A Cloud-Based Robust Semaphore Mirroring System for Social Robots

Nan Tian*,[1,2] Benjamin Kuo*,[2,3], Xinhe Ren,[1], Michael Yu [2,4], Robert Zhang,[2], Bill Huang, [2]
Ken Goldberg[1] and Somayeh Sojoudi[1,4,5]

***Abstract*— We present a cloud-based human-robot interaction system that automatically controls a humanoid robot to mirror a human demonstrator performing flag semaphores. We use a cloud-based framework called Human Augmented Robotic Intelligence (HARI) to perform gesture recognition of the human demonstrator and gesture control of a local humanoid robot, named Pepper. To ensure that the system is real-time, we design a system to maximize cloud computation contribution to the deep-neural-network-based gesture recognition system, OpenPose, and to minimize communication costs between the cloud and the robot. A hybrid control system is used to hide latency caused by either routing or physical distances. We conducted real-time semaphore mirroring experiments in which both the robots and the demonstrator were located in Tokyo, Japan, whereas the cloud server was deployed in the United States. The total latency was 400ms for the video streaming to the cloud and 108ms for the robot commanding from the cloud. Further, we measured the reliability of our gesture-based semaphore recognition system with two human subjects, and were able to achieve 90% and 76.7% recognition accuracy, respectively, for the two subjects with open-loop when the subjects were not allowed to see the recognition results. We could achieve 100% recognition accuracy when both subjects were allowed to adapt to the recognition system under a closed-loop setting. Lastly, we showed that we can support two humanoid robots with a single server at the same time. With this real-time cloud-based HRI system, we illustrate that we can deploy gesture-based human-robot globally and at scale.**

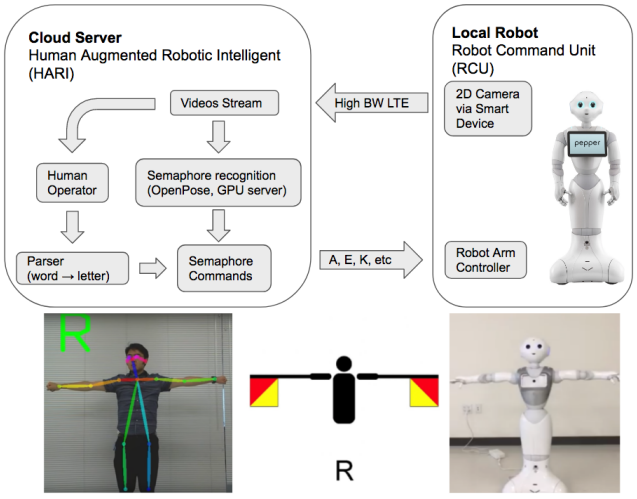

Fig. 1. **Block diagram: (Top)** Architecture diagram of the cloud-based semaphore mirroring system built on HARI, which supports AI assisted teleoperation. We deploy semaphore recognition system in the cloud, and used a discrete-continuous hybrid control system to control Pepper, a humanoid robot, to imitate semaphore gestures. **(Bottom)** An example of our system when the demonstrator performs semaphore "R", and the robot imitate him

I. INTRODUCTION

Humanoid social robots are available commercially, including Softbank's Pepper [1] and Nao [2], AvatarMind's iPal, and iCub from RobotCub [3]. These robots have similar appearances as humans, and have potentials in professional services, such as retail, hospitality, and educations. Pepper, a humanoid social robot designed by Soft-bank [1], is well liked because it has a human voice, Astro boy liked body design, and generous full body movements. It is also well equipped with cameras, LIDAR, ultrasound sensors, and a chest input pad, integrated either through Pepper's android SDK or ROS based system.

In this work, we focus on using Pepper to mirror a person performing semaphore, a hand-held flag language. It is a system conveying information at a distance by visual signals with hand-held flags. Alphabetical letters are encoded by the positions of the flags with respect to the body. It is still used by lifeguards around the world.

To perform semaphore recognition, we use 2D video stream that is common in low-cost humanoid robots. However, it is impossible to deploy a real-time gesture-based recognition system locally on humanoid robots due to limited computation power. Deep learning based gesture recognition systems such as OpenPose [4] can only achieve real-time performance on state-of-the-art deep learning GPU servers. At the same time, compared to Microsoft Kinect based gesture recognitions [5], OpenPose is preferable as it can work outdoors, can track more people, and is camera agnostic. To use OpenPose for robot gesture mirroring in real-time, one option is to stream videos from local robots to the cloud to perform OpenPose gesture recognition and send control signals back to local robots.

To illustrate this concept, we use CloudMinds Inc. HARI, Human Augmented Robot Intelligence, to support large-scale cloud service robot deployment. HARI is a hybrid system that combines the power of artificial intelligence (AI) and human intelligence to control robots from the cloud.

Mitigating network latency is essential to make cloud-

The AUTOLab at Berkeley (automation.berkeley.edu)
[1]Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA; {neubotech, jim.x.ren, goldberg, sojoudi}@berkeley.edu
[2]Cloudminds Technology Inc.; {robert, bill}@cloudminds.com
[3]Carnegie Mellon University, The Robotics Institute;
[4]Department of Mechanical Engineering, University of California, Berkeley, USA;
[5]Tsinghua-Berkeley Shenzhen Institute;
*Both authors contributed equally to this work

based motion control robust, because network latency is unavoidable at times due to routing and physical distance even with a reliable network. We pre-generate semaphore motion trajectories and store them in the local robot command unit (RCU). We execute these trajectories based on commands sent from the cloud. This way, the cloud AI only needs to use alphabetical letters to control robots to perform semaphore, which hides the network latency while the robot moves from one semaphore to another.

To test network speeds, we conducted cross-Pacific experiments to control a Pepper robot in Japan from a HARI server located in the United States while the human subject was standing in front of the robot in Japan. We further showed that one HARI server could control two robots located in two different locations simultaneously, which demonstrates scalability.

II. RELATED WORK

An anthropomorphic robot, or humanoid robot, refers to a robot whose shape resembles the human body. Many famous robots are humanoids, including Atlas from Boston Dynamics [6], Honda research's Asimo [7], and the first space humanoid robot–NASA's Robonaut 2 [8], just to name a few.

Socially interactive robots [9], though not necessarily anthropomorphic, often take humanoid forms, since many researchers believe humans must be able to interact with robots naturally, or as human like as possible [10], [11]. Others find them easier to control by emulating natural human robot interactions [9]. Therefore, many works have been dedicated to imitating human gestures [12]–[17], human demonstrations [14], [18] and human social interactions [19], [20] using humanoid robots. There are also successful works to learn human behavior policy to a virtual humanoids from in simulation [21]

Imitating full body human gesture requires a 3D positioning camera like Microsoft Kinect and a gesture recognition system such as Kinect SDK [5] or deep neural network based OpenPose [4]. The imitating controller maps position and orientation estimation of human joints into a series of robot joint angles–trajectory. These generated gesture trojectories are then used to control humanoids directly [5], indirectly after editing [22], or with a model learned from machine learning [12], [14]. More recent work also used videos from multiple 2D camera views, to train a control policy to imitate human motions directly from 2D video stream rather than from 3D video stream [23]

In Goodrich's HRI survey [24], he separates HRI into two types: remote interaction and proximate interaction. In either case, perception feedback–visual, audio, verbal, or even tactile–is important to control and program robot for interaction. In this work, we focus on large-scale cloud-based social human robot interaction using a humanoid robot, named Pepper [1]. We enabled both remote interaction–teleoperation–and proximate interaction by deploying both visual perception–gesture recognition, and high-level robot command–semaphore–in the cloud. A similar cloud robotic

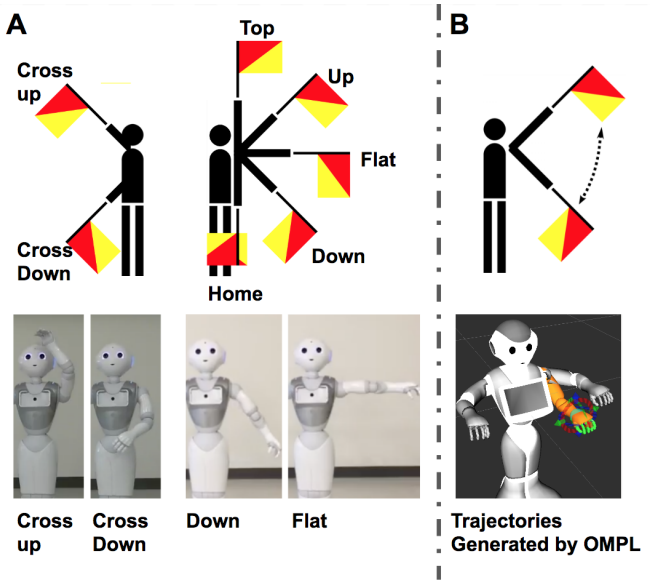

Fig. 2. **Programming Single Side Semaphore: (A) Semaphore positions (Top)** Seven different positions–cross up, cross down, top, up, flat, down, home–demonstrated with cartons. **(Bottom)** Selected examples of single side semaphore positions implemented on Pepper. **(B) Semaphore Motion** Motion trajectories in between different semaphore positons are generated first using MOVEIT! and OMPL in ROS, and then stored on local RCU.

system was built by Kehoe, et. al. before [25] when an object recognition system was deployed in the cloud to support the PR2 robot grasping system.

James Kuffner coined the term "Cloud Robotics" to describe the increasing number of robotics or automation systems that rely on remote data or code for effective operation [26]. A wide variety of models for connecting robots to the cloud have been developed [27]. Some studies have used an SaaS model that moves a robot motion controller entirely into the cloud and used it for manipulation planning [28], while others, such as RoboEarth's Rapyuta system [29], follow a Platform as a Service (PaaS) such that others would easily program on the platform. A third kind of cloud robotic system, aiming for a more energy efficient robot system, distributes the computation of robot motion plannings to both robot's embedded computer and a cloud-based compute service [30]. Some view such model as combination of cloud and edge computing systems for robotics [31], [32]. HARI is a combination of the three and aims at providing both AI and human intelligence control for large deployment of service type robots.

III. SYSTEM DESIGN

We describe three basic components of our cloud-based semaphore mirroring system: humanoid robot, robot command unit, and cloud intelligent HARI; and explain how cloud-based semaphore system was implemented, including semaphore motion generation in ROS, trajectory execution at RCU, command line interface on HARI, semaphore recognition, and semaphore mirroring with hybrid control.

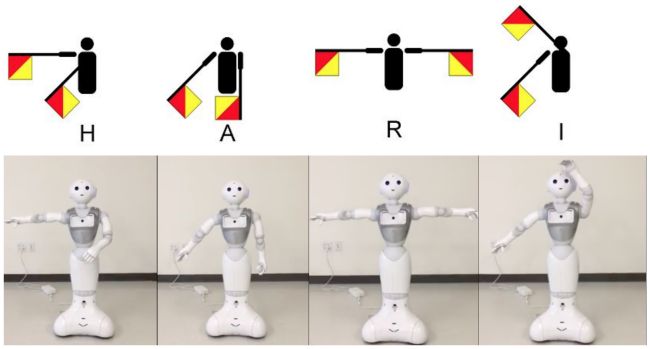

Fig. 3. **Selected Example of Semaphore Performed by both Arms of Pepper** Combination of the seven positions of each arm can be used to generate the full alphabet with 26 letters. The "HARI" sequence was teleoperated via command line interface on HARI, (video: `https://youtu.be/XOeiDrG4sAQ`)

A. Humanoid Robot Pepper

We use Pepper, a humanoid social robot designed by Softbank, for all implementation. Pepper has two arms. Each arm has five degrees of freedom (DoF), with one more degree of freedom for hand closing and opening. Further, Pepper has two DoFs for head movements, two DoFs for hip movements, and one additional DoF for knee movements.

The robot is natively controlled through an on-board Atom E3845 CPU card. This is where robot sensors and motion commands are integrated. An experimental ROS node is available to control Pepper directly via the CPU over a wireless network.

B. Robot Command Unit (RCU)

The Pepper robot has a smart input pad with an Android based system on its chest. We refer to this smart device as robot command unit (RCU). The smart device communicates with the CPU to integrate sensor information and send robot commands via hardwired connections. It serves as the primary central control of the robot. The robot manufacturer allows the user to program the RCU directly with Android API. The user can alternatively program Pepper from another PC via WiFi using ROS, which can be convenient for roboticists.

However, we chose to directly program the robot semaphore execution on RCU rather than over WiFi with ROS for the following reasons. First, a hardwired connection is fast, reliable, and free from any interruption caused by poor WiFi connections. Further, the smart device has wireless connectives, for both indoors or outdoors using WiFi or mobile LTE, and it can act as a natural connection to the cloud services. Additionally, touch-screen-based GUIs are easy to use for local users. Finally, onboard sensors, especially cameras, can be used as perception inputs for the robot. Therefore, we position the RCU as a local robot controller as well as the local network gateway from cloud robotic services. This is illustrated in Figure 1.

C. Cloud Intelligence HARI

Currently, HARI has three major modules: (1) cloud-based AI engine and HI teleoperation interface, (2) RCU in the form of smart devices as WiFi and mobile LTE gateway to cloud services, and as mid-level controller for local robot, and (3) a high bandwidth, reliable, secure private-LTE network connecting the cloud AI directly to local robot via RCU (see figure 1).

As mentioned before, the key idea in building a real-time, scalable cloud application with wide geological coverage is to maximize the cloud computation for valuable AI services and minimize the communication costs for each robot via RCU. We use semaphore mirroring robotic task as an example to demonstrate how to use these ideas to build such cloud-based applications in real life. First, we move the deep learning based gesture recognition system, OpenPose, entirely to HARI's cloud AI GPU servers. The GPU server includes four Nvidia Titan Xp graphics cards. Second, to minimize the communication costs, we stream 320x320 videos with highly efficient video compression standard H.264 from RCU to Cloud HARI through private LTE.

Third, mitigating the network latency is important to ensure a robust motion control over long-range communication, because network latency is unavoidable at times due to routing and physical distances even with a reliable LTE network. Therefore, we store pre-generated semaphore motion trajectories in the local RCU, and execute these trajectories based on commands sent from the cloud. This way, we hide the network latency caused by video streaming and cloud control during the trajectory execution when the robot is moving from one semaphore to another. (Fig. 5). Additionally, the cloud AI only needs to send intermittent alphabetical letters to control robots to perform semaphore. Please refer to Section IV and V for an exposition of these three design decisions.

D. Semaphore Motion Generation in ROS

Semaphore is the telegraphy system conveying information at a distance using visual signals with hand-held flags. Flags encode alphabet information with positions of both arms. Each of the two arms has the same set of different arm positions: home, down, flat, up, top, cross down, and cross up (see Fig. 2A). The combination of these seven positions using two arms forms 49 possible stationary positions. These positions are used to represent the alphabet (Fig. 3A), and additional symbols such as home, space, etc. (Fig. 4C, 4D).

Based on the above definition, we first program the seven different positions of each arm for the Pepper robot (fig. 2A). We then use the Pepper's ROS interface to generate trajectories that would move each of the Pepper's arms between any two of these seven positions. (fig. 2B) We use SBL, short for Single-Query Bi-Directional Probabilistic Roadmap Planner [33], to generate these trajectories in MOVEIT! using OMPL. We record the resulting trajectories to ROS bags, and convert them into Android readable format.

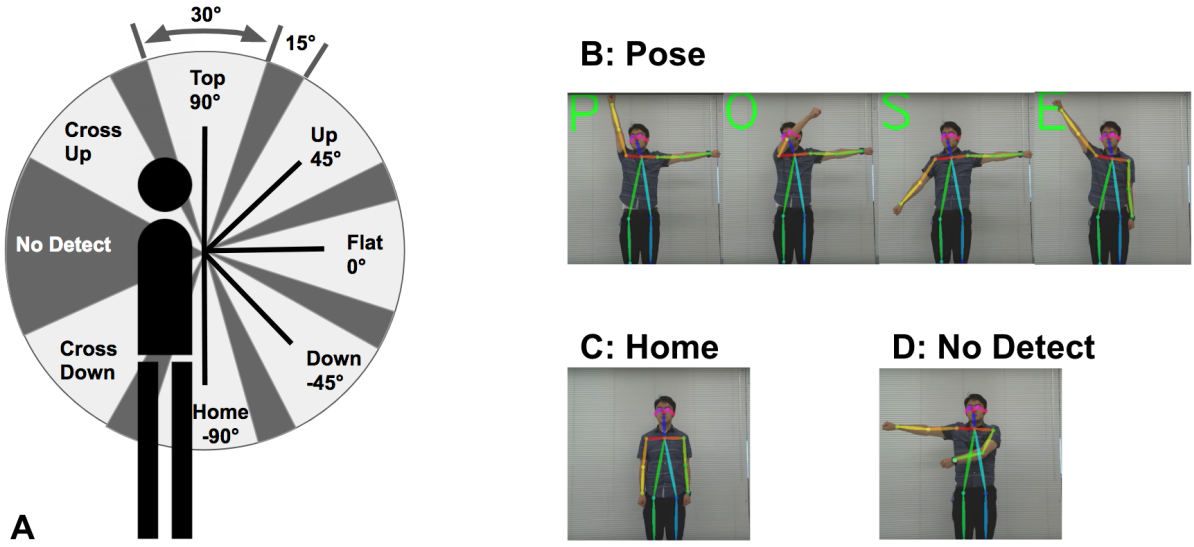

Fig. 4. **Semaphore Recognition using Openpose: (A) Single Side Position Recognition** Seven positions are recognized based on a unit circle centered at the shoulder joint; the angle between the line formed by the hand and the shoulder and a horizontal line is used to recognize the seven positions of a single arm. **Light zones** indicate the detection zones, whereas **dark zones** are "No Detect" zones. No detect zones are used to avoid false detection between two neighboring positions. We can reliably recognize the full **semaphore alphabet** in real-time (video: `https://youtu.be/arC6OZGgkgE`). **(B)** the user spells out "POSE" and the semaphore recognition system recognizes the letters. **(C) "Space"** is represented by both arms rest in the "home" position. **(D) "No Detect"** is recognized because the left arm of the demonstrator is in the dark "No Detect" zone.

E. Trajectory Execution at RCU

We store pre-generated trajectories in RCU. Upon receiving an alphabetical letter command, RCU on Pepper would look up the trajectory that corresponds to the current hand position as start and the received semaphore letter position as goal. It will then execute this trajectory locally until the robot arms stop at the expected commanding positions. This is programmed using Pepper's Android API.

Note that the execution is a blocking function. The robot arm would not change its execution trajectory or take any other command until the motion is finished. We later take advantage of this feature to hide the network latency for responsive user experiences.

F. Command Line Interface on HARI

We test this local robot semaphore controller through a command line interface in HARI. In this interface, a cloud operator would input a letter command via HARI. When RCU receives the letter, it performs semaphore as described above.

We further modified this command line interface so that a word or a sequence of letters can act as a command to Pepper. When a word is sent, Pepper will perform semaphore for each of the letters in sequence. It would also spell out the letters as it moves and pronounces the word at the end of the sequence. Fig. 3 illustrates Pepper performing "HARI" in sequence. This demonstrates a form of AI-assisted teleoperation in which a cloud operator can communicate to the person in front of the robot in the form of combined semaphore motion and spoken word by using Pepper as the media. (see supplemental video for a demonstration: `https://youtu.be/XOeiDrG4sAQ`) This interface becomes a part of HARI teleoperation module during the developments (Fig 1Top).

G. Semaphore Recognition

Semaphore mirroring is built on top of the OpenPose based semaphore recognition system and the command line teleoperation system mentioned above. To program semaphore recognition, we use only the largest skeleton extracted from OpenPose. This skeleton corresponds to the person standing in front of the robot. All other skeletons in the background are filtered out. Based on this skeleton, we measure the relative angle from the left-hand position to the horizontal line crossing the right shoulder. If this angle falls into the perspective gray detection zone (Fig. 4A), then the left arm's semaphore position is recognized. Note that the black area indicates a "no-detect" zone. We setup the no-detect zone to prevent false detections between the two neighboring positions since the skeleton extracted by OpenPose can be noisy and the detected hand position can easily cross the border at times.

The right-hand semaphore recognition is programmed the same way, except that the detection zone map was horizontally flipped from the one illustrated in figure 4A. Combination of the detected left-and-right-hand position is used to infer semaphore based on its alphabet. In Figure 4B, we show that the system could detect multiple letters of the alphabet, and the subject spells out the word "POSE" using semaphore. Figure 4C shows the home position which corresponds to a "space" in the alphabet, and Figure 4D shows a case of "no-detect" because the left hand of the subject is in the "no-detect" zone.

H. Semaphore Mirroring with Hybrid Control

When the cloud AI detects a change in the human gesture in semaphore, HARI will use the command line interface to send a letter to Pepper's RCU. RCU takes this letter command and move the robot arms to that semaphore position. This completes the semaphore mirroring pipeline. To minimize communications, HARI will not send in semaphore command at every frame of semaphore detection. It only sends a command to RCU when it detects a change in semaphore letter recognition, see fig. 5.

However, there is a problem if we use the raw command signal sequence sent from HARI. The semaphore motion executed by RCU is a blocking function. The RCU would not process the command buffer during the motion. If we choose to program the receiving buffer at RCU using a stack, the original command sequence were to be reserved. So when multiple semaphore changes are to be detected during the semaphore motion execution, then the letter representing the "first" detection are executed first, then followed by the other ones, until the most "current" semaphore command is detected. It surprises many users as the robot executes several semaphores before it starts to execute the semaphore they are intended to demonstrate.

An alternative is to program the receiving buffer as a queue, so that when a motion was finished, RCU jumps to the most current semaphore gesture demonstrated by the user. Users would feel more natural since they perceive the robot is mirroring them faithfully. Therefore, the queue-based receiving buffer is chosen.

The semaphore mirroring pipeline is a form of hybrid control–Discrete, high level command is sent from the cloud intermittently to a local RCU where continuous motion is executed. As we will discuss later, this technique helps hide latency between the cloud and the robot. It makes the cloud-based real-time semaphore mirroring Pepper more natural to interact with, even under extreme network conditions.

IV. EXPERIMENTS AND RESULTS

A. Cross Pacific Experiments

We conducted cross Pacific robot control experiments. We used a US HARI server located in Oregon, United States to control a Pepper robot located in Tokyo, Japan, with a human subject standing in front of Pepper as a demonstrator. We also performed the same test to a Pepper robot located in Santa Clara, United States, with the same HARI server in the United States (see video illustrates US-to-US test case: `https://youtu.be/NXrdoSXXxqY`). Furthermore, to show scalability, we successfully used the US based HARI server to support both Peppers at the same time, one in Japan and the other in the United States.

B. Reliability of the Cloud Semaphore Recognition System

We first performed reliability tests of semaphore recognition system in the United States. To test reliability, we randomly showed a semaphore position for the demonstrator to perform, and check the recognized semaphore results obtained from the cloud server. We conducted two sets of experiments with two different demonstrators with thirty experiments each. We positioned the robot camera in such a way that both shoulders of the subject were at the center horizontal line of the image. Further, no arm, when fully stretched, should go out of the camera's frame. The background of the demonstrators were uniformly white, and the experiments were conducted in a well-lit room.

During the first set of experiments, we hid the real-time recognition results from the demonstrator. The recognition accuracy was 90.0% (27/30) for subject one, and 76.7% (23/30) for subject two. (see the first row of table I) The accuracy were high, but not reliable, and not consistent across subjects.

However, we noticed that all of the failures were caused by "No Detect" error, meaning one or both of subject's hand was in the no detect zone. None of these recognition errors was due to confusion between two different semaphores. This made us think that the failed trails were caused by habits of different person when performing semaphore rather than system errors such as a high noises level of OpenPose's recognized joints positions. The actual recognition accuracy can be significantly higher if the demonstrators were trained, or if the real-time results were shown to the demonstrators.

Therefore, we conducted a second set of experiments for the same subjects by allowing them to look at the semaphore recognition screen to adjust their pose for a better recognition result. However, they could not adjust their arm so that their arms moves outside the quadrant they were in. A quadrant was defined as the uni-circle made by the vertical and horizontal lines in these experiments.

We obtained perfect accuracy of 100% for both subjects in the second set of experiments. (see the second row of table I) Therefore, it can be concluded that the semaphore recognition system is highly reliable for the purpose of remotely teleoperating a robot to perform semaphore mirroring (see video demonstration: `https://youtu.be/arC6OZGgkgE`).

TABLE I
OPENPOSE SEMAPHORE RECOGNITION ACCURACY (%)

	Subject 1	Subject 2
Open Loop	27/30	23/30
with Feedback	30/30	30/30

C. Real-Time performance of the Cloud Semaphore Recognition System

We used semaphore mirroring as an example to demonstrate the benefit of using a cloud-based AI platform. We measured the real-time performance of the cloud-based OpenPose system. We used a single Titan Xp GPU with a single thread to launch OpenPose in the cloud. We fed a local video stream with resolutions of 640x480 to OpenPose. We observed 10-12 fps (frames per second) inferencing framerate while using a pre-trained 320x320 neural network model provided by OpenPose.

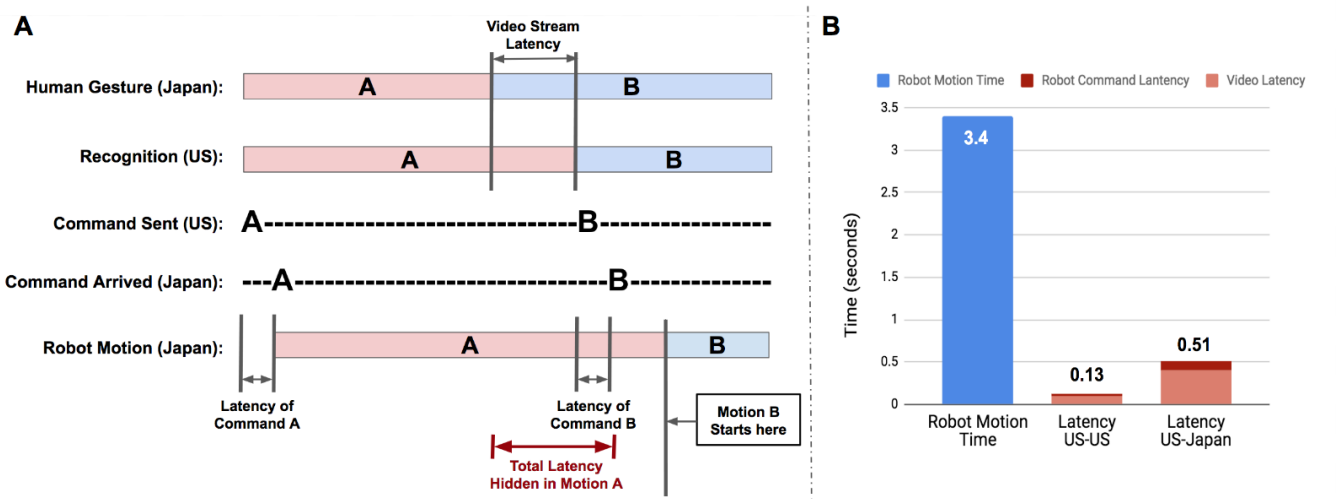

Fig. 5. **A Hybrid System Hide the Network Latency: (A) Timing Diagrams of the Hybrid System** The diagram shows that, starting from the actual human gesture in Japan, the video latency and latency of sending in a command after recognition are consecutive. The sum of the two forms of latencies is marked in red. The total latency is hidden inside motion A, but occurs before the start of motion B. **(B) Timing Comparison** Robot semaphore execution time is much higher than either the US-to-US network latency or the US-to-Japan network latency.

The resolution parameter on the neural network model indicates the picture resolution of the dataset that was used to train the neural network. The video stream fed into the neural network should be as similar to this resolution to guarantee gesture positional accuracy. Resolutions of 640x480 is the minimum resolution on HARI, therefore, we used it for the latency benchmark later. A neural network model trained on larger resolution pictures, such as 640x640, can be used. However, larger picture size requires neural network model with more weights, thus more computation during inferencing, which affected the real-time performances of the OpenPose system. Only 5-6 fps can be achieved with a 640x640 neural network model while we gain little accuracy via the change. Therefore, we chose to only benchmark network performance of the case when 640x480 video stream and 320x320 neural network model were used.

We measured the communication costs to stream video to HARI. To minimize communication costs, we streamed in a 640x480 resolution with a highly efficient video compression standard H.264 from RCU to Cloud HARI through private LTE. We measured consistent 22-30 frames per second video stream both from the US-to-US test case and from Japan-to-US test case. The difference between the two cases are video streaming latency. There is around 100 ms latency for the the US-to-US case, and around 400 ms latency for the Japan-to-US case (Table II)

This suggests that semaphore recognition is the bottleneck for the cloud-based recognition system, which still provides 10-12 fps real-time performances even if we stream video from Japan to the US. The cloud-based OpenPose system has an order of magnitude better performances compared to OpenPose on an iPad 2017, which can only process a little more than one frame per second or less [4].

The results from these experiments support our claim that communication costs are lower compared to the benefits gained from using a cloud robotic AI framework, in our semaphore mirroring system.

D. Hybrid System Design Hides Network Latency

A hybrid system can hide the network latency in the robot motion in this cloud-based system. The mechanism is illustrated in Fig. 5. The semaphore motion time average is 3.4 seconds. It is long compared to the total latency contributed by video streaming and robot commanding (Fig. 5B). After the demonstrators are used to the system, they tend to start the next semaphore demonstration before the robot finishes executing the last semaphore. The semaphore recognition system in the cloud would recognize any change of gesture earlier despite the delay from video streaming. The change in gesture would trigger HARI to send the next semaphore command, which can be delivered to RCU before the end of the last semaphore execution. Therefore, the next command would sit in RCU's receiving buffer until the end of the last motion, so that the next command is executed immediately after the end of the last motion.

TABLE II
NETWORK LATENCY (MS)

	Video	Robot Command	Total
Japan to US	400	108	508
US to US	98	31	129

E. Compared to a Local Full Gesture Imitation System

We also built a general gesture imitation system for Pepper using a 3D Kinect camera with a local PC in ROS (see video: `https://youtu.be/fPqF1xwqRcY`. To compare this

local general gesture imitation system and our cloud-based semaphore mirroring system, we performed semaphore mirroring using both systems.

The general gesture imitation system finishes a semaphore within 5-10 seconds which is slower but close to the performance of our cloud-based system. However, it failed when the human demonstrator drove Pepper into unreachiable areas in its arm. This failure cases happens more often when the start and goal positions of the robot is far away, for example from home to top, or when any of the cross positions are involved. In these cases, Pepper would stuck in those places for a while and the user would need to find a way to drive the arm out via demonstration. There are also times when the user drives two arms to collide to each other, though this does not happen often during semaphore demonstrations. In contrast, the cloud-based semaphore system never failed because the trajectory was generated from an automatic path planning algorithm. The system checks for kinematic constrains and self collisions during the generation. We will discuss the trade-offs between the two systems next.

V. DISCUSSION AND FUTURE WORK

We showed that, with a private-LTE network and a highly compressed video stream, we can achieve a 22-30 fps video streaming rate even if we stream from Japan to the United States. This is faster than the cloud-based gesture recognition rate–10-12 fps, therefore, the current state of the art network is sufficient to support real-time perception even across long distance. This confirmed our hypothesis that the value of running a more powerful AI service in the cloud would reduce the communication costs, as even better and cheaper communication technologies, such as 5G LTE network, would become available in the future.

In the HARI cloud-based hybrid system, discrete command is sent from the cloud and the robot performs pre-generated motion upon receiving the command. Therefore, only a robot control unit (RCU), not a full motion planner, is necessary at the local robot, dramatically reducing the computation requirements for a low-budget robot. Further, we only send a semaphore command when there is a change in semaphore recognition, makes the communication highly compact. As we illustrate in Fig. 5A, the hybrid system help hide network latency during robot motion, making the human robot interaction responsive even if the cloud server is far away.

Such cloud-based hybrid system would help scalability in two ways. One, it scales up the number of robots that can be controlled by a single server. By leveraging the latency hiding phenomenon and sending compact robot commands, multiple robots at different locations can be controlled in real-time for responsive interaction with humans. Two, by putting perception and control into the cloud, we can duplicate the servers easily and deploy them on a large scale, theoretically, over different continents across the globe. These perception and control services can be updated and managed in a centralized fashion, and can be scaled up or scaled down based on demand.

There are, however, limitations of the current hybrid, cloud based, gesture imitating system. It can only imitate a discrete set of 2D gestures, semaphore, which is a sub-set of general human gesture. It also requires roboticist to program and generate these motions before-hands and store them in the RCU, which can be labor intensive if more general gestures are needed. Further, the number of pre-generated motion grow quadratically in respect to the number of discrete gestures needed. When the set of discrete gestures becomes large, true in the case of general gesture followings, the memory requirements on a local robot grows significantly, which is less ideal for a low-budget design.

As future work, we aim to generalize the semaphore mirroring system so that general gesture following can be achieved with such cloud-based hybrid system. It is desirable to create discrete gesture segments from a large human activity database using machine learning techniques, which obviates the needs for explicitly programming poses and gestures by hand. This can also help reduce local robot storage needed to save pre-generated motion. A promising gesture imitation system from annotated human gesture data is recently explored in physical simulation [21]. We would want to extending these simulated results for a physical humanoid robot for similar performance. Furthermore, it is important to explore human robot interaction rather than gesture imitating with human compliance hardware.

VI. ACKNOWLEDGMENTS

We thank all members of the AUTOLAB and members at Cloudminds – Mas Ma and Arvin Zhang–for their contributions and support. We also thank Mark Mueller from Berkeley HiPeRLab for his valuable discussion on latency effects of HRI with cloud robotics. The authors would like to acknowledge fundings from Cloudminds Inc. and the Office of Naval Research. 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] S. Robotics, "Pepper," *Softbank Robotics*, 2016.

[2] ——, "Nao www. ald. softbankrobotics. com/en/robots/nao," *Last accessed*, vol. 20, 2017.

[3] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The icub humanoid robot: an open platform for research in embodied cognition," in *Proceedings of the 8th workshop on performance metrics for intelligent systems*. ACM, 2008, pp. 50–56.

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[5] K. K. Biswas and S. K. Basu, "Gesture recognition using microsoft kinect®," in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*. IEEE, 2011, pp. 100–103.

[6] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[7] M. Hirose and K. Ogawa, "Honda humanoid robots development," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 11–19, 2007.

[8] M. A. Diftler, J. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. Permenter *et al.*, "Robonaut 2-the first humanoid robot in space," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2178–2183.

[9] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[10] F. Iida, M. Tabata, and F. Hara, "Generating personality character in a face robot through interaction with human," in *7th IEEE International Workshop on Robot and Human Communication*, 1998, pp. 481–486.

[11] M. Inaba, "Remote-brained robots," in *Proc. International Joint Conference on Artificial Intelligence*, 1997, pp. 1593–1606.

[12] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 299–304.

[13] ——, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[14] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[15] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," in *Proc. Australasian Conference on Robotics and Automation*, 2012.

[16] K. Qian, J. Niu, and H. Yang, "Developing a gesture based remote human-robot interaction system using kinect," *International Journal of Smart Home*, vol. 7, no. 4, 2013.

[17] D. Matsui, T. Minato, K. F. MacDorman, and H. Ishiguro, *Generating natural motion in an android by mapping human motion*. Springer, 2018.

[18] 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.

[19] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in cognitive sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[20] C. Breazeal, "Social interactions in hri: the robot view," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 181–186, 2004.

[21] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)*, vol. 37, no. 4, 2018.

[22] S. Waldherr, R. Romero, and S. Thrun, "A gesture based interface for human-robot interaction," *Autonomous Robots*, vol. 9, no. 2, pp. 151–173, 2000.

[23] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," *arXiv preprint arXiv:1704.06888*, 2017.

[24] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[25] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the google object recognition engine," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4263–4270.

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

[27] 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.

[28] A. Vick, V. Vonásek, R. Pěnička, and J. Krüger, "Robot control as a service—towards cloud-based motion planning and control for industrial robots," in *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on*. IEEE, 2015, pp. 33–39.

[29] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2015.

[30] J. P. Jeffrey Ichnowski and R. Alterovitz, "Cloud based motion plan computation for power constrained robots," in *2016 Workshop on the Algorithmic Foundations of Robotics*. WAFR, 2016, .

[31] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Networks and Communications (EuCNC), 2017 European Conference on*. IEEE, 2017, pp. 1–6.

[32] P. Simoens, M. Dragone, and A. Saffiotti, "The internet of robotic things: A review of concept, added value and applications," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, p. 1729881418759424, 2018.

[33] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics Research*. Springer, 2003, pp. 403–417.