SPOT Poachers in Action: Augmenting Conservation Drones with Automatic Detection in Near Real Time

Elizabeth Bondi[1], Fei Fang[2], Mark Hamilton[3], Debarun Kar[1], Donnabell Dmello[1], Jongmoo Choi[1]
Robert Hannaford[4], Arvind Iyer[4], Lucas Joppa[3], Milind Tambe[1], Ram Nevatia[1]
[1]University of Southern California, Los Angeles, CA, 90089 USA
{bondi, dkar, ddmello, jongmooc, tambe, nevatia}@usc.edu
[2]Carnegie Mellon University, Pittsburgh, PA, 15213 USA, feifang@cmu.edu
[3]Microsoft, Redmond, WA, 98052 USA, {marhamil, lujoppa}@microsoft.com
[4]AirShepherd, Berkeley Springs, WV, 25411 USA
rob@coolideassolutions.com, arvind.iyer@lindberghfoundation.org

Abstract

The unrelenting threat of poaching has led to increased development of new technologies to combat it. One such example is the use of long wave thermal infrared cameras mounted on unmanned aerial vehicles (UAVs or drones) to spot poachers at night and report them to park rangers before they are able to harm animals. However, monitoring the live video stream from these conservation UAVs all night is an arduous task. Therefore, we build SPOT (Systematic POacher deTector), a novel application that augments conservation drones with the ability to automatically detect poachers and animals in near real time. SPOT illustrates the feasibility of building upon state-of-the-art AI techniques, such as Faster RCNN, to address the challenges of automatically detecting animals and poachers in infrared images. This paper reports (i) the design and architecture of SPOT, (ii) a series of efforts towards more robust and faster processing to make SPOT usable in the field and provide detections in near real time, and (iii) evaluation of SPOT based on both historical videos and a real-world test run by the end users in the field. The promising results from the test in the field have led to a plan for larger-scale deployment in a national park in Botswana. While SPOT is developed for conservation drones, its design and novel techniques have wider application for automated detection from UAV videos.

Introduction

Poaching has recently been on the rise, particularly poaching of elephants and rhinoceroses in Africa (Great Elephant Census 2016). With elephant and rhino numbers dropping rapidly, it is imperative that we swiftly act before they are hunted to extinction. Multiple strategies exist to combat poaching, including park ranger patrols, and more recently, the use of unmanned aerial vehicles (UAVs or drones) (Ivošević et al. 2015). In particular, UAVs equipped with long wave thermal infrared (hereafter referred to as thermal infrared) cameras can be used for nighttime surveillance to notify park rangers of poaching activity because there is increased poaching activity at night, and because animals and humans are warm and emit thermal infrared light even at night. However, the video stream from these UAVs must be monitored at all times in order to notify park rangers of poachers. Monitoring of streaming footage is an arduous task requiring human supervision throughout the night, and is also prone to systematic lapses in quality as human detection often degrades with fatigue (Porikli et al. 2013). Furthermore, as more drones are added to the system, more resources are required to monitor the additional videos.

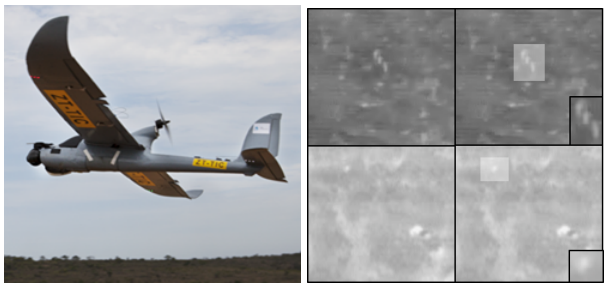


Figure 1: Example UAV and thermal frames from UAV, with white boxes surrounding poachers.

Whereas previous work in AI has focused on game theory for patrol planning (Xu et al. 2017; Wang, Zhang, and Zhong 2017) and machine learning-based poaching prediction (Gholami et al. 2017; Critchlow et al. 2015) to assist human patrollers in combating poaching, little effort has focused on decision aids to assist the UAV crew in detecting poachers and animals automatically. Given the tedious work of monitoring UAV videos, such a decision aid is in high demand. It could help reduce the burden of the monitoring personnel and the probability of missing poachers by simply notifying personnel or park rangers of a detection. In the future, the decision aid could also be integrated with existing tools that predict poaching activity and guide human patrols. For example, the system could scout ahead for poachers to protect park rangers, monitor in other directions than human patrollers, or gather more information about the location of wildlife for better predictions. The integration would lead to a new generation of machine learning and game theoretic tools to guide rangers and UAVs simultaneously.

In building this decision aid, there are several major challenges. First, automatic detection in thermal infrared videos

captured aboard UAVs is extremely difficult, because (i) the varying altitude of the UAV can lead to extremely small humans and animals, possibly less than 20 pixels in the images, (ii) the motion of the UAV makes stabilization, and consequently human and animal motion detection, difficult, and (iii) the thermal infrared sensor itself leads to lower resolution, single-band images, much different from typical RGB images. Second, we must provide notification in near real time so the UAV can immediately start following humans in order to provide park rangers with current locations. Real-time detection is an especially difficult challenge because we have limited computing power and Internet in the field.

In this paper, we present SPOT (Systematic POacher deTector), a novel AI-based application that addresses these issues and augments conservation drones with the ability to automatically detect humans and animals in near real time. In particular, SPOT consists of (i) offline training and (ii) online detection. During offline training, we treat each video frame as an image, and use a set of labeled training data collected for this application (Bondi et al. 2017) to fine-tune a model which has shown success in detecting objects of interest in images, Faster RCNN. During online detection, the trained model is used to automatically detect poachers and animals in new frames from a live video stream, *showing that modern computer vision techniques are capable of conquering difficulties that have not been addressed before*.

We also use a series of efficient processing techniques to improve the online detection speed of SPOT in the field. Online detection can be completed either on the cloud or on a local computer. Therefore, we have experimented with several architectures that trade off between local and remote computers, depending on network strength. Finally, we evaluate SPOT on both historical videos and a real-world test run in the field by the end users, a conservation program called AirShepherd (AirShepherd 2017). The promising field test results have led to a plan for larger-scale deployment, and encourage its use in other surveillance domains.

Problem Domain and Current Practice

Conservation programs such as AirShepherd (AirShepherd 2017) send crews to fly UAVs (Fig. 1) in national parks in Africa, including Liwonde National Park in Malawi and Hwange National Park in Zimbabwe, in order to notify park rangers of poaching activity. Teams of people are required for UAV missions, including several UAV operators and personnel capable of repairing the UAVs should anything happen. The UAV is a fixed-wing aircraft with a range of 50 km and a flight time of 5 hours with one battery. It carries a FLIR 640 Vue Pro thermal infrared camera. The UAV flight path is pre-programmed based on typical poaching hotspots or tips. While flying at night, the UAV operators monitor the live video stream, transmitted via radio waves, for any signs of poachers. Should anyone be spotted, the team will manually take control to follow the suspects, notify nearby park rangers, who are sometimes on patrol or in a van with the team, and guide them to the poachers.

However, as we already mentioned, monitoring these videos all night is difficult. Several example frames from thermal infrared videos are shown in Fig. 1, with objects of interest highlighted in transparent white boxes on the right. Notice that these frames are grayscale, with few pixels on objects of interest and many objects that look similar to those of interest. It is often difficult for humans to recognize objects in these videos because of this, leading to recognition errors and hours of tedious work. As such, there is great need for a tool that automatically detects poachers and animals, the objects of interest in these videos for conservation. This tool should provide detections with as much accuracy as possible in near real time speeds on a laptop computer in the field with a potentially slow Internet connection.

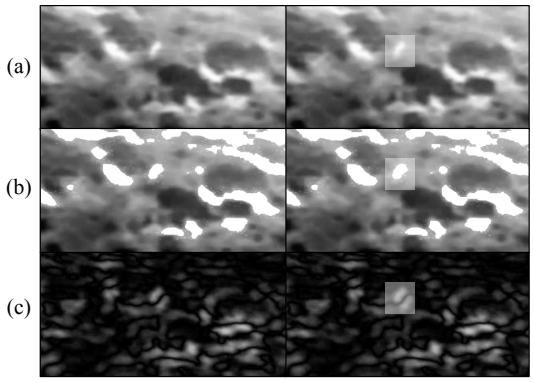


Figure 2: Traditional computer vision techniques. (a): original image, (b): thresholded, where white pixels are above the threshold, (c): stabilized frame difference. Original results (left), manually added transparent white boxes around true poachers (right). These figures illustrate the difficulty these techniques face in locating poachers.

There has been some effort towards automatic detection. EyeSpy (Hannaford 2017), the application that is used in current practice, detects moving objects based on edge detection. When in use, it first asks the monitoring personnel to provide parameters such as various edge detection thresholds and sizes of humans in pixels. EyeSpy then requires information such as altitude and camera look angle throughout the flight to complete detection. Three limitations restrict the use of this tool as a result. First, EyeSpy relies heavily on a well-trained expert who can manually fine-tune the parameters based on the UAV and camera information. Novices are often unable to find the correct settings. Second, the parameters need to be compatible with flight altitude and camera look angle. To make this tool usable, the UAV crew either needs to restrict the way the UAV flies by keeping the flight altitude and camera look angle almost the same throughout the mission, or have the expert monitoring personnel manually adjust the parameters from time to time as the settings change. Third, this tool cannot differentiate between wildlife and poachers, and thus cannot highlight the detection of poachers to the monitoring personnel or the patrol team. We will examine this tool further in Evaluation.

Related Work and Design Choices

We arrive at the current framework of SPOT after several rounds of trials and errors. As humans and animals are typ-

ically warmer than other objects in the scene, and consequently brighter, we first consider automatic thresholding techniques such as Otsu thresholding (Otsu 1979). However, other objects such as vegetation often have similar digital counts and lead to many false positives (Fig. 2(b)). Because humans and animals tend to move, we also consider motion using algorithms such as the Lucas-Kanade tracker for optical flow (Lucas, Kanade, and others 1981) and general correlation-based tracking (Ma et al. 2015). Again, other objects such as vegetation look similar to the objects we want to track, which often leads to lost or incorrect tracks (Fig. 1). Assuming a planar surface, small moving objects can also be detected by a background subtraction method after applying video stabilization (Pai et al. 2007). Motion is unfortunately detected incorrectly by this method in the case of complex terrain such as tall trees (Fig. 2(c)). More complex algorithms to track moving objects throughout videos rely on high resolution, visible spectrum videos or videos taken from a fixed camera (Kristan et al. 2015; Milan et al. 2016).

Given the limitations of these traditional computer vision techniques and the great strides in object detection using convolutional neural networks, we turn to deep learning-based approaches. We treat each frame of the video as an image, and apply techniques to localize and classify the objects of interest in the images. Faster RCNN and YOLO (Ren et al. 2015; Redmon et al. 2016) are two state-of-the-art algorithms suitable for this purpose. They both propose regions automatically for classification. Faster RCNN tends to have higher accuracy than YOLO, particularly for smaller objects, although YOLO tends to be faster (Redmon et al. 2016). A newer version, YOLOv2, (Redmon and Farhadi 2016), has improved performance over YOLO and could be used as an alternative to Faster RCNN. In this work, we focus on using Faster RCNN for detection.

Other emerging techniques such as deep learning-based optical flow or tracking (Zhu et al. 2017; Fan and Ling 2017) may fail due to drastic UAV motion and low resolution frames, and they do not classify the objects, only localize. Tubelets (Kang et al. 2017) propose bounding boxes over time, but are not yet performing in real time even on GPUs. Recently, there has also been some work on automatic wildlife detection and counting based on videos from UAVs using other traditional computer vision or machine learning techniques, but they either rely on RGB images in high resolution (Olivares-Mendez et al. 2015) or do not consider real-time detection (van Gemert et al. 2014). Due to the unique challenges of our problem, these techniques cannot be applied to detecting poachers during flights at night.

SPOT

Overview

SPOT includes two main parts: (i) offline training and (ii) online detection (Fig. 3). In this section, we introduce both parts in detail, with an emphasis on the robust and faster processing techniques we use to improve the online detection efficiency and provide detections in near real time.

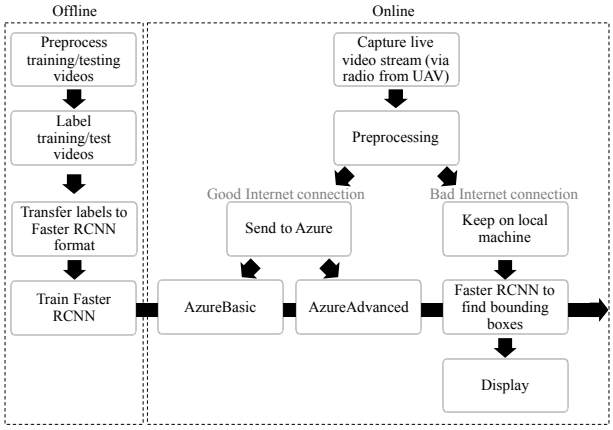


Figure 3: SPOT Overview.

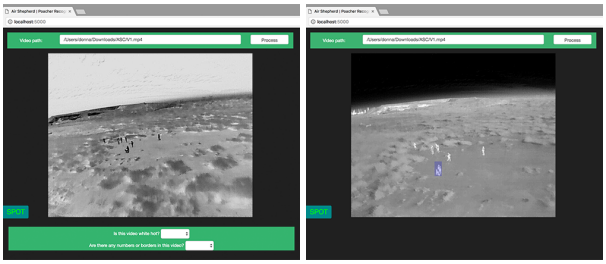
Offline Training

In our problem, detection means to localize the objects of interest in the scene, and classify them as poachers or animals. We choose a state-of-the-art object detection algorithm, Faster RCNN, to serve our purpose. Faster RCNN is composed of a region proposal network (RPN) and Fast Region-based Convolutional Network method (Fast RCNN) (Girshick 2015), which is used to classify regions from the RPN, thereby giving us the location and class of objects. The RPN shares the convolutional layers of Fast RCNN, which is VGG-16 (Simonyan and Zisserman 2014) in our system.

To train the Faster RCNN model, we first initialize the VGG-16 network in the Faster RCNN model with pre-trained weights from ImageNet. Then, we use a set of videos in this application domain with annotated labels for each frame, collected using a framework described in (Bondi et al. 2017). A small team of students (not Amazon Mechanical Turk users in order to protect sensitive information such as flight locations and strategies) used this framework to label all frames in 70 videos containing animals and poachers. Because consecutive frames are similar, we do not have enough heterogeneous data samples to train VGG-16 from scratch. This is the reason we start with pre-trained weights and fine-tune VGG-16 by treating each video frame as a separate image. Furthermore, we fine-tune different models for poacher and animal detection, so that depending on the mission type, whether monitoring a park for poachers or counting animals, for example, the user may choose a model to provide better detection results. For the poacher-specific model, we fine-tuned using 4,183 frames, and for the animal-specific model, we used 18,480 frames, as we have more animal videos.

Online Detection

Preprocessing Thermal infrared images can be "black-hot" or "white-hot", meaning warm objects are darker or lighter, respectively. During the online detection, we first ask the user if the video is white-hot, and if the answer is no, we will invert every frame we receive from the UAV. In addition, there is occasionally a border or text on the videos, consisting of date, flight altitude, and other metadata. We ask users



(a) (b)

Figure 4: GUI created for SPOT for use in the field. 4(a): inquiries about video, 4(b): detection.

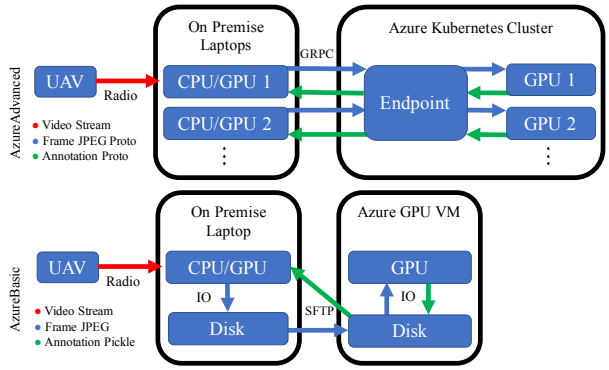


Figure 5: AzureBasic and AzureAdvanced overview.

to provide the area of interest at the beginning and only display detections in this area of interest throughout the flight.

Detection We treat each frame of the video stream as an image and input it to Faster RCNN. The trained model computes regions and classes associated with each region.

User Interface Fig. 4 shows the user interface of SPOT for online detection. A file can be selected for detection, or a live stream video. In Fig. 4(a), we gather preprocessing information about the video, and then begin detection in Fig. 4(b).

Architectures and Efficiency Faster RCNN runs at 5 frames per second (fps) on a K40 GPU (Ren et al. 2015). Efficiency and computation speed are paramount for similar performance in the field where there may be limited computing power, especially since videos are 25 fps. We therefore examine different Microsoft Azure architectures (Fig. 5), and discuss techniques to improve performance in the field and trade off between local and remote compute.

The first and simplest cloud architecture we investigate, which we will refer to as AzureBasic, is an NC-6 Series Virtual Machine (VM) with a Tesla K80 NVIDIA GPU hosted on Microsoft Azure. We simply transfer frames from the local laptop to this VM using Paramiko, a Python SFTP client. Once frames are transferred to the remote machine, we detect objects in the frame using our stored, fine-tuned Faster RCNN model in a running Python instance on the remote machine. We then display the annotated frame locally using X forwarding. For the purposes of testing, we send frames in batches, and we use Paramiko to transfer annotated frames instead of displaying. Speed could be improved by transferring annotations instead of annotated frames.

Although AzureBasic allows us to improve our throughput through cloud GPU acceleration over a CPU laptop, it is limited to a single Azure GPU VM and a single local laptop linked together by SFTP. To scale out SPOT, we utilize Tensorflow Serving, a framework for efficiently operationalizing trained Tensorflow computation graphs. Tensorflow Serving provides a way to evaluate Faster RCNN without the overhead of a running Python instance and file IO from SFTP. Furthermore, Tensorflow Serving communicates through Protocol Buffers, a flexible and efficient data representation language that significantly reduces the size of large tensors. For serving scenarios with large requests and responses, such as video processing, this reduces network communication and improves performance on slow networks. Tensorflow Serving also supplies tools for creating multi-threaded clients. We use four threads for our testing. Like AzureBasic, we also process images in batches to ensure that there is no downtime between uploading frames and downloading the results. Finally, we use azure-engine to create a cluster of NC-6 series GPU VMs managed with Kubernetes, a fault tolerant load balancer for scalable cloud-based services. This keeps the latency of SPOT low in potential compute intensive multi-UAV scenarios. It also provides a single REST endpoint so the client code can use a single web URL for sending images regardless of the number of machines in the cluster. We deploy on a GPU-enabled docker image with Tensorflow Serving, and add tools for convenient re-deployment of models hosted on Azure Blob Storage. We refer to this architecture as AzureAdvanced.

Evaluation

To provide a working prototype system, SPOT needs to meet two major criteria: (i) detection accuracy and (ii) efficiency. Detection accuracy is most important for poacher identification, particularly to make sure we have few false negatives and false positives. Speed is critical to being able to quickly notify monitoring personnel and the ranger team. In this section, we evaluate SPOT in the lab using six historical videos, consisting of 15,403 frames in total, as test video streams. We will first evaluate the performance of the object detection, and then the efficiency, where we compare the different methods discussed in earlier sections.

EyeSpy (Hannaford 2017), the application that is used in current practice, requires users to tune eight parameters to correctly identify objects of interest, plus six flight metadata parameters such as altitude and camera angle. Because of so many parameters, it is often difficult to successfully tune all of these parameters as a novice. On the other hand, our application does not require the user to fine-tune any parameters – it can be used as is. We therefore consider EyeSpy as used by a novice (ESN). Of our six test videos, only the three animal videos have average flight metadata records (i.e., not flight metadata per frame). For analysis of ESN, we use flight metadata parameters if present, and make educated guesses for altitude if not, because this is the baseline only. Otherwise, we utilize default values for all parameters. We

Table 1: Precision-Recall for SPOT and EyeSpy Novice (ESN) for animals.

	Precision		Recall	
Video	SPOT	ESN	SPOT	ESN
SA	0.5729	0.1536	0.0025	0.0072
MA	0.1497	0.0008	0.0073	0.0004
LA	0.5584	0.0235	0.2293	0.0694

Table 2: Precision-Recall for SPOT and EyeSpy Novice (ESN) for poachers.

	Precision		Recall	
Video	SPOT	ESN	SPOT	ESN
SP	0	0.00003	0	0.0007
MP	0.0995	0.0004	0.0073	0.0009
LP	0.3977	0.0052	0.0188	0.0159

also include results from EyeSpy as used by an expert (ESE). These parameters are adjusted by our collaborators at AirShepherd who created EyeSpy. We do not make educated guesses for ESE because a lack of exact parameters could drastically reduce performance of EyeSpy, which would not be a fair comparison. We record the output from EyeSpy, which is a video with red outlines around objects of interest, and place bounding boxes around any red outlines obtained. We then use an IoU threshold of 0.5 as is typical in (Ren et al. 2015). Finally, we choose a low confidence threshold for SPOT because missing a poacher detection is extremely undesirable, and we report the precision and recall.

We compare the performance of SPOT and ESN on videos containing animals or poachers with labels of small, medium, or large average sizes in Tables 1 and 2. We also compare the performance of SPOT and ESE in Table 3. We perform better than the novice in both precision and recall for medium- and large-sized poachers and animals. We also perform better than the expert for large-sized animals, and comparably for small- and medium-sized animals. Because we perform better than ESN and similarly to ESE, we thus reduce significant burden. For small poachers, which is a challenging task for object detection in general, both tools perform poorly, with EyeSpy being able to identify a small number of poachers correctly. Small animals also prove to be a challenge for SPOT. To improve performance for small objects in the future, we expect pooling the results of video frames and incorporating motion will be beneficial.

Next, we evaluate efficiency by comparing CPU performance to the initial Azure system, to the improved Azure system, and finally to the single GPU performance. The GPU laptop is a CUK MSI GE62 Apache Pro, with Intel Skylake i7-6700HQ, 32GB RAM, and the NVIDIA GTX 960M with 2GB RAM. It is deployed in the field. The CPU laptop has an Intel i5-3230M CPU at 2.60GHz. In order to compare the Azure systems, we time how long it takes from the frame being sent to Azure, to the prediction, to the return back to the local machine, and finally to reading the final image back into memory. We conducted these tests in two different networking environments: 533.20 Mbps upload and 812.14 Mbps download, which we will call "fast", and 5.33 Mbps upload and 5.29 Mbps download, which we will call "slow". We repeat the experiment for several images and show the final time per image in Table 4. The results show that both AzureAdvanced and the GPU laptop perform detection almost 100 times faster than the CPU laptop, and AzureAdvanced drastically improves over AzureBasic when a slower network is present. Therefore, we can achieve detection in near real time.

Table 3: Precision-Recall for SPOT and EyeSpy Expert (ESE) for animals.

	Precision		Recall	
Video	SPOT	ESE	SPOT	ESE
SA	0.5729	0.6667	0.0025	0.0062
MA	0.1497	0.1615	0.0073	0.0014
LA	0.5584	0.0433	0.2293	0.0832

Table 4: Timing Results for CPU, AzureAdvanced (AA), AzureBasic (AB), and GPU.

	# GPUs	Network	s/img
CPU	0	-	10.4354
AB	1	fast	0.5785
AB	1	slow	2.2783
GPU	1	-	0.3870
AA	2	fast	0.3484
AA	2	slow	0.4858

Implementation in the Field

We also evaluate the in-field performance of SPOT. So far, these tests have been run by AirShepherd at a testing site in South Africa, where training exercises take place. Fig. 6 shows a screenshot from a 30 minute test of AzureBasic at the site. For a full video, sped up 20 times, please visit http://bit.ly/SPOTVideo. Precision and recall results are shown for this in Table 5, which shows that SPOT performs better than both ESN and ESE. Our collaborators at AirShepherd reported that SPOT performed poacher detection well during this test flight, and was so promising that they want to move forward with further development and deployment in Botswana. They also showed excitement because SPOT requires no tuning from the user. Although the network connection was poor for some of the flight and caused detection to occur slowly, especially because AzureBasic was used, AzureAdvanced will perform better in these situations, and the GPU laptop can now provide consistent detection speeds with slow networks, which our collaborators found encouraging as well. With the promising results from the field test, a wider deployment is being planned.

Lessons Learned and Conclusion

In conclusion, we developed a system, SPOT, to automatically detect poachers as well as animals in thermal infrared UAV videos taken at night in near real time, which shows that modern computer vision techniques are capable of con-

Table 5: Precision-Recall for SPOT, EyeSpy Novice (ESN), and EyeSpy Expert (ESE) for poachers in test video.

Precision			Recall		
SPOT	ESN	ESE	SPOT	ESN	ESE
0.4235	0.0024	0.0573	0.3697	0.0432	0.2836

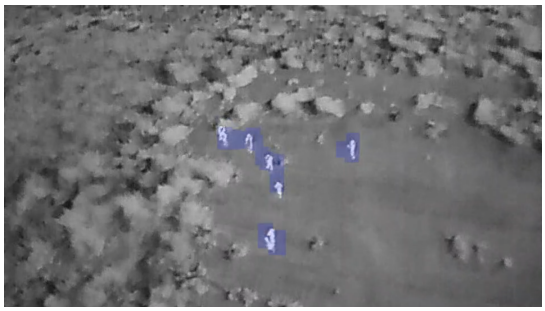


Figure 6: A screenshot of the field test environment with annotated figures.

quering difficulties that have not been addressed before. This system works in varying situations and does not require the users to adjust any parameters when they use it. Thus, it is easily accessible to non-expert users. Furthermore, the system can detect poachers in near real time with either good or bad network connectivity. The system has been tested in the field, and will be deployed in the near future in several national parks in Africa, including one in Botswana. SPOT opens the door for exciting new research questions in object detection in difficult videos, and for new anti-poaching strategies utilizing UAVs in the field.

Acknowledgments

This work was supported by UCAR N00173-16-2-C903, primary sponsor Naval Research Laboratory (Z17-19598). It was partially supported by the Harvard Center for Research on Computation and Society Fellowship and the Viterbi School of Engineering Ph.D. Merit Top-Off Fellowship.

References

AirShepherd. 2017. Airshepherd: The lindbergh foundation. `http://airshepherd.org`. Accessed: 2017-09-11.

Bondi, E.; Fang, F.; Kar, D.; Noronha, V.; Dmello, D.; Tambe, M.; Iyer, A.; and Hannaford, R. 2017. Viola: Video labeling application for security domains. In *GameSec*.

Critchlow, R.; Plumptre, A.; Driciru, M.; Rwetsiba, A.; Stokes, E.; Tumwesigye, C.; Wanyama, F.; and Beale, C. 2015. Spatiotemporal trends of illegal activities from ranger-collected data in a ugandan national park. *Conservation biology* 29(5):1458–1470.

Fan, H., and Ling, H. 2017. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. *arXiv preprint arXiv:1708.00153*.

Gholami, S.; Ford, B.; Fang, F.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; and Mabonga, J. 2017. Taking it for a test drive: A hybrid spatio-temporal model for wildlife poaching prediction evaluated through a controlled field test. In *ECML PKDD*.

Girshick, R. 2015. Fast r-cnn. In *ICCV*, 1440–1448.

Great Elephant Census. 2016. The great elephant census, a paul g. allen project. Press Release.

Hannaford, R. 2017. Eyespy. Private Communication.

Ivošević, B.; Han, Y.-G.; Cho, Y.; and Kwon, O. 2015. The use of conservation drones in ecology and wildlife research. *Ecology and Environment* 38(1):113–188.

Kang, K.; Li, H.; Xiao, T.; Ouyang, W.; Yan, J.; Liu, X.; and Wang, X. 2017. Object detection in videos with tubelet proposal networks. *arXiv preprint arXiv:1702.06355*.

Kristan, M.; Matas, J.; Leonardis, A.; Felsberg, M.; Cehovin, L.; Fernández, G.; Vojir, T.; Hager, G.; Nebehay, G.; and Pflugfelder, R. 2015. The visual object tracking vot2015 challenge results. In *ICCV Workshops*, 1–23.

Lucas, B. D.; Kanade, T.; et al. 1981. An iterative image registration technique with an application to stereo vision.

Ma, C.; Yang, X.; Zhang, C.; and Yang, M.-H. 2015. Long-term correlation tracking. In *CVPR*, 5388–5396.

Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; and Schindler, K. 2016. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

Olivares-Mendez, M. A.; Fu, C.; Ludivig, P.; Bissyandé, T. F.; Kannan, S.; Zurad, M.; Annaiyan, A.; Voos, H.; and Campoy, P. 2015. Towards an autonomous vision-based unmanned aerial system against wildlife poachers. *Sensors* 15(12):31362–31391.

Otsu, N. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* 9(1):62–66.

Pai, C.-H.; Lin, Y.-P.; Medioni, G. G.; and Hamza, R. R. 2007. Moving object detection on a runway prior to landing using an onboard infrared camera. In *CVPR*, 1–8. IEEE.

Porikli, F.; Bremond, F.; Dockstader, S. L.; Ferryman, J.; Hoogs, A.; Lovell, B. C.; Pankanti, S.; Rinner, B.; Tu, P.; and Venetianer, P. L. 2013. Video surveillance: past, present, and now the future [dsp forum]. *IEEE Signal Processing Magazine* 30(3):190–198.

Redmon, J., and Farhadi, A. 2016. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *CVPR*, 779–788.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 91–99.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

van Gemert, J. C.; Verschoor, C. R.; Mettes, P.; Epema, K.; Koh, L. P.; Wich, S.; et al. 2014. Nature conservation drones for automatic localization and counting of animals. In *ECCV Workshops (1)*, 255–270.

Wang, B.; Zhang, Y.; and Zhong, S. 2017. On repeated stackelberg security game with the cooperative human behavior modelfor wildlife protection. In *AAMAS*.

Xu, H.; Ford, B.; Fang, F.; Dilkina, B.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; and Nsubaga, M. 2017. Optimal patrol planning for green security games with black-box attackers. In *GameSec*.

Zhu, Y.; Lan, Z.; Newsam, S.; and Hauptmann, A. G. 2017. Guided optical flow learning. *arXiv preprint arXiv:1702.02295*.