Real-Time, Cloud-based Object Detection for Unmanned Aerial Vehicles

Jangwon Lee, Jingya Wang, David Crandall, Selma Šabanović, and Geoffrey Fox
School of Informatics and Computing
Indiana University
Bloomington, Indiana 47408
Email: {leejang, wang203, djcran, selmas, gcf}@indiana.edu

***Abstract*—Real-time object detection is crucial for many applications of Unmanned Aerial Vehicles (UAVs) such as reconnaissance and surveillance, search-and-rescue, and infrastructure inspection. In the last few years, Convolutional Neural Networks (CNNs) have emerged as a powerful class of models for recognizing image content, and are widely considered in the computer vision community to be the de facto standard approach for most problems. However, object detection based on CNNs is extremely computationally demanding, typically requiring high-end Graphics Processing Units (GPUs) that require too much power and weight, especially for a lightweight and low-cost drone. In this paper, we propose moving the computation to an off-board computing cloud, while keeping low-level object detection and short-term navigation onboard. We apply Faster Regions with CNNs (R-CNNs), a state-of-the-art algorithm, to detect not one or two but hundreds of object types in near real-time.**

I. Introduction

Recent years have brought increasing interest in autonomous UAVs and their applications, including reconnaissance and surveillance, search-and-rescue, and infrastructure inspection [1]–[5]. Visual object detection is an important component of such UAV applications, and is critical to develop fully autonomous systems. However, the task of object detection is very challenging, and is made even more difficult by the imaging conditions aboard low-cost consumer UAVs: images are often noisy and blurred due to UAV motion, onboard cameras often have relatively low resolution, and targets are usually quite small. The task is even more difficult because of the need for near real-time performance in many UAV applications.

Many UAV studies have tried to detect and track certain types of objects such as vehicles [6], [7], people including moving pedestrians [8], [9], and landmarks for autonomous navigation and landing [10], [11] in real-time. However, there are only a few that consider detecting multiple objects [12], despite the fact that detecting multiple target objects is obviously important for many applications of UAVs. In our view, this gap between application needs and technical capabilities are due to three practical but critical limitations: (1) object recognition algorithms often need to be hand-tuned to particular object and context types; (2) it is difficult to build and store a variety of target object models, especially when the objects are diverse in appearance, and (3) real-time object detection demands high computing power even to detect a single object, much less when many target objects are involved.

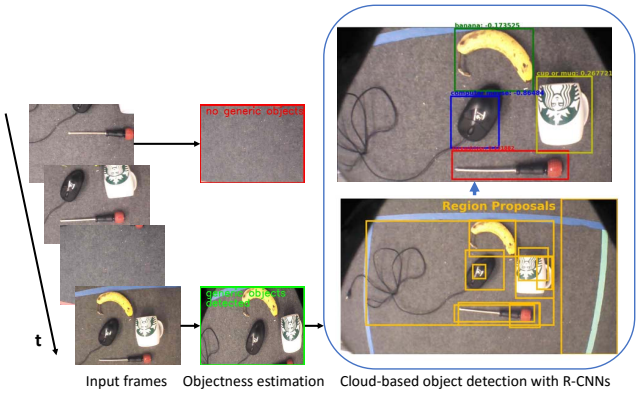

Fig. 1. A drone is able to detect hundreds of object categories in near real-time with our hybrid approach. Convolutional Neural Network-based object detection runs on a remote cloud, while a local machine plays a role in objectness estimation, short-term navigation and stability control.

However, object recognition performance is rapidly improving, thanks to breakthrough techniques in computer vision that work well on a wide variety of objects. Most of these techniques are based on "deep learning" with Convolutional Neural Networks, and have delivered striking performance increases on a range of recognition problems [13]–[15]. The key idea is to learn the object models from raw pixel data, instead of using hand-tuned features as in tradition recognition approaches. Training these deep models typically requires large training datasets, but this problem has also been overcome by new large-scale labeled datasets like ImageNet [16]. Unfortunately, these new techniques also require unprecedented amounts of computation; the number of parameters in an object model is typically in the millions or billions, requiring gigabytes of memory, and training and recognition using the object models requires high-end Graphics Processing Units (GPUs). Using these new techniques on low-cost, light-weight drones is thus infeasible because of the size, weight, and power requirements of these devices.

In this paper, we propose moving the computationally-demanding object recognition to a remote compute cloud, instead of trying to implement it on the drone itself, letting us take advantage of these breakthroughs in computer

978-1-5090-6724-4/17 $31.00 © 2017 IEEE
DOI 10.1109/IRC.2017.77

vision technology without paying the weight and power costs. Commercial compute clouds like Amazon Web Services also have the advantage of allowing on-demand access to nearly unlimited compute resources. This is especially useful for drone applications where most of the processing for navigation and control can be handled onboard, but short bursts of intense computation are required when an unknown object is detected or during active object search and tracking. Using the cloud system, we are able to apply Faster R-CNNs [17], a state-of-the-art recognition algorithm, to detect not one or two but *hundreds* of object types in near real-time. Of course, moving recognition to the cloud introduces unpredictable lag from communication latencies. Thus, we retain some visual processing locally, including a triage step that quickly identifies region(s) of an image that are likely to correspond with objects of interest, as well as low-level feature matching needed for real-time navigation and stability. Fig. 1 shows the image processing dataflow of this hybrid approach that allows a low-cost drone to detect hundreds of objects in near real-time. We report on experiments measuring accuracy, recognition time, and latencies using the low-cost Parrot AR Drone 2.0 as a hardware platform, in the scenario of the drone searching for target objects in an indoor environment.

Fig. 2. We use the Parrot AR.Drone2.0 as our hardware platform (top), adding a mirror to the front-facing camera in order to detect objects on the ground (bottom).

II. Related work

A. Deep Learning Approaches in Robotics

We apply object detection based on Convolutional Neural Networks (CNNs) [13], [18] for detecting a variety of objects in images captured from a drone. These networks are a type of deep learning approach that are much like traditional multi-layer, feed-forward perceptron networks, with two key structural differences: (1) they have a special structure that takes advantage of the unique properties of image data, including local receptive fields, since image data within local spatial regions is likely to be related, and (2) weights are shared across receptive fields, since the absolute position within an image is typically not important to an object's identity. Moreover, these networks are typically much deeper than traditional networks, often with a dozen or more layers [18]. CNNs have been demonstrated as a powerful class of models in the computer vision field, beating state-of-the-art results on many tasks such as object detection, image segmentation and object recognition [13]–[15].

Recent work in robotics has applied these deep learning techniques to object manipulation [19], hand gesture recognition for Human-Robot Interaction [20], and detecting robotic grasps [21]. These studies show the potential promise of applying deep learning to robotics. However, it is often difficult to apply recent computer vision technologies directly to robotics because most work with recognition in the computer vision community does not consider hardware limitation or power requirements as an important factors (since most applications are focused on batch-mode processing of large image and video collections like social media). In our work we explore using cloud computing to bring near real-time performance to robotics applications, without having to compromise on accuracy or the number of object classes that can be detected.

B. Cloud Robotics

Since James Kuffner introduced the term "Cloud Robotics" in 2010, numerous studies have explored the benefits of this approach [22], [23]. Cloud computing allows on-demand access to nearly unlimited computational resources, which is especially useful for bursty computational workloads that periodically require huge amounts of computation. Although the idea of taking advantage of remote computers in robotics is not new, the unparalleled scale and accessibility of modern clouds has opened up many otherwise unrealistic applications for mobile robot systems. For example, automated self-driving cars can access large-scale image and map data through the cloud without having to store or process this data locally [22]. Cloud-based infrastructures can also allow robots to communicate and collaborate with one another, as in the RoboEarth project [24].

However, a key challenge in using remote cloud resources, and especially commodity cloud facilities like Amazon Web Services, is that they introduce a number of variables that are beyond the control of the robot system. Communicating with a remote cloud typically introduces unpredictable network delay, and the cloud computation time itself may depend on which compute resources are available and how many other jobs are running on the system at any given moment in time. This means that although the cloud may deliver near real-time performance in the average case, latencies may be quite high at times, such that onboard processing is still needed for critical tasks like stability control. Here we move target recognition to the cloud, while keeping low-level detection,

short-term navigation and stability control local. This hybrid approach allows a low-cost quadcopter to recognize hundreds of objects in near real-time on average, with limited negative consequences when the real-time target cannot be met.

C. Objectness Estimation

While modern object recognition may be too resource-intensive to run on a lightweight drone, it is also unrealistic to transfer all imagery to a remote cloud due to bandwidth limitations. Instead, we propose locally running a single, lightweight "triage" object detector identifies images and image regions that are likely to contain some object of interest, which then can be identified by a more computationally-intensive, cloud-based algorithm. To do this, we evaluate *'objectness'* [25], which is measure of how likely a certain window of an image contains an object of any class. Most recent object detectors in the computer vision field use one of the objectness estimation techniques (or object proposal methods) for reducing computation instead of using brute-force sliding windows that run detectors at every possible image location [13], [26].

Several object proposal methods have been recently proposed, each with strengths and weaknesses [27]. We apply the Binarized Normed Gradients (BING) algorithm to measure objectness on input frames as a first step process in our hybrid object detection system [28]. While it is not the most accurate technique available [27], it is one of the simplest and fastest proposal methods (1 ms / image on a CPU), and thus can run in real-time on local machine.

III. Hardware Platform

We use a Parrot AR.Drone 2.0 as a low-cost hardware platform [29] to test our cloud-based recognition approach. The AR.Drone costs about US$300, is small and lightweight (about 50cm $\times$ 50cm and 420g including the battery), and can be operated both indoors and outdoors.

A. Hardware Specifications

The AR.Drone 2.0 is equipped with two cameras, an Inertial Measurement Unit (IMU) including a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer, and pressure- and ultrasound-based altitude sensors. The front-facing camera has a resolution of 1280 $\times$ 720 at 30fps with a diagonal field of view of 92°, and the lower-resolution downward-facing camera has a resolution of 320 $\times$ 240 at 60fps with a diagonal field of view of 64°. We use both cameras, although we can only capture images from one of the two cameras at the same time due to firmware limitations.

Because the front-facing camera has a higher resolution and wider field of view than the downward-facing one, we use the front-facing camera for object detection. To allow the drone to see objects on the ground, which is needed for most UAV applications like search and rescue, we mounted a mirror at a 45° angle to the front camera (see Fig. 2).

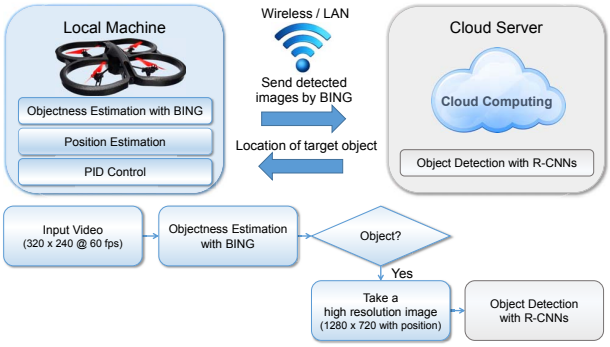

Fig. 3. System Overview: Our approach consists of four main components: BING based objectness estimation, a position estimation for localization, PID control for navigation, and R-CNNs based object detection. All components are implemented under the ROS framework, so each component can communicate with every other via the ROS network protocol (top). Given input video, the local machine detects generic objects in every frame with BING, then takes a high resolution image and sends it to the cloud server if the frame contains generic objects. The cloud server then runs R-CNNs based object detection to find a target object (bottom).

B. Embedded Software

The AR.Drone 2.0 comes equipped with a 1 GHz ARM Cortex-A8 as the CPU and an embedded version of Linux as its operating system. The embedded software on the board measures horizontal velocity of the drone using its downward-facing camera and estimates the state of the drone such as roll, pitch, yaw and altitude using available sensor information. The horizontal velocity is measured based on two complementary computer vision features, one based on optical flow and the other based on tracking image features (like corners), with the quality of the speed estimates highly dependent on the texture in the input video streams [29]. All sensor measurements are updated at 200Hz. The AR.Drone 2.0 can communicate with other devices like smartphones or laptops over a standard WiFi network.

IV. Approach

A. System Overview

Our approach consists of four main components shown at top in Fig. 3. Each component is implemented as a node in the Robot Operating System (ROS), allowing it to communicate with others using the ROS transport protocol [30]. Three components, the objectness estimator, the position estimator and PID controller, are run on a laptop (with an Intel Core i7 Processor running at 2.4 GHz), connected to the drone through the AR.Drone device driver package of ROS, over a WiFi link. The drone is controlled by the control commands with four parameters, the roll Φ, the pitch Θ, the vertical speed z, and the yaw Ψ. The most computationally demanding component, the R-CNN-based object detection node, runs on a remote cloud computing server that the laptop connects to via the open Internet.

The bottom of Fig. 3 shows the pipeline of image processing in our hybrid approach. The drone takes off and starts to

search for target objects with the downward-facing camera. Given input video taken from this downward-facing camera, the objectness estimator node runs the BING algorithm to detect generic objects on every frame, and then takes a high resolution image with the front-facing camera if it detects candidate objects in the frame [28]. Consequently, only the "interesting" images that have a high likelihood to contain objects are sent to the cloud server, where the R-CNN-based object detection node is run to recognize the target objects in the environment.

B. Position Estimation and PID Controller for Navigation

We employ an Extended Kalman Filter (EKF) to estimate the current position of the drone from all available sensing data. We use a visual marker detection library, ArtoolkitPlus, in our update step in order to get accurate and robust absolution position estimation results within the test environment [31]. (It would be more realistic if the drone estimated its current position without these artificial markers, but position estimation is not the focus of this paper so we made this simplification here.)

Furthermore, since our test environment is free of obstructions, we assume that the drone can move without changing altitude while it is exploring the environment to look for target objects. This is a strong assumption but again is reasonable for the purposes of this paper, and it makes the position estimation problem much easier because this assumption reduces the state space from 3D to 2D. Note that this assumption does not mean that the drone never changes its altitude — in fact, it can and does change altitude to get a closer view of objects, when needed, but it does so in hovering mode and returns back to the canonical altitude before flying elsewhere in the environment.

In order to generate the control commands that drive the drone towards its desired goal locations, we employ a standard PID controller. Thus, the PID controller generates the control commands to drive the drone according the computed error values, and finally, the drone changes operation mode to hovering mode when the drone reaches within a small distance of the desired goal position.

C. Objectness Estimation with BING

The quadrocopter starts its object detection mission with the downward-facing camera, which takes video at 60fps with 320×240 image resolution. Given this video input, the local objectness estimation node decides whether the current input frame contains a potential object of interest. We apply the Binarized Normed Gradients (BING) algorithm to measure this objectness on every input frame [28].

We trained the BING parameters on the Pascal VOC 2012 dataset [16], and used the average score of the top 10 bounding boxes for making a decision. In order to set a decision threshold for our approach, we collected background images having no object using our quadrocopter. Using the threshold, the object estimator node measures the objectness of each frame, then takes a high resolution image with the front-facing camera if the score is above the threshold. Finally, the node sends the images to the cloud server with its position information.

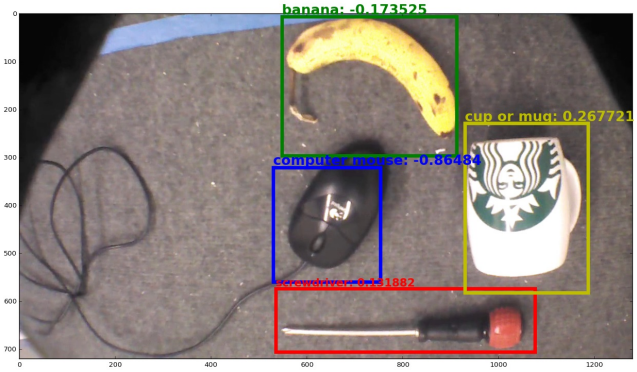

Fig. 4. An example of R-CNNs-based object detection with an image taken by our drone.

D. Cloud-based R-CNNs for Object Detection

After receiving an image of a candidate object, we apply the Faster R-CNN algorithm for object detection [17]. R-CNNs are a leading approach for object detection that combines a fast object proposal mechanism with CNN-based classifiers [13], and Faster R-CNN is a follow-up approach by the same authors that increases accuracy while reducing the running time of the algorithm. Very briefly, the technique runs a lightweight, unsupervised hierarchical segmentation algorithm on an image, breaking the image into many (hundreds or thousands of) overlapping windows that seem to contain "interesting" image content that may correspond to an object, and then each of these windows is classified separately using a CNN. R-CNNs have shown leading performance in datasets for object detection challenges, but these images are usually collected from social media (e.g. Flickr), and to our knowledge, have not been applied to robotic applications. The main reason for this is probably that CNNs demand very high computational power, typically in the form of high-end GPUs, even though their recent approach only requires around 200 ms for processing per image with GPUs. We therefore move the R-CNNs based object detection part to a cloud system.

Besides the computational cost, another major challenge with using CNNs is their need for very large-scale training datasets, typically in the hundreds of thousands or millions of images. Because it is unrealistic for us to capture this scale dataset for our application, we used R-CNN models trained for the 200 object types of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC13) dataset [16]. A disadvantage of this approach is that the training images were mostly collected from sources like Google Images and Flickr, and thus are largely consumer images and not the aerial-type images seen by our drone. We could likely achieve much better recognition accuracies by training on a more representative dataset; one option for future work is to take a hybrid approach that uses the ILSVRC13 data to bootstrap a classifier fine-tuned for our aerial images. Nevertheless, our approach has

TABLE I
OBJECT DETECTION RESULTS ON OUR AERIAL IMAGES COLLECTED BY THE DRONE.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fast YOLO	**87.5**	84.6	0.0	50.0	65.5	**100.0**	87.9	80.0	**92.3**	47.1	60.0	75.0	88.9	**100.0**	76.6	**100.0**	54.5	**100.0**	66.7	84.6	78.3
YOLO	60.9	88.2	80.0	80.0	92.3	**100.0**	87.2	**100.0**	70.4	50.0	40.0	**81.0**	77.8	93.4	88.7	**100.0**	45.2	**100.0**	81.8	79.2	79.4
SSD300	60.0	**94.1**	20.0	**100.0**	90.0	**100.0**	**100.0**	**100.0**	75.0	47.8	50.0	66.7	81.3	92.9	**92.9**	**100.0**	66.7	**100.0**	85.7	82.6	81.6
SSD500	66.7	88.2	50.0	88.9	**100.0**	92.9	93.2	**100.0**	72.1	**65**	**85.7**	69.6	88.9	93.8	81.7	**100.0**	66.7	89.5	66.7	**100.0**	82.6
Faster R-CNN	70.6	93.8	**83.3**	85.7	91.9	92.9	89.7	**100.0**	87.2	62.5	-	77.3	**100.0**	93.8	81.7	66.7	**72.7**	**100.0**	**100.0**	62.5	**83.9**

the advantage of giving our robot the ability to detect several hundred types of objects "for free," without much additional investment in dataset collection. We use the Faster R-CNN implementation in Caffe [32], a C++ deep learning framework library.

An example of our detection results with an image taken by the drone is shown in Fig. 4. Here, the numbers above the box are the confidence scores of detected object, with greater score meaning greater confidence. The drone detected four different types of objects correctly, even though one object, a computer mouse, has a relatively low confidence. However, an advantage of robotic applications is that when such uncertainty is detected, the drone can choose to approach the computer mouse and take more pictures from different angles and distances, in order to confirm the detection. For example, if a detection score decreases while approaching the object and falls under some threshold, the drone can decide that the object is not the target.

V. EXPERIMENTAL RESULTS

We conducted three sets of experiments to demonstrate that our approach performs successfully in a realistic but controlled environment. In the first set of experiments, we focus on testing the accuracy of recent deep network based object detectors with aerial images taken by the drone, and specifically the viability of our idea of applying object models trained on consumer images (from ImageNet) to a robot application. In the second set of experiments, we evaluate the speed of our cloud based object detection approach, comparing with running time of the fastest deep learning based object detector on a local laptop. Finally, we verify our approach with the scenario of a drone searching for a target object in an indoor environment, as a simple simulation of a search-and-rescue or surveillance application.

The first two sets of experiments were conducted on our aerial image dataset and the last experiment was conducted in an indoor room of about $3m \times 3m$. We did not make any attempt to control for illumination or background clutter, although the illumination was fixed (overhead fluorescent lighting) and the background was largely composed of the navigation markers mentioned above.

A. Object Detection Accuracy

We first compared the ability of Faster R-CNNs with two recent state-of-the-are object detectors (YOLO [33] and SSD [34]) to recognize aerial images taken by the drone. YOLO and SSD are approaches that are designed to speed up classifier-based object detection systems through eliminating the most computationally demanding part (generating region proposals and computing CNN features for each region). Both methods showed accurate mean average precision (mAP) on Pascal VOC 2007 dataset (YOLO: 69.0% vs. SSD300: 74.3%) with real-time performance (faster than 30 FPS) on GPU.

To make a fair comparison, we used models that were all pre-trained on the same dataset (Pascal VOC 2007 and Pascal VOC 2012). We collected 294 aerial images of 20 object classes and annotated 578 objects in the images. The images had the same object classes as the Pascal VOC 2007 dataset and were collected from two sources (some of them taken by ourselves and the others were collected from 31 publicly available Youtube videos taken by the same drone as ours). Table I shows average precision of each algorithm on this dataset. Here, the SSD300 model and SSD500 model have the same architecture and the only difference is the input image size (300×300 pixels vs. 500×500 pixels). YOLO and Fast YOLO also use similar architectures except Fast YOLO uses fewer convolutional layers (24 convolutional layers vs. 9 convolutional layers for Fast YOLO).

On this dataset, Faster R-CNN acheived 83.9% mAP compared to YOLO models (78.3% and 79.4%) and two SSD models (81.6% and 82.6%). All models achieved higher mAP on our aerial image dataset than their detection results on Pascal VOC2007 since images of some object classes such as cats and plants are very distinctive with clean backgrounds. The first row of Fig. 8 shows these "easy" images on this dataset, and the second row presents some "hard" examples which were taken at high altitude.

As discussed above, we applied Faster R-CNN trained on ImageNet consumer images and fine-tuned with Pascal VOC dataset to our drone scenario. This time, we did not limit the objects to those 20 object categories of VOC 2007, but instead we looked at the results among the 200 categories Faster R-CNN provided. We did this though the aerial drone images look nothing like most consumer images, because we did not have the large-scale dataset needed to train a CNN from scratch. This can be thought of as a simple case of transfer learning, and likely suffers from the usual mismatch problem when training sets and testing sets are sampled from different distributions. We took other 74 images like bottom two rows of Fig. 8, and achieved 63.5% of accuracy.

B. Recognition Speed on Cloud System

Our second set of experiments evaluated the running time performance of the CNN-based object recognition, testing the extent to which cloud computing could improve recognition times, and the variability of cloud-based recognition times due to unpredictable communication times. For these experiments we used the same set of images and objects collected in

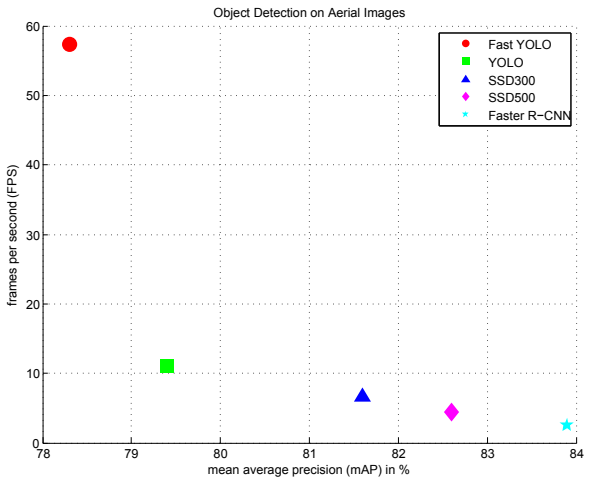

Fig. 5. A running time comparison of recent state-of-the-art object detectors on our aerial images.

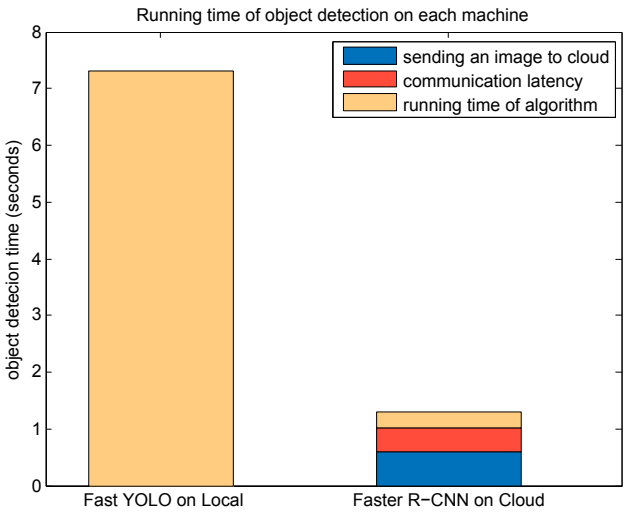

Fig. 6. Running time of object detection on each machine.

the previous section, and compared the speed of each algorithm using Graphics Processing Unit (GPU) on a simulated cloud machine at first. We measured the running time including image loading, pre-processing, and output parsing (post-processing) time, since those times are important in real-time applications.

Fig. 5 shows the running time of each algorithm as a function of its accuracy. Even though all recent state-of-the-art methods showed reasonable speed with high-accuracy, for instance, SSD 300 models showed 6.55 FPS with mAP 81.6, the result shows detection speed and accuracy are still in inverse related. Fast YOLO showed the highest speed (57.4 FPS) with the lowest accuracy (mAP 78.3), while Faster R-CNN had the lowest speed (3.48 FPS) with the highest accuracy (mAP 83.9).

In the second experiment, we thus compared Fast YOLO on a local laptop versus Faster R-CNN on a remote server as a simulated cloud. A comparison of these computing facilities are shown in Table II. Fig. 6 shows the running time of Fast YOLO and Faster R-CNN on the two different machines.

The average running time of Fast YOLO on the local machine was 7.31 seconds per image, while the average time on the cloud machine based Faster R-CNN was 1.29 seconds, including latencies for sending each image to the cloud computer (which averaged about 600ms), and for exchanging detected results and other command messages (which averaged 0.41ms). Thus the cloud-based recognition performed about 5.7 times faster than the local Fast YOLO on average. The average running time on our single-server simulated cloud is not fast enough to be considered real time, but is still fast enough to be useful in many applications. Moreover, recognition could be easily made faster by parallelizing object model evaluations across different machines.

C. *Target Search with a Drone*

In this section, we demonstrate our approach with a simple scenario of the drone searching for a target object in an indoor environment. We assume that a drone is supposed to find a single target object in a room in a building. There are several different types of objects in the room, but fortunately there are no obstacles.

In the test scenario, we used a screwdriver as a target object and scattered various distractor objects on the floor in the indoor test room. The drone started this object searching mission with lower-resolution downward-facing camera, and ran the BING algorithm for finding generic objects given the input video. At the same time, the position estimator node estimated the drone's position continuously. When the drone found any "interesting" objects on the floor, it switched to the front-facing camera to capture a photo at a higher resolution and with a wider angle, then took picture of the candidate area and sends it to the cloud system (t = 3 s and t = 8 s). Then, the drone switched the camera back to the downward-facing camera for localization and stability control, and proceeded to the other candidate positions. In the meantime, the cloud system performed recognition then sent results to the drone. The drone performed the same steps until it found a target object, at which point the mission was completed (t = 17 s).

TABLE II
HARDWARE COMPARISON BETWEEN LOCAL AND CLOUD MACHINE

	local computer	cloud computer
CPUs	one Intel Core i7-4700HQ @ 2.4GHz	two Intel Xeon E5-2680 v3 @ 2.5GHz
GPUs	one Nvidia GeForce GTX 770M	two Nvidia Tesla K40
RAM	16 GB	128 GB

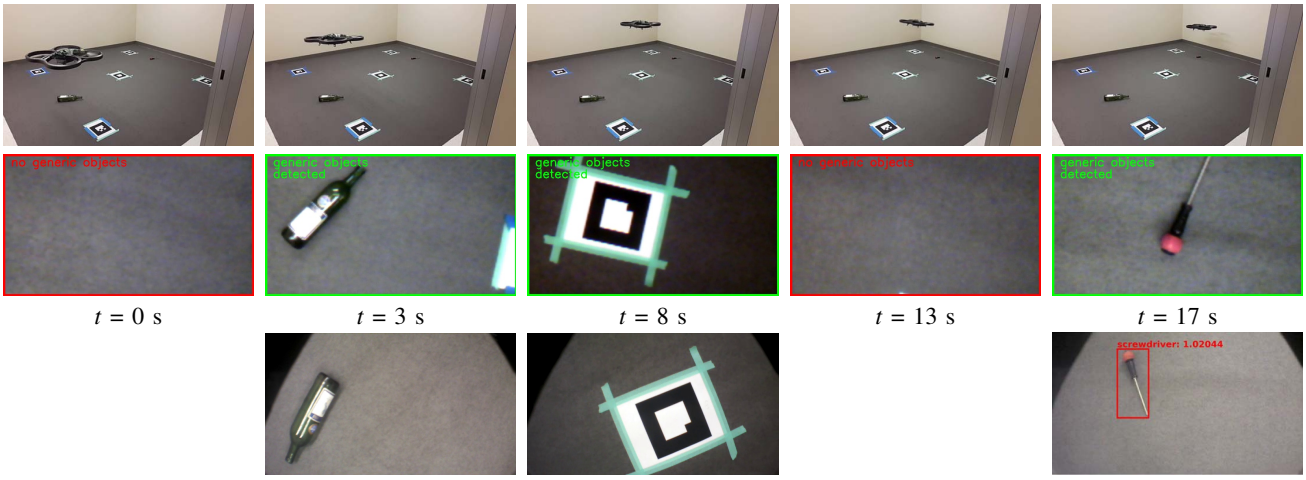

Fig. 7. Target Search with a Drone: First rows show movements of the drone during the experiment, and second and third rows indicate detection results from BING and R-CNNs respectively. At t = 0 s the drone started to search for a target object and did not find generic objects with BING. At t = 3 s, t = 8 s, the drone found generic objects with BING, thus took high resolution pictures and sent them to cloud server. However, R-CNNs did not detect a target object in those images. At t = 17 s, the drone found generic objects again, thus it took the high resolution picture and sent it to cloud server. Then, finally R-CNNs based object detector found a target object.

Fig. 7 shows a sequence of images taken during the drone's search for a target object in our test scenario. It shows that the drone only took pictures and sent them when there were "interesting" objects on the floor, and finally found the target object, a screwdriver, with the cloud-based R-CNNs object detector.

VI. Conclusion

In this paper, we proposed to use Convolutional Neural Networks to allow UAVs to detect hundreds of object categories. CNNs are computationally expensive, however, so we explore the hybrid approach that moving the recognition to a remote computing cloud while keeping low-level object detection and short-term navigation onboard. Our approach enables the UAVs, especially lightweight, low-cost consumer UAVs, to use state-of-the-art object detection algorithms, despite their very large computational demands. The (nearly) unlimited cloud-based computation resources, however, come at the cost of potentially high and unpredictable communication lag and highly variable system load. We tested our approach with a Parrot AR.Drone 2.0 as a low-cost hardware platform in a real indoor environment. The results suggest that the cloud-based approach could allow speed-ups of nearly an order of magnitude, approaching real-time performance even when detecting hundreds of object categories, despite these additional communication lags. We demonstrated our approach in terms of recognition accuracy and speed, and in a simple target searching scenario.

Acknowledgment

The authors wish to thank Matt Francisco for helping to design and fabricate the forward-facing camera mirror, Supun Kamburugamuve for helping with the software interface to the cloud infrastructure, and Bruce Shei for configuring the cloud servers.

References

[1] M. Bhaskaranand and J. D. Gibson, "Low-complexity video encoding for uav reconnaissance and surveillance," in *Military Communications Conference (MILCOM)*. IEEE, 2011, pp. 1633–1638.

[2] P. Doherty and P. Rudol, "A uav search and rescue scenario with human body detection and geolocalization," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2007, pp. 1–13.

[3] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[4] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple uavs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006.

[5] I. Sa, S. Hrabar, and P. Corke, "Outdoor flight testing of a pole inspection uav incorporating high-speed vision," in *Field and Service Robotics*. Springer, 2015, pp. 107–121.

[6] T. P. Breckon, S. E. Barnes, M. L. Eichner, and K. Wahren, "Autonomous real-time vehicle detection from a medium-level uav," in *Proc. 24th International Conference on Unmanned Air Vehicle Systems*, 2009, pp. 29–1.

[7] J. Gleason, A. V. Nefian, X. Bouyssounousse, T. Fong, and G. Bebis, "Vehicle detection from aerial imagery," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2065–2070.

[8] A. Gaszczak, T. P. Breckon, and J. Han, "Real-time people and vehicle detection from uav imagery," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2011, pp. 78 780B–78 780B.

[9] H. Lim and S. N. Sinha, "Monocular localization of a moving person onboard a quadrotor mav," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2182–2189.

[10] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014.

[11] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 111–118.

Fig. 8. Sample images collected by our drone. R-CNNs based object recognition are able to detect a wide variety of different types of objects.

[12] F. S. Leira, T. A. Johansen, and T. I. Fossen, "Automatic detection, classification and tracking of objects in the ocean surface from uavs using a thermal camera," in *2015 IEEE Aerospace Conference*. IEEE, 2015, pp. 1–10.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[14] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

[15] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[20] J. Nagi, A. Giusti, F. Nagi, L. M. Gambardella, and G. A. Di Caro, "Online feature extraction for the incremental learning of gestures in human-swarm interaction," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3331–3338.

[21] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[22] K. Goldberg and B. Kehoe, "Cloud robotics and automation: A survey of related work," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5*, 2013.

[23] 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.

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

[25] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[26] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for generic object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 17–24.

[27] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 814–830, 2016.

[28] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3286–3293.

[29] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1477–1484, 2011.

[30] 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*, 2009.

[31] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," in *Computer Vision Winter Workshop (CVWW)*, 2007.

[32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *roceedings of European Conference on Computer Vision (ECCV)*, 2016.