CORE: A Cloud-based Object Recognition Engine for Robotics

William J. Beksi, John Spruth and Nikolaos Papanikolopoulos

***Abstract*— An object recognition engine needs to extract discriminative features from data representing an object and accurately classify the object to be of practical use in robotics. Furthermore, the classification of the object must be rapidly performed in the presence of a voluminous stream of data. These conditions call for a distributed and scalable architecture that can utilize a cloud computing infrastructure for performing object recognition. This paper introduces a Cloud-based Object Recognition Engine (CORE) to address these needs. CORE is able to train on large-scale datasets, perform classification of 3D point cloud data, and efficiently transfer data in a robotic network.**

I. Introduction

As robots become more and more prevalent in household and workplace environments, the recognition of objects plays a fundamental role in the overall ability of an intelligent machine. Object recognition is defined as the extraction and classification of meaningful representations (features) from high dimensional data. The data may take the form of images, videos, and 3D point clouds. Performing object recognition is challenging since classification is performed in a high dimensional space where large intra-class variance may exist. The object recognition task is further challenged by a data space that changes with each new viewpoint and the fact that objects can be deformable entities.

3D point cloud data provided by low-cost RGB-D sensors [1], has allowed for the collection of feature rich datasets. These widely available sensors provide synchronized color and depth images and are increasingly being used in robotic applications such as object detection and classification. Although these datasets have opened new avenues of research, it remains undetermined as to which feature descriptors and machine learning classifiers provide optimal object recognition performance. However, current research suggests that robotic applications can benefit by using different descriptors and classifiers in varying situations.

The data processing steps required to perform object recognition are computational demanding in terms of memory and processor cycles. Furthermore, robots may not have the necessary on-board resources to perform object recognition tasks. Under these conditions, a robotic network with access to remote computing facilities can make use of vast computational resources and data. The idea of a cloud-based object recognition infrastructure for groups of robots enabled with RGB-D point cloud capture ability has received little attention.

The authors are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA. Emails: {beksi, sprut006, npapas}@cs.umn.edu.

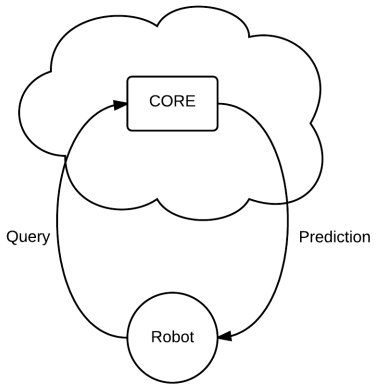

Fig. 1. An overview of the cloud-based object recognition engine (**CORE**). Networked robots send 3D point cloud data of an object to a remote server followed by a query message. The object recognition engine responds with a prediction message containing the label of the classified object.

To address the problem of performing object recognition tasks among robots that lack sufficient on-board resources we propose **CORE**, a distributed, modular, and scalable software architecture that is able to effectively execute data transfers in a robotic network, Figure 1. **CORE** lives in the Robot Operating System (ROS) [2] ecosystem. It wraps existing data filters, feature descriptors, and segmentation techniques in the Point Cloud Library (PCL) [3], along with machine learning classifiers, without the necessity of writing additional source code. The goal of **CORE** is to leverage cloud computing facilities for the purpose of storing data, training classifiers, and performing computations offloaded by robots in a network.

The remainder of this paper is organized as follows. Related work is mentioned in Section II. In Section III, a description of the architecture and design choices are presented. The communication protocols used are described in Section IV. Use cases and performance are discussed in V. We conclude in Section VI with an outlook of ongoing and future work.

II. Related Work

Many specialized recognition systems have been created in the past in both academia and industry. Researchers often use their own feature descriptors with a suitable selection of machine learning classifiers in their work. Although comparisons are usually made to other descriptor and classifier combinations, there has been little incentive to merge descriptors and classifiers into a generalized framework. Motivated by

this situation, this work seeks to establish an open source cloud-based infrastructure for performing object recognition on 3D point cloud data.

Industrial object recognition systems are apt to be domain specific and limited to controlled environmental conditions (ambient lighting, sensor position, object position, etc.). For example, Cognex's machine vision library [4] for factory inspection. Google Goggles, a closed source black box 2D image recognition application, was used on the PR2 robot for the purpose of performing pose estimation and grasp selection of common household items [5].

Attempts at creating cognitive perceptual architectures have been made in the past such as the Cog project at MIT [6]. The STAIR Vision Library [7], developed at Stanford, provides a software infrastructure for computer vision, machine learning, and probabilistic graphical models to support the STanford AI Robot project. CMU has developed a system [8] that uses descriptors to match online stored models for object recognition and full pose registration from a single image for robotic manipulation.

A fast, robust, scalable, recognition infrastructure (ReIn) [9] proposes a robot perception system that can accurately identify objects and their poses at high speeds. It combines 2D/3D object recognition and pose estimation techniques as dynamically loadable plugins. A cloud-based object recognition system for 2D images is considered in [10]. RoboEarth [11] is an ambitious project with the aim of using the Internet to create an open source database that can be accessed and continually updated by robots around the world. The idea is to provide a powerful feed forward to any robot's 3D sensing, acting, and learning capabilities.

III. Architecture and Design Choices

The architecture of **CORE** is composed of three main components. The first component allows for acquiring and storing large-scale data sets for the purpose of training machine learning classifiers. The second component consists of a modular system of feature descriptors and classifiers for performing an object recognition task. The third component functions to intelligently transfer data over a robotic network. The first two components reside remotely in the cloud while the third component is located on-board the robot. Submodules of the individual components are shown in Figure 2. Below we outline the details of each part.

A. Large-scale Object Training

In addition to extra processing power, a cloud infrastructure can provide robots with data that would otherwise not fit into on-board memory. Problems in object recognition deal with high dimensional data such as images, videos, and 3D point clouds. Such data can be used to construct training sets for the purpose of building machine learning models for object classification.

Object recognition tasks can be carried out using manually labeled training. However, to create a scalable object recognition engine, a sufficiently large amount of labeled training data is necessary to learn good classifiers. Lai *et al.* [12] investigated how to significantly reduce the need for manually labeled training data by leveraging datasets available on the Internet. There exist accessible large-scale 3D datasets such as the RGB-D object dataset [13], SHREC [14], 3DNET [15], and BigBIRD [16] that can be used for object training.

CORE can be configured to make use of RGB-D datasets. The pulled datasets are the basis for resource intensive classifier training. Classifier models are then used to facilitate object classification queries by robots pushing RGB-D point cloud data over the network. Robots can also contribute to the cloud datasets in the situation where a new object is discovered. In this scenario, a mobile robot can provide point clouds of varying viewpoints for classifiers within **CORE** to train on.

B. On Demand Machine Learning Classifiers and Feature Descriptors

The ability to load from an assortment of machine learning classifiers and feature descriptors, on demand, can be an asset to a robot seeking to perform object classification as certain classifier and descriptor combinations exhibit better performance on different tasks. **CORE** bundles together multiple machine learning classifiers and object feature descriptors which can be accessed according to the parameters of a classification query. The architecture currently makes use of 3D point cloud covariance descriptors [17] for their low dimensionality and high performance, making them desirable feature descriptors in a robotic network. However, **CORE** is designed to interoperate with existing feature descriptors in PCL and to allow researchers to easily integrate their own choice of machine learning classifier.

1) Support Vector Machine: Object classification can be performed using a support vector machine (SVM) [18]. Model building occurs in the cloud by training on large scale datasets as outlined in subsection III-A. Robots can then query the SVM model with captured point cloud data represented by a covariance descriptor. Additional details on the use of an SVM classifier with RGB-D covariance descriptors are given in [19].

2) Dictionary Learning: Dictionary learning classification consists of learning a set of dictionaries (one for each object) for a specified dataset in the cloud [20]. Robot queries are then performed against the dictionary sets in parallel. These dictionaries, readily transferred over a network, can also be pulled by robots for on-board storage. This allows object classification to continue in the event of a broken connection to the cloud. It also has the potential to enable sharing of disjoint dictionary sets among robots within communication range. Further information on dictionary learning using RGB-D covariance descriptors can be found in [21].

C. Point Cloud Culling

Robots equipped with RGB-D sensors can capture both color and depth images for performing object classification tasks. These sensors operate at high frame rates producing over 10 MB/s of data. This can easily lead to the saturation

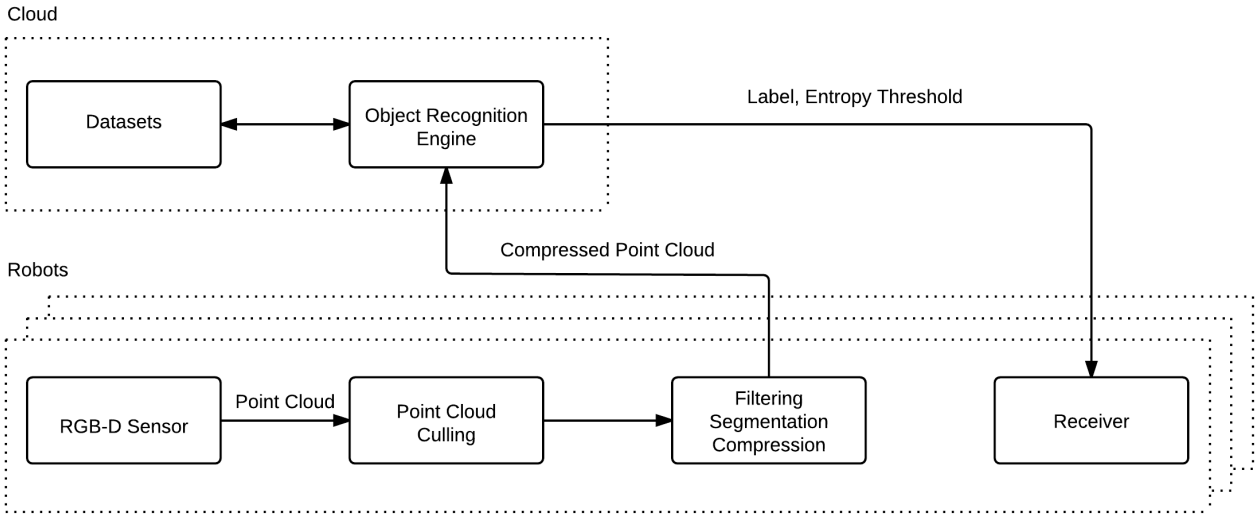

Fig. 2. The architecture of the cloud-based object recognition engine. Robots perform as much on-board processing (e.g. culling, filtering, segmentation) as possible before sending compressed point clouds to the object recognition engine. The object recognition engine can be configured to train classifiers on large-scale datasets, and passes the results of object recognition queries along with a transmission entropy threshold back to connected robots.

of a network due to the transmission of large amounts of generated data. In this subsection, we describe methods for controlling data transmission in a robotic network based on a remote computing infrastructure. This allows for a highly efficient transfer of RGB-D data from a client (robot) to a server (cloud). A design goal of **CORE** is to conserve network bandwidth and reduce latency while permitting a mobile robot the ability to recognize objects in the environment. Details on these algorithms can be found in [22].

1) Scene Entropy Calculation: In a cloud infrastructure, limited bandwidth and latency in the network can deteriorate the performance of an object recognition engine. Therefore, to reduce load on the network, point clouds are selectively culled prior to transmission from the robot to the cloud. This culling is performed by measuring the scene entropy of sequential point cloud frames.

The density of the voxels is used as a metric to detect changes in the overall information content when performing the entropy calculation. For example, movement of the robot and/or objects in the scene results in a change in entropy. Conversely, a static scene results in no change in entropy. This allows the transmission of sequential frames with a significant difference in entropy while discarding frames that have similar entropy values.

2) Robot Point Cloud Culling: A robot's on-board resources are used to perform filtering of the incoming point cloud frame from the RGB-D sensor. This filtering includes removing range outliers, and the estimation and extraction of a planar model in order to reduce the scene to the objects being classified. An octree data structure is used for downsampling and storing each filtered point cloud frame.

For each frame, the computed entropy is compared to a threshold. The current frame is dropped if the entropy value falls below the threshold. If the threshold is exceeded, then the octree of the frame is compressed [23] and sent to the cloud.

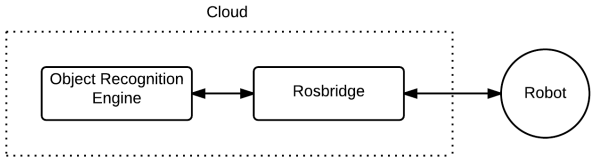

Fig. 3. Rosbridge serializes all robot to cloud communications over a single socket interface.

3) Remote Transmission Throttling: The cloud is used to throttle the robot's transmission of RGB-D data by employing a sliding window method. The server acknowledges the point cloud frames received by the robot by adjusting the entropy threshold. Correct classification of the objects in the current frame causes the server to increase the entropy threshold while incorrect labeling results in the server decreasing the robot's entropy threshold.

IV. Communication Protocols

In this section, we describe the communication protocols of **CORE** and provide an example of a data transfer session between a robot and the cloud.

A. Robot to Cloud Communication

Communication between processes within the **CORE** framework is done using UNIX sockets and utilizes rosbridge [24], a middleware abstraction layer that provides socket-based programmatic access to robot interfaces and algorithms

by ROS, Figure 3. Messages used for robot to cloud communication consist of four types. The first type of message initiates a connection between the robot and remote server. Once a robot receives an acknowledgment from the server it can then begin to send object point cloud data followed by a query, the second message type. The third type of message is the predicted object label from the server. Closure of the connection between the robot and the cloud is handled by the fourth message type.

All messages are transported as serialized ROS messages represented as JSON[1] objects. These messages have the following top-level structure

```
{ "op": "...", "service": "..."}
```

made up of an unordered collection of key/value pairs. The value of the *op* key is a string that denotes the type of operation to be instantiated:

- `advertise_service` - Advertises the availability of **CORE** services
- `call_service` - An RPC[2] invoked by a robot calling a specific **CORE** service
- `service_response` - A response from **CORE** to a robot RPC
- `unadvertise_service` - Stops advertising the availability of **CORE** services

The *service* key when calling and responding to a service consists of the string value `"query"` sent by a robot proceeded by the remote server issuing a predicted object label. The advertise and unadvertise operations are used internally between **CORE** and rosbridge.

B. Robot to Robot Communication

On the local network, robot to robot communication is performed by publishing and subscribing to ROS topics. Topics allow for unidirectional exchange of messages between nodes in ROS. TCP-based and UDP-based message transport mechanisms are supported. ROS multimaster support [25] enables communication independence among groups of robots. Using the multimaster setup, robots can share select data without saturating the network.

C. Sending Point Cloud Messages

CORE supports the transmission of point cloud data as binary blobs. The payload is encoded in the PNG (lossless data compression) image format. Messages containing compressed point clouds are published by setting the *op* key to "publish", assigning the topic name, and affixing of the encoded *msg* key and value pair:

```
{ "op": "publish",
  "topic": "/core_client_pointcloud/compressed"
  "msg": "Xc4p81H1cO8P+KsmbbHXJajErmah7bfo..." }
```

D. Basic Communication Example

In this subsection, we present a simple example of a robot interacting with the cloud-based server to carryout an object recognition task. The communication takes place as follows:

1) Initialize Connection: A robot begins the initialization process by sending an HTTP request to the remote rosbridge interface. Rosbridge then passes the robot information to the object recognition engine. Once the connection setup is finalized, an acknowledgement is sent back to the robot. At this point, the robot can advertise a ROS topic and begin sending point cloud messages.

2) Query: After beginning to send point cloud frames of an object to the server, the robot may perform a query. A query message takes the following form:

```
{ "op": "call_service",
  "service": "query",
  "args": "dl" }
```

In this example, the value "dl" associated with the *args* key specifies that the object recognition engine should use the dictionary learning classifier to perform the query.

3) Prediction: In response to a successful robot query, the server sends back the result of the object recognition engine:

```
{ "op": "service_response",
  "service": "query",
  "values": "apple_1",
  "result": "true" }
```

In the above message, the key *values* holds the predicted label identifying the object. A query match results in setting the *result* key to "true".

When point cloud data of an object cannot be matched by the selected classifier the server replies with the message:

```
{ "op": "service_response",
  "service": "query",
  "values": "",
  "result": "false" }
```

The value of the *values* key is left empty and the *result* key is set to "false".

4) Close Connection: A robot terminates the remote connection through an HTTP request to rosbridge. Rosbridge acts by notifying the object recognition engine of the pending disconnection which in turn responds by releasing held resources and unadvertising services.

V. Use Cases and Performance

A. Use Cases

The first use case involves multiple robots connected to the cloud with the objective of carrying out object recognition tasks, Figure 4. Initially, each robot connection is handed off to the object recognition engine which allocates resources. Incoming point cloud messages are queued, and then processed in FIFO order in the cloud. Object queries made by robots using a requested classifier are performed against the buffered point cloud data. A response message containing the predicted object label is sent back to the requesting robot.

The second use case considers the event of a slow, noisy, or unavailable connection to the cloud. In this scenario, robot to robot communication permits object recognition tasks to continue on the local network. Using the dictionary learning classifier, dictionaries can be created by robots when they encounter new objects. The current architecture provides for the potential sharing of dictionaries within a group of robots, Figure 5. This setup consists of robots publishing

[1]JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write and for machines to parse and generate.

[2]RPC is an inter-process communication protocol that allows client and server programs to communicate.

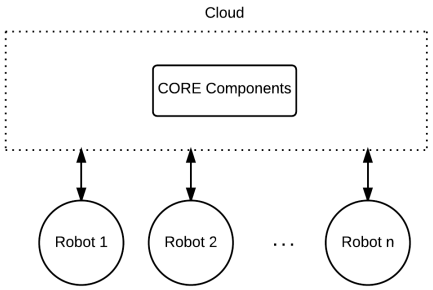

Fig. 4. Use case 1: Multiple robots connect to the cloud to make use of computationally demanding and data intensive object recognition services.

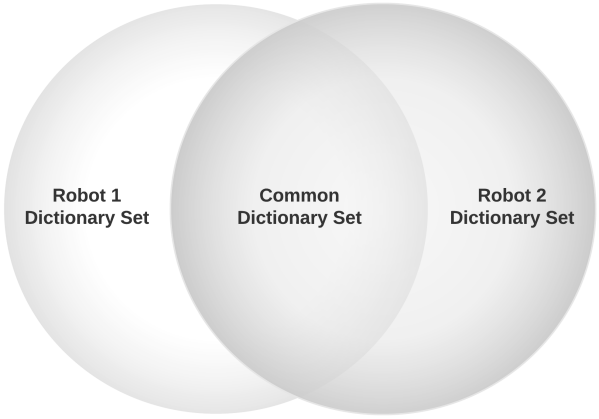

Fig. 5. Use case 2: When no cloud connection is available, robots may transfer the symmetric difference of their dictionary sets thus enhancing their object recognition capabilities.

ROS messages that contain the MD5 hashes of their on-board dictionaries. Subscribed robots may initiate a transfer of dictionaries missing from their own set resulting in the expansion of the overall knowledge pool of the group. New dictionaries can later be pushed to the cloud when a reliable connection becomes available.

B. Performance Evaluation

This subsection presents the experimental results of the measured round-trip times when transmitting point cloud data in **CORE**. The communication path consists of the University of Minnesota campus network and CloudLab[3] [26], a large-scale distributed infrastructure that allows researchers to construct clouds and test cloud applications. The robot was a ROS client with a wireless connection on the campus network, and the object recognition engine and rosbridge interface were deployed on the Wisconsin Cloudlab cluster. A profile (disk image) that captures the entire cloud environment has been created and shared within CloudLab under the project 'core-robotics'. It consists of one x86 node running Ubuntu 14.04 with ROS Indigo installed.

[3]CloudLab: www.cloudlab.us

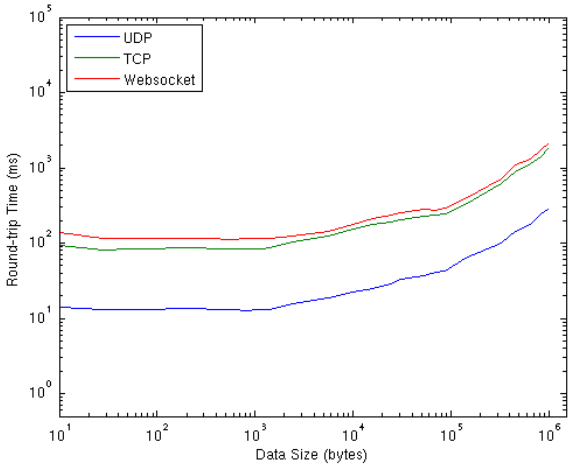

Fig. 6. Round-trip times (RTT) using the UDP, TCP, and Websocket protocols for transferring compressed point cloud data from a robot to the cloud. UDP has the lowest RTT; however, there is no guarantee all messages will arrive at the server, in order, with no duplicates. The RTT is higher for TCP and Websocket, but these protocols provide reliable data transmission. The Websocket protocol has the added advantage of allowing for an interactive communication session between a robot and the cloud.

To evaluate the round-trip times, messages containing compressed point clouds of varying sizes were passed from the client to the remote server using the following protocols: UDP, TCP, Websocket [27]. The results in Figure 6 show:

1) UDP provides the fastest round-trip times. Under this protocol, there is no guarantee that the server will receive all point cloud messages, in order, with no duplication. For many object recognition jobs this may not be a concern.
2) Although the overhead of the Websocket protocol is slightly more than plain TCP, the bidirectional full duplex communications provided by Websockets allow not only ROS-based robots, but also the potential to connect future mobile devices with RGB-D sensors to the object recognition engine.

VI. Conclusion and Future Outlook

In this paper, we described the architectural design and implementation of **CORE**, a cloud-based object recognition engine for robotics. The architecture provides access to large-scale datasets for training machine learning classifiers, offers the capability to load different feature detector and classifier combinations, and intelligently throttles RGB-D data within a robotic network. We've showed how a cloud computing environment enables networked robots to offload computations, access greater quantities of memory, and tap supplementary data. These conditions make **CORE** an ideal framework for performing object recognition with groups of RGB-D sensor equipped robots.

The communication protocols used in this architecture were presented, and an example was provided to illustrate how the different types of messages work together. We showed the flexibility of **CORE** by giving two specific use cases involving the framework, one of which deals with

the situation where cloud services are not available. Finally, we provided some performance results when evaluating different message transport mechanisms. An alpha release is publicly available at https://github.com/wjbeksi/core under the Apache License 2.0.

Future plans include the development of additional machine learning classifiers and feature descriptors into **CORE**. A key motivating factor behind this project is to allow researchers the ability to load a variety of machine learning classifiers and feature descriptors on demand, and simplify comparisons between different published works. We also want to make it easier to incorporate large-scale datasets into **CORE**. Plans are being developed to further extend the dictionary learning framework for performing object recognition. We believe dictionary learning, as a classification scheme, has many nice qualities when used in a robotic network.

Software containers, such as Docker [28], can be used for reproducible research and ease of deployment of **CORE**. With regards to other cloud-based robotic frameworks, we are especially interested in integrating **CORE** with Rapyuta, the RoboEarth Cloud Engine [29]. Rapyuta is an open source Platform-as-a-Service framework that supplies customizable computing environments in the cloud. The modularity of the **CORE** software architecture makes it a fitting candidate to be hosted by the Rapyuta platform. Moreover, object recognition services provided by **CORE** could greatly benefit from access to the RoboEarth knowledge repository.

ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation through grants #IIP-0934327, #IIS-1017344, #CNS-1061489, #CNS-1138020, #IIP-1332133, #IIS-1427014, and #IIP-1432957.

REFERENCES

[1] Kinect, 2015. [Online]. Available: http://www.xbox.com/en-US/xbox-one/accessories/kinect-for-xbox-one

[2] 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, 2009, p. 5.

[3] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1–4.

[4] Cognex Vision Library, 2015. [Online]. Available: http://www.cognex.com/products/machine-vision/cognex-vision-library

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

[6] P. M. Fitzpatrick, "From first contact to close encounters: A developmentally deep perceptual system for a humanoid robot," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[7] S. Gould, O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Y. Ng, and D. Koller, "The stair vision library (v2.4)," 2010. [Online]. Available: http://ai.stanford.edu/ sgould/svl

[8] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 48–55.

[9] M. Muja, R. B. Rusu, G. Bradski, and D. G. Lowe, "Rein-a fast, robust, scalable recognition infrastructure," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2939–2946.

[10] D. Lorencik, M. Tarhanicova, and P. Sincak, "Cloud-based object recognition: A system proposal," in *Robot Intelligence Technology and Applications 2*. Springer, 2014, pp. 707–715.

[11] O. Zweigle, R. van de Molengraft, R. d'Andrea, and K. Häussermann, "Roboearth: connecting robots worldwide," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. ACM, 2009, pp. 184–191.

[12] K. Lai and D. Fox, "Object recognition in 3d point clouds using web data and domain adaptation," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1019–1037, 2010.

[13] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1817–1824.

[14] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. Van Nguyen, R. Ohbuchi, *et al.*, "Shrec'11 track: Shape retrieval on non-rigid 3d watertight meshes." *3DOR*, vol. 11, pp. 79–88, 2011.

[15] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: large-scale object class recognition from cad models," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 5384–5391.

[16] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 509–516.

[17] D. Fehr, A. Cherian, R. Sivalingam, S. Nickolay, V. Morellas, and N. Papanikolopoulos, "Compact covariance descriptors in 3d point clouds for object recognition," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1793–1798.

[18] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[19] D. Fehr, W. J. Beksi, D. Zermas, and N. Papanikolopoulos, "Rgb-d object classification using covariance descriptors," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5467–5472.

[20] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[21] W. J. Beksi and N. Papanikolopoulos, "Object classification using dictionary learning and rgb-d covariance descriptors," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1880–1885.

[22] ——, "Point cloud culling for robot vision tasks under communication constraints," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 3747–3752.

[23] J. Kammerl, N. Blodow, R. B. Rusu, M. Gedikli, S. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 778–785.

[24] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Proceedings of the 15th International Symposium on Robotics Research*, 2011.

[25] ROS Community Project, "Multimaster special interest group (sig)," 2015. [Online]. Available: http://wiki.ros.org/sig/Multimaster

[26] R. Ricci and E. Eide, "Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications," *; login:*, vol. 39, no. 6, pp. 36–38, 2014.

[27] I. Fette and A. Melnikov, "The websocket protocol," 2011. [Online]. Available: http://tools.ietf.org/html/rfc6455

[28] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[29] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The roboearth cloud engine," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 438–444.