Fast, Deep Detection and Tracking of Birds & Nests

Qiaosong Wang Christopher Rasmussen Chunbo Song

University of Delaware, Dept. Computer & Information Sciences
cer@cis.udel.edu

Abstract. We present a visual object detector based on a deep convolutional neural network that quickly outputs bounding box hypotheses without a separate proposal generation stage [1]. We modify the network for better performance, specialize it for a robotic application involving "bird" and "nest" categories (including the creation of a new dataset for the latter), and extend it to enforce temporal continuity for tracking. The system exhibits very competitive detection accuracy and speed, as well as robust, high-speed tracking on several difficult sequences.

1 INTRODUCTION

Visual object detection is a complex task which entails recognizing, localizing, and counting objects within an image. The human ability to rapidly detect natural objects in a scene has long been studied in neuroscience and cognitive psychology [2], but this task is particularly challenging for computers. Until recently, the best-performing detectors for objects such as people and cars used combinations of handcrafted image features such as histograms of oriented gradients [3, 4].

Our motivation in this paper is not general object detection, but rather to rapidly and accurately detect and track *birds* and *bird nests* in forest scenes for a environmental robotic application. "Bird" is a category in the well-known PASCAL VOC dataset [5], a widely-used benchmark in visual category classification, detection, and segmentation. However, there is very little previous work on visual bird tracking or bird nest detection: [6] applies morphological analysis to analyze overhead images of poultry, [7] using saliency methods on visible-wavelength and infrared images to find ground nests in agricultural fields, and [8] uses shape analysis to find nests as outliers on power poles adjacent to high-speed rail.

In the last few years, standard detection pipelines have been dramatically outperformed by *deep learning* representations. Deep convolutional neural network (CNN) architectures such as [9, 10] are able to generate high-level image representations that are effective for a variety of tasks. However, most CNN-based object detectors operate either in a sliding window fashion [11] or by generating object "proposals" separately [12, 13] and then evaluating these hypotheses. These approaches can achieve good results on difficult detection benchmarks, but are typically fairly slow and not well-suited to real-time deployment when compared to very fast and accurate special-purpose detectors (such as for pedestrians [14] and traffic signs [15]) built with other machine learning methods.

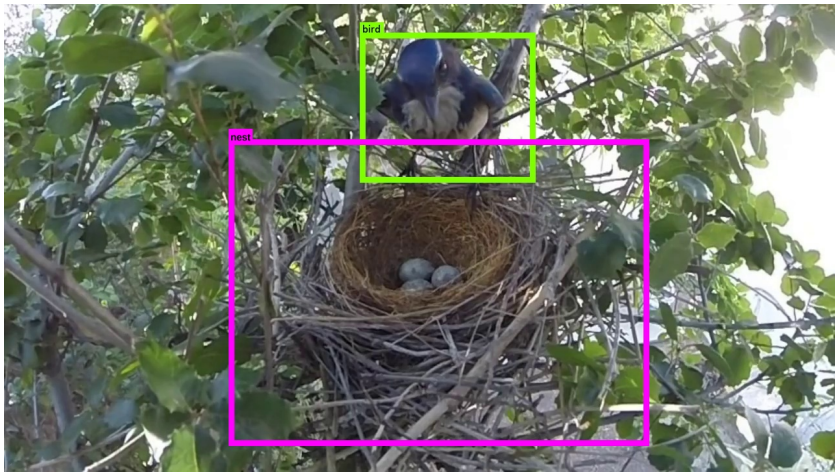


Fig. 1. Sample bird and nest detections from a test video sequence

More recently, it has been shown that current CNNs have sufficient power to represent geometric information for localizing objects, opening the possibility of building state-of-the-art object detectors that rely exclusively on CNNs free of proposal generation schemes [1, 16, 17]. In such approaches, the network is trained end-to-end to predict both the appearance and geometric information of an object. At test time, given an input image, the entire network is only evaluated once instead of evaluating at different locations and scales of the image, enabling a large speed-up.

Inspired by these examples, in Sec. 2 we build on the general-purpose "YOLO" detection network [1], which exhibits excellent accuracy and runs at up to 150 Hz by directly outputting detection bounding boxes with confidences. We improve upon the original network by making several modifications, and specialize it by training on only our two classes "bird" and "nest." To this end, we contribution a new dataset for nest detection, described in Sec. 4.1.

The speed of the detector permits it to be integrated into a real-time tracker. One advantage of a deep CNN tracker vs. most standard template-based trackers [18] is "automatic" initialization: because it has an *a priori* class concept, it can find the object(s) itself, and refind it/them if occlusions or mistracking occurs. There has been some recent work on applying deep learning techniques to visual tracking, or so-called "deep tracking" [19–21], but these are still relatively slow. In Sec. 3 we extend the baseline YOLO detector to improve the temporal smoothness of the localization estimate while retaining robustness to object appearance and pose changes.

2 DETECTION

We adapt the 24-layer YOLO network [1] for detection tasks, which we term $\mathbf{YOLO_{B+N}}$ ("YOLO Birds + Nests"). $\mathrm{YOLO_{B+N}}$ has approximately the same architecture as the GoogLeNet proposed by [10], except that the inception modules are replaced by 1×1 reduction layers $+$ 3×3 convolutional layers. The full network structure is shown in

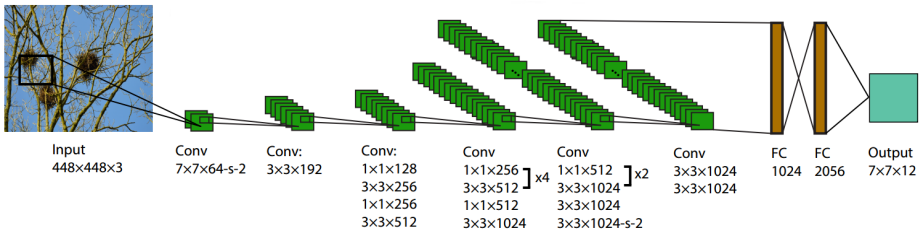


Fig. 2. $\mathrm{YOLO_{B+N}}$ detection pipeline (here $S = 7$, $B = 2$, and $C = 2$)

Fig. 2: it takes a raw input image, resizes it to 448×448, and outputs the size and location of bounding boxes for all C classes.

The resized input image is divided into an $S \times S$ grid of cells, each of which contains information on B hypothetical object bounding boxes. Each bounding box is parametrized by a 5-D vector $[x, y, w, h, P(Obj)]$, where $P(Obj) = 1$ if the center of any ground-truth object bounding box is inside the cell and $P(Obj) = 0$ otherwise. Each grid cell also includes a conditional class probability: $\Pr(c \mid Obj)$, where $c \in \{C\}$. Accordingly, the class-specific confidence is given by: $P(c) = Pr(c \mid Obj)P(Obj)$. For each presented image, the output layer of the network is an $S \times S \times (B * 5 + C)$ tensor. Non-maximal suppression is used to remove duplicate detections, followed by thresholding on $P(c)$.

Our modifications are as follows. First, during the training stage, except the final layer which uses a linear activation function ϕ, all other layers in $\mathrm{YOLO_{B+N}}$ use *softplus* activation [22]: $\phi_{\mathrm{YOLO}_{B+N}}(x) = ln(1+e^x)$. This gives a smoother approximation than the leaky activation function in the original implementation in [1]. Second, in [1] a term in the network loss function containing the square roots of the bounding box width and height is used to address the fact that small deviations in large boxes should weigh less than in small boxes. We got better results using normalized coordinates to equally weigh errors between large and small boxes:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(\frac{w_i - \hat{w}_i}{\hat{w}_i}\right)^2 + \left(\frac{h_i - \hat{h}_i}{\hat{h}_i}\right)^2 \tag{1}$$

3 TRACKING

A "naive" $\mathrm{YOLO_{B+N}}$ tracker consists of running the detector on each successive frame independently. As seen in Table 1, this approach surpasses a number of recent trackers benchmarked in [18]. However, it still misses detections in isolated frames, and the localization is a little noisy, suggesting the introduction of a temporal filter.

We accomplish this by running $\mathrm{YOLO_{B+N}}$ and a template-based tracker simultaneously, combining them to create a hybrid detector-tracker which we call $\mathbf{TrackYOLO_{B+N}}$. For the single-object tracker here, we use the very fast kernelized correlation filter (KCF) [23] (coded as "CSK" in [18]), which maintains a trained linear classifier for all frames since last initialization. The two threads are combined as follows:

– When $\mathrm{YOLO_{B+N}}$ first detects an object, KCF is initialized using the highest-probability detected bounding box
– Let $B^t_{\mathrm{YOLO_{B+N}}}$ and B^t_{KCF} denote the output bounding boxes at time t of the base detector and the template tracker, respectively, where $B = [x, y, w, h]$
– Let $\Delta(t) = ||B^t_{\mathrm{YOLO_{B+N}}} - B^t_{KCF}||$ be a "disagreement" measure for each frame that $\mathrm{YOLO_{B+N}}$ has at least one detection, and define a threshold $\epsilon = 0.5 \times \max(W/S, H/S)$ where $W \times H$ are the image dimensions and S is the YOLO detection grid size
– If $\Delta(t) < \epsilon$, the hybrid tracking solution is a linear combination $B^t_{\mathrm{TrackYOLO_{B+N}}} = \lambda_1 B^t_{\mathrm{YOLO_{B+N}}} + \lambda_2 B^t_{KCF}$. Else, clear the training buffer of KCF and fall back to the detector $B^t_{\mathrm{TrackYOLO_{B+N}}} = B^t_{\mathrm{YOLO_{B+N}}}$
– Finally, if $\mathrm{YOLO_{B+N}}$ does not detect an object, the template tracker alone is used: $B^t_{\mathrm{TrackYOLO_{B+N}}} = B^t_{KCF}$

4 RESULTS

4.1 Data sets

Bird nests We collected 114 images from the web, each containing at least one nest from a variety of species, for a total of 169 nest instances[1]. A wide range of scales were included, from close-ups to very distant views, and image resolutions ranged from 500×333 pixels to 4000×3000. Some examples (with results overlaid) can be seen in Fig. 4(a).
Birds We used the *bird* object category from the 2012 PASCAL VOC dataset [5], a widely-used benchmark in visual category classification, detection, and segmentation. VOC 2012 has 20 classes; there are 765 images containing birds in the *trainval* portion of the data, with 1,165 bird instances present. Samples are in Fig. 4(b).
Tracking Neither birds nor nests are in standard tracking benchmarks [18], so we prepared several image sequences from YouTube videos; 3 are presented here. For each, we manually chose ground truth bounding boxes every 10 frames and linearly interpolated them to generate annotations for all frames. The first sequence has 540 frames at 1920×1080 resolution. It is taken from a ground-based camera and the dominant motion is a zoom-in on a distant nest. The second sequence has 310 frames at 1280×720, and is from a drone flying around a tree containing a large bird nest. The third sequence has 564 frames at 854×450, and is a pan to follow a single bird flapping in front of a complex background. Samples from these sequences can be seen in Fig. 4(c)-(e).

4.2 Detection

We set the following parameters for $\mathrm{YOLO_{B+N}}$ training: batch size = 64, momentum = 0.6, decay = 0.001, learning rate = 0.0001, iterations = 5,000. For robustness, we perform data perturbation during training via random scaling and translations of up to 30% of the original image size and random adjustment of the exposure and saturation of the image by up to a factor of 2 in the HSV color space. Our model is pre-trained

[1] Full nest dataset available here: `http://nameless.cis.udel.edu/data/nests`

Table 1. mAP %, time on detection, tracking datasets

	Nest	Bird	Ground nest	UAV nest	Flying bird	s / im
$\mathbf{YOLO_{B+N}}$	97.9	77.2	36.8	62.8	27.8	0.07
ImageNet-CNN	34.5	18.2				32.0
DenseBox [26]		28.8				≥ 2
YOLO [1]		57.7				0.07
ResNet [27]		84.8				≥ 2
$\mathbf{TrackYOLO_{B+N}}$			63.8	45.6	77.4	0.07
KCF ("CSK") [23]			15.9	54.6	74.3	0.001
CT [28]			19.9	59.8	81.4	0.03
SCM [29]			17.5	6.0	75.3	9.61

on the 1000-class ImageNet classification training set [9] and fine-tuned on the VOC 2012 trainval set containing only bird images and half of the nest dataset. For testing, detection threshold on P(c)$= 0.2$, and the correctness threshold on *Intersection over Union* (IoU) $= 0.5$.

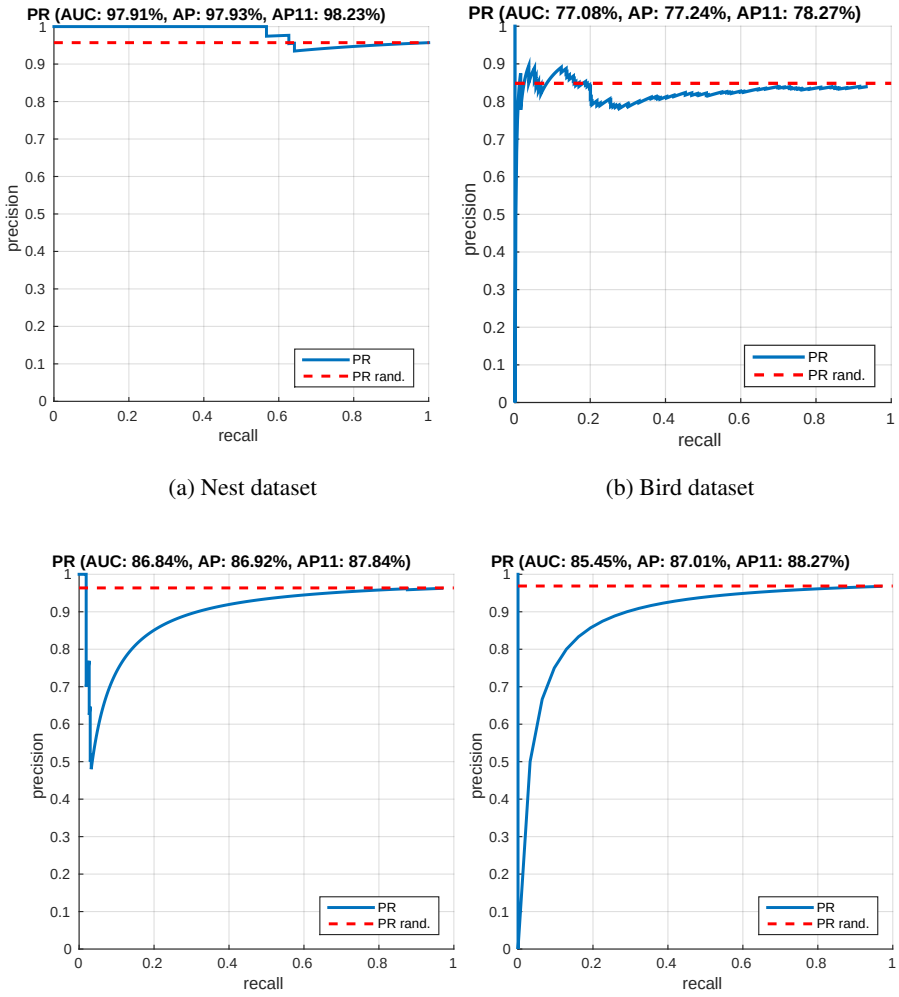
Results are summarized in Table 1 in terms of mean Average Precision (mAP) [5] and time in seconds to process each image. For a baseline comparison (denoted "ImageNet-CNN"), we used the Caffe reference network [24] with approximately the same architecture as [9] and selective search to generate 4,000 object proposals per image. The network was trained and tested on the nest and bird data separately. mAP on both categories was quite low, and processing time very long. For a more competitive comparison, we refer to the PASCAL VOC 2012 detection task submissions [25]. At the time of submission, the leader using only PASCAL VOC data is "DenseBox", a VGG16-like CNN which performs end-to-end object detection [26]. DenseBox's mAP on the "bird" category is well below ours, and it is fairly slow and thus is not suitable for tracking.

When external training data is allowed, the current VOC 2012 detection leader is "ResNet", based on a residual network with a depth of over 100 layers [27]. Its "bird" mAP is the *only* submission higher than that of $\mathrm{YOLO_{B+N}}$, but at a cost of considerably more processing time. However, this number is not directly comparable to ours. All of the detectors submitted to [25] are attempting a harder task in that they are trained for $C = 20$ classes rather than $C = 2$ as we do. To capture the difference in difficulty, we note the lower mAP for the original YOLO [1], also using external training data.

We were only able to directly compare ImageNet-CNN to $\mathrm{YOLO_{B+N}}$ on the "nest" category, but we obtained a higher mAP for it than *any* algorithm on any other category in the VOC dataset. This may be because nests are rigid objects with relatively less appearance variation than other categories.

4.3 Tracking

Table 1 also shows tracking results for $\mathrm{YOLO_{B+N}}$ and $\mathrm{TrackYOLO_{B+N}}$ ($\epsilon = 60$ and $\lambda_1 = \lambda_2 = 0.5$) as compared to several trackers benchmarked in [18] (others were measured, but left out for space reasons). The comparison trackers and $\mathrm{TrackYOLO_{B+N}}$



(a) Nest dataset

(b) Bird dataset

(c) Ground cam nest sequence

(d) UAV nest sequence

Fig. 3. Quantitative PR-curves for different datasets

were started on the ground truth bounding box in the first frame, whereas $\mathrm{YOLO_{B+N}}$ has to find the object by itself.

In all of the sequences, both YOLO-based trackers found and followed the object throughout the sequence, as seen in Figure 4(c)-(e). We observe that the ground nest sequence was most difficult for the comparison trackers, most likely because of its extreme scale change. $\mathrm{TrackYOLO_{B+N}}$ provided the most improvement on the flying bird sequence, because $\mathrm{YOLO_{B+N}}$ did not reliably detect the bird in certain phases of its flapping cycle.

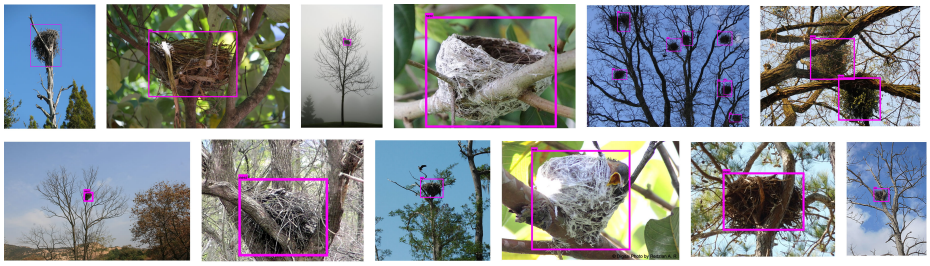
5 CONCLUSION

We have presented a deep CNN system specialized for bird and nest detection and tracking that exhibits excellent accuracy and speed. Current work focuses on incorporating scene context (sky/ground/tree segmentations) into the detection process, bringing more online learning into the tracking process without impacting speed severely, and extending the tracking to multi-object/class scenarios.

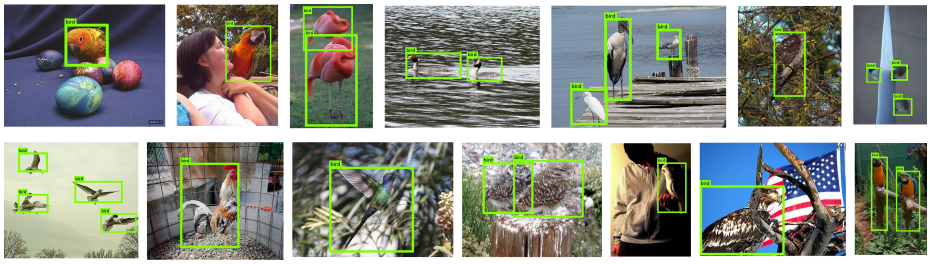
References

1. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640 (2015)
2. Treisman, A., Gelade, G.: A feature-integration theory of attention. Cognitive psychology **12** (1980) 97–136
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition (CVPR). (2005)
4. Han, F., Shan, Y., Cekander, R., Sawhney, H., Kumar, R.: A twostage approach to people and vehicle detection with hog-based svm. In: Proceedings of the Performance Metrics for Intelligent Systems Workshop. (2006)
5. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88** (2010) 303–338
6. Sergeant, D., Boyle, R., Forbes, M.: Computer visual tracking of poultry. Computers and Electronics in Agriculture **21** (1998)
7. Steen, K., Therkildsen, O., Green, O., Karstoft, H.: Detection of bird nests during mechanical weeding by incremental background modeling and visual saliency. Sensors **3** (2015)
8. Wu, X., Yuan, P., Peng, Q., Ngo, C., He, J.: Detection of bird nests in overhead catenary system images for high-speed rail. Pattern Recognition **51** (2016) 242–254
9. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems (NIPS). (2012) 1097–1105
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR). (2015)
11. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)
12. Uijlings, J., van de Sande, K., Gevers, T., Smeulders, A.: Selective search for object recognition. International Journal of Computer Vision **104** (2013) 154–171
13. Zitnick, C., Dollár, P.: Edge boxes: Locating object proposals from edges. In: European Conference on Computer Vision (ECCV). Springer (2014) 391–405
14. Benenson, R., Mathias, M., Timofte, R., Gool, L.V.: Pedestrian detection at 100 frames per second. In: Computer Vision and Pattern Recognition (CVPR). (2012)
15. Mathias, M., Timofte, R., Benenson, R., Gool, L.V.: Traffic sign recognition how far are we from the solution? In: Int. Joint Conf. on Neural Networks. (2013)
16. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS). (2015) 91–99
17. Lenc, K., Vedaldi, A.: R-cnn minus r. arXiv preprint arXiv:1506.06981 (2015)

18. Wu, Y., Lim, J., Yang, M.: Online object tracking: A benchmark. In: Computer Vision and Pattern Recognition (CVPR). (2013)
19. Wang, N., Yeung, D.: Learning a deep compact image representation for visual tracking. In: Advances in Neural Information Processing Systems (NIPS). (2013)
20. Li, H., Li, Y., Porikli, F.: Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In: British Machine Vision Conference (BMVC). (2014)
21. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: International Conference on Computer Vision (ICCV). (2015)
22. Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., Garcia, R.: Incorporating second-order functional knowledge for better option pricing. Advances in Neural Information Processing Systems (NIPS) (2001) 472–478
23. Henriques, J., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: European Conference on Computer Vision (ECCV). (2012)
24. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia, ACM (2014) 675–678
25. : Pascal voc challenge performance evaluation and download server – detection. `http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4` (January 31, 2016)
26. Huang, L., Yang, Y., Deng, Y., Yu, Y.: Densebox: Unifying landmark localization with end to end object detection. arXiv preprint arXiv:1509.04874 (2015)
27. He, K., Zhang, X., Ren, R., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
28. Zhang, K., Zhang, L., Yang, M.: Real-time compressive tracking. In: European Conference on Computer Vision (ECCV). (2012)
29. Zhong, W., Lu, H., Yang, M.: Robust object tracking via sparsity-based collaborative model. In: Computer Vision and Pattern Recognition (CVPR). (2012)



(a) Our nest dataset



(b) PASCAL VOC 2012 bird dataset

125 162 235 520

(c) Ground camera nest video sequence with frame numbers

10 87 210 312

(d) UAV nest video sequence

5 164 318 476

(e) Flying bird video sequence

Fig. 4. (a-b) Sample detection results of $\mathrm{YOLO_{B+N}}$. (c)-(e) Sample tracking results, where blue bounding box is output of hybrid tracker $\mathrm{TrackYOLO_{B+N}}$, yellow is naive tracker $\mathrm{YOLO_{B+N}}$, and red is KCF [23] alone (initialized manually)