Point Cloud Culling for Robot Vision Tasks Under Communication Constraints

William J. Beksi and Nikolaos Papanikolopoulos
{beksi, npapas}@cs.umn.edu
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455

***Abstract*— In this paper, we present two real-time methods for controlling data transmission in a robotic network that utilizes a remote computing infrastructure. The proposed algorithms use information and communication theory concepts to perform a highly efficient transfer of RGB-D data from a client (robot) to a server (cloud). We show that this approach makes it possible to conserve bandwidth and reduce network latency while allowing a mobile robot to perform vision tasks.**

I. Introduction

Visual perception is employed by many robots to interpret the surrounding environment. The development of affordable RGB-D sensors has sparked interest in the robotics community, especially in the area of 3D point cloud processing. An RGB-D sensor is able to simultaneously capture both color and depth images. The sensor operates at high frame rates, and can produce over 10 MB/s of data, allowing for potential bottlenecks in robotic networks.

Robot vision tasks, such as detecting, segmenting, and classifying objects, are inherently data and processing intensive. Small robot platforms often do not have the on-board resources to perform these tasks. However, by making use of a remote computing infrastructure, data and computations can be offloaded thus expanding the capabilities of a standalone robot.

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort of the service provider [1]. Recently, the field of robotics has begun to make use of cloud computing resources. Cloud robotics exploits advances in cloud computing and big data, and allows for the potential of developing a new generation of robotics applications.

In this work, we present two algorithms that allow for the intelligent throttling of RGB-D data between a client (robot) and server (cloud) for the purpose of object classification and tracking. The client makes use of redundant information in point cloud frames to reduce the amount of data transmitted. The server analyzes the received data and makes adjustments to the client's rate of transmission by employing an adaptive threshold setting. Together, the client and server maintain the usability of the network while performing the vision task at hand.

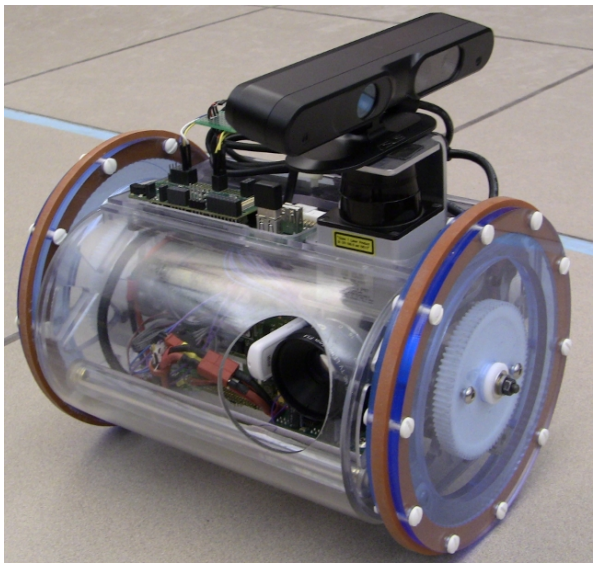


Fig. 1. The Microvision robot equipped with an RGB-D sensor.

The remainder of this paper is organized as follows. Presented in Section II is related work. Section III details the client-side point cloud culling technique. The server-side throttling of RGB-D data based on object classification performance is described in Section IV. Section V introduces the Microvision robot used in this paper followed by experimental results (Section VI). The paper concludes in the last section and presents future work.

II. Related Work

In this work, we use ideas from information and communication theory to reduce unnecessary data packet traffic in a robotic cloud infrastructure. Cloud robotics is an emerging area. This is the first work that we are aware of that incorporates a vision-based solution to the problem associated with transmitting large amounts of sensor data over a network for robot vision tasks such as object classification.

Entropy is a measure of information content based on the uncertainty of a random variable, and has a long history in the field of information theory. Shannon first defined the entropy of an information source [2]. An increase in entropy corresponds to an increase in the amount of information present in the system. Likewise, a decrease in entropy means

there is a decrease in the total amount of information. The use of entropy as a metric for measuring image information is used in the area of image compression [3]. Entropy is used in this work to measure of the overall information content of an RGB-D point cloud.

In the communication theory literature, much work has been done in the field of limited feedback wireless communications [4]. These systems allow a transmitter to adapt to the propagation conditions based on knowledge of the wireless conditions. This knowledge is acquired through feedback using a low rate data stream on the reverse side of the link to pipe information to a transmitter on the forward side of the link. This information may provide details on the forward link condition (e.g., received power, channel state, interference level, etc.), and the transmitter uses the information to adjust the forward link transmission. Our work does not seek to replace nor outperform existing feedback methods. Instead, we introduce a system that can complement these methods by the addition of vision-based feedback.

Covariance descriptors are used for object classification and tracking. Introduced by Tuzel *et al.* [5] and Porikli *et al.* [6] for people tracking in the area of image processing, covariance descriptors present a new model for the classification of point cloud data. These descriptors have not only been used for people tracking, but also in other domains such as facial recognition. Object classification using covariance based descriptors on 3D point cloud data was first developed by Fehr *et al.* [7]. Features of the covariance based descriptors are further developed in [8]. Comprehensive coverage of covariance descriptors can be found in [9]. We also use the descriptors in this work for tracking object clusters within a point cloud frame.

III. Client-side Point Cloud Culling

A. Scene Entropy

The Shannon entropy of a discrete random variable $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is defined by

$$H(\mathcal{X}) = -\sum_{i=1}^{N} p(x_i) \log p(x_i), \tag{1}$$

where $N = |\mathcal{X}|$, and $p(x_i) = \Pr[\mathcal{X} = x_i]$ for $i \in \{1, \ldots, N\}$. In a cloud infrastructure, significant difficulties are encountered due to limited bandwidth and network latency. Under these communication constraints, we selectively cull point cloud data prior to transmission. This culling is done by measuring the scene entropy of sequential point cloud frames.

To define the entropy of a scene S, the density of the voxels in the point cloud provided by the robot's current point of view is used as the probability distribution. Thus, the scene entropy is defined as

$$H(S) = -\sum_{i=1}^{N} \frac{d_i}{d_t} \log \frac{d_i}{d_t}, \tag{2}$$

where N is the total number of voxels in the scene, d_i is the density of voxel i, and $d_t = \sum_{i=1}^{N} d_i$ is the total density of all the voxels. Here, $\frac{d_i}{d_t}$ represents the density of the ith voxel with respect to the robot's observation of scene S, and maximum entropy is obtained when all voxels have the same density d_i.

Voxel density is used as the metric for the entropy calculation as it provides a direct way to detect changes in the overall information content. For example, movement of the robot and/or objects in the scene causes a corresponding change in entropy. On the contrary, a static scene results in no change in entropy. We are interested in transmitting sequential frames with a significant difference in entropy while discarding frames that have similar entropy values.

B. Robot Entropy Calculation

We make use of the robot's on-board resources 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 structure is used for downsampling and storing each filtered point cloud frame. Let T be an octree representing the current frame and let l be a leaf (voxel) of T containing 3D point data where d is the density of the leaf. The density of all leafs in T, d_t, is computed prior to the entropy. We iterate over the leaves of the T and accumulate the total entropy of the frame (Alg. 1).

Algorithm 1 computeEntropy(T)

1: **for all** $l \in T$ **do**
2: $\quad d \leftarrow l.size$
3: $\quad entropy \leftarrow entropy + \frac{d}{d_t} \log \frac{d}{d_t}$
4: **end for**
5: **return** $-entropy$

In Alg. 1, the for loop is computed in parallel by using a reduction clause on the entropy variable. This results in a substantial speedup in the per leaf calculation. After computing the entropy, a comparison is made to the currently set threshold. If the entropy value falls below the threshold, the frame is discarded. Otherwise, the octree of the frame is compressed [10] and sent to the remote server. In the next section, we explain how the entropy threshold is determined by the server.

IV. Server-side Transmission Throttling

A. Object Classification and Tracking

For each filtered point cloud frame received by the server from the client, we use scale invariant covariance descriptors to both classify and track the objects from frame to frame. The point cloud frames processed by the server provide nine different features per point. This feature vector is represented as

$$f = [x, y, z, r, g, b, n_x, n_y, n_z]. \tag{3}$$

The entries of the feature vector are composed of the Cartesian coordinates (x, y, z), the color channel values (r, g, b), and the coordinates of the surface normals (n_x, n_y, n_z) at the specific point.

From the feature vector f of each point, the covariance C of an object can be computed

$$C = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \mu_f)(f_i - \mu_f)^T, \tag{4}$$

where N is the number of points in the object, i the point's index in the objects list, and μ_f is the mean of the feature vector.

These covariance matrices characterize the objects and form the descriptors on which the classification is performed. The classification process uses a support vector machine (SVM) [11] with a radial basis function, $\exp\left(-\gamma\, d^2\left(C_1, C_2\right)\right)$, on the log Euclidean distance d between the covariances C_1 and C_2. The distance is defined as the Frobenius norm of the difference of the covariance matrix logarithms

$$d\left(C_1, C_2\right) = \left\| \log\left(C_1\right) - \log\left(C_2\right) \right\|_F. \tag{5}$$

Covariance descriptors are also used to track the objects from frame to frame. After the segmentation and extraction of the object clusters in the current frame, we iterate over the objects and match them to their corresponding clusters in the previous frame. Specifically, for a given object i in the current frame, we select object j from the previous frame such that the distance between the covariance matrices is minimized

$$\arg\min_{j} d\left(C_i, C_j\right). \tag{6}$$

Using covariance descriptors for object tracking provides computational savings versus the use of other tracking methods. Since the covariance matrices of the objects have already been computed for classification in the previous frame, they are readily recalled for matching in the current frame. In addition, the covariance matrix of each object can be compactly stored in memory making the descriptor an ideal choice for tracking objects.

B. Entropy Threshold Setting

For the robot vision task of recognizing common objects, we allow the server to throttle the client's transmission of RGB-D data. By utilizing a sliding window method, the server acknowledges the point cloud frames received by the client by adjusting the entropy threshold. Correct classification of the objects in the current frame causes the server to increase the entropy threshold. Incorrect labeling of the objects in the current frame results in the server decreasing the client's entropy threshold.

In Alg. 2, let F_i represent the ith frame of 3D point cloud data received by the server. For each frame, we compute the covariance descriptor of the extracted object cluster c. The predicted label for cluster c is then obtained from the SVM model based on the computed descriptor. Next, cluster c is matched to the cluster in F_{i-1} with minimum log Euclidean distance (Eq. (5)). Finally, the entropy threshold is decreased by δ if there is a label mismatch between the current and previous frame, or increased by δ if the labels match and the current entropy threshold is less than the maximum threshold.

Algorithm 2 setEntropyThreshold(F_i)

```
for all c ∈ F_i do
    computeCovariance(c)
    getPredictedLabel(c)
    findClusterMatch(c)
    if c.label_i ≠ c.label_{i-1} then
        threshold ← threshold − δ
    else
        if theshold < max_threshold then
            threshold ← threshold + δ
        end if
    end if
end for
return threshold
```

Sensor noise and the results of poor segmentation of the clusters can cause misclassification in the labeling of the objects. By decreasing the entropy threshold, we allow the client to transmit more frames with the expectation that additional frames will decrease the overall misclassification rate. When the system is performing optimally (i.e., no misclassified objects in the current frame), we can decrease the sending rate of the client by increasing the entropy threshold thus reducing packet traffic in the robotic network.

V. Robot Description

The Microvision robot platform was developed at the University of Minnesota's Center for Distributed Robotics. The robot is equipped with a scanning laser range finder, RGB and RGB-D sensors, and audio capture ability. The Microvision is used for experimental research in robot vision within the laboratory.

The Asus Xtion Pro Live RGB-D sensor is mounted on top of the Microvision. The depth sensor has a range of 0.8 m to 3.5 m. It has viewing angles of 45° in the vertical direction, 58° in the horizontal direction, and 70° in the diagonal direction.

The Robot Operating System (ROS) [12] is deployed on the Microvision. A Microvision ROS node functions as the driver for the robot and provides control to the robot's wheels and tail. Point cloud frames are transmitted via wireless signal to a remote server for real-time processing. Feedback from the server is also received over the wireless communication medium.

VI. Experimental Results

A. Object Training

The objects used in the experiment are shown in Fig. 2. The top row ((a) - (f)) shows the color image of each object. The middle row ((g) - (l)) provides the robot's viewpoint,

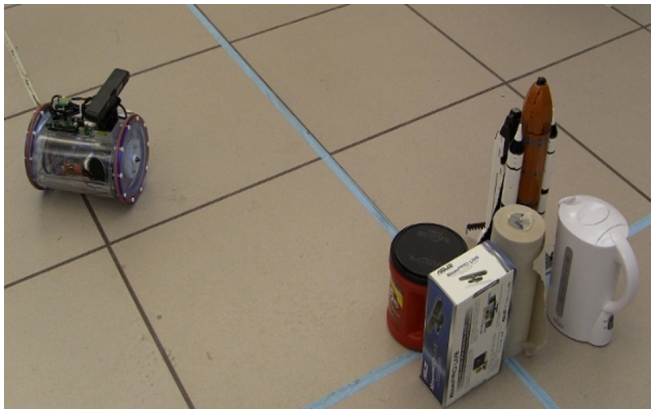


Fig. 3. The robot and an object set.

and the bottom row ((m) - (r)) shows the classification color coding of the objects. Fig. 3 shows the experimental setup. Prior to the experiment, the robot was trained on each object individually by capturing a series of point clouds while completing one revolution around the object. These point clouds were then used in the creation of the SVM model.

B. Motion Control and Data Transmission

The motion control scheme involves the robot navigating around a set of objects to obtain different points of view. It was first developed and utilized in [13] and [14], then later refined in [15]. Positioned along the y_R axis of the robot is the RGB-D sensor. The convex hull of the detected objects and the aiming point used for velocity control are computed from the sensor data.

To perform point cloud culling, a ROS node implements Alg. 1 in Section III. This node runs on-board the robot with the following functionality. First, it contains a callback function that receives point cloud data from the RGB-D sensor. Then, the node filters the point cloud data and computes the entropy of the remaining points in the frame. Finally, the data is compressed and sent to the server or dropped based on the current entropy threshold setting. At all times, the client node maintains a subscription to messages published by the server.

C. Classification and Transmission Throttling

Object classification is performed on the server. To perform the transmission throttling, a ROS node implements Alg. 2 in Section IV. The node first extracts the object clusters from the received point cloud based on a Euclidean distance segmentation method. Next, the covariance descriptor of each cluster is computed. The SVM classifier is then used to predict the object label of the cluster based on the covariance distance metric. Each cluster is matched to its closest cluster from the previous frame. If a mismatch occurs between the predicted label and matched cluster from the preceding frame, and the entropy threshold is less than the maximum threshold, then the server node updates and publishes the entropy threshold. The subscribed client node adjusts its entropy threshold upon receiving the updated value from the server.

Fig. 4. A correctly classified frame from object set A.

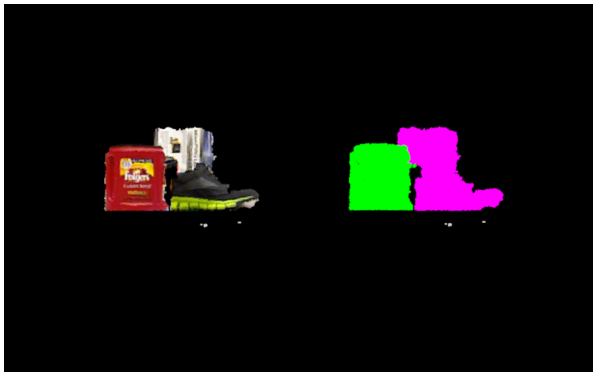


Fig. 5. An example of a misclassified frame from object set A. The box, occluded by the shoe, is merged with it.

For the experiment, we performed object classification using the Microvision robot on three sets of objects labeled A, B, and C. Set A consists of the box, coffee can, and shoe. Set B is made up of the coffee can, water heater, paper roll, and shuttle. Set C is comprised of all six objects: box, coffee can, water heater, paper roll, shoe, and shuttle. A starting entropy threshold value of $\delta = 0.1$ was used. Figs. 4 through 9 show results of the experimental runs.

While circumnavigating the object set, the robot is able to capture RGB-D data and classify the different items. When

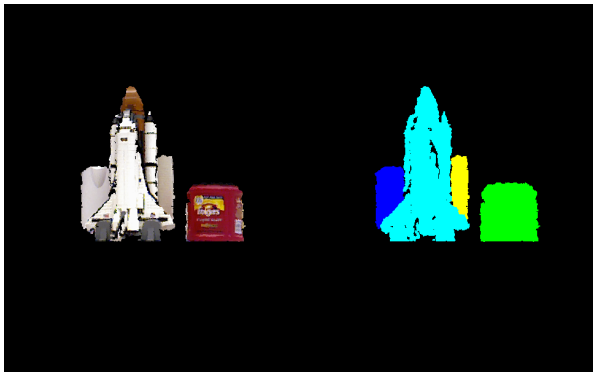
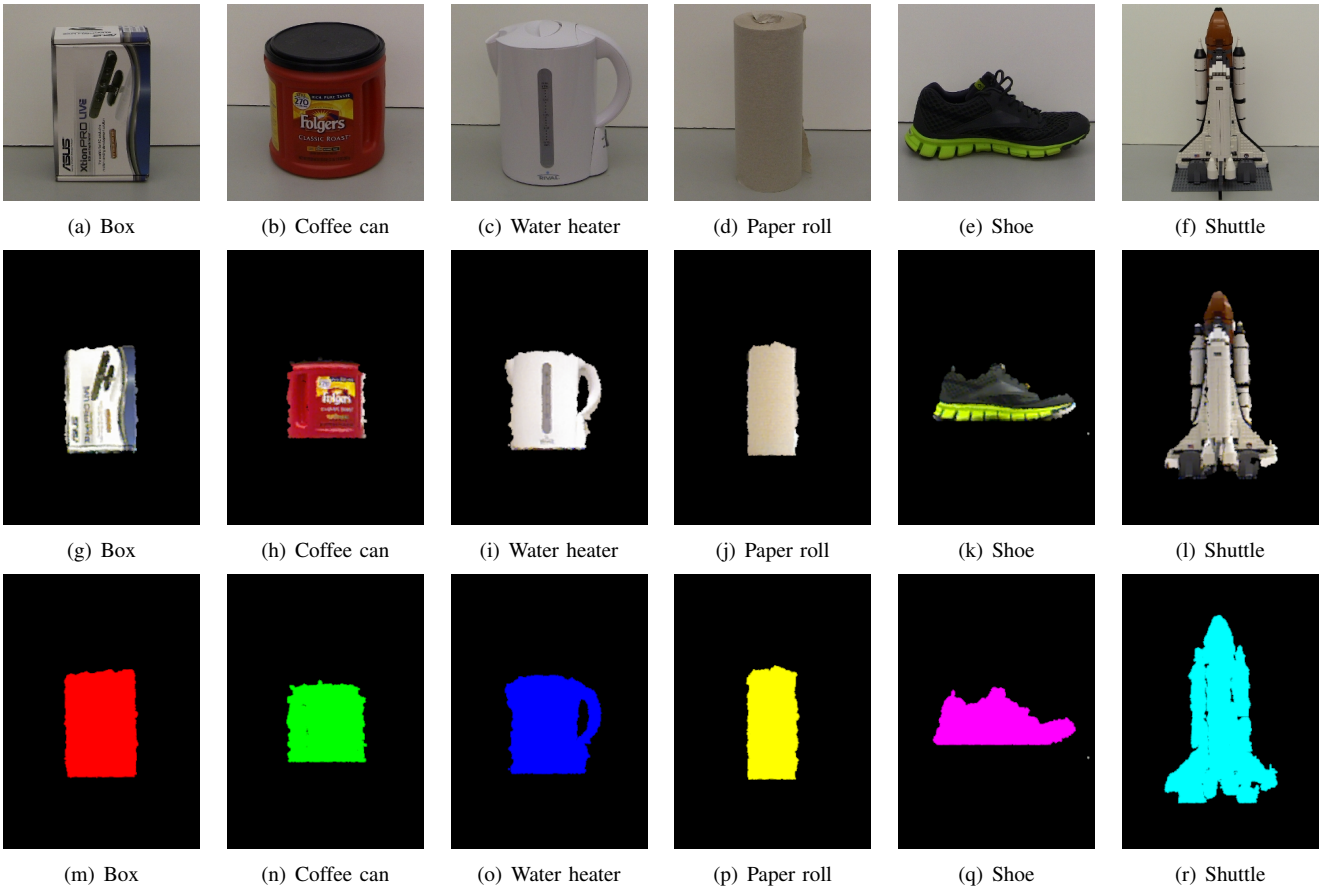


Fig. 6. A correctly classified frame from object set B.



(a) Box (b) Coffee can (c) Water heater (d) Paper roll (e) Shoe (f) Shuttle

(g) Box (h) Coffee can (i) Water heater (j) Paper roll (k) Shoe (l) Shuttle

(m) Box (n) Coffee can (o) Water heater (p) Paper roll (q) Shoe (r) Shuttle

Fig. 2. The collection of objects used in the experiment. The top row shows the RGB image, the middle row is the robot view, and the bottom row is the color classification label of each object.

Fig. 7. An example of a misclassified frame from object set B. The shuttle is incorrectly labeled.

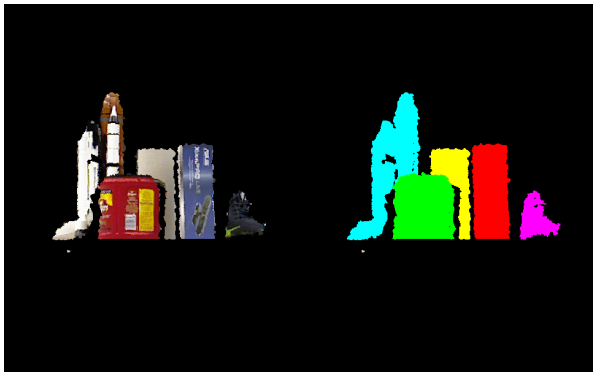


Fig. 8. A correctly classified frame from object set C.

the objects are cleanly segmented and the predicted label corresponds to the matched cluster (Fig. 4, 6, 8), the server increases the entropy threshold thus throttling the data transmission from the robot. Occlusion and/or excessive noise in the data causes the segmentation to break down (Fig. 5, 7, 9). Under these circumstances, a mismatch in the label predicted and matched cluster results in the server decreasing the entropy threshold allowing the client to transmit more frames.

In Table I, we present the results of performing point cloud culling. For each object set, the table shows the ratio of point cloud frames transmitted from the client to the server versus the total number of frames captured, the amount of data saved by not transmitting the similar frame data, and the accuracy of the classification. Accuracy is measured by the number of frames with each object cluster correctly classified, divided by the total number of frames captured. By working together, the client and server were able to reduce the number of frames sent over the network by 45.1% for object set A

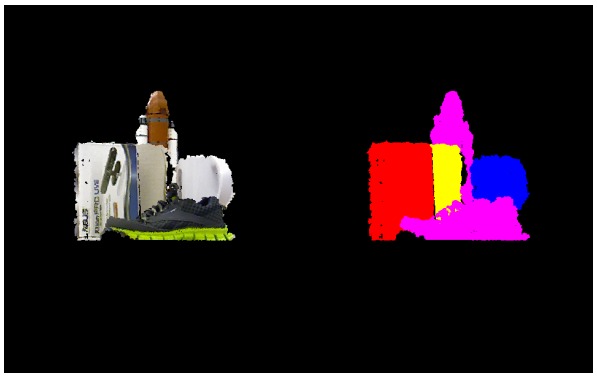


Fig. 9. An example of a misclassified frame from object set C. The majority of the shuttle is occluded which causes the segment to be classified as part of the shoe.

TABLE I

RESULTS OF POINT CLOUD CULLING

Object set	Frames transmitted	Savings (MB)	Accuracy (%)
A	17/31	168	94.44
B	19/32	156	84.72
C	24/31	84	85.61

and 40.6% for object set B. Object set C, consisting of all objects placed within close proximity to each other and with many occlusions among the objects, saw a 22.6% decrease in the number of frames transmitted from the robot to the server. With point cloud culling, the performance of the classification over each set of objects remained consistent (compared to not performing culling) while reducing the overall number of frames transmitted.

VII. CONCLUSION AND FUTURE WORK

Clients (robots) ideally make use of processing and data resources when connected to a server in a cloud environment. The role of the server can also be used to provide guidance to the actions of the client. Classifying and tracking objects is both a data and computationally intensive robot vision task. It can overwhelm the on-board resources of a robot and strain the network. We've presented a novel framework that can autonomously prevent the saturation of the network by reducing the number of point cloud frames transmitted from the client to the server.

As the robot moves around an object set, the experimental results show that it is unnecessary for the robot to transmit every point cloud frame; the classification performance is maintained by transmitting a subset of the frames. This work shows the capability of using a mobile robot for performing real-time vision tasks in a cloud computing environment.

Future work includes the development of a more robust segmentation scheme. With improved segmentation of the object clusters, we can improve the classification performance and further decrease the RGB-D data transmission rate. Extraction of the object clusters remains an important and difficult problem in this area. The incorporation of additional information metrics, such as mutual information, is part of our work going forward.

Finally, we are interested in applying the ideas of cloud computing to vision tasks among heterogeneous teams of robots. At the Center for Distributed Robotics, we are currently working on the development of robotic algorithms that make use of a cloud computing infrastructure. We believe the field of cloud robotics will open new avenues for the utilization of small and low-cost robots in an increasing number of applications.

ACKNOWLEDGEMENTS

This work was carried out in part using computing resources at the University of Minnesota Supercomputing Institute. This material is based upon work supported by the National Science Foundation through grants #IIP-0934327, #IIP-1032018, #IIS-1017344, #CNS-1061489, #CNS-1138020, #IIP-1127938, #IIP-1237259, and #IIP-1332133.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," *NIST special publication*, vol. 800, no. 145, p. 7, 2011.

[2] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[3] R. Gonzales and R. Woods, *Digital Image Processing (Third Edition)*. Pearson Education, Inc., 2008.

[4] D. J. Love, R. W. Heath, V. K. Lau, D. Gesbert, B. D. Rao, and M. Andrews, "An overview of limited feedback in wireless communication systems," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 8, pp. 1341–1365, 2008.

[5] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, vol. 3952, pp. 589–600.

[6] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 728–735.

[7] 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.

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

[9] D. Fehr, "Covariance based point cloud descriptors for object detection and classification," Ph.D. dissertation, University of Minnesota, 2013.

[10] 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.

[11] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[12] ROS. [Online]. Available: http://www.ros.org

[13] D. Fehr and N. Papanikolopoulos, "Using a laser range finder mounted on a microvision robot to estimate environmental parameters," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2009, pp. 733 211-1–733 211-9.

[14] P. Tokekar, V. Bhatawadekar, D. Fehr, and N. Papanikolopoulos, "Experiments in object reconstruction using a robot-mounted laser range-finder," in *17th Mediterranean Conference on Control and Automation, 2009. MED'09*. IEEE, 2009, pp. 946–951.

[15] D. Fehr, W. J. Beksi, D. Zermas, and N. Papanikolopoulos, "Occlusion alleviation through motion using a mobile robot," *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3179–3184, 2014.