# A Fog Robotic System for Dynamic Visual Servoing

Nan Tian,[1,2] Jinfa Chen[2], Mas Ma[2], Robert Zhang[2], Bill Huang[2]
Ken Goldberg[1] and Somayeh Sojoudi[1,3,4]

***Abstract*— Cloud Robotics is a paradigm where distributed robots are connected to cloud services via networks to access "unlimited" computation power, at the cost of network communication. However, due to limitations such as network latency and variability, it is difficult to control dynamic, human compliant service robots directly from the cloud. In this work, by leveraging asynchronous protocol with a "heartbeat" signal, we combine cloud robotics with a smart edge device to build a Fog Robotic system. We use the system to enable robust teleoperation of a dynamic self-balancing robot from the cloud. We first use the system to pick up boxes from static locations, a task commonly performed in warehouse logistics. To make cloud teleoperation more efficient, we deploy image based visual servoing (IBVS) to perform box pickups automatically. Visual feedbacks, including apriltag recognition and tracking, are performed in the cloud to emulate a Fog Robotic object recognition system for IBVS. We demonstrate the feasibility of real-time dynamic automation system using this cloud-edge hybrid, which opens up possibilities of deploying dynamic robotic control with deep-learning recognition systems in Fog Robotics. Finally, we show that Fog Robotics enables the self-balancing service robot to pick up a box automatically from a person under unstructured environments.**

## I. INTRODUCTION

Service robots are robots that operate semi- or fully autonomously to perform services useful to the well-being of humans and equipments [1]. International Federation of Robotics (IFR) predicts that 32 million service robots are to be deployed between 2018-2022 [2]. Some popular service robot applications include elderly care, house cleaning, cooking, patrol robots, robot receptionists, entertainment, and education. A few famous examples of service robots are Roomba by iRobot, Pepper by Softbank Robotics, "the robotic chef" by Moley Robotics, and Spotmini and Atlas by Boston Dynamics.

Different from industrial robots, service robots need to interact and cooperate with people safely under dynamic unstructured environments. Therefore, two key requirements for service robot operations are (1) accurate, general visual perceptions and (2) intelligent, dynamic, human compliant robot controls.

With recent breakthroughs in deep neural networks and robotic learning, robot visual perceptions [3] [4] [5] [6] and intelligent controls [7] [8] [9] [10] [11] under unstructured environments have become readily available . However, these learning-based technologies come with a high computation cost, and it is hard to deploy them directly on native robot controllers that have limited computation power. In our previous work on gesture based semaphore mirroring using a humanoid robot [12], we addressed this problem by moving deep-learning-based gesture inferencing into the cloud.

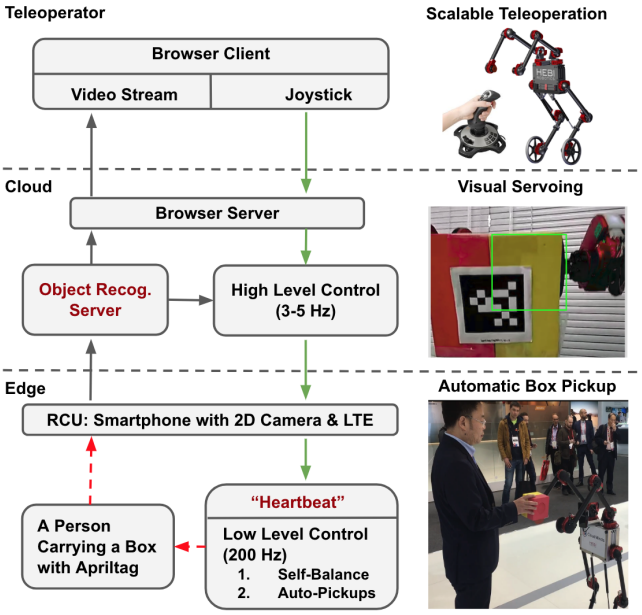


Fig. 1. **The Fog Robotic system for dyanmic visual servoing: (Left)** Architecture diagram and information flow–visual perceptions (black arrows), control signals (green arrows), human-robot interactions (red arrows). **(Right)** Illustrations of three major contributions of this work, teleoperation, visual servoing, and auto box-pickups from a human. They are positioned to their related functional blocks in the Fog Robotic system

However, scalable cloud robotics systems are associated with high network communication costs, in the form of privacy, security, bandwidth, latency, and variability. Specifically, network latencies and variabilities, bounded by speed-of-light and inconsistent network routings, prevent cloud-based robotic controller from controlling dynamic robots directly for interactive, human-compliant robot tasks, especially those requiring visual feedbacks.

In this work, we combine both powerful cloud services and agile edge devices to build an intelligent Fog Robotic control system. Our Fog Robotic system is implemented under Human Augmented Robotic Intelligence Platform, or HARI [12], provided by Cloudminds Inc. We integrate this system with a dynamic, dual arm, dual leg, self-balancing

[1]Department of Electrical Engineering and Computer Science (EECS), University of California, Berkeley, USA; {neubotech, goldberg, sojoudi}@berkeley.edu
[2]Advanced Technology Department, Cloudminds Technology Inc., Santa Clara, US; {robert, bill}@cloudminds.com
[3]Department of Mechanical Engineering, University of California, Berkeley, USA;
[4]Tsinghua-Berkeley Shenzhen Institute.

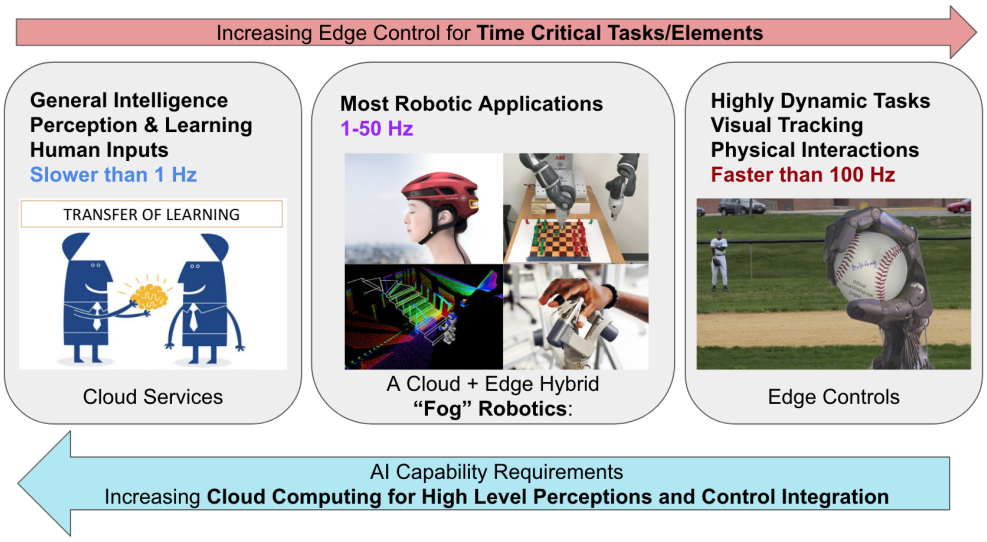


Fig. 2. **Intelligence vs. Reflexes: Fog Robotics Balances Cloud and Edge Computation for Different Robotic Applications. (Left) Cloud Services** provide high-level intelligence, such as deep-learning based perception systems and cloud based human teleoperation. **(Right) Edge controller** fastest control loops, control highly dynamic robotic tasks, but often lacks high performance computer. **(Middle) Fog Robotics** combine the intelligence of the cloud and the responsiveness of the edge, covers 80% of the robotic applications. **(Top and Bottom) Move the computing to the cloud for more intelligence**, and **move the control to the edge for highly dynamic, closed-loop control**. Together, Fog Robotic system can handle most service applications that require both intelligence and fast reflexes.

robot, named Igor, made by HEBI robotics.

Leveraging a "heartbeat" communication protocol between cloud and edge, we are able to teleoperate Igor robustly using HARI. We choose to perform a robotic task that is commonly performed in warehouse logistics, namely box pickups from a human carrier.

While it is intuitive to teleoperate Igor during navigation, we found that it was extremely difficult and inefficient to teleoperate Igor to pick up objects as simple as a box. To make the cloud-based teleoperation more intuitive, we program a dynamic automatic box-pickup module based on visual detection of an apriltag [13], [14]. Apriltage detections are performed in the cloud to emulate how a cloud-based deep learning object recognition system would affect the performance of this Fog Robotic visual feedback loop. Furthermore, to avoid performing time-consuming camera calibration and registration whenever the robot performs a box pickup, we choose to implement the module using 2D Image Based Visual Servoing (IBVS). With the Fog Robotic IBVS, we demonstrate that Igor can perform reliable, automatic box pickups from a human carrier under unstructured environments.

## II. CONTRIBUTION

The main contributions of this work are as follows:

1) **A "heartbeat" protocol** that enables robust teleoperation of a dynamic robot in Fog Robotics.
2) **A Fog Robotic visual servoing module** that enables automatic box-pickups to assist cloud teleoperators
3) **Automatic box-pickups from a human** to demonstrate dynamic human robot interaction (HRI) under unstructured service environments.

## III. RELATED WORK

**Cloud Robotics** encompasses any robot or automation system that relies on either data or code from a network to support its operation [15]. The term was introduced by James Kuffner in 2010. It was evolved from *Networked Robotics* [16]. Well-known Cloud Robotic Systems includes: RoboEarth's Rapyuta [17], motion planning for services at both cloud [18] and edge [19], Berkeley robotics and automation as a service (Brass) [20], and Dex-Net as a Service (DNaaS) [21], just to name a few. However, network costs in the form of privacy, security latency, bandwidth, and reliability present a challenge in Cloud Robotics [22].

**Fog Robotics** was recently introduced by Goldberg et al, and is defined as an extension of Cloud Robotics that balances storage, compute and networking resources between the Cloud and the Edge [22]. It is inspired by Fog Computing, originally introduced by Cisco Systems in 2012 [23]. In Fog Robotics, cloud computing resources is brought closer to the robot so that learning can be done close to where data is created. It has found its applications in service robots where robot can learn surface grasps from nearby unstructured environments [22]. In this work, we use Fog Robotics for (1) cloud-based teleoperation of a dynamic robot; (2) host vision servoing server to provide robot object recognition and localization feedbacks under unstructured environments.

**Robotic Vision** for service robots to operate under unstructured environment is challenging if traditional industrial robotic approaches were used, because these methods were developed for precision under highly structured manufacturing environments. Registrations [24] and calibration [25] are often done before a robotic task, and they can be time consuming and do require additional expertise for system maintaining.

**Image Based Visual Servoings (IBVS)** [26], [27] uses camera 2D image space to measure relative distances between the robot and the target. The measurements are independent of the exact 3D locations of the robot and the target. Therefore, camera registrations and calibrations are not required before each robotic task, desirable for service robots deployments under unstructured environments for human robot interactions.

Furthermore, recent development in deep-learning-based vision systems allow recognition [4], object detection [3], segmentation [5], and human gesture recognition [6] to be performed for unstructured environments in semi-real-time (5 - 10Hz) on a GPU server. Previously, we successfully implemented a cloud-based gesture perception system for a humanoid robot gesture mirroring task [12]. More advanced robotic learning systems based on visual feedbacks has been developed for grasping [28], [7], visual servoing [11], guided policy search [8], [9], visual foresight [10], and domain randomization for transfer learning from simulation to reality [29], [30]. All of these can be deployed in Fog Robotics to provide visual feed-backs for large scale service robot deployments.

## IV. SELF-BALANCING ROBOT IGOR

We use a 14 degrees of freedom (DoF), dual-arm, dual-leg, dynamic self-balancing robot named Igor (shown in Fig. 3). It is designed and made by HEBI robotics. Each DoF is built with a self-contained, series elastic X-series servo modules. These servos can be controlled with position, velocity, and torque commands simultaneously, and can provide accurate measurements of these three quantities to a central computer at high speed ( >1KHz) with minimum latency. These modules are connected with Ethernet to an on-board Intel Nuc computer in the metal control box for self-balancing control.

The self-balancing is achieved by modeling the system as an inverted pendulum (see Fig. 3 bottom). To estimate the center of mass (CoM) of the robot, the CoM of the two arms and two legs are first measured through HEBI's API in real-time using forward kinematics. The position of the total CoM is then estimated as the average of the CoMs of the four extremities plus the CoM of the control box weighted by the mass distribution:

$$x_{CoM} = \frac{\sum_i m_i x_i}{\sum_i m_i} \qquad i = \text{arms, legs, box} \tag{1}$$

Igor also uses accelerometer measurements from the four servo modules attached to the control box to estimate the direction of gravity ($G$) at all times. With CoM of Igor, center of wheels ($o_w$), and direction of gravity, we can calculate the length and direction of the inverted pendulum:

$$L = x_{CoM} - o_w \tag{2}$$

The lean angle ($\psi$), which is the angle between gravity and the inverted pendulum can then be estimated in real-time:

$$\psi = cos^{-1}\Big(\frac{L \cdot G}{|L||G|}\Big) \tag{3}$$

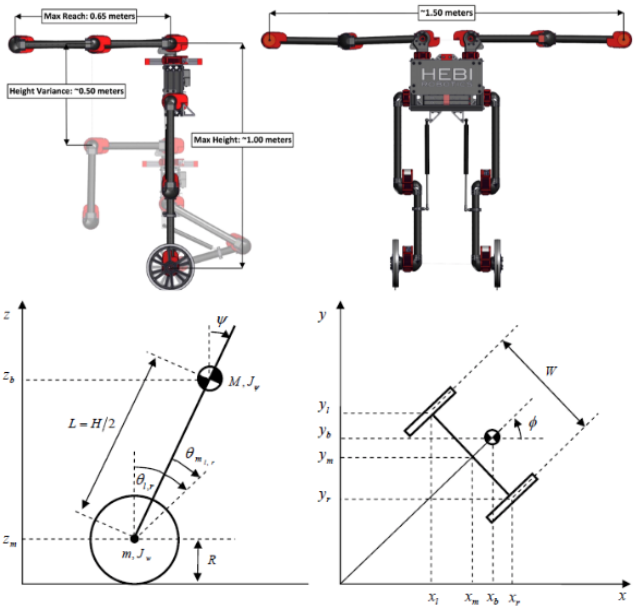


Fig. 3. **Self-Balancing Robot Igor: (Top)**14 Dof, dual-arm, dual-leg robot built with series elastic servo modules. **(Bottom)** free body diagram of the inverted pendulum balancing control (**see supplemental video for details**)

To keep the robot balancing, assuming that the lean angle ($\psi$) is small so that we can linearize the system:

$$\psi \approxeq sin(\psi) \tag{4}$$

a torque ($\mathcal{T}$) in the direction of falling is applied to the wheel with radius ($R$) and angular velocity ($\omega$) to counteract the effects of gravity on the robot's center of mass:

$$\mathcal{T} = R\omega = v - \dot{\psi}L \tag{5}$$

where $v$ is the velocity of the robot's CoM. Furthermore, the derivative of the lean angle ($\dot{\psi}$) can be controlled by a proportional controller with coefficient ($K_v$), and is related to the velocity of the robot as follows:

$$\dot{\psi} = K_v v \tag{6}$$

The real-time measurements of both robot CoM and direction of gravity are important, because the Igor controller needs to compensate for dynamic movements of the four extremities for robust self-balance control. We can also rotate the robot by varying velocity applied to the two wheels, and control the angle of rotation based on inertia measurement unit (IMU) readings in real-time.

## V. AN INTELLIGENT FOG ROBOTIC CONTROLLER

### A. Edge Controllers

There are two edge controllers in our system. The first one is an Intel Nuc computer in the Igor control box. It collects all sensor information from the 14 modular servos, and it controls all servos in real-time. It hosts high-speed feedback control loops (200Hz or above) to maintain robot posture

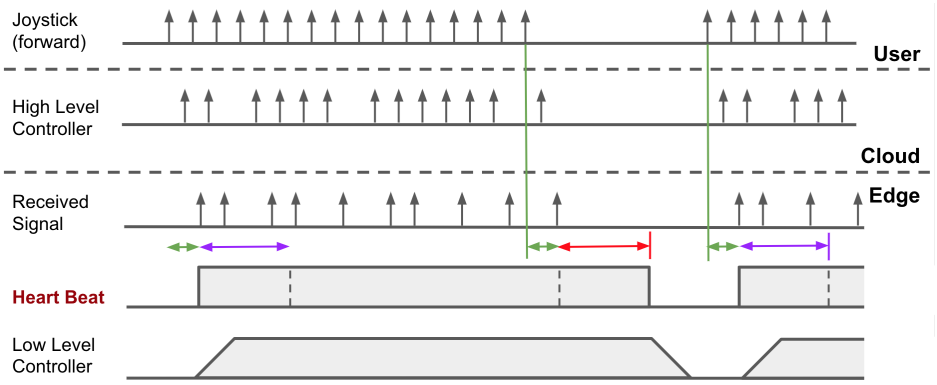


Fig. 4. **"Heartbeat" Protocol with Asynchronous Communication to Send a Forward Command: (Top) Teleoperator with a Joystick** sending a series of forward command signals. **(Middle) Cloud High-Level Controller** receiving most of the packages with a little latency. It then forwards received commands to the edge device. **(Bottom) Edge device** receives the commands with a lot of lost packages and some delays. A "heartbeat" signal for forward motion is turned on upon receiving the first command, and will stay on the duration of the sliding window **(250 ms, purple arrow)**. The "heartbeat" will turn off when there is no command received during the sliding window **(red arrow)**. The **green arrow** marks the network delay from the joystick to the edge device. Finally, at the **low-level controller**, a ramp-up and ramp-down function is used to smooth out the start and stop forward velocity perturbation to the balancing system so that the robot can move forward without jitters or instability.

and self-balancing. We refer to it as the low-level controller in Fig. 1.

The other edge device is the robot command unit (RCU). It is a smart android phone with a private LTE connection and a 2D camera (Fig. 1). RCU serves as the gateway between the high-level cloud robotic platform and the low-level self-balancing controller. It uses the private LTE connection to stream live videos to the cloud, and it receives and forwards high-level intelligent controls from the cloud to the low-level controller with minimum delay. RCU works both indoors and outdoors with a good LTE reception.

Furthermore, we are in the process of integrating HEBI's self-balancing controller into the smart phone. It can replace the native Intel Nuc computer so that it will serve as both RCU and low-level controller. By making the edge controller more compact, we gain more battery room in the control box for a longer robot operation time.

### *B. Cloud Controller*

A high-level intelligent robot controller is placed in the cloud to work with the edge controller (see Fig. 1). It operates at a lower speed (3-5Hz), yet it commands the robot based on HARI's Artificial Intelligence (AI) and Human Intelligence (HI) services, which is critical for robots to operate under unstructured environments. Depending on the situation, it can either extract commands based on the object recognition server or forward commands sent from a cloud teleoperator. These high-level commands are sent to RCU and are executed in a different form on the low-level controller.

### *C. Hybrid Control with "Heartbeat"*

When controlling Igor, commands sent from the high-level cloud controller act as perturbations to a time-invariant, stable system maintained by the low-level self-balancing controller at the edge. This is a form of hybrid control where discrete signals are sent from the cloud to control a dynamical system at the edge.

Minimum delays in the cloud to edge robot command delivery are desirable for intuitive teleoperation. We choose to use the asynchronous network protocol UDP to implement the cloud-edge communication. However, since deliveries are not guaranteed in UDP, packages can be lost during communication. Further, the packages can arrive at the designation in different orders from their original sequence. Both problems can create variable controls at the edge controller, which can cause instability in self-balancing. This would affect user experiences during teleoperation as well, and can be dangerous to people around the robot.

We implement a "heartbeat" signal at the edge controller to solve these problems (shown in Fig. 4). The "heartbeat" is a switch signal that is turned on when the first signal arrives at the edge. It will remain on for a period of time (t) and will only turn off if there is no package received for the selected command during this time. We can view the "heartbeat" design as performing a "convolution" with a moving window on the signal received. Finally, we turn the "heartbeat" signal into an edge control signal with a ramping function at the beginning and the end of the control to ensure a smooth start and end action when the controlled perturbation hits the stable self-balancing system.

## VI. DYNAMIC VISUAL SERVOING

To assist teleoperation with automation, we focus on using Fog Robotics to control a dynamic robot with Image Based Visual Servoing (IBVS) to automatically pick up a box. We choose IBVS because it eliminates extensive camera calibration that is hard to maintain on a dynamic robotic system. The goal of our IBVS is to navigate the robot to an optimal box pickup location where the apriltag lay within the green target box, which has the same size as the apriltag (Fig. 5)

The aim of visual-servoing-based control is to minimize the relative error between the measured target position and

desired target position $\boldsymbol{e}(t)$:

$$\boldsymbol{e}(t) = \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}) - \boldsymbol{s}^* \tag{7}$$

where $\boldsymbol{m}(t)$ is a set of image measurements and $\boldsymbol{a}$ is a set of parameters, such as camera intrinsics, that represents additional knowledge about the system. $\boldsymbol{s}$ is the measured values of image features/object locations, such as pixel coordinates in the picture frame, and $\boldsymbol{s}^*$ is the desired values of image features/object locations.

The change of feature error $\dot{\boldsymbol{e}}$ and camera velocity $\boldsymbol{v_c}$ is related by *interactive matrix* $\boldsymbol{L}$:

$$\dot{\boldsymbol{e}} = \boldsymbol{L}\boldsymbol{v_c} \tag{8}$$

For IBVS, which is done in 2D image space, 3D points $\boldsymbol{X} = (X, Y, Z)$ are projected onto 2-D images with coordinates $\boldsymbol{x} = (x, y)$:

$$x = X/Z \tag{9}$$
$$y = Y/Z \tag{10}$$

which creates an interactive matrix for 2D image based servoing:

$$\boldsymbol{L} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}$$

With the interactive matrix, camera velocity can be estimated by:

$$\boldsymbol{v_c} = -\lambda \boldsymbol{L}^+ \boldsymbol{e} = -\lambda \boldsymbol{L}^+ (\boldsymbol{s} - \boldsymbol{s}^*) \tag{11}$$

where $L^+$ is the Moore-Penrose pseudo-inverse of $L$:

$$\boldsymbol{L}^+ = (\boldsymbol{L}^T \boldsymbol{L})^{-1} \boldsymbol{L}^T \tag{12}$$

The final control law is set as a robot velocity effort $v_s$ opposite to the camera velocity $v_c$ because the target moves in the opposite direction of the camera in the image frame:

$$\boldsymbol{v_s} = -\boldsymbol{v_c} \tag{13}$$

Notice that the interactive matrix depends only on $x$ and $y$, that is the 2D pixel coordinate of the target, and $Z$ which is the depth of the target. In our system, $Z$ is measured as the size of the apriltag. Therefore, the IBVS measurement is independent of the exact 3D position of the target measurement, which is attractive to our system because the exact 3D camera registration is not required.

### A. IBVS Implementations for Automatic Box Pickup

The automatic visual servoing controller executes a box pickup in three phases. Phase 1 uses IBVS to move the robot to a position where the aprialtag has the same size as the green box shown in left side of Fig 5. The robot also need to position apriltag on the center purple line of the video frame after phase one, but not at the center. In phase 2, the robot adjusts its own height by changing the joint angles of the two "knee" joints so that the apriltag would lay at the center of the video frame where the green box is. After the robot reaches the optimal picking position when apriltag is at

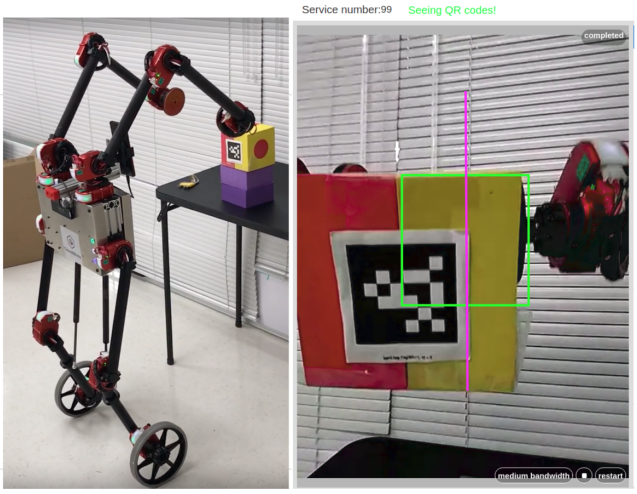


Fig. 5. **Image Based Visual Servoing using Fog Robotics: (Left) Igor Picking Up a Static Box; (Right) Image Based Visual Servoing (IBVS)** the purple central line and the size of the green box are used in the phase 1 of the IBVS. Phase 1 controls how to navigate the robot an optimal position. Phase 2 controls the optimal height of the robot. The robot changes its height by the two "knee" joints of the two legs so that the apriltag would fit directly into the green box in the center of the 2D image frame. **Videos: (1) Pickup a box from a table: `https://youtu.be/b0mr5GHHjBg` (2) Pickup a box from a human: `https://youtu.be/K9R4y2w1uPw`**

the center of the video, phase 3 begins. The robot controller commits to perform a box-pickup with a pre-defined, hard-coded, dual arm, grasping motion.

The size of the target green box encodes depth information ($Z$) of the target. We find this optimal size by teleoperating the robot to different positions that is close to the box. From locations, we select the position that has the highest box-pickup successful rate using the hard-coded grasping motion.

### B. Fog Robotic IBVS

Although we use simple apriltags for object recognition, we aim to anticipate the design of a deep-learning-based Fog Robotic visual system for robotic pickups. Therefore, to emulate the latency effects under such system, we deploy apriltag recognition in the cloud and use "heartbeat" protocol to stream apriltag's geometric locations to low-level controller via RCU. Together, we build a robust Fog Robotic IBVS controller for box pickups.

## VII. EXPERIMENTS AND RESULTS

With the "heartbeat" design, we are able to navigate the self-balancing robot reliably (see video) from the cloud-based teleoperation interface. We also pre-program arm actions such as clapping, and "disco" for human robot communications and entertainment (video: `https://youtu.be/1H1VEpkbG_E`).

We further hardcode a box pickup motion for the two arms, and attempt to pick up a box via the cloud-based teleoperation. However, even with a reliable teleoperation module and with pre-programmed pick up motions, we find it extremely difficult and inefficient to pick up a box using cloud teleoperation. We suspect that it is caused by the

lack of natural, immersive 3D visual perception for the teleoperator,

To quantify the observation, we perform two different teleoperation experiments with 10 trials each: 1) control Igor locally so that the operator can see the robot and the box; 2) control Igor from the cloud to pick up the box. In both cases, the box is positioned on the table in a stable location. The robot is about 2 meters from the object and faces the front of the box (see video `https://youtu.be/b0mr5GHHjBg`). We observe that local teleoperator (at least 2 meters away from the target) can perform box pickups much faster with a higher success rate than the cloud teleoperator (see table I)

After implementing the automatic IBVS module, we perform the same experiments and benchmark the automatic module with human in the loop. We allow the cloud operator to first teleoperate the robot as fast as possible to a location where apriltag is recognizable, which can be up to 20 degrees from the surface normal direction of the apriltag. Then, the human triggers the Fog Robotic IBVS module, allowing the robot to pick up the box automatically. We observe that the speed of this third case is on-par with human local teleoperation, but the reliability is even higher at 100% (see table I)

TABLE I
TELEOPERATION VS. AUTOMATIC BOX PICKUPS

| | Average Duration (s) | Success Rate |
|---|---|---|
| Local Teleop | 43 | 9/10 |
| Cloud Teleop | 340 | 4/10 |
| Auto Pickups | 46 | 10/10 |

Finally, we leverage the flexibility of visual servoing to perform a proximate human robot interaction (HRI) task. During the task, the human carries the box with an apriltag. They can show Igor the apriltag while moving around. Igor can recognize and localize the tag with a distance as far as 6 meters. As soon as the robot recognizes the apriltag, the teleoperator can release the robot so that it enters automatic mode. We observe that as long as the robot can recognize the apriltag and the box is in a reachable height for the robot, the robot will follow the human around, and eventually pick up the box from the person with a high successful rate (see video: `https://youtu.be/K9R4y2w1uPw`)

## VIII. DISCUSSION AND FUTURE WORK

In this work, we take advantage of both intelligent cloud robotic platform and edge controller to build an intelligent Fog Robotic system that can perform human-compliant, automatic box pickups using visual servoing.

A "heartbeat" protocol with asynchronous communication is introduced to mitigate network latencies and variabilities effects on the dynamic hybrid self-balancing controller in Fog Robotics. However, the current "heartbeat" protocol is not perfect. There is an increased delay at the end of command signal that would cause a delayed reaction after the last command signal is received (see red arrow in Fig. 4). This imposes a significant safety concern, because even if the "heartbeat" time window is short, i.e. 250 ms in our case, the robot will not stop completely until after 250 ms plus the ramping down period. To compensate, we implement a sharper ramp function at the stop compared to the ramp function at the start, but 250 ms is the hard limit for the delay on the current system (shown as red arrow in Fig. 4).

Our future work includes a better modeling of package drops in asynchronous communication, so that we can build a probability model to measure variabilities of time intervals between packages. This way, we can further reduce this delayed reaction by adjusting the "heartbeat" window size based on the predicted time of last package.

Like other service robots, Igor needs to interact and cooperate with human beings. We demonstrate the advantages of visual servoing: (1) it requires no calibrations before each robotic task; (2) it can handle dynamic human robot interaction, such as following a human to pick up a box from that person. Our automatic system works even in unstructured environments when obstacles are present between the human and the robot. The self-balancing control and the compliant servos can correct themselves when small obstacles are encountered. The human box carrier or the cloud teleoperator can also help the robot avoid obstacles by guiding it to a path with more clearance.

One failure case is when the human carrier tricks the robot. It happens if the target is moved after the robot commits to the final phase of box picking, which is hard-coded.

In the future, we can program a more dynamic automatic object pickup so that the robot can pick up a moving object with a continuous motion, without hard-codings. We also plan to deploy deep-learning recognition pipelines such as mask-RCNN (cite) together with intelligent grasping systems such as dex-net [28] [7] [21] using Fog Robotic systems, so that it can guide both dynamic robots such as Igor and static robots such as YuMi [20] and HSR [31] to perform generalized, human compliant object pickups and manipulations.

## IX. ACKNOWLEDGMENTS

We thank members from the AUTOLAB– Ron Berenstein, Ajay Tanwani–for discussions, and members at Cloudminds– Arvin Zhang, Havelet Zhang, Mel Zhao–for their technical support. Special thanks Prof. Joseph Gonzalez for discussions on hybrid synchronous and asynchronous Systems. We thank Matthew Tesch, David Rollinson, Curtis Layton, and Prof. Howie Choset from HEBI robotics for continuous support, training, and discussion on the Igor self-balancing research robot.

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] I. F. of Robotics, *Introduction into Service Robots*, 2016 (accessed September 15, 2018). [Online]. Available: https://ifr.org/img/office/Service_Robots_2016_Chapter_1_2.pdf

[2] ——, *Executive Summary World Robotics 2017 Service Robots*, 2017 (accessed September 15, 2018). [Online]. Available: https://ifr.org/downloads/press/Executive_Summary_WR_Service_Robots_2017.pdf

[3] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

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

[7] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[8] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[9] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint arXiv:1603.02199*, 2016.

[10] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2786–2793.

[11] A. X. Lee, S. Levine, and P. Abbeel, "Learning visual servoing with deep features and fitted q-iteration," *arXiv preprint arXiv:1703.11000*, 2017.

[12] 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," *learning*, vol. 12, p. 14.

[13] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[14] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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, .

[20] N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, and K. Goldberg, "A cloud robot system using the dexterity network and berkeley robotics and automation as a service (brass)," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1615–1622.

[21] P. Li, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Goldberg, "Dex-net as a service (dnaas): A cloud-based robust robot grasp planning system."

[22] J. K. K. G. Ajay Kumar Tanwani, Nitesh Mor, "A fog robotics architecture for distributed learning of surface decluttering," *Robotics and Automation Letters (under review)*.

[23] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[24] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.

[25] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[26] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[27] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[28] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)*, 2016.

[29] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 23–30.

[30] OpenAI, :, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning Dexterous In-Hand Manipulation," *ArXiv e-prints*, Aug. 2018.

[31] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," *arXiv preprint arXiv:1703.09327*, 2017.