A Cloud Robot System Accessing Learned Grasps from Dexterity Network 1.0 and Berkeley Robotics and Automation as a Service (Brass)

Nan Tian*,[1,2] Matthew Matl*,[1] Jeffrey Mahler,[1] Yu Xiang Zhou,[2,3] Samantha Staszak,[1] Christopher Correa,[1] Steven Zheng,[1,2] Qiang Li,[2] Robert Zhang,[2] Ken Goldberg[1]

***Abstract*— Robotics and Automation as a Service (RAaaS), the robotics-equivalent of Software as a Service (SaaS), can serve as an important component in a cloud robotics framework by avoiding complex software installation and maintenance, reducing application development time, and facilitating sharing of data. However, RAaaS may introduce network latency and security issues. This paper describes an implemented RAaaS system architecture and reports on physical grasping experiments. The system uses Berkeley RAaaS Software (Brass) to remotely host an instance of Dex-Net 1.0, a robust grasp-planning system that samples grasps on 3D object meshes and computes stochastic robustness metrics for each grasp. The system links a local ABB YuMi human-safe robot with Brass via a cross-border, secure, and low-latency network provided by Cloudminds, Inc. We study grasp performance under this architecture by programming the YuMi to grasp and lift a set of non-standard, asymmetric chess pieces. Results suggest that the RAaaS system can provide significant improvements in grasp robustness with reasonable mean network latency times of $30\,\mathrm{ms}$ and $200\,\mathrm{ms}$ for servers 500 and 6000 miles away from the robot, respectively.**

I. INTRODUCTION

“Cloud Robotics and Automation” describes robots and automation systems that share data, re-use code, and perform necessary but expensive computation on remote cloud servers. It builds on emerging research in Cloud Computing, Deep Learning, and Big Data and government/industry initiatives such as the “Internet of Things”, “Industry 4.0”, and “Made in China 2025”.

In earlier work [1], we proposed the concept of Robotics and Automation as a Service (RAaaS), the robotics-equivalent of Software as a Service (SaaS) – a model in which software components are hosted on central cloud servers and made available to end users and robots over the internet. Other internet-based software services, such as GMail and Google Docs, facilitate software installation, allow for centralized software maintenance and upgrades, and promote the sharing of data and the parallelization of computation within cloud datacenters. However, these benefits come at the cost of network latency and security concerns.

The AUTOLAB at Berkeley (`automation.berkeley.edu`)

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the AMP Lab, BAIR, and the CITRIS “People and Robots” (CPAR) Initiative: http://robotics.citris-uc.org.

[1]University of California, Berkeley, USA; {`neubotech, mmatl, jmahler, samstaszak, goldberg`}`@berkeley.edu`

[2]Cloudminds Technology Inc.;`robert@cloudminds.com`

[3]CMU School of Computer Science;

*Both authors contributed equally to this work

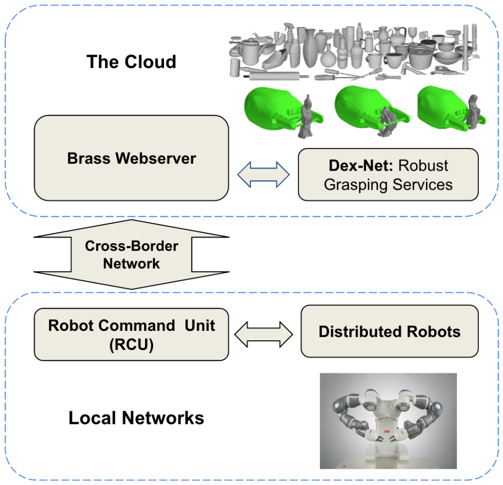

Fig. 1. **Block Diagram of BRASS (Berkeley RAaaS Software) Framework: (Top)** Brass webserver and associated robotics services, like Dex-Net. **(Center)** Cross-Border Network that connects Brass to local systems. **(Bottom)** Local robot command unit (RCU) for querying Brass and controlling robots.

Under a RAaaS framework, similar benefits and drawbacks exist. Developers of robotics software would be able to access useful tools like path planning, object recognition, and robust grasp planning through web-based interfaces, which would reduce development time, allow end users to receive instant updates when algorithms are improved in the cloud service, and enable robots to take advantage of the significant computational and data storage resources that the cloud offers. However, network latency could hamper the execution of time-sensitive robotic tasks, and security becomes a larger concern – especially when physical robots are involved.

This paper reports on the architecture of an implemented RAaaS system and results from experiments that explore the costs and benefits of using such a system. Berkeley RAaaS (Brass) currently includes the Dexterity Network (Dex-Net 1.0), a robust grasp planning package. Dex-Net includes a database of over 10,000 3D object models and uses over 1500 cloud computing nodes to perform Monte-Carlo integration to pre-compute stochastical robustness properties for thousands of parallel-jaw grasps per object [2]. End-users can connect to Brass via a standard network or through the proprietary new Cloudminds cross-border, secure, low-latency network. Once a connection is established, a robot can retrieve hundreds of grasp candidates and their associated robustness metrics from Dex-Net to use as a guide for planning manipulation tasks. Because robust grasp locations are pre-computed and stored in the cloud, Dex-Net can empower robots with very limited memory and computing resources to manipulate the complex objects encountered in tasks such as warehouse order fulfillment and home decluttering.

As examples of parts that are non-trivial to grasp, we use a set of six asymmetric, non-standard chess pieces downloaded from Thingiverse [3] (see Figure 2, bottom). The experimental setup uses an ABB YuMi bilateral human-safe robot to manipulate the chess pieces, and the YuMi's controller accesses Dex-Net's grasp recommendation system through Brass to facilitate robust grasp planning.

Although it is of course possible to pre-fetch several hundred robust candidate grasps for each of the six unique chess pieces from Dex-Net and store them locally, it is not possible to pre-fetch the grasps for every object encountered in a warehouse or home. The six chess pieces thus serve as examples of parts that could be encountered and for which robust grasps would be requested over the cloud.

We analyze how using Dex-Net through Brass affects grasp reliability and repeatability when compared against local, hard-coded grasping movements along the x and y axes of the chessboard that do not take piece geometry into account. We compare network latency and variance for Brass in three locations: on site, 500 miles away in an Amazon EC2 instance, and 6000 miles away in China.

II. RELATED WORK

The concept of cloud robotics and automation can be traced back at least two decades to the advent of "Networked Robotics" [4]. In 1997, Inaba et al. described the advantages of using remote computing for robot control [5], and in 2001 the IEEE Robotics and Automation Society established the Technical Committee on Networked Robotics [6]. Work continued throughout the decade, and in 2009 the RoboEarth project envisioned the construction of "a giant network and database repository where robots can share information and learn from each other about their behaviour and environment" [7], [8]. This project developed cloud computing resources for generating 3D models of environments, speech recognition, and face recognition [9], and the idea of using the cloud for computation and data aggregation in robotics continued to pick up traction. In 2010, James Kuffner first used the term "Cloud Robotics" to describe the increasing number of robotics or automation systems that rely on remote data or code for effective operation [10]. Since then, a wide variety of models for connecting robots to the cloud have been developed, implemented, and tested [1].

Some, like RoboEarth's Rapyuta system [11], offer secure, optimized platforms in the cloud for offloading robotic computational tasks. These Platform as a Service (PaaS) systems do not offer robotic services explicitly, but instead make it easier for developers to push existing code into the cloud for parallelization. Adoption of PaaS systems is made easier by the ubiquity of ROS, the Robot Operating System [12]. Individual computational units in ROS are organized into nodes that communicate via the ROS messaging protocol,

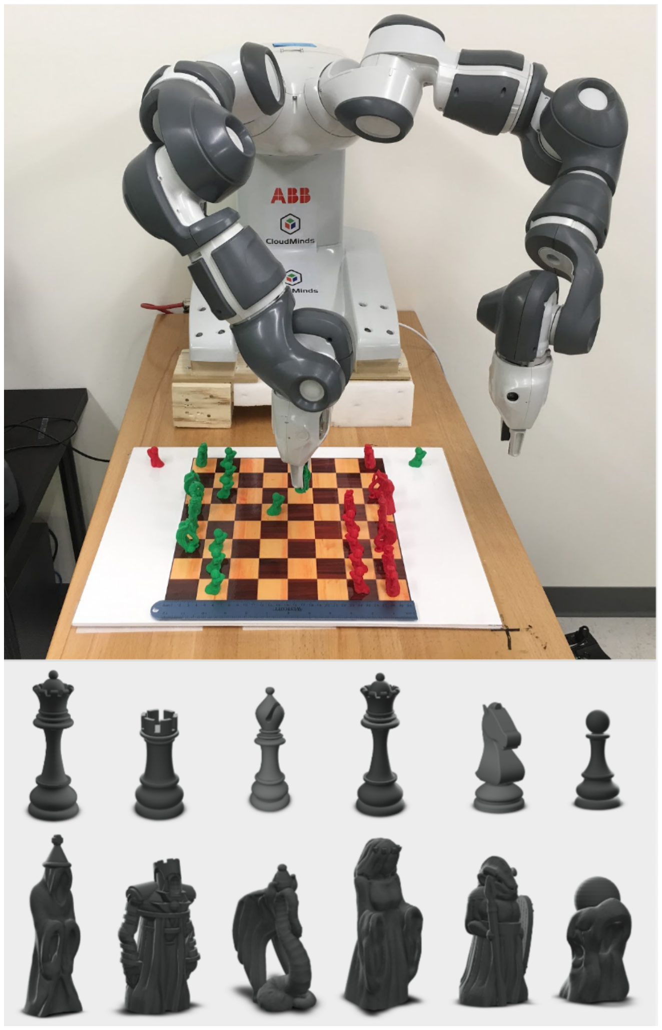

Fig. 2. **(Top)** The YuMi robot replaying the classic 1997 chess game where IBM's Deep Blue defeated world champion Garry Kasparov. **(Bottom)** CAD model renderings of standard chess pieces and the equivalent non-standard "Wizard" pieces.

and moving these nodes to a remote PaaS server is easy. However, these remote nodes are not available as shared services for public users.

On the other hand, some systems have adopted a SaaS-like model for robot motion planning. Vick et al. moved a robot motion controller into the cloud and used it for manipulation planning [13], and Zieliński et al. created a system that provides cloud-hosted task planning agents for exploratory robots [14]. Additionally, Bekris et. al explored the tradeoff between path quality and computational efficiency for a cloud-based motion planner for industrial grippers [15]. In the realm of robotic mapping, the DAvinci framework offered a highly-parallelized implementation of FastSLAM in the cloud and allowed many robots to share generated environment maps through a SaaS model [16].

Brass builds on ideas from these SaaS-style frameworks as well as related work on robotic grasping systems. In particular, Brass directly uses Dex-Net 1.0 [2] as a service and draws inspiration from Ben Kehoe's work on cloud-based robotic grasping, which used a variety of cloud-hosted services like the Google Object Recognition Engine, OpenRAVE, and the Point Cloud Library in an integrated object manipulation pipeline [17], [18]. Additionally, Brass is based on several ideas from Arjun Singh's dissertation on benchmarks for cloud robotics [19].

Our work is also closely related to robust grasp planning. For a review of grasping, see Bicchi and Kumar [20]. Early research on grasp planning focused on maximizing analytic quality metrics such as force closure [21] or the Ferrari-Canny metric (also known as epslion metric) [22]. However, these metrics depend on precise knowledge of geometry, material properties, contact locations, and surface normals, which may not be known due to imprecision in sensing and control. This motivated the development of robustness metrics, which measure the expectation of quality metrics under uncertainty in variables such as object pose and friction [17], [23], [24]. Since evaluating robustness may be computationally expensive, recent research has studied reducing the number of samples required, for example by using Multi-Armed Bandits (MAB) [25]. Recently, Mahler et al. [2] developed Dex-Net 1.0, a cloud-based dataset of over 10,000 3D models annotated with parallel-jaw grasps and robustness metrics, and showed that this dataset could be used to accelerate robust grasp planning with MAB. Another approach is learning a predictive model of grasp robustness from featurizations such as heightmaps [26] or depth images [27]. In this this work we access robust grasps computed by Dex-Net 1.0 over a network and study their success on a physical system.

III. SYSTEM DESIGN

Our experimental setup consists of the five primary components listed below (Figure 1):

1) A local dual-arm ABB YuMi robot. This robot receives commands and executes commanded motions to manipulate objects.
2) A robot command unit (RCU). The RCU plans chess moves, makes requests to Brass over the internet to retrieve suggested grasps, and sends motion commands to the robot.
3) A proprietary new cross-border network service. This network is global, secure, and provides low latency, and it is used to connect the local RCU to Brass.
4) Brass, a server that exposes a universal API to clients. Internally, Brass starts up instances of the services it offers, performs queries to each of those services when a web request arrives, and coordinates responses to clients.
5) An instance of Dex-Net 1.0, run within Brass. Dex-Net is a service for robust grasp planning that takes as input an RGBD image of an object, matches it to similar objects, and returns a set of pre-computed parallel-jaw grasps ranked by robustness. It currently includes over 2 million grasps for 10,000 object models [2].

Each of these components is described in more detail below.

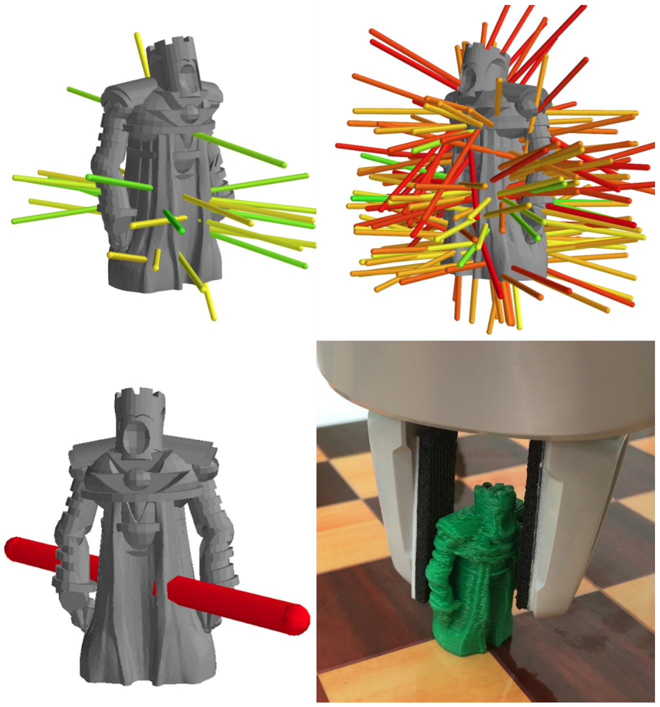

Fig. 3. **(Top)** Parallel-jaw grasp configuration candidates generated by Dex-Net ranked by robustness (probability of force closure). The left image shows the grasp axis for the 15 most robust grasps, at right are the 200 most robust grasps. **(Bottom)** The left image shows the most robust grasp for the rook piece, while the right shows that grasp being executed by the YuMi robot gripper.

A. *Dual-Arm YuMi Robot*

In experiments, an ABB YuMi was used as the local robot. This robot has a pair of 7-DOF arms, each of which can move at a rate of $1500\,\mathrm{mm\,s^{-1}}$ with sub-millimeter repeatability and can carry a payload of up to $250\,\mathrm{g}$. Each arm is equipped with a parallel-jaw gripper whose jaws are $5\,\mathrm{cm}$ in length and can open to $10\,\mathrm{cm}$ apart. Furthermore, the robot is human-safe in compliance with ISO/TS 15066, which means that its controller will immediately stop each arm within milliseconds of contacting a human or any other part of the robot itself. The YuMi was designed for manipulating small parts in collaboration with humans, which makes it an excellent candidate for applications with human-robot interaction (HRI) [28] or humans in proximity.

In our experimental setup, we use the YuMi's ethernet service port to stream RAPID commands for controlling its arms, and we use YuMi's built-in inverse kinematic solver for planning joint angles.

B. *Robot Command Unit (RCU)*

The robot command unit, or RCU, is the mid-level software component that creates task plans and then utilizes Brass to plan lower-level commands for controlling the YuMi robot. The RCU's task planner parses a series of chess moves from a PGN-format chess game file and then begins to execute them sequentially. Throughout the game, the RCU keeps track of each piece's position on the chess board.

When a move is processed, the RCU identifies the target piece and then queries Dex-Net via Brass to retrieve candidate grasps for that piece. These grasps are parametrized by a center point in $\mathbb{R}^3$ that lies halfway between the target locations of two gripper jaws and an axis in $\mathcal{S}^2$ that points from one jaw tip to the other. Additionally, each grasp is returned with its probability of force closure under uncertainty in object pose, gripper pose, and friction, which serves as our primary quality metric for each grasp.

To reduce the probability of collisions with other pieces, the RCU constrains the YuMi to grasp pieces from directly above so that the gripper jaws are perpendicular to the table. These grasps can be represented as a rotation of the jaws around the z-axis and a translation of the gripper. From this parametrization, the RCU can directly command the YuMi's built-in motion planner to execute the grasp, lift the piece, and place it in its target location.

C. *Cross-Border Network Service*

To connect a local RCU to Brass and its services, we use Cloudminds, a new proprietary, secure, low-latency, global network designed to send files rapidly and securely across great distances. This network differs from traditional networks in two primary ways. First, it has control over all routers and access points within its range of service. Second, the network requires every user to authenticate before use so that every packet can be traced back to its owner.

This network has demonstrated a 10 to 100 times increase in throughput for long-distance file transfers when compared to traditional global networks. However, robotics applications are generally more sensitive to latency, reliability, and security than throughput. In our chess-playing experiments, typical Brass packets containing Dex-Net grasp recommendations are roughly $33\,\mathrm{kB}$ in size, so throughput is not a huge concern. However, latency directly affects the responsiveness of the robot.

The Cloudminds' Cross-Border Network achieves significant improvements in latency by optimizing the number of hops and distance traveled during routing. The system is also more reliable, as it uses advanced traffic control to handle congestion and packet drops. Furthermore, mandatory authentication makes it much more secure and more resilient to router attacks than traditional networks. Any denial-of-service attack launched on one of the network's routers will be ineffective, as packets from unauthorized sources are simply dropped.

D. *Brass*

Brass is a custom experimental webserver that presents a generic API for robotic services over the internet. By presenting a uniform, well-defined API, Brass abstracts away the complexities of its internal services and enables end-users to build code against a stable interface. Internally, Brass starts up an instance of each of its services and connects to them using their own built-in communication protocols. When a request arrives, Brass queries these services, aggregates results, and returns them to the client in a well-defined format. Each request contains an object identifier, and Brass will return either a raw list of Dex-Net's suggested grasps for that object, sorted by their probability of force closure, or a single grasp that Brass selects from that set by applying filters specific to the chess-playing task.

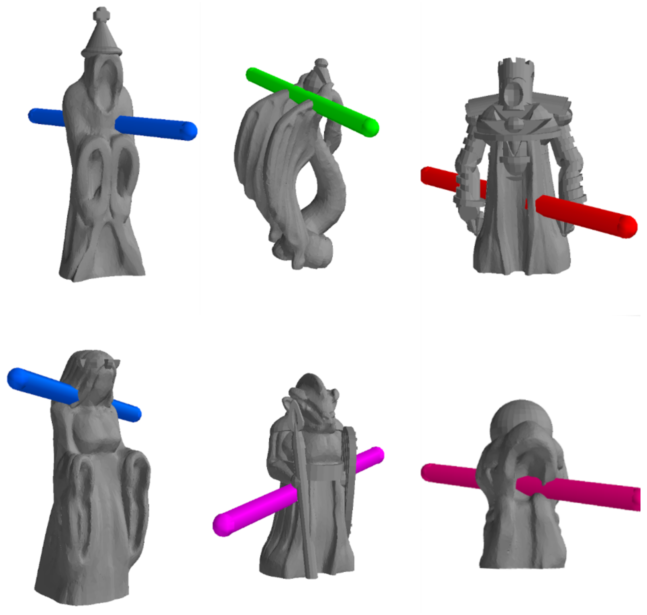

Fig. 4. Visualization of optimal Dex-Net grasps selected by Brass. (top) king, bishop, rook (bottom) queen, knight, pawn. When these grasps are executed, the axis shown is projected onto a plane that contains its center point and is parallel to the table. The object frame is defined as negative y-axis is the direction that the pieces are facing, and positive x-axis is from right of the piece to the left.

This filter eliminates grasps where the grasp axis is not nearly parallel to the game board and prioritizes grasps whose center point is close to the vertical line that passes through the object's center of mass to minimize gravitational torque on the piece when it is lifted. The filter also removes grasps that are too close to the table as they risk collisions between the top of the piece and the palm of the YuMi gripper. The steps in this filtering process are described as follows:

1) Filter out grasps whose probability of force closure is below 30% of the highest probability of force closure over all grasps.
2) Filter out grasps that are within $10\,\mathrm{mm}$ of the work-surface to avoid collisions with the YuMi gripper.
3) Filter out grasps whose center point is more than $5\,\mathrm{mm}$ from the line that runs through the piece's center of gravity perpendicular to the table.
4) From the remaining grasps, choose the grasp whose axis is most parallel to the table.

Application of these filtering steps led to the selection of grasps shown in Figure 4.

When accessing Brass, the RCU makes HTTP requests and the resulting grasps, parametrized by a center point and a grasp axis vector, are returned to the RCU as a JSON structure.

E. Dexterity Network

Dex-Net 1.0 is a robust cloud-based grasp-planning service [2]. Given a particular object's 3D mesh model, Dex-Net computes all stable poses for the object on a planar work-surface using static analysis [29]. Then, Dex-Net generates thousands of grasp candidates for each registered gripper type using a multi-armed bandit model to leverage prior grasps for increased sampling efficiency. Individual grasps are represented as a center point and a 3D vector that indicates the axis between the jaws. Finally, robustness metrics, such as probability of force closure [30] and expected Ferrari-Canny quality [31] [32] are computed for each grasp under uncertainty in object pose ξ, gripper pose ν, and friction coefficient γ, and these metrics are stored with each grasp for later access.

The grasp sampling and metric calculation processes are computationally expensive, requiring 10-15 minutes on a standard desktop, so Dex-Net takes advantage of cluster computing and uses 1500 nodes to pre-compute grasps. Dex-Net currently stores data for over 10,000 unique objects, which amounts to over 2.5 million grasp candidates. Dex-Net can then serve these pre-computed grasps over the internet via Brass in small, kilobyte-sized packages. This means that nearly any system – even ones constrained heavily in both memory and computational power – can use Dex-Net to plan robust grasps for thousands of objects.

In this work we model object pose as a zero-mean Gaussian over SE(3) $\xi = \exp(v^\wedge)$ [2] with rotational uncertainty of 0.01 radians and positional uncertainty of $1.0cm$, where v is a member of the Lie Algebra. We modeled gripper pose as a zero-mean Gaussian over SE(3) with rotational uncertainty of 0.001 radians and positional uncertainty of $0.1cm$, and we modeled $\gamma \sim \mathcal{N}(0, 0.1)$, truncating γ to be in $[0, 1]$ using rejection sampling. The values were chosen based on the robot specification and the material properties of the gripper and objects.

For these experiments, the 3D mesh model for each chess piece was added to Dex-Net and grasps were pre-computed before the experiments were initiated.

IV. EXPERIMENTS AND RESULTS

To evaluate the use of Dex-Net via Brass, we set up a series of experiments to determine how using Dex-Net affects grasp robustness and how serving Dex-Net grasps via Brass from remote servers affects latency.

A. Individual and Sequential Grasp Success With and Without Dex-Net

Our first set of experiments was designed to evaluate if access to Dex-Net via Brass improves grasp quality and robustness. The experiments are divided into two scenarios:

1) Dex-Net is not used. Instead, grasps are identical for all pieces, with axes between the gripper jaws placed parallel to either the x-axis or y-axis (y-axis is in the direction where each piece is facing, whereas x-axis runs from left to right through each piece).
2) Dex-Net is used via Brass to provide robust grasp candidates. Brass performs grasp filtering to provide the RCU with a single target grasp.

For each scenario, we ran two experiments. First, we attempted to grasp each piece 50 times and move it from a start square to a target square. This experiment was designed to characterize the probability of correctly manipulating each individual piece under each of the listed scenarios. An attempt was considered successful if the piece was successfully lifted from its start square and placed in the target square without being dropped or knocked over. After each test, the target piece was re-registered in its start square, with its center of mass placed directly above the center of the square. The results from these experiments are displayed in Table I.

TABLE I
GRASP SUCCESS RATES (PERCENTAGE) OVER 50 ATTEMPTS

	Hardcoded x-axis	Hardcoded y-axis	Dex-Net
King	94	100	100
Queen	84	100	98
Rook	96	98	100
Bishop	98	72	98
Knight	100	100	100
Pawn	94	76	100

TABLE II
AVERAGE NUMBER OF CONSECUTIVE SUCCESSFUL MOVES AND (PARENTHESIS) NUMBER OF INTERRUPTIONS WITHIN 50 CHESS MOVES

	Hardcoded x-axis	Hardcoded y-axis	Dex-Net
King	12.5 (3)	50.0 (0)	50.0 (0)
Queen	5.0 (9)	50.0 (0)	50.0 (0)
Rook	16.7 (2)	25.0 (1)	25.0 (1)
Bishop	25.0 (1)	3.1 (15)	25.0 (1)
Knight	50.0 (0)	50.0 (0)	50.0 (0)
Pawn	12.5 (3)	3.9 (12)	25.0 (1)

Next, we attempted to play through 50 consecutive moves with each piece. Whenever a grasp failed, we re-registered the piece in its target square before allowing the sequence to continue. The mean number of moves between failures and the total number of failures during each 50-move sequence are shown in Table II. For a Bernoulli random variable with probability of success p, if all trials are independent, the number of expected consecutive successful trials is $1/(1-p)$. Our data is somewhat consistent with this model but there is some dependency between consecutive grasps since small errors in part position and orientation accumulate.

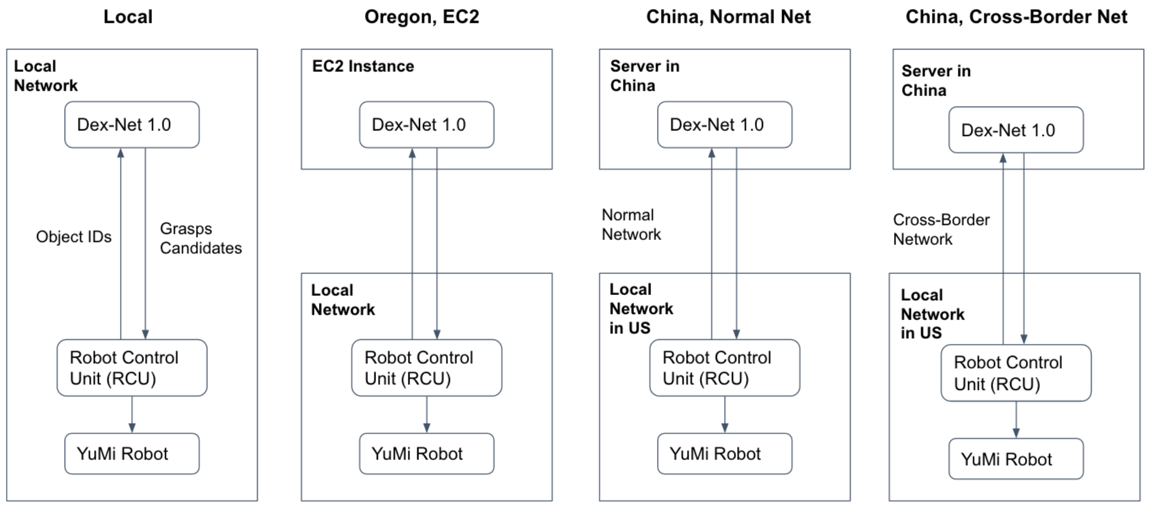

Fig. 5. Block diagrams for four experimental network scenarios. In case 1, both Dex-Net and the RCU are served locally. In case 2, Dex-Net is served from an Amazon EC2 instance in Oregon and is accessed over an ordinary network. In case 3, Dex-Net is hosted on a server in China and is accessed via an ordinary network. In case 4, Dex-Net is also hosted on a server in China and is accessed via the Cross-Border Network.

TABLE III
ROUND-TRIP TIME TO RECEIVE GRASPS FROM DEX-NET

	Mean (ms)	Variance (ms)
Local	0.11	0.0011
Oregon, EC2	31.50	0.0018
China, normal net	303.10	54.2180
China, cross-border net	197.03	0.3239

B. *Chess Movement Time vs. Network Latencies*

Our next set of experiments was designed to measure the effect of moving Brass and Dex-Net away from the robot and hosting them on remote servers. These experiments were performed for four primary scenarios (see Figure 5):

1) Brass, Dex-Net, and the RCU are all deployed on a local machine directly connected to the YuMi.
2) Brass and Dex-Net are deployed on an Amazon EC2 instance in Oregon, while the RCU remains directly connected to the robot. An ordinary ISP's network is used for packet transportation.
3) Brass and Dex-Net are deployed on a server in China, and the RCU is still hosted locally. An ordinary ISP's network is used for packet transportation.
4) Brass and Dex-Net are deployed on a server in China, and the RCU is still hosted locally. Cloudminds' Cross-Border Network is used for packet transportation.

For each scenario, we benchmarked the round-trip time to perform a Dex-Net query via TCP/IP. Each query contained the name of a particular piece, and each response contained a raw list of Dex-Net's pre-computed grasps for that piece in JSON format. Responses were generally around $33\,\mathrm{kB}$ in size. One hundred trials were performed per scenario, and the mean and variance for each are shown in Table III.

V. DISCUSSION AND FUTURE WORK

Grasp performance for three of the chess pieces was excellent across all methods, but grasp performance for the pawn, bishop, and the rook was significantly better with Dex-Net than without. These experiments suggest that utilizing Dex-Net via Brass transforms the system from nearly unusable to fairly robust at a relatively mild cost in terms of network latency.

However, this cost must be explored, since network latency is an important consideration for cloud-enabled robots. For time-sensitive on-line and real-time applications, large latencies for critical services can render a system unresponsive or unstable. For example, if an assembly-line robot needed to access Dex-Net grasps for objects moving on a conveyor belt, a $200\,\mathrm{ms}$ network delay could significantly impede performance. Fortunately, the Amazon EC2 configuration demonstrated $32\,\mathrm{ms}$ latency with relatively low variance (see Figure III). A RAaaS service placed in this datacenter in Oregon is capable of supporting $30\,\mathrm{Hz}$ control signals for physical robots in San Francisco, which is adequate for most on-line applications and some real-time robotic tasks. If the RAaaS server is placed in China instead, we can achieve $197\,\mathrm{ms}$ latencies with low variance if the cross-border network is used. Under this setup, the system is capable of supporting $5\,\mathrm{Hz}$ control signals, which is inadequate for real-time applications but should be sufficient for applications without tight real-time constraints.

In many such applications, however, robotic movement times will often dominate over network latencies. For example, when playing through a full 52-move game with the YuMi, moving Brass and Dex-Net from a local machine to a server in China only increased the average time per movement from $4.0\,\mathrm{s}$ to $4.3\,\mathrm{s}$. Furthermore, the network latency could be hidden entirely by requesting grasp suggestions for the next move during the execution of the current one.

If latency is dominated by robot motion time, using a RAaaS system is essentially free in terms of performance cost, and even if latency-hiding is impossible, the increased robustness and reduced development time brought by RAaaS systems is often worth the small increase in latency. In summary, while RAaaS services may not be useful in all real-time applications, many applications with looser constraints could benefit greatly from using a RAaaS framework with very little downside.

In future work, we will perform experiments to test how well the Brass system performs when parts are moving on a conveyor belt to further evaluate the effects of network latency. We will also explore Cloud-based object identification and pose detection, where parts are presented at random and RGBD images are uploaded to a remote perception server that identifies the object, position, and orientation before selecting grasps. We will also introduce other robot services such as path planning and calibration into the Brass framework.

ACKNOWLEDGMENTS

We thank all members of the AUTOLAB and the student interns at Cloudminds – Shao Zhou, Aashna Garg, and Michael Yu – for their advice and contributions during system setup. We also thank Ben Kehoe, Pieter Abbeel, and Arjun Singh for discussions about Cloud Robotics and James Kuffner of Toyota Research and Kai Kohlhoff and Torsten Kroeger of Google. The authors were supported in part by the U.S. National Science Foundation under NRI Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems and by funding from Google, Siemens, and Cloudminds. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Sponsors.

REFERENCES

[1] 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.

[2] 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.

[3] Robots versus wizards chess set. [Online]. Available: http://www.thingiverse.com/thing:351119

[4] K. Goldberg and R. Siegwart, *Beyond Webcams: An Introduction to Online Robots*. MIT Press, 2002.

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

[6] Ieee networked robots technical committee. [Online]. Available: http://www-users.cs.umn.edu/~isler/tc/

[7] What is roboearth? [Online]. Available: http://www.roboearth.org/what-is-roboearth

[8] M. Waibel, M. Beetz, J. Civera, R. dAndrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo *et al.*, "A world wide web for robots," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[9] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "The roboearth language: Representing and exchanging knowledge about actions, objects, and environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1284–1289.

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

[11] 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.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[13] A. Vick, V. Vonásek, R. Pěnička, and J. Krüger, "Robot control as a servicetowards 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.

[14] C. Zieliński, W. Szynkiewicz, M. Figat, M. Szlenk, T. Kornuta, W. Kasprzak, M. Stefańczyk, T. Zielińska, and J. Figat, "Reconfigurable control architecture for exploratory robots," in *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on*. IEEE, 2015, pp. 130–135.

[15] K. Bekris, R. Shome, A. Krontiris, and A. Dobson, "Cloud automation: Precomputing roadmaps for flexible manipulation," *IEEE Robotics & Automation Magazine*, vol. 22, no. 2, pp. 41–50, 2015.

[16] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3084–3089.

[17] 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.

[18] B. Kehoe, "Cloud-based methods and architectures for robot grasping," Ph.D. dissertation, Univ. of California, Berkeley, 2014.

[19] A. Singh, "Benchmarks for cloud robotics," Ph.D. dissertation, Univ. of California, Berkeley, 2016.

[20] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *ICRA*. Citeseer, 2000, pp. 348–353.

[21] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[22] C. Ferrari and J. Canny, "Planning optimal grasps," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 2290–2295.

[23] J. Kim, K. Iwamoto, J. J. Kuffner, Y. Ota, and N. S. Pollard, "Physically based grasp quality evaluation under pose uncertainty," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1424–1439, 2013.

[24] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 557–562.

[25] M. Laskey, J. Mahler, Z. McCarthy, F. T. Pokorny, S. Patil, J. Van Den Berg, D. Kragic, P. Abbeel, and K. Goldberg, "Multi-armed bandit models for 2d grasp planning with uncertainty," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015, pp. 572–579.

[26] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4304–4311.

[27] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," *arXiv preprint arXiv:1608.02239*, 2016.

[28] *ABB YuMi Datasheet*, ABB Group, 2015.

[29] J. Wiegley, A. Rao, and K. Goldberg, "Computing a statistical distribution of stable poses for a polyhedron," in *In 30th Annual Allerton Conf. on Communications, Control and Computing*, 1992.

[30] J. Weisz and P. Allen, "Pose error robust grasping from contact wrench space metrics," in *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)'12*, 2012.

[31] J. Kim, K. Imwamoto, J. Kuffner, Y. Ota, and N. Pollard, "Physically based grasp quality evaluation under uncertainty," in *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)'12*, 2012.

[32] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)'92*, 1992.