Real-Time Object Detection for Unmanned Aerial Vehicles based on Cloud-based Convolutional Neural Networks

Jangwon Lee, Jingya Wang, David Crandall, Selma Šabanović and Geoffrey Fox

***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. We apply 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

In recent years, there has been increasing interest in autonomous UAVs and its applications such as reconnaissance and surveillance, search-and-rescue, and infrastructure inspection [1, 2, 3, 4, 5]. Visual object detection is an important component in such applications of UAVs, 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, such as when objects are used for navigation.

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, the main reasons for this gap between application needs and technical capabilities are due to two 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 single objects, much less when many target objects are involved.

School of Informatics and Computing, Indiana University, Bloomington, IN 47408, USA. {leejang, wang203, djcran}@indiana.edu. This work was supported in part by The Air Force Office of Scientific Research (AFOSR) and by NVidia.

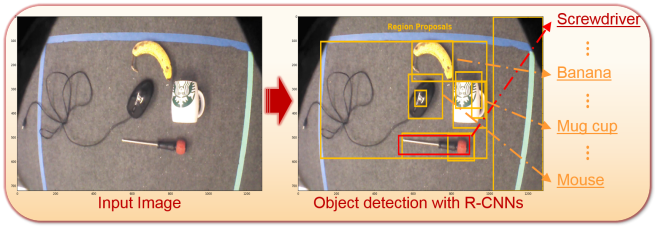

Fig. 1. A drone is able to detect hundreds of object categories in near real-time using Convolutional Neural Networks running on a remote cloud.

However, the first of these problems is eroding due to new 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, 16]. 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 [29]. 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, lightweight 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 vision technology without paying the weight and power costs. 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 R-CNNs [13], a state-of-the-art recognition algorithm, to detect not one or two but *hundreds* of object types in near real-time (see Fig. 1). Of course, moving recognition to the cloud introduces unpredictable lag from communication latencies. 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.

II. Related work

A. Deep Learning Approaches in Robotics

We apply object detection based on Convolutional Neural Networks (CNNs) [13, 14] for detecting a variety of objects in images captured from a drone. These networks are type of deep learning approach that are much like traditional multi-layer, feed-forward perceptron networks, except that 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 shared weights 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 [14]. 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, 16]. Recent work in robotics has applied these deep learning techniques, such as in object manipulation [17], hand gestures recognition for Human-Robot Interaction [18], and detecting robotic grasps [19]. These studies show the potential promise of applying deep learning approaches to robotics fields. In many cases, however, there is difficulty in applying recent computer vision technologies directly to robotics, despite the state-of-the-art performance. This is because most work with recognition in the computer vision community does not consider computation time as an important factor, since most applications are focused on batch-mode processing of large image and video collections (e.g. for organizing social media collections). 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 and research have explored the benefits of this approach [20, 21]. 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, for accurate mapping and localization, without having to store or process this data locally [20]. Cloud-based infrastructures can also allow robots to communicate and collaborate with one another, as in the RoboEarth project [22].

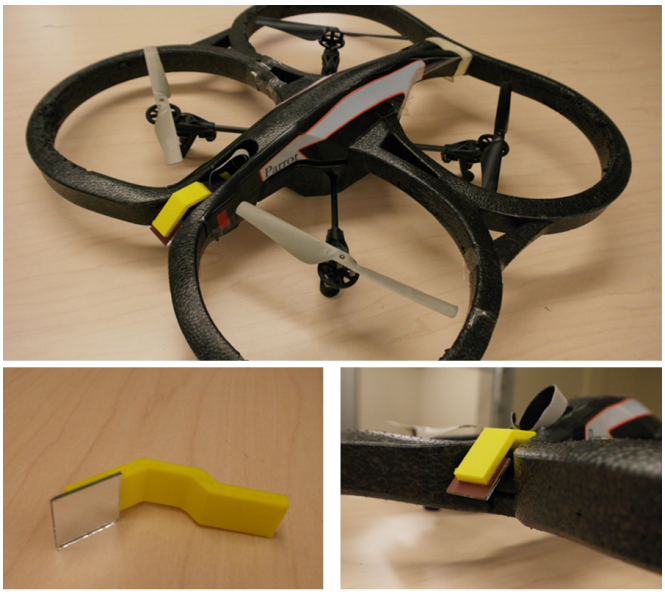

Fig. 2. We use the Parrot AR.Drone2.0 as our hardware platform (top), adding a mirror to the front-facing camera in order to detect objects on the ground (bottoms).

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 propose moving target recognition to the cloud, while allowing keeping 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.

III. Hardware platform

We use a Parrot AR.Drone 2.0 as a low-cost hardware platform [25] 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 Units (IMUs) 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).

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 [26]. 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 three main components as shown in Fig. 3. Each component is implemented as a node within the Robot Operating System (ROS) framework, allowing each component to communicate the others using the ROS transport protocol [27].

Two components, 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 CNN-based object detection node, runs on a remote cloud computing server that the laptop connects to via the open Internet.

B. Position Estimation with Extended Kalman Filter

We employ an Extended Kalman Filter (EKF) to estimate the current position of the drone from all available sensing data. Here, the height and horizontal velocity measurements are often inaccurate due to the fast motion of the drone, and the low-cost sensors and actuators of our hardware platform, which together create a high degree of noise. We therefore 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 [28]. (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.) We place 25 markers in a 5 × 5 grid on the ground of the test environment, covering an area of 2m × 2m.

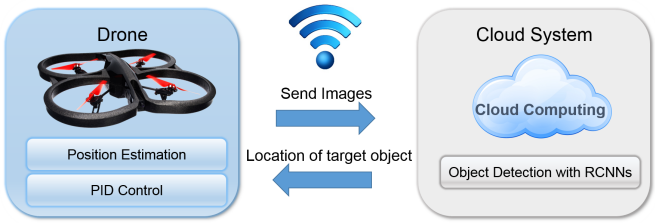

Fig. 3. System Overview: Our approach consists of three main components: a position estimation for localization, PID control for navigation, and R-CNN-based object detection. All components are implemented under the ROS framework, so each component can communicate with every other via the ROS network protocol.

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.

The state space of the drone is given by,

$$\mathbf{x}_t = (x_t, y_t, \Psi_t)^T \in \mathbb{R}^3, \tag{1}$$

where x_t and y_t are position of the drone along axes parallel to the ground plane at time t, and Ψ_t is the yaw angle of the drone at time t. Then, the prediction function $g(\mathbf{u}_t, \mathbf{x}_{t-1})$ and update function $h(\mathbf{x}_t)$ of the EKF are,

$$g(\mathbf{u}_t, \mathbf{x}_{t-1}) = \begin{pmatrix} x_{t-1} + (\cos(\Psi_{t-1})x_o - \sin(\Psi_{t-1})y_o)\Delta_t \\ y_{t-1} + (\sin(\Psi_{t-1})x_o + \cos(\Psi_{t-1})y_o)\Delta_t \\ \Psi_{t-1} + \Psi_o\Delta_t \end{pmatrix}, \tag{2}$$

$$h(\mathbf{x}_t) = \begin{pmatrix} \cos(\Psi_t)(x_t - x_m) + \sin(\Psi_t)(y_t - y_m) \\ -\sin(\Psi_t)(x_t - x_m) + \cos(\Psi_t)(y_t - y_m) \\ \Psi_t - \Psi_m \end{pmatrix}, \tag{3}$$

where $\mathbf{u}_t = (x_o, y_o, \Psi_o)^T$ denotes odometry information, measured by the horizontal velocity and yaw rotation speed of the drone, (x_m, y_m, Ψ_m) represents the position of visual marker, and Δ_t indicates time intervals.

C. PID Controller for Navigation

We employ a simple PID controller to generate the control commands that drive the drone towards its desired goal locations. We compute the error values of each degrees-of-

freedom as,

$$\mathbf{x}_{error} = \begin{pmatrix} e_x \\ e_y \\ e_\Psi \end{pmatrix} = \mathbf{x}_{goal} - \mathbf{x}_{current}, \quad (4)$$

where $\mathbf{x}_{goal}$ and $\mathbf{x}_{current}$ denote goal and current state of the drone respectively. Then, we apply the error values to the discrete form of PID controller,

$$\mathbf{u}_t = \mathbf{K}_p \mathbf{e}_t + \mathbf{K}_i \mathbf{e}_{i,t} + \mathbf{K}_d \mathbf{e}_{d,t}, \quad (5)$$

where $\mathbf{K}_p$, $\mathbf{K}_i$, and $\mathbf{K}_d$, denote the gains for the proportional, integral, and derivative terms respectively, and $\mathbf{e}_t$ are computed error values at time t. Then, $\mathbf{e}_{i,t}$, and $\mathbf{e}_{d,t}$ are computed as follows

$$\mathbf{e}_{i,t} = \mathbf{e}_t \Delta_t + \mathbf{e}_{i,t-1}, \quad (6)$$

$$\mathbf{e}_{d,t} = \frac{\mathbf{e}_t - \mathbf{e}_{t-1}}{\Delta_t}, \quad (7)$$

where Δ_t denotes time intervals, but it is a constant value since we control the drone with same frequency.

Thus, the PID controller generates the control commands to drive the drone with fast velocity when the error values are large, and reduces the drone speed in proportion to decreasing error values when the drone approaches the desired goal location correctly. Finally, we change operation mode of the drone to hovering mode when the drone reaches within a small distance of the desired goal position.

D. Cloud-based R-CNNs for Object Detection

After the drone captures an image of a candidate object, we apply the R-CNN algorithm for object detection [13]. R-CNNs are a leading approach for object detection, that combines a fast object proposal mechanism with CNN-based classifiers. 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 several 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. 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 [29]. A disadvantage of this approach is that the training images were mostly collected from sources like Google Images and Flickr,

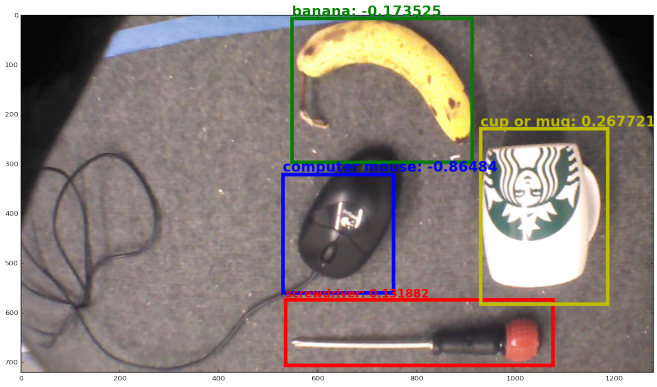

Fig. 4. An example of R-CNN-based object detection with an image taken by our drone.

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 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 R-CNN implementation in Caffe [30], 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. Experiment 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 object recognition using R-CNNs, 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 R-CNNs for object recognition, comparing running times on a local laptop versus a compute cloud, where communication time is much less predictable but computational resources are much greater. 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.

All experiments were 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 Recognition Accuracy

We first tested the ability of the CNNs trained on ImageNet data to recognize different objects appearing in our test environment. We collected 18 different objects, scattered them around the room, and used the drone to collect 90 aerial images of them from random orientations and perspectives. The objects consisted of typical household items, including: a helmet, cocktail shaker, hammer, hair spray, pitcher, tennis ball, dumbbell, wine bottle, face powder, computer keyboard, banana, coffee mug, screwdriver, water bottle, perfume, binder, tennis racket, and backpack. Of the 90 images, about half (41) were blank or blurry beyond human recognition, so we excluded them for the remainder of the tests. We passed the remaining images to the recognition module, which used R-CNNs to classify the object according to its 200 object models. The result of this process is 200 confidence scores per image, indicating the classifier's estimate of the likelihood that each object is in each image.

Of the 49 images, the CNN identified the correct object as the highest confidence category about 35% of the time (N=17), compared to a random baseline of 0.5%. For about 88% of images (N=43), its top 10 most confident categories included the target object, relative to a baseline of 5%. Fig. 8 presents some samples of our test images. A few interesting patterns emerged in the results across object categories. For example, dumbbells, computer keyboards, bananas and screwdrivers were consistently detected reliably. Helmets, on the other hand, were confused with soccer balls and neck braces, presumably because of the similar texture patterns on these objects. Water bottles and hair spray bottles were also frequently confused, again because of similar visual appearance. The ImageNet-trained models sometimes hallucinated false positives in background clutter; hammers and filing cabinets were sometimes incorrectly found because of the similarity of their appearance to some of the fiducial markers we use for navigation.

As discussed above, we took the approach of using CNNs trained on ImageNet consumer images and applying them to our drone scenario, even 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 found that besides different backgrounds and object appearances, a key difference between the two datasets is that drone-based images have much more variation in object orientation: many objects have a canonical orientation when photographed from the ground, but can appear in a wider range of orientations from an aerial perspective. To investigate this further, we conducted another experiment in which we carefully aligned objects to appear upright, on another set of 53 images. On this set, the top-1 recognition rate increases to 43% (N=23), suggesting that rotation variation could be a significant factor. For the top-10 results, the recognition rate fell to about 70% (N=37), suggesting that most of the errors caused by orientation variation may be mostly contributing to confusion among the few most confident objects (see Fig. 5).

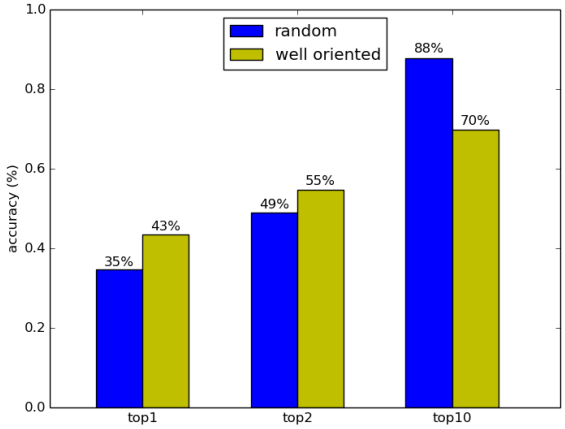

Fig. 5. Accuracy of object recognition with R-CNNs.

In future work, we plan to investigate using the parameters trained from ImageNet to bootstrap new models fine-tuned on smaller scale training datasets from our drone.

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 the previous section, and compared the speed of CNNs running on a local laptop versus those running on a remote server as a simulated cloud. A comparison of these compute facilities are shown in Table I. The cloud machine has more powerful CPUs as well as two high-end Graphical Processing Units (GPUs) that the CNN software takes advantage of.

Fig. 6 shows the running time of object recognition on

TABLE I
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

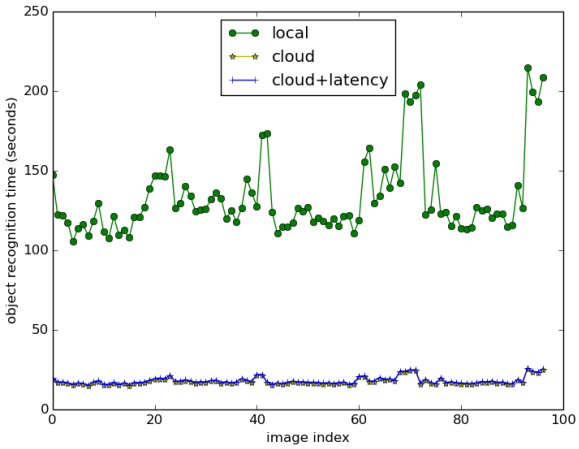

Fig. 6. Running time of object recognition on each machine.

the two machines, for each of the 97 images in our dataset. The cloud running times included 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). The average time on the cloud machine for running all 200 models on an image was 18.07 seconds, including the latencies, in contrast to the local machine which took an average of 133.14 seconds. Thus the cloud-based recognition performed about 7.4 times faster 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

Object recognition is useful in a variety of potential applications of UAVs, including search-and-rescue and reconnaissance. 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. The drone has a knowledge about possible locations of the target object, but does not know which one it is in.

In the test scenario, we put four different objects on the floor in the indoor test room, then assumed that the drone knows the four possible locations of the target object. We designed the four possible locations as the four corner points of a square in order to make the trajectory more clearly, but the objects could be put anywhere in the room. The drone approaches each possible location, using the downward-facing, lower-resolution camera for navigation and control. It then switches to hovering mode and switches to capturing images from the front-facing camera in order to capture at higher resolution and with a wider angle. The drone then takes a picture of the candidate area and sends it

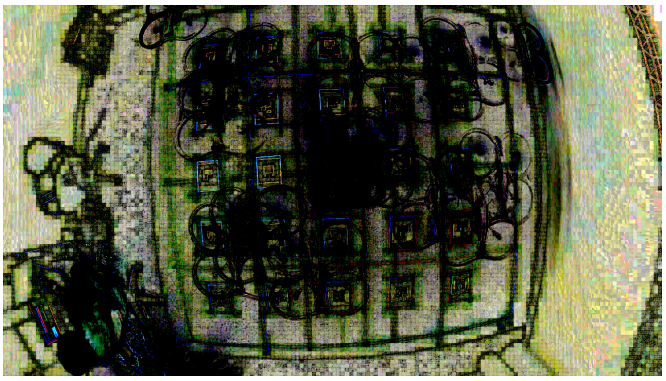

Fig. 7. Trajectory of the drone in test scenario: The drone searching for a target object in an indoor environment.

to the cloud system. Then, the drone switches the camera back to the downward-facing camera for localization and stability control, and proceeds to the other candidates. In the meantime, the cloud system performs recognition and sends results to the drone. The position with the highest confidence score for the target object is then declared to be the estimated location.

Fig. 7 shows the trajectory of the drone in our test scenario. The location where the drone stayed longer marked as black. It shows that the drone reached the four candidate area, then stayed some time to take a picture, then moved back to the center of the room according to our test scenario.

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 approach of moving the recognition to a remote computing cloud. 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.

VII. Acknowledgments

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.

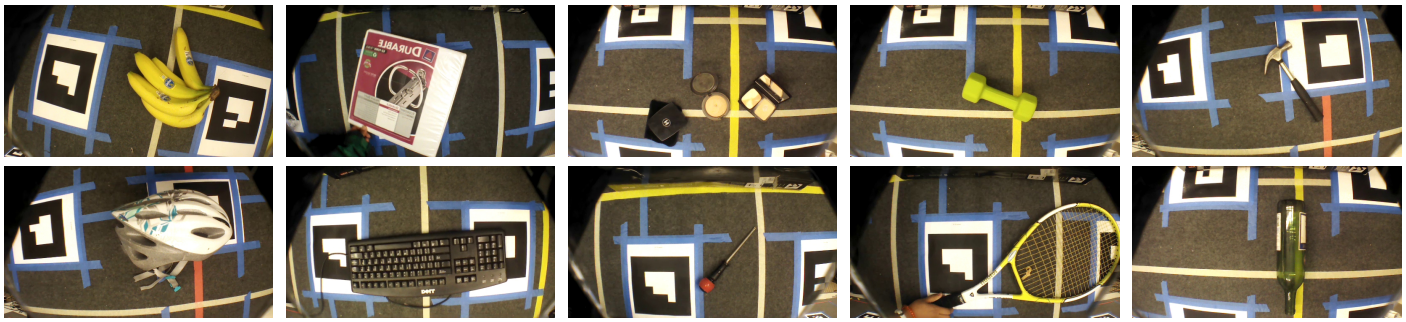

Fig. 8. Sample images collected by our drone. R-CNNs based object recognition are able to detect a wide variety of different types of objects.

REFERENCES

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

[2] P. Doherty and P. Rudol, "A UAV search and rescue scenario with human body detection and geolocalization," in *AI 2007: Advances in Artificial Intelligence,* pp. 1-13, 2007.

[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," in *Robotics & Automation Magazine, IEEE,* vol. 19, no. 3, pp. 46-56, 2012.

[4] L. Merino, F. Caballero, J. R. Martinez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires," in *Journal of Field Robotics 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 *Springer Tracts in Advanced Robotics,* pp. 107-121, 2015.

[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,* pp. 29.1-29.9, 2009.

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

[8] A. Gaszczak, T. P. Breckon, and J. Han, "Real-time people and vehicle detection from UAV imagery," in *Proc SPIE Conference Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques,* vol. 7878, 2011.

[9] H. Lim, and S. N. Sinha, "Monocular Localization of a moving person onboard a Quadrotor MAV," in *Proc. IEEE International Conference on Robotics and Automation (ICRA),* pp. 2182-2189, 2015.

[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 *Proc. IEEE International Conference on Robotics and Automation (ICRA),* pp. 111-118, 2015.

[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 *Proc. IEEE Aerospace Conference,* pp. 1-10, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 580-587, 2014.

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

[15] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *arXiv preprint arXiv:1411.4389,* 2014.

[16] 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,* pp. 2843-2851, 2012.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," in *arXiv preprint arXiv:1504.00702,* 2015.

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

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

[20] K. Goldberg and B. Kehoe, "Cloud robotics and automation: A survey of related work," in *EECS Department, University of California, Berkeley,Tech. Rep. UCB/EECS-2013-5, [Online] Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-5.html,* 2013.

[21] B. Kehoe, S.Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," in *Automation Science and Engineering,* IEEE Transactions on 12, no. 2, pp. 398-409, 2015.

[22] RoboEarth, *[online] Available: http://roboearth.org/*

[23] K. Ok, D. Gamage, T. Drummond, F. Dellaert, and N. Roy, "Monocular Image Space Tracking on a Computationally Limited MAV," in *Proc. IEEE International Conference on Robotics and Automation (ICRA),* pp. 6415-6422, 2015.

[24] K. Schauwecker and A. Zell, "On-board dual-stereo-vision for the navigation of an autonomous MAV," in *Journal of Intelligent & Robotic Systems 74,* no. 1-2, pp. 1-16, 2014.

[25] AR.Drone 2.0, *[online] Available: http://ardrone2.parrot.com/*

[26] P. J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the ar.drone micro UAV," in *18th IFAC World Congress,* vol. 18, no. 1, pp. 1477-1484, 2011.

[27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A.Y. Ng, "ROS: An open-source robot operating system," in *Proc. IEEE International Conference on Robotics and Automation (ICRA), Open-Source Software Workshop,* 2009.

[28] D. and D. Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices," in *Proc. 12th Computer Vision Winter Workshop (CVWW),* 2007.

[29] Large Scale Visual Recognition Challenge 2013 (ILSVRC2013), *[online] Available: http://image-net.org/challenges/LSVRC/2013/*

[30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM International Conference on Multimedia,* pp. 675-678, 2014.