Bandwidth efficient object recognition for drone swarms

MARCO ZOVERALLI
Semester Project at Laboratory of Intelligent Systems
École polytechnique fédérale de Lausanne

December 22, 2018

TODO: 1-page summary.

***Abstract*—abstract goes here**

I. Introduction

Object detection can sometimes suffer from non-optimal viewpoints of cameras. For instance, if the angulation of a camera hides important features of an image that represents a certain object, its prediction could be wrong: false positives and false negatives can occur. The goal of this project is to improve object recognition in the context of drones swarms, in terms of precision and recall measures. In particular, we aim at improving a system that, given a specific object, determines the presence/absence of this object. Instead of a single-host system, we want to get advantage of multiple viewpoints of different devices. We would also like to have a swarm of devices (drones) to behave as an autonomous system that is able to trigger events to more complex systems, based on whether the object is detected or not. This implies having a way to communicate with an external base station that is not part of the swarm, in order to propagate the decision of the group of devices to the external world.
One important constraint that we impose to our solution relates to the usage of bandwidth: it should be kept as low as possible. Therefore, attention will be put to the size of the exchanged messages and the transmission frequency.

A. Related Work

Distributed object recognition is a problem that has been studied extensively, under different perspectives and by considering different issues.
Rahimpour *et al.* [4] have proposed a distributed object recognition system in the context of wireless camera networks. Their approach aims at extracting features from each camera, sending them to a base station, and delegate the whole object recognition phase to this node. The problem with this solution is that a significant amount of bandwidth is consumed – even if their histogram compression and feature selection framework are based on Sparse Non-negative Matrix Factorization. Furthermore, their approach does not imply any autonomous set of devices that can trigger events.
Medeiros *et al.* [5] have proposed an algorithm to aggregate data coming from different cameras and improve object tracking. Their protocol implies using a distributed Kalman filter in order to estimate the position of the object. While this approach is interesting, object tracking is not our goal, since the focus of this project is object detection.
Giusti *et al.* [6] have proposed a mechanism to perform recognition of hand gestures through a statistical classifier and distributed consensus protoco to make the nodes of the network converge to a single decision. The decision process takes into account the quality of the prediction of each node. The quality is defined as the certainty that the predicted hand gesture corresponds to the observed one. This approach is the main source of inspiration for our system, for what concerns the communication and consensus protocol. However, the setup and the goal of their solution differ from ours: they use infrared, they rely on Support Vector Machine (SVM) classifiers, communication, and they aim at classifying an object that they assumed to be there. Instead, we rely on wifi communication, we use neural network models, and we aim at determining whether a specified object is present or absent.

II. System Setup

A general target scenario is the one shown in Figure 1. Multiple drones, each one with a camera mounted on top

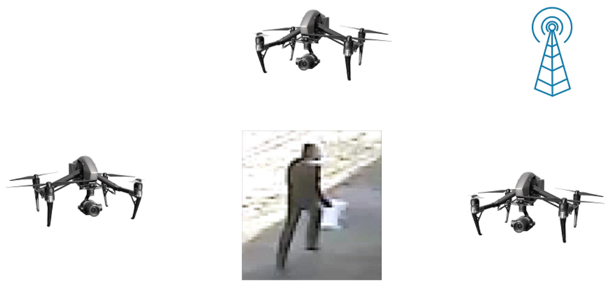


Figure 1: System setup

of it, point to the same direction and try to understand whether a target object is present. Our approach consists in having these devices communicating in order to share their opinions with each other: they all run a local object detection algorithm [1], but the object is classified as present if and only if a certain amount of predictions is positive about its presence. In particular, our design involves having a leader that collects all the predictions of the hosts. Then, data fusion occurs and the leader sends the final result to the base station. More details are provided in section IV.
Since the swarm of hosts is considered as an independent system, the communication occurs directly between hosts, without any third device as intermediary. Therefore, the obvious choice is to use an adhoc network for exchanging messages between the network nodes. In this way, there is no single point of failure. Furthermore, if broadcast is used, the number of exchanged messages is very low.
We now give additional information about the setup of our system and the hardware that was involved.

A. Hardware

1) Embedded boards: The single-board computers that were used in this project were the Odroid XU4. Thanks to the acceptable power of this device [2], it is possible

[1]A pre-trained neural network algorithm: SSD Mobilenet, trained on MSCOCO dataset
[2]https://wiki.odroid.com/odroid-xu4/hardware/hardware

to run relatively complex algorithms, including neural networks, in hundreds of milliseconds. Ubuntu MATE is the chosen operating system, which allows to easily run modern software platforms, such as various Python libraries/toolkits and Go. The board has three USB interfaces and an ethernet one. Each drone will have one of these devices attached. Any additional piece of hardware, such as connectivity modules or cameras, will be attached directly to this board.

2) Connectivity: The WiFi Module 5 will be used in order to provide wifi connectivity among the Odroid boards. The choice of this component is supported by the fact that connectivity tests have shown a positive outcome for the adhoc mode too, which is the wifi mode that we use in order to make the nodes of the network communicate.

3) Image Acquisition: Image capture is performed by the OpenMV Cam M7 camera. It easily allows the capture of images of various sizes and their transfer to the Odroid. This is the only use of this camera, although it would allow to perform some on-camera image processing algorithm. The connection with the board is done via USB.

III. Object Detection Setup

As already mentioned, we use neural networks in order to perform object detection. Although these models perform better than several classifiers, we cannot use too complex algorithms because execution time can become an issue if we consider that each drone must do continuous predictions in order to keep its opinion up-to-date and that our single-board computers do not have the computational power of state-of-the-art computer clusters. Therefore, the idea is to compensate the lack of state-of-the-art networks (but still performing!) through the presence of several devices trying to detect the same object.
The same holds for cameras: since we have a swarm we can mount cameras of less quality/weight and still get good performance thanks to the different viewpoints.

A. Image Acquisition

Two parameters that must be tuned during the image acquisition phase are the capture rate and the frame size.

1) Capture rate: it defines how often images are captured and sent to the odroid board. In order to have an accurate prediction, it would be nice to have this interval as small as possible (i.e. very high rate), so that the local view is always up-to-date. However, an important constraint relates to the prediction rate of the Odroid XU4, which should not be lower than the capture rate of the camera. In fact, this would mean sending a higher amount of data than the one that can be handled, which would result in having predictions that relate to old frames.

2) Frame size: it defines how big the taken images are. Ideally, we would like to capture images whose size is similar to the ones that were used to train the dataset, in order to have better performance. However, a larger frame size implies higher computational time as well, which can have a bad impact in terms of the ”capture rate vs prediction time” issue. Also, it depends on the camera resolution as well. In our setup we tested 64x64 and 128x128 images.

B. Model Selection and Preliminary Object Detection Tests

Among the available pre-trained models, the zoo models trained with the COCO dataset caught our attention, since they offer several alternatives in terms of performance and execution speed. We tested few of them. At fist glance, it might seem that a classification algorithm would be suitable for our task of detecting the presence of an object. However, classifiers usually handle – and are trained with – simple images with one object. In a real scenario, there would usually be several objects and the target should be detected among them: this means that a single image could contain two objects of interest. Traditional classifiers do not handle setups like this.

1) SSD Mobilenet: Performance and computation time vary depending on the image size and on the detected object. With 64x64 and 128x128 images, the average computation time on the Odroid boards is around 400ms. There is no high difference in terms of computation time, but performance is improved by 128x128 images. Also, since the model was trained with the MSCOCO dataset [3], whose samples are 640x480, better performance would be achieved with images of comparable size. This should be kept in mind for any future use of our approach: re-training the model might be necessary in order to have a functioning system. Since our goal was to develop a new technique, we used the pre-trained neural network and used it out-of-the-box. This implied having very accurate predictions for some kinds of objects 2, but not so good ones for others 3. Another consequence was that several false negatives, apparently unjustified [4], are generated for some of these detections. This is why we conducted our experiments with the classes of objects that proved to be easily detectable under ideal conditions. In order to further motivate our setup, figure 4 shows that even classes that are detected very well can be badly recognized from some points of view.

[3]http://cocodataset.org

[4]But explained by the difference in terms of images used for training and the ones taken by the cameras

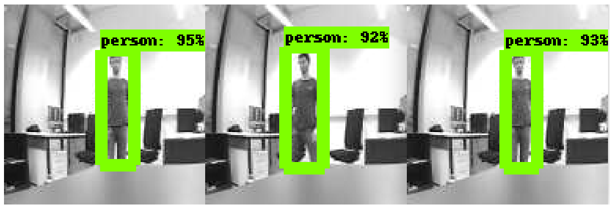


Figure 2: Preliminary test of detection of people in a noisy environment: the confidence scores are extremely high.

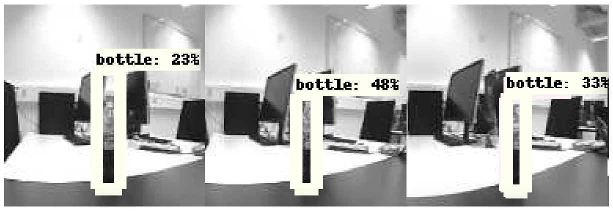


Figure 3: Preliminary test of detection of bottle in a noisy environment: the confidence scores lower than the ones that relate to people, although the quality of the view is comparable.

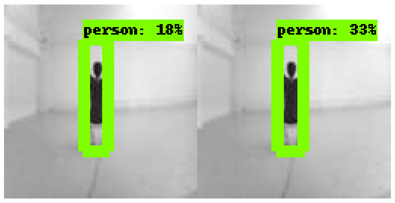


Figure 4: A person not detected very well.

2) Faster RCNN: These models represent the state of the art in terms of accuracy [1]. However, one big issue relates to the long prediction time, which is around 200ms on state-of-the-art hardware [5], and therefore it becomes unfeasable to use them on our hardware, which has much lower computational power.

C. Measures of Interest

Whenever a prediction is performed, the algorithm provides an output that contains a set of predictions. Each prediction contains the following information:

- the class of the object that was localized.
- the confidence score of the prediction.
- the area in which the object is located (i.e. the bounding box)

Differently from classification algorithms, here the output is not represented by a vector of probabilities whose sum is 1. In fact, object detection algorithms make several predictions on different regions, which relate to different objects and are therefore independend from each other: each chosen area has a classifier that runs in order to do a prediction that relates to the specified region. Although we are not interested in object tracking, one first thought that we had was that, under some assumptions, the bounding boxes might have given some useful information in order to give proper weights to different predictions. The idea was that, if we assume that an object has symmetric features, having a larger bounding box means being closer to an object. While this makes sense, the size of the bounding box is not really related to the confidence score that the model provides. This implies that noisy predictions may cause several false negatives. This is why we are considering the confidence score as the only value for the quality of predictions. As the next section IV shows, we will rely on a mechanism that considers an object as present if a subset (whose size is chosen) considers it as present.

[5]NVIDIA Tesla K40 GPU Computing Processor Graphic Cards, as mentioned in [1]

IV. Design

Designing the protocol was one of the main steps of this project. Besides being able to improve object detection, our goal is also to ensure that little bandwidth is used and that a set of devices is able to take a decision as if it was a single, independent, system. The main challenges involved are: how a single object detection prediction is obtained and how it is communicated to other hosts, what kinds of messages are transmitted, how data is aggregated and combined together, how the faults of the network nodes are handled, and how the information is transferred for external purposes. Our protocol is divided in three different phases, that involve significantly different activities:

- capture the image with the camera, and transfer it to the Odroid
- perform object detection on the Odroid, and propagate the result to the other hosts of the network
- aggregate information together in order to determine a final prediction

A. Consensus Protocol

Whenever a consensus problem is faced, there are few standard approaches that define how the nodes cooperate in order to achieve a result. One way consists in having a leader that takes the local predictions from all the nodes in the network, aggregates the results, takes a final decision, and propagates the final decisions to all the nodes in the network. An advantage of this approach is that the communication scheme is quite simple: all the nodes provide their prediction to a single node, who is the only one that needs to propagate the final result. This also implies that the bandwidth usage is kept low, since only the leader needs to see other nodes' messages. However, this solution requires that the leader is elected dynamically in case of failures (leader election). Furthermore, there should be some criteria to determine the eligibility of nodes to become leaders (e.g. the node that is able to communicate with more hosts). A second approach would be to have every node to receive all the values and computing the (same) solution. While this ensures robustness (especially against faults), it also means that unnecessary work is performed: the result is the same as the previous approach, with the difference there may be several more messages transmitted between hosts. Finally, a third approach consists in having each node to compute part of the final solution and then all the computations have to be aggregated. This approach is potentially very flexible and smart, because it allows to define very complex data fusion mechanisms. However, these mechanisms are usually very complex and guarantee better results only with some specific assumptions. For instance, many approaches make sense if the network is separated in clusters and if each node knows who is in its cluster. We chose the first approach because it is the easiest to implement and it is optimal if there is no packet loss. and there is a direct communication between all the nodes of the network. Since in real systems there are packet losses, and for scalability issues, we switch leader at each round, in order to have some spatial diversity to the network.

B. Protocol Overview

Our approach got inspired from the family of consensus protocols Paxos. However, our usage is different from the typical ones, which often aim to reach **state machine replication**. In fact, we want to get advantage of the hosts' predictions just to determine whether the target object is present or not. As a consequence, many of the constraints of classical consensus protocols are relaxed and do not need to be met. The idea is to define a majority (M) as a fraction of the total number of hosts of the network (N). The majority should express a meaningful lower bound in terms of number positive predictions that are needed in order to trigger a positive final prediction. A high majority implies having less false positives, while a low majority will cause less false negatives. We tested M=N/2+1, which is fair in terms of false positives and false negatives, and M=N/3+1, which favors false positives.

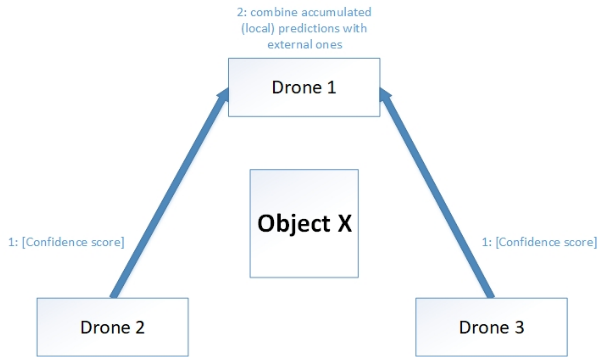


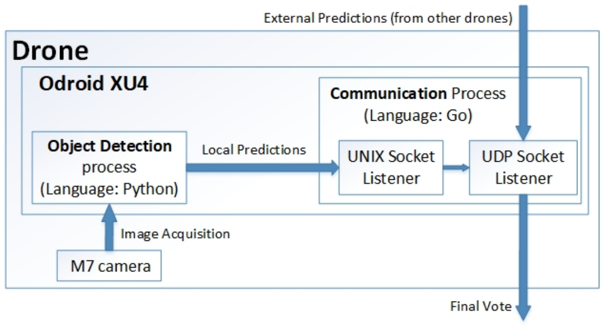
Figure 5: Inter-host communication

The decision process takes place in rounds: during a round the leader must take a decision that is based on its local prediction and the other hosts' predictions. The leader probes all the other hosts in order to receive their predictions: the transmission of the probe message is repeated periodically (T_{probe}) in order to mitigate the impact packet losses. Each host (including the leader) has a timeout ($T_{timeout}$) after which it declares the object as absent (for that round) and it advances to the next round. $T_{timeout}$ = K * T_{probe}. The leader advances to the next round after a decision for the current round is taken. This implies three cases:

- Timeout: the object is declared as absent for that round. This occurs if not enough positive predictions are received or if not all predictions are received
- Predictions from all hosts are received, but there is no positive majority: the object is declared as absent for that round.
- A majority of positive predictions is received: the object is declared as present for that round.

C. Prediction Obtainment and Inter-process communication

Once the image is captured by the camera, it is immediately transferred to the Odroid, which allows the usage of Linux and the Tensorflow library. This has a set of benefits, including a broad flexibility in terms of choice of object detection models and language for inter-host communication.
Figure 6 summarizes the computations that occur within a single host. First, the Odroid takes care of performing object detection. This model runs over a Python process, which uses Tensorflow. Then, we decided to separate the communication phase and to have another process handling it. Therefore, the prediction is forwarded to a Go process, which takes care of handling any kind of communication with the other hosts. The local communication occurs via a UNIX socket [6], which is a simple and reliable way to do inter-process communication. The communication with the rest of the networks occurs via UDP.
The reason behind Go's usage is that this language is much more robust, flexible and scalable than Python, and its concurrency [7] and networking features make it an excellent choice for building any kind of distributed system.


External Predictions (from other drones)
Drone
Odroid XU4
Communication Process (Language: Go)
Object Detection process (Language: Python)
Local Predictions
UNIX Socket Listener
UDP Socket Listener
Image Acquisition
M7 camera
Final Vote

[6]http://man7.org/linux/man-pages/man2/socket.2.html
[7]Channels: https://www.sohamkamani.com/blog/2017/08/24/golang-channels-explained/

Figure 6: Intra-host communication: this figure summarizes the leader perspective, since only the leader takes into account the external predictions, computes a final vote, and propagates it. Follower nodes behave similarly, but they do not compute/send any final vote.

D. Exchanged Messages

There are different kinds of messages that the hosts of our system handle and propagate. They ensure that a final prediction can be obtained and sent to a base station for external usage.

1) Probe: it is used to ask for the status of another host. A host propagating a Probe message expects a Status message as reply. Probes have a field indicating the round ID.

2) Status: it is used to propagate the status of the host that forwards the message. The status includes the round in which the sending host is, and the current local prediction that relates to the presence/absence of the target object.

3) Final Prediction: it contains the final prediction, after considering the other hosts' predictions as well.

4) Acknowledgement: a message for confirming the reception of a Final Prediction message.

5) Start Round: a message for forcing another host to start a new round. It contains the ID of the round that has to be started.

E. Communication Strategy

We decided to use connectionless communication between hosts (UDP). This allows us to have a lighter communication process, because there is no need to establish and mantain communications between hosts. In order to handle faults and packet losses, each node has some timeout (described above) and resynchronization mechanism (described below).

F. Propagation frequency

Since each camera sends frames quite frequently (every 400ms) to the Odroid board – which runs the object detection algorithm on each of these received frames – , having each network host propagating its decisions as soon as they are performed could cause an explosion of messages in the network. Therefore, each node propagates its decision after being probed. This implies the following advantages: drones' decisions are more reliable, bandwidth usage is reduced, and the resulting protocol is simpler to handle, test and develop.

G. Hosts synchronization

All the nodes that vote in the consensus must be sure that they are voting in the same consensus round as the others. In an ideal environment, with no crashes, faults, and delays, the leader should trigger the beginning of a new round for all the other hosts (with a Start Round message), after it takes a decision for the current round. However, the aforementioned events occur in a real environment. This is why the advancement to a new round can be triggered by other messages as well:

- Probe message: the receiver updates its status if the received message has higher Round ID that its one.
- Status message: the receiver updates its status if the received message has higher Round ID that its one. If the received message has lower Round ID, then a probe message with the current Round ID is sent back (here the leader is trying to force the follower to update its round).
- Start Round message: as described above, it forces a host to start a new round.

Leader Election: If we assume that all the nodes have good reachability with each other, it is not important who the leader is. Instead, it is important that the whole swarm does not rely always on the same node to be the leader, in order to avoid a single point of failure. For this reason, the leader is simply determined by the Round ID: each leader has a Node ID, and at round i, the leader is the node with Node ID = i % N, where N is the number of hosts in our network.

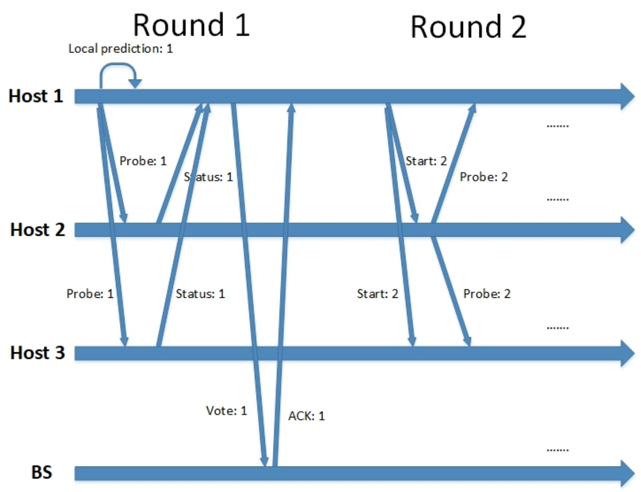


Figure 7: Example of execution of one round of the protocol, without packet losses.

V. Experiments

In this section we describe the results of our approach, by showing experiments performed on few well-known objects. We consider two different setups: one that involves a noisy environment and one that was performed in a clean environment without surrounding objects. Our experiments aim to validate different aspects of our solution:

- data fusion mechanism
- communication for the consensus protocol convergence

We separated the two procedures, because with the hardware that we had it was hard to fully validate the protocol on a real setup.

A. Data Fusion Validation

In order to validate the data fusion mechanism we compared our multi-agent system with the single-host system. The measure that we are using relates to the Precision and the Recall. Since Precision = $\frac{TP}{TP+FP}$ and Recall = $\frac{TP}{TP+FN}$, we need two different scenarios (with the same setup, in order to have consistent conditions) to evaluate them. We assumed to be in an ideal environment. This allowed us to consider an arbitrary number of devices in order to compare the two approaches (single-host system and multi-host system). Let N be the number of hosts that we consider in our system. If N=1, we are in the special case of the single-host system. Let K be the number of different views that we consider for our experiments. For each point of view, we captured some pictures with the camera while it was pointing to the object. The predictions of these pictures are aggregated in the same way as the consensus protocol does for the local prediction process. The first setup that we evaluated consisted in a clean environment, with only one object. The object could be either what we were looking for, or something else (that could be confused with what we were looking for). In this way, we could compare the two solutions in terms of false positives and false negatives. Figure 8 shows two views of the setup that we used in order to validate our method: the camera is positioned 3 meters away from the object, and multiple viewpoints are considered. In this particular setup, 9 different viewpoints are considered, on a range of 180 degrees. This means that the angular distance between two viewpoints is 22.5 degrees.

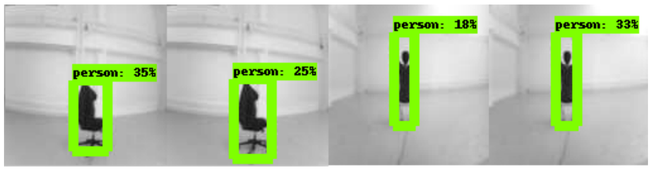


Figure 8: Setup: we are investigating the presence/absence of a person. In the two leftmost images, a chair is misclassified as a person. In the two rightmost images, a person is not recognized too well.

Figure 9 and 10 show the comparison between the single-host system (first row of both figures) and our multi-host system, with M=N/2+1 and M=N/3+1, respectively.

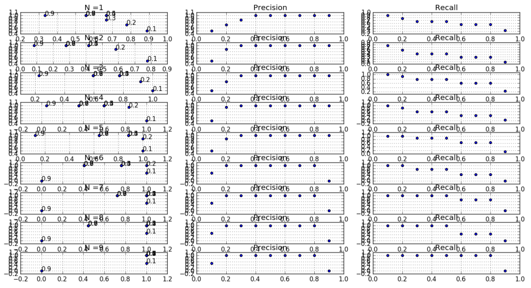


Figure 9: Majority M = N/2 + 1. This is the number of hosts that need to surpass the threshold with their prediction in order to declare the object as present.

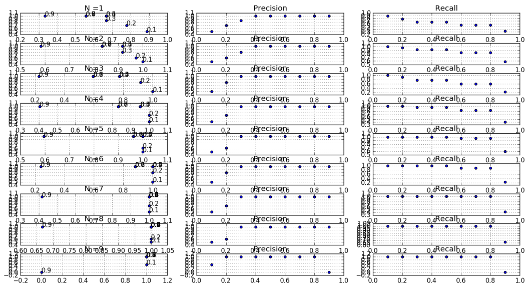


Figure 10: Majority M = N/3 + 1. This is the number of hosts that need to surpass the threshold with their prediction in order to declare the object as present.

It is interesting to notice that in both cases, under some circumstances, we can find advantages with our approach. The general trend for any system is to have a high precision for high thresholds and a high recall for low thresholds. If a high threshold is picked with the single-host system, we can see that the recalls starts degrading as soon as the threshold goes over 0.2 and 0.3, which are pretty low values. The same reasoning can be done if we decide to pick a low threshold: the recall will benefit from this choice, but the precision is very low for threshold values below 0.3. If we compare this with the multi-host approach, we can see that the choice of the threshold can be more flexible. In fact, if we pick any threshold, we can see that if the number of hosts in the system is appropriate, we can improve the precision (recall) without degrading the recall (precision) too much. For instance, if our threshold is 0.4, the single-host system will have a high precision. However, if N=1 we can see that the recall is degraded. If we pick a higher N (e.g. N=3), we can see that the recall still has an acceptable value **TODO: specify some more details (numbers) instead of "acceptable" and create the plots again (to make the more readable).

1) Single-host system: There are K samples (one per viewpoint), and the precision and recall are computed on the basis of the outcomes of the predictions on these samples, in the two distinct scenarios.

2) Multi-host system: Here, the number of samples (from which precision and recall are derived) depends on N. We started with a clean setup, sketched in figure XXX (TODO: put an image that shows the setup. Here, each sample is represented by a set of local predictions from N hosts. This means that for each scenario we have the number of samples that is equal to the combinations of K over N: $\binom{K}{N}$.

B. Communication and Consensus Test

In order to test the communication and the convergence of the consensus protocol, we used a setup like the one shown in figure 1: there are three drones that point at the center, and they are trying to detect a person. We aim at showing that having a minority of devices (one out three, in this case) that has a non-optimal viewpoint does not impact on the correctness of the system's output.
We firstly put a chair instead of a person, and we saw that two out of three devices detected the absence of a person quite accurately: their confidence score was around 10%. The third drone's confidence score fluctuated between 30% and 40%. During the experiments, we saw that the system was accurately detecting the person as absent although a threshold of 25% was being used: no false positives.
The same procedure was followed again, but with a person with outstretched arms, and the outcome was similar: two drones were correctly detecting the presence of the person (confidence score between 80% and 90%) and one drone with a lower confidence (between 50% and 60%). Again, setting the confidence threshold to 80% did not lead to false negatives.

C. Methodology

Our approach As figure XXX shows, the cameras are pointing in the same direction. Depending on the nature of the object, the detection accuracy is improved in comparison of a single viewpoint prediction.

VI. Results

TOOO

VII. Conclusion

TODO

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, June 2015.

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg *SSD: Single Shot MultiBox Detector*, December 2016.

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, April 2017.

[4] Alireza Rahimpour, Ali Taalimi, Jiajia Luo, and Hairong Qi , *Distributed Object Recognition in Smart Camera Networks*, 2016.

[5] Henry Medeiros, Johnny Park, and Avinash C. Kak, *Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks*, b.

[6] Alessandro Giusti, Jawad Nagi, Luca Gambardella, and Gianni A. Di Caro, *Cooperative Sensing and Recognition by a Swarm of Mobile Robots*, a.