Surveying Areas in Developing Regions Through Context Aware Drone Mobility

Alessandro Montanari
University of Cambridge
Cambridge, United Kingdom
alessandro.montanari@cl.cam.ac.uk

Fredrika Kringberg
KTH Royal Institute of Technology
Stockholm, Sweden
frekri@kth.se

Alice Valentini
University of Bologna
Bologna, Italy
alice.valentini7@studio.unibo.it

Cecilia Mascolo
University of Cambridge
Cambridge, United Kingdom
cecilia.mascolo@cl.cam.ac.uk

Amanda Prorok
University of Cambridge
Cambridge, United Kingdom
amanda.prorok@cl.cam.ac.uk

ABSTRACT

Developing regions are often characterized by large areas that are poorly reachable or explored. The mapping of these regions and the census of roaming populations in these areas are often difficult and sporadic.

In this paper we put forward an approach to aid area surveying which relies on drone mobility. In particular we illustrate the two main components of the approach: an efficient on-device deep learning object identification component able to capture and infer image content with acceptable performance (latency and accuracy) and a dynamic path planning approach, informed by the object identification component, which exploits potential fields to navigate between waypoints.

We report some initial performance results of the framework and describe our implementation.

CCS CONCEPTS

• **Computing methodologies** → **Robotic planning**; • **Computer systems organization** → **Robotic autonomy**;

KEYWORDS

Unmanned aerial vehicles, UAV, Autonomous vehicles, Area surveying, Convolutional neural network, Object detection, Artificial potential field

ACM Reference Format:
Alessandro Montanari, Fredrika Kringberg, Alice Valentini, Cecilia Mascolo, and Amanda Prorok. 2018. Surveying Areas in Developing Regions Through Context Aware Drone Mobility. In *DroNet'18: 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications , June 10–15, 2018, Munich, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3213526.3213532

1 INTRODUCTION

Area surveying is the detailed study or inspection of an area which employs all the essential measurements to determine the characteristics of a particular territory. Traditional area surveying through human observation can be a costly and difficult operation. This is particularly true in developing regions, where areas are often hard to reach because of limited road access, danger from wildlife or simply unknowns.

Recent developments in Unmanned Aerial Vehicles (UAVs or drones) sparked research in using small vehicles, remotely controlled or autonomous, for area surveying and remote sensing. UAV-based solutions have many benefits. The employment of drones facilitates reaching remote and dangerous areas without any risk to people. Drone surveying is also a lot more immediate as operations can be started easily with limited restrictions and thus can be carried out more frequently. For these reasons, it often constitutes a cheaper solution compared to traditional methods, such as helicopters or ground based surveys. Typical UAV applications for remote sensing include the study of morphological characteristics of a specific terrain [13, 17], creation of 3D models of the environment [15], infrastructure monitoring and many more [5].

In addition to the applications mentioned, the advantages of UAVs make them a perfect tool to monitor the movements and settlements of populations and animals in developing regions. The information gathered is very useful, for example, in emergency situations as it permits rescuers to know where to send aid or supplies like medicines, food and water. Also, if the number of people living in an area is known, rescue efforts could be scaled to the specific needs. Area survey is especially important to monitor nomad populations living in those areas. Their constant migrations require rescuers to be in possession of timely and accurate pictures of the conditions of the area and the location of the settlements. In addition, the gathered information could also be used to study the behaviour and the culture of these populations and eventually to understand how they adapt to harsh conditions.

Despite UAVs' benefits there are still several unsolved problems. Even if they are simpler to operate than other vehicles (i.e., full size airplanes or helicopters), they still require someone with enough knowledge to pilot them and to process the generated data. If we consider a typical workflow for UAV-based surveying, the two main phases are:

(1) Piloting the UAV manually or automatically to capture images of a certain area. When piloted automatically the UAV follows a predefined path (waypoint navigation) configured by the operator from a ground station controller.
(2) After the flight(s), downloading the images from the drone and processing them locally or with cloud services to produce the desired output (e.g., 3D reconstruction). Usually the results are available a few hours or days later.

The main issue in this context regards the availability of knowledgeable operators capable of efficiently flying the UAV, especially when considering developing regions. Power consumption constitutes a considerable limit as UAVs could fly only for a limited period of time and inefficient paths or maneuvers could further reduce their flight time [1]. Additionally, in applications that involve image capture, smoother flights are required to ensure high quality images [15]. While autonomous waypoint navigation could guarantee a more streamlined operation, it might miss important features on the ground because it blindly follows the pre-defined waypoints. Also the data processing usually requires trained personnel and the availability of results few days later might not be convenient when timely information is crucial, for example, in emergency situations or to monitor illegal activities.

The fundamental problem of these approaches is that they rely heavily on manual operation and the data processing is done offline. The goal of our work is to develop a system to mitigate these problems. Our vision, is to devise *a completely autonomous, efficient and affordable system which combines context sensing and on-device processing in order to autonomously generate optimal flight paths with respect to energy, dynamic goals and latency trade offs*. The system should be simple enough to be used by local populations to gather data periodically or in case of exceptional conditions. Processed data will be available immediately after each flight and could be used for various decision making processes.

In this paper we present the architecture design and initial results of our system to efficiently detect human settlements in developing regions with UAVs following an adaptive flying path. The foundation of our system consists of the on board visual perception through the use of Convolutional Neural Networks (CNNs) [11] for the accurate and robust detection of ground-level objects. The detection output is subsequently used by a reactive navigation approach based on Artificial Potential Fields [4] to dynamically adapt the flight in order to gather useful information of the environment, while keeping the flight smooth and efficient. We show also a prototype platform based on a fixed-wing aircraft and initial results for the deep learning-based perception.

2 RELATED WORK

In the context of conservation in developing regions satellite images are usually employed for mapping and monitoring of land use [3]. However, satellite sensing or airborne manned systems are costly and inaccessible, especially for researchers in developing countries. Satellite images also suffer from adverse weather conditions that might affect the quality of the images. For example, tropic areas around the equator are often covered by persistent clouds that completely prevent a clear view of the ground [9]. Ground surveys, by contrast, are typically employed for biodiversity monitoring. Nevertheless, by being expensive and time consuming they are impossible to be conducted at the proper frequency for studying animal population trends [7] and they are further limited by difficult or inaccessible terrain.

To mitigate these issues, over the years, UAVs have been used extensively for various remote sensing tasks for conservation [10]. For example, they have been used for counting various species of animals [24], for vegetation monitoring [6, 10] and to gather illegal hunting evidence for law enforcement [20]. However, the issue with these systems is that they are not fully autonomous and do not take complete advantage of the benefits offered by unmanned vehicles. In fact, in these works UAVs have been flown with pre-programmed flight paths and the images gathered have been inspected and analysed manually by researchers. This results in a time consuming process which needs trained personnel in order to be accomplished, limiting the usability for local researchers.

Other works explored the possibility of automatic detection of salient features in the images. Van Gemert et al. analysed methods to automatically count animals in aerial images [23]. The authors concluded that approaches relying on deformable part-based models (DPM) should be suitable for UAV applications given their reduced computational cost but offer a limited detection speed (up to 5 frames per second). However, they did not benchmark these algorithms but reported only previous results. Recent works considered thermal cameras to overcome some of the limitations imposed by traditional ones (i.e., simplify the distinction of objects from the background and monitoring during the night). In this area, Longmore et al. evaluated a traditional computer vision technique which makes use of Histogram of Oriented Gradients (HOG) and Support Vector Machines (SVM) to detect cows and humans in thermal images [12]. However, even in this case the authors did not report any result about the speed of the detection and how the systems would run on an actual aircraft. Thermal images have also been used by Bondi et al. who developed a system to detect humans (e.g., poachers) using convolutional neural networks (Faster RCNN) running on a laptop or on the Azure cloud service [2]. This approach has two main issues: firstly, the UAV needs a constant and robust link with the base station in order to send video frames to be processed, a condition that might be difficult to satisfy and considerably limits the vehicle's maximum range. Secondly, the detection speed (up to 3 frames per second) is limited and not sufficiently high to support autonomous navigation.

The system we are proposing here by contrast, takes advantage of modern deep-learning and hardware architectures in order to perform object detection completely onboard, without any connection with a base station. This way the maximum range of the UAV is only limited by its endurance (e.g. battery capacity). Additionally, we use the detection of objects of interest to autonomously and efficiently adapt the flight path of the UAV in order to gather additional images of those objects which could be useful at a later stage for additional analysis. These two characteristics make our system simple to use also by non-expert people because it simply involves launching the UAV and gathering the results once it is back from the autonomous mission.

Only one work evaluated the YOLOv2 and TinyYOLO architectures on the Nvidia Jetson TX2, as we do in this work [22]. The

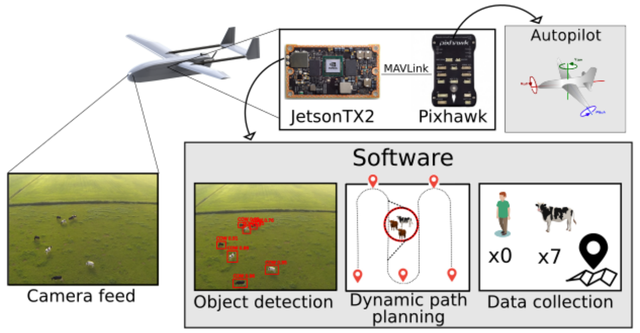

Fig. 1: System overview. The software uses the camera feed as input to perform object detection. When an object is detected, its location is used by the path planning module to dynamically alter the UAV's path. Data (such as the number of detected objects and their location) is collected, along with images captured during the flight, for later analysis.

authors were able to achieve a higher frame rate for both architectures (up to 12fps for TinyYOLO) because they adopted an optimised C implementation while we run Keras and Tensorflow using Python. We plan to move to a more efficient implementation in the near future. However, the authors did not consider how to use the detections for the autonomous navigation of the aircraft as we do in this work, where we propose a comprehensive system.

3 SYSTEM ARCHITECTURE

In this section we describe our approach in detail. At a glance, the system is composed of two main components: *(i)* the image capturing and inference component and *(ii)* the autonomous and dynamic flight planning which is informed by *i*). The UAV will perform autonomous flight with the purpose of gathering information and images of the territory. Perception and sensing of ground-level objects will be used to take instant decisions regarding path planning. Specifically, the UAV will detect and locate people or cattle present on the ground using deep learning techniques. The path planning component is informed by the object identification and dynamically alters the path of the UAV with the overall objective of increasing the information gained within the area. An overview of the system architecture is shown in Figure 1.

This type of approach based on real-time detection has the benefit of running autonomously without relying on cloud computation or network coverage, which is often unattainable in rural and inaccessible areas. However, low latency in the end-to-end system is essential to successfully perform the task. Therefore, the system was designed with the aim to perform object detection with a speed of $\sim$ 10 Hz. Furthermore, the embedded system was optimized with respect to weight and computational power.

We now describe the two main components of the system in detail.

3.1 Deep Learning-based Perception

In order to accurately and reliably detect people and animals in frames from the onboard camera, we directed our attention towards convolutional-based models for object detection. Object detection is a computer vision task that provides the coordinates of objects in the frame along with their class and confidence. The result is commonly visualized with a bounding box, as shown in figure 1. Our choice was motivated by the performance of these models; they proved to have very good accuracy results, outperforming previous years' approaches, since the first implementations [8, 21]. Additionally, given the availability of sufficient training data, these architectures benefit from the possibility of being trained end-to-end, from raw pixels to bounding boxes, removing the need for hand-crafted feature extraction and hence simplifying their application to different detection tasks.

The main issue that prevented the use of convolutional neural networks on small and lightweight UAVs, so far, is their requirements both in terms of memory and processing power. This is particularly true for object detection networks which are typically more complex than networks used only for classification. Usually these models run on large clusters or data centres using multiple graphic cards (GPUs). However, the power consumed by these high-end cards is in the order of hundreds of Watts[1] and therefore are prohibitive for a fixed-wing UAV that usually has a power consumption in the order of tens of Watts. Additionally, their big form factor limits the usability on small aircrafts where space and weight are premium.

Our system requires a network design which provides a good trade off between detection accuracy, speed and resources usage. High detection speed is required since the object detection is used to autonomously adapt the path during flight. At the same time the onboard computer running the architecture has to be limited in size and therefore also in computational power, meaning that it might not be able to run large networks. These two requirements of having small and efficient but accurate networks are interdependent given that smaller networks generally run faster but are less accurate.

Considering our constraints we decided to opt for the architecture YOLO v2 (You Only Look Once) proposed by Redmon et al. [18], which represents the state-of-the-art in terms of accuracy and detection speed. The model, also called Darknet-19, is in fact composed of a reduced number of convolutional layers (19) and only 5 max-pooling layers. The final layer predicts both class probabilities and bounding boxes coordinates from the feature map given in output by the previous layers. YOLO was designed to be simpler than approaches based on region proposal [19]. Region based algorithms scan the image for possible objects generating region proposals. Every region proposal is then run over a convolutional neural network for extracting the features which are then fed into a classifier. Computing thousands of region proposals and then run CNN over them is computationally expensive, making the detection process too slow for real-time processing. YOLO overcomes this limitation by treating the detection problem as a regression one and running a CNN over the input image just once.

A reduced version of YOLO v2, TinyYOLO was also implemented [18]. TinyYOLO is based off of the Darknet reference network. It is composed by fewer convolutional layers (9) and fewer filters in each layer. This structure allows TinyYOLO to run much faster at a cost of a lower detection accuracy. We considered both YOLO v2 and TinyYOLO as candidates for the real-time object detection pipeline (more details about initial results are in Section 5).

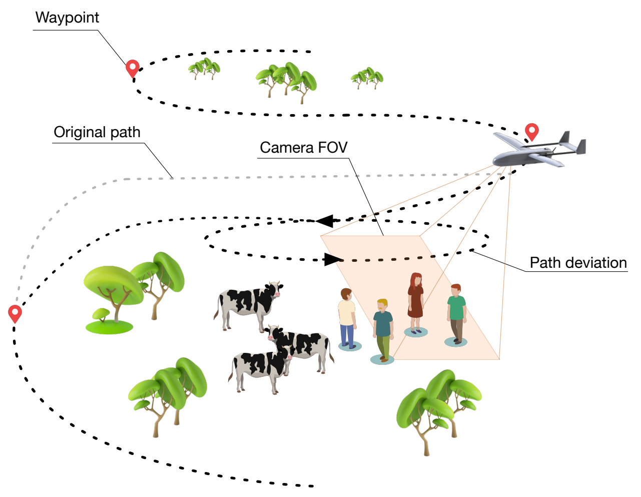

Fig. 2: Approach for dynamic path planning. The UAV deviates from its pre-defined path to circle around the detected objects to maximise the information gained about them. Then it continues the mission to explore additional areas.

3.2 Reactive Navigation

The just described object identification system is pivotal to the driving of the dynamic path planning component of the system which we now illustrate. To efficiently gather information of the area, the UAV must dynamically adapt its path at runtime. In the event of detected people or cattle, the UAV will fly closer to the objects to collect more images, and then continue to monitor the area. The proposed solution has the following outline (figure 2):

(1) The user defines the area that shall be surveyed.
(2) Waypoints are calculated to cover the entire area, since there is no prior information of the whereabouts of people and cattle. The waypoints generate an area coverage path that the UAV follows unless interesting objects are detected.
(3) If objects are detected by the camera their position in the frame is mapped to real world coordinates. If there is more than one object in a single frame, we can ensure that the objects are densely located and therefore, a weighted average of their locations is used.
(4) The UAV adjusts its heading towards the location of the object. Once the location is reached the UAV circles around the objects to gather information and take images for later analysis. Once sufficient information has been gathered, the UAV continues towards the next waypoint.
(5) Once a waypoint has been reached, the remaining flight time is estimated based on the battery voltage. If there is sufficient battery power to go to the next waypoint and back to the starting position, the drone will go to the next waypoint. Otherwise, it will return to launch.

The chosen approach uses an artificial potential field [4] for navigation, a method that models the configuration space as a field of forces. The UAV moves towards the point in the field with lowest potential, and moves away from points with high potential. This method was chosen due to its computational speed and ability to model different aspects of the system, such as waypoints, detected objects and the motion constraints of the prototype platform.

Waypoints and detected objects are modeled as points with low potentials, and will influence the UAV with an attractive force. Given the location of the UAV $q = [x, y]^\top$ and a goal location, such as a waypoint or a detected object, $q_{goal} = [x_{goal}, y_{goal}]^\top$, the attractive potential is given by the function

$$U_{att}(q) = \begin{cases} \frac{1}{2}\zeta d^2(q, q_{goal}) & d(q, q_{goal}) \leq d^* \\ d^*\zeta d(q, q_{goal}) - \frac{1}{2}\zeta d^{*2} & d(q, q_{goal}) > d^* \end{cases} \tag{1}$$

where ζ is a scaling factor and $d(q, q_{goal})$ is the distance between the UAV's location q and a goal location q_{goal}. Here, a quadratic function is used for locations closer to the goal than a distance d^* and a conical function for locations further away than d^*. The quadratic function is needed because of the discontinuity of the conical function at the goal position.

Obstacles and other places that are undesirable to visit are modelled with a repulsive potential, according to the equation

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{d(q, q_{obs})} - \frac{1}{Q^*})^2 & d(q, q_{obs}) \leq Q^* \\ 0 & d(q, q_{obs}) > Q^* \end{cases} \tag{2}$$

where η is a scaling factor and $d(q, q_{obs})$ is the distance between the UAV's location and the obstacle. When the UAV is within a radius Q^* from the obstacle, it is affected by the repulsive force.

The total potential of the UAV's location in the field is given by the sum of the attractive and repulsive potential, i.e.

$$U(q) = U_{att}(q) + U_{rep}(q). \tag{3}$$

The force affecting the UAV is inversely proportional to the gradient of the potential;

$$F = -\nabla U(q). \tag{4}$$

The UAV moves towards the minimum point in the field using gradient descent. Objects detected by the vision system are given a larger scaling factor than the initial waypoints, ensuring detected objects to be prioritized over the predefined path.

When the UAV has reached an object and gained sufficient information by circling around it, the location of the object is given a repulsive potential, making the UAV to exit the circular path and continue surveying the area following the planned waypoints. The repulsive potential also prevents the UAV from returning to the same object at a later stage.

4 EXPERIMENTAL PLATFORM

This section describes the hardware and software used to realize the functionality of the UAV.

Firstly, an appropriate airframe was chosen. UAVs are divided into two main categories; fixed-wing and multirotor. Fixed-wing UAVs are less maneuverable than multirotors, but superior in endurance and flight time. Due to the lift force generated from the wings, all thrust from the motor can be used for forward propulsion. This makes them energy efficient, and suitable for surveying larger areas. The airframe used in this project is a X-UAV Mini Talon, shown in Figure 3. It has a generous wing area that promotes long endurance

[1]Maximum 250 W for Nvidia Tesla P100 [16]

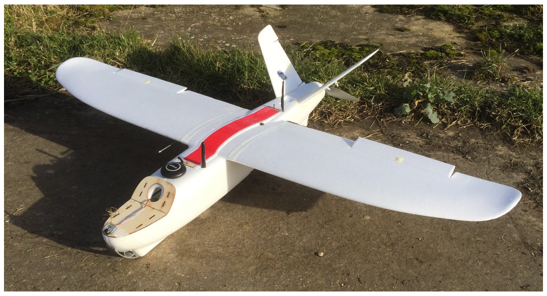

Fig. 3: The selected airframe, X-UAV Mini Talon.

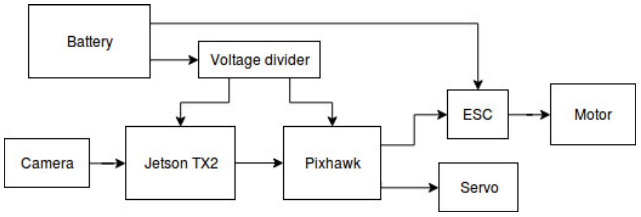

Fig. 4: Hardware overview. The lines correspond to the physical cabling of the system.

flight and a large fuselage with space for extra components, allowing embedded applications.

To achieve autonomous flight with on-device computation, additional hardware was used. As shown in Figure 4, in the end-to-end system the camera provides a stream of images to the embedded Jetson TX2 platform, which runs object detection and path planning. The flight controller (Pixhawk) uses the Ardupilot[**?**] code base to execute a low level controller for path execution as well as stabilize the UAV and reads inputs from GPS and onboard sensors to estimate the UAV's location. The flight controller outputs signals to the electric speed controller (ESC), which provides the correct current to the motor, and to the servos that are used to keep the control surfaces in the right position. The system is powered by a 7500 mAh li-ion battery to support long endurance flight, and a voltage divider is used to supply the correct voltage to the components.

The embedded platform is a NVIDIA Jetson TX2[2], chosen for its computational power and support for deep learning technologies. It is equipped with a NVIDIA Pascal-family GPU which makes it suitable for running image inference in real time. The deep learning models were implemented and deployed using Keras[3] with TensorFlow[4] backend.

Future work focuses on full system integration. Robot Operating System[5] (ROS) is a framework for robotics applications, which will be used for integrating the object detection, path planning and flight control. Communication between the Jetson TX2 and the Pixhawk will use the MAVLink[6] protocol, a lightweight and redundant library for UAVs. The aim of the system integration is to maintain a low latency in the end-to-end system, so that navigation decisions can be made during flight.

[2]https://developer.nvidia.com/embedded/buy/jetson-tx2
[3]https://keras.io/
[4]https://tensorflow.org/
[5]https://ros.org/
[6]https://mavlink.io/

5 PRELIMINARY RESULTS

This section reports preliminary results about the performance of the object detection networks (see Section 3.1 for details) on NVIDIA Jetson TX2 with respect to framerate, measured in frames per second (fps), and mean average precision (mAP).

5.1 Model Training

The deep learning models (YOLO v2 and TinyYOLO) were trained with a dataset composed by two classes; cattle and people. The training dataset was selected in order to reflect the environment where the UAV will operate. In other words, the selected images were taken from a UAV's perspective, rather than from a satellite or from the ground. The class *cattle* was represented by the Verschoor Areal Cow Dataset [23]. After manual removal of incorrect samples, the dataset contained 4809 images. For the class *people*, the UAV123 dataset [14], was used. Also this dataset required modifications; several images contained people for whom the corresponding ground truth bounding boxes were not available, therefore these images were discarded. This resulted in a dataset containing 11309 images with one person each.

Both models were trained with an image input resolution of 416×416 in order to keep the size of the network small with the objective of achieving the fastest possible speed. The imgaug[7] library for Python was used to implement real-time augmentation during training, in order to artificially generate new images starting from the existing ones with the purpose of improving the model's capability to generalize and prevent overfitting. The transformations were chosen in order to reflect all the possible UAV's operating cases. They include: scale, rotation, translate, change in brightness, blurness or contrast and Gaussian noise addition.

The anchor boxes, i.e, the prior bounding boxes used to simplify training and hence the final accuracy [18], were recomputed on our dataset, using k-means clustering, to ensure they reflect the sizes of the objects we want to detect. Five anchor boxes were used for both models.

5.2 Model Testing

For our application we are interested in evaluating two aspects of the object detection models: their accuracy in detecting objects on the ground and the speed at which they can process images (i.e., frames per second). The latter is particularly important because the convolutional network should not introduce excessive latency which might negatively impact the responsiveness of the navigation module.

The detection accuracy was tested using a dataset disjoint from the training set containing 1612 images. The metric we used for this evaluation is the mean over all classes (two in our case) of the Average Precision [**?**]. This is a common metric used for the evaluation of object detection methods and it is usually called *mAP*.

The detection speed was tested by deploying the Keras+Tensorflow models on the Jetson TX2, feeding 500 still images of size 416×416 pixels to the network and measuring the average time needed to process an image. This evaluation was performed for both networks, one at the time, and while the Jetson was mounted on the developer kit,

[7]https://github.com/aleju/imgaug

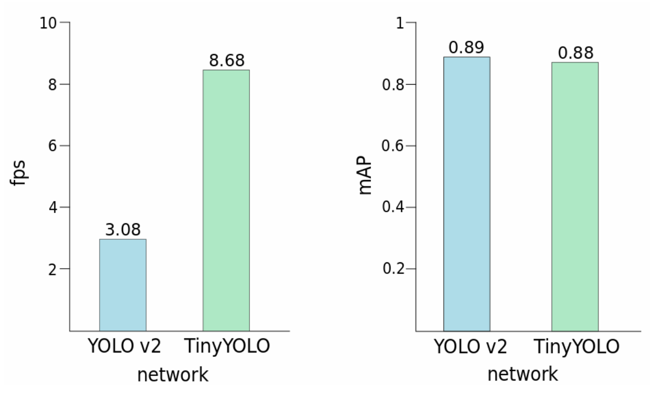

Fig. 5: Comparison of YOLO v2 and TinyYOLO.

hence not inside the aircraft. Jetson TX2 has five different frequency modes that were tested. By choosing a frequency model, the user can make a tradeoff between speed and power consumption. In the results presented below, the mode with the highest GPU frequency was used to attain maximum inference speed. The average fps and mAP for YOLO v2 and TinyYOLO are presented in Figure 5.

As seen in the figure, the highest framerate was 8.68 fps, achieved by the TinyYOLO network. TinyYOLO is almost three times faster than YOLO v2, but it achieved similar results in terms of accuracy. Therefore, TinyYOLO is the preferable network to use, due to the strict latency constraints of our application. For a more complex detection task, e.g. with more classes, the performance in terms of accuracy may differ more, and the larger network might be needed. Other aspects that might be tuned to improve the detection accuracy are the input image size and the number of anchor boxes. We leave the exploration of these parameters for future work.

As mentioned in the previous sections, the desired framerate for our system is $\sim$ 10 fps. The achieved results are slightly lower, but it could still be considered enough for real-time detection applications. When compared with results from similar works, such as in [22] where a framerate of almost 12 fps was reached, we conclude that the detection rate could be improved, if required by the latency constraints.

6 CONCLUSION

We have presented a framework for dynamic drone mobility with two main components: an efficient object identification system which uses deep learning on drone collected images on device and a dynamic path planning component which relies on potential fields to incorporate the inference information of the object detection component dynamically to inform the path planning. Future work include the full integration of the path planning component and large scale testing of the framework.

ACKNOWLEDGEMENTS

We want to greatly thank the NGO Africa Mission – Cooperation and Development (www.africamission.org) for the discussions that led to the ideation and refinement of this project. The project was partially funded through an Institutional GCRF EPSRC grant.

REFERENCES

[1] G. A. Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.
[2] E. Bondi et al. Spot poachers in action: Augmenting conservation drones with automatic detection in near real time. 2018.
[3] M. Broich et al. Time-series analysis of multi-resolution optical imagery for quantifying forest cover loss in sumatra and kalimantan, indonesia. *International Journal of Applied Earth Observation and Geoinformation*, 13(2), 2011.
[4] H. Choset et al. *Principles of robot motion*. MIT press, 2005.
[5] I. Colomina and P. Molina. Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92, 2014.
[6] Q. Feng et al. UAV remote sensing for urban vegetation mapping using random forest and texture analysis. *Remote Sensing*, 7(1), 2015.
[7] T. A. Gardner et al. The cost-effectiveness of biodiversity surveys in tropical forests. *Ecology letters*, 11(2), 2008.
[8] R. Girshick et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.
[9] M. C. Hansen et al. A method for integrating modis and landsat data for systematic monitoring of forest cover and change in the congo basin. *Remote Sensing of Environment*, 112(5), 2008.
[10] L. P. Koh and S. A. Wich. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. *Tropical Conservation Science*, 5(2), 2012.
[11] Y. LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.
[12] S. Longmore et al. Adapting astronomical source detection software to help detect animals in thermal images obtained by unmanned aerial systems. *International Journal of Remote Sensing*, 38(8-10), 2017.
[13] F. Mancini et al. Using unmanned aerial vehicles (UAV) for high-resolution reconstruction of topography: The structure from motion approach on coastal environments. *Remote Sensing*, 2013.
[14] M. Mueller et al. A benchmark and simulator for uav tracking. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016.
[15] F. Nex and F. Remondino. UAV for 3D mapping applications: a review. *Applied geomatics*, 6(1), 2014.
[16] NVIDIA. *NVIDIA Tesla P100 GPU Accelerator*, 2016.
[17] S. Rathinam et al. Autonomous searching and tracking of a river using an uav. In *American Control Conference*. IEEE, 2007.
[18] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.
[19] S. Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015.
[20] R. Schiffman. Drones flying high as new tool for field biologists, 2014.
[21] P. Sermanet et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.
[22] N. Tijtgat et al. Embedded real-time object detection for a uav warning system. In *ICCV2017, the International Conference on Computer Vision*, 2017.
[23] J. C. van Gemert et al. Nature conservation drones for automatic localization and counting of animals. In *Workshop at the European Conference on Computer Vision*. Springer, 2014.
[24] C. Vermeulen et al. Unmanned aerial survey of elephants. *PloS one*, 8(2), 2013.