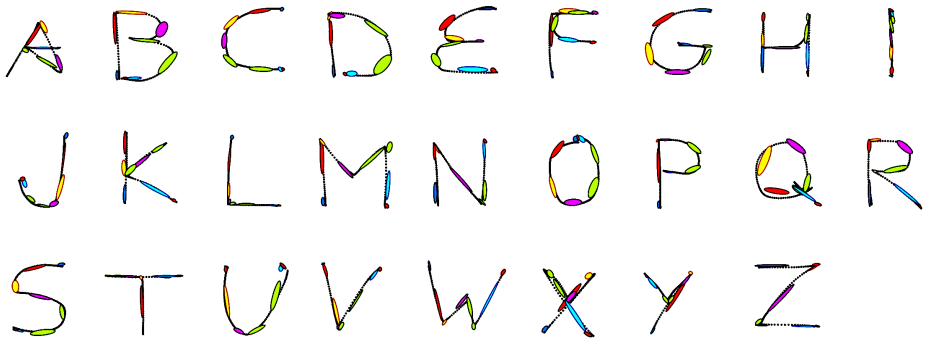

Fig. 4. Motion Segmentation of All Handwritten Letters using GMM for spatial encoding/decoding: colored 2D Gaussian clusters overlaying on demonstration trajectory. GMM clusters are used to represent motion segments. They are also used to generate synthetic motions.

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

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

6.2 Temporal Encoding/Decoding for Letter Recognition

We use both hidden Markov model(HMM) or its generalization hidden semi-Markov model(HSMM) [14] to encode and decode temporal state sequences. GMMs obtained from above are used as latent states $z_t = i$ in HMM at time t. The GMM-based HMM model is parameterized by $\theta = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \mu_i, \Sigma_i\}_i$. During encoding, it 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

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

HSMM generalizes to HMM by explicitly modeling an additional state duration probability, so that the state transition depends not only on 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 represents state duration steps in HSMM. For more details, please refer to [14] and [15].

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:

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

6.3 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:

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

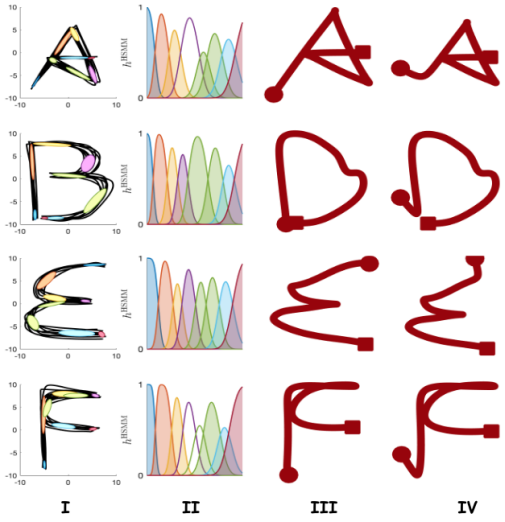

Fig. 5. Trajectory generation with HSMM:
(I) HSMM mixtures overlay on data We learn a HSMM for each letter from eight trajectory samples per letter.
(II) HSMM State Probabilities of a given trajectory inferred through forward-backward Viberbi algorithm
Generated Trajectories: (III) from the same start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

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 $\boldsymbol{R}$.

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

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 [14] and [16].

6.4 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 teleoperator 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 teleoperation.

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

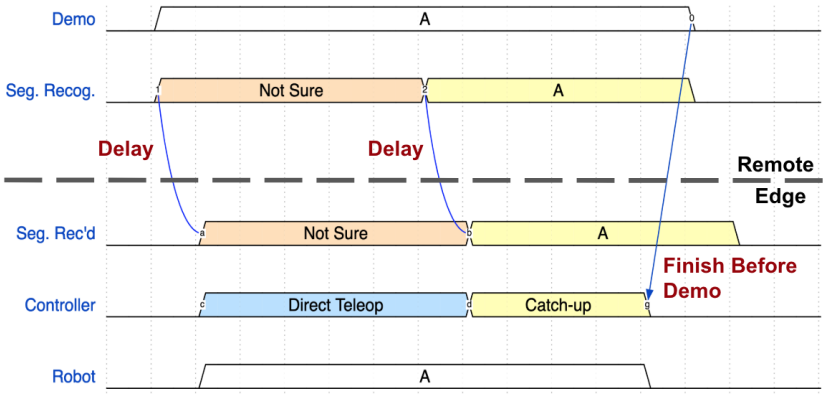

Fig. 6. Latency Mitigation Protocol II for teleoperating handwritten letters: Phase I: the Robot performs direct teleoperation when the Cloud is not sure about which letter the teleoperator is demonstrating. There are network delays associated with this phase. **Phase II:** the robot finishes the letter 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 latency. **See video demo of Protocol II: in simulation and teleoperating a dynamic robot arm**.

7 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. We extract additional velocity and acceleration features from position data via differentiation. Together with position, they are used to encode motion segmentation and synthesis models. We learn 26 HSMM models (one for each letter) in the encoding/training stage. We then use the models to decode the state sequences so that we can regenerate trajectories for the Edge robot controller. To show the advantage of regenerating a trajectory over replaying pre-recorded trajectory, we put the initial state as either the same or different starting points as the original trajectory. (Fig. 5 III and IV).

7.1 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. Each is associated with recognition and synthesis phase in the latency mitigation protocol.

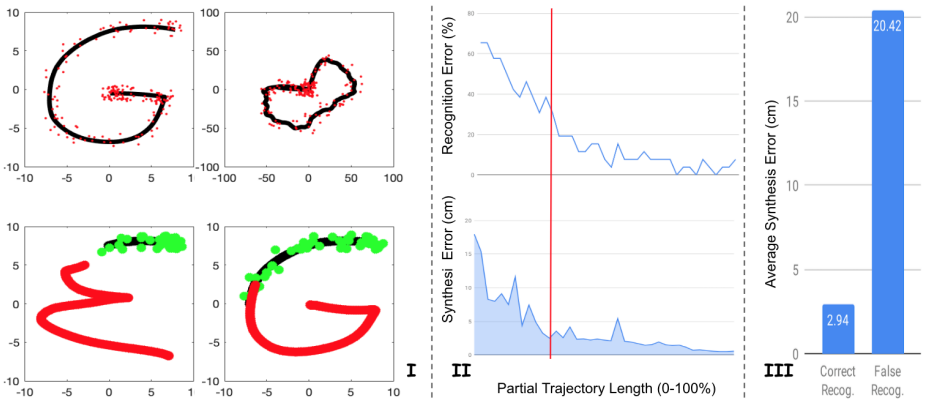

Fig. 7. (I) Interactive Recognition and Synthesis Trials: (Top) Uniformly Distributed Noise Injected to position (left, $\boldsymbol{\sigma} = 2$ cm) and velocity (left, $\boldsymbol{\sigma} = 20$ cm/sample) of trajectory "G" for benchmarks. **(Bottom) Synthesized Trajectory** (red trajectory) based 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. **See video demo in simulation.**

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

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

Note that the synthesis error, accounts for only second part of the trajectory, is normalized by the number of generated samples. This way, L_2 distance of each sample contributes equally to synthesis error, so that we can compare synthesis error across partial trajectories generated 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. 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 errors 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).

This 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 recognition are compared against each other.

7.2 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 allows it to be autonomous over the entire second phase of the protocol, after successful letter recognition. In the video demos of Protocol II in simulation:, we show the interaction between the Remote demonstrator and the Edge controller when drawing letters "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 dynamic robot arm can finish the motion even before the teleoperator, as can be seen from video: teleoperating a dynamic robot arm with Protocol II.

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.

8 DISCUSSION AND FUTURE WORK

We present an intelligent latency mitigation teleoperation 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

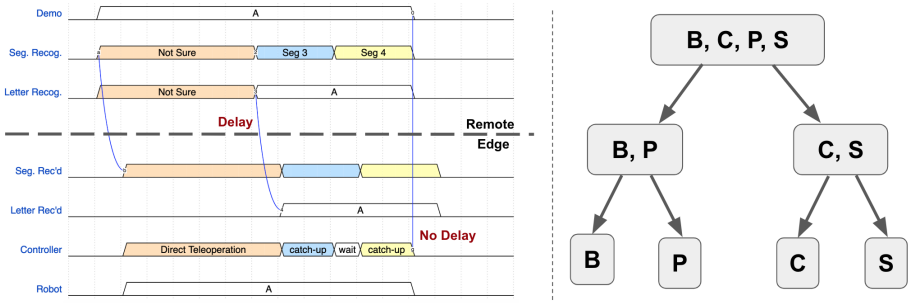

Fig. 8. (Left) Protocol III that recognizes both letters and motion segments for intelligent latency mitigation **(Right) future hierarchical GMMs/HSMMs** that can recognize and generate longer motion segments for the entire alphabet. These videos of 'B,P' and **'C, S'** illustrate the concept. Notice 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.

motion segments. We use 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 more desirable for motion synthesis. We introduce and evaluate latency mitigation communication protocols based on recognition and synthesis errors for drawing hand-written letters.

There are trade-offs 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 teleoperator. Consequently, longer motion segments are more desirable, leaving more room for the Remote to recognize the segment and for the Edge controller to autonomously generate the movement.

We further generalize Protocol II into a concept Protocol III (without actual implementation). As illustrated in Fig. 8 left, the Edge controller in Protocol III 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.

In future work, we plan to hierarchically group Gaussian mixtures and HSMMs to represent mega-segments of the entire dataset, so that they can be recognized and regenerated mega-segments based on a execution tree (Fig. 8 Right). For example, 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. Under this setting, 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 should 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 teleoperator.

Moreover, we can use the existing motion segmentation and generation system to generate synthetic trajectory samples to train a hierarchical variational auto-encoder (VAE)[7] so that a single model can (1) handle position invariants

better to improve recognition and (2) encode and decode motion datasets with more sophisticate and general motion concepts. Lastly, 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 thank William Wong, Shivin Devgon, Ryan Hoque, and Andy Yan for proof-reading the manuscript. We also thank Matthew Tesch, David Rollinson, Curtis Layton, and Prof. Howie Choset from HEBI robotics for supports on the robot arm.

Fundings from Office of Naval Research, NSF EPCN, and NSF ECDI Secure Fog Robotics Project 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] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[2] Richard Baldwin. *Globotics Upheaval: globalisation, robotics and the future of work*. Oxford University Press, 2019.

[3] Daniel Berio et al. "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.

[4] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2011.

[5] S. Calinon et al. "Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression". In: *IEEE Robotics and Automation Magazine* 17.2 (2010), pp. 44–54.

[6] SLKC Gudi et al. "Fog robotics: An introduction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017.

[7] David Ha and Douglas Eck. "A neural representation of sketch drawings". In: *arXiv preprint arXiv:1704.03477* (2017).

[8] Ben Kehoe et al. "A survey of research on cloud robotics and automation". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (2015), pp. 398–409.

[9] Sanjay Krishnan et al. "Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning". In: *Robotics Research: Volume 2*. Ed. by Antonio Bicchi and Wolfram Burgard. Cham: Springer International Publishing, 2018, pp. 91–110. DOI: 10.1007/978-3-319-60916-4_6.

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

[11] Franziska Meier, Evangelos Theodorou, and Stefan Schaal. "Movement segmentation and recognition for imitation learning". In: *Artificial Intelligence and Statistics*. 2012, pp. 761–769.

[12] Franziska Meier et al. "Movement segmentation using a primitive library". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 3407–3412.

[13] D. Song, A. K. Tanwani, and K. Goldberg. "Networked-, Cloud- and Fog-Robotics". In: *Robotics Goes MOOC*. Ed. by B. Siciliano. Springer, 2019.

[14] A. K. Tanwani. "Generative Models for Learning Robot Manipulation Skills from Humans". PhD thesis. Ecole Polytechnique Federale de Lausanne, Switzerland, 2018.

[15] 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. DOI: 10.1109/IROS.2017.8202136.

[16] A. K. Tanwani et al. *Generalizing Robot Imitation Learning with Invariant Hidden Semi-Markov Models*. 2018. arXiv: 1811.07489 [cs.RO].

[17] Ajay Kumar Tanwani et al. "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.

[18] A.K. Tanwani and S. Calinon. "Learning Robot Manipulation Tasks With Task-Parameterized Semitied Hidden Semi-Markov Model". In: *Robotics and Automation Letters, IEEE* 1.1 (2016), pp. 235–242. DOI: 10.1109/LRA.2016.2517825.

[19] Nan Tian et al. "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.

[20] Nan Tian et al. "A Fog Robotic System for Dynamic Visual Servoing". In: *arXiv preprint arXiv:1809.06716* (2018).

[21] HP Wang, Y Tian, and N Christov. "Event-triggered observer based control of networked visual servoing control systems". In: *Journal of Control Engineering and Applied Informatics* 16.1 (2014), pp. 22–30.

[22] Haiyan Wu et al. "Cloud-based networked visual servo control". In: *IEEE Transactions on Industrial Electronics* 60.2 (2013), pp. 554–566.

[23] S.-Z. Yu. "Hidden semi-Markov models". In: *Artificial Intelligence* 174 (2010), pp. 215–243.

[24] Wei Zhang, Michael S Branicky, and Stephen M Phillips. "Stability of networked control systems". In: *IEEE Control Systems* 21.1 (2001), pp. 84–99.