# Computer Vision for Small UAS Onboard Pedestrian Detection

Xuxi Yang*, Marc Brittain†, and Peng Wei‡
*Iowa State University, Ames, IA, 50011, USA*

Michael Murphy§
*AirMap, Santa Monica, CA, 90401, USA*

**Computer vision is a key component for safe autonomous flying of small UAS. For example, in a package delivery mission, or an emergency landing event, pedestrian detection could help small UASs with safe landing zone identification. In this research, we focus on deep-learning-based computer vision on a small UAS for pedestrian detection and tracking. In contrast with existing research with ground-level pedestrian detection, our contribution is that we achieve highly accurate multiple pedestrian detection from a birds-eye view, when both the pedestrians and the UAS platform are moving.**

## I. Introduction

With the rapid development of drone technology, small UASs have become popular in a wide range of applications such as cargo delivery, agriculture, construction inspection, weather surveillance, news reporting, firefighting, disaster response, border patrol, infrastructure monitoring, and law enforcement. However, safety concerns are a significant barrier to entry for enterprise-level drone applications such as food or package delivery in densely populated areas. Since the vehicles are unmanned, a primary safety consideration is not to strike people on the ground underneath a drone's flight path; minimizing damage to the drone is a second-order consideration. To improve the safety of drone operations, here we investigate computer-vision-based pedestrian-detection models. This work is motivated by many anticipated benefits for both online (real-time) and offline pedestrian detection in drone-video feeds.

For offline uses, let us consider the difficult process of obtaining a FAA waiver for a beyond-visual-line-of-sight mission. In order to make the case that a proposed mission is safe, the applicant must quantify the risk to people on the ground and in moving vehicles in the mission area. One way to do this is to survey the underlying ground area with a series of safe visual line of sight drone missions. We can then identify pedestrians and moving cars in the video footage from each flight using our proposed object detection models. By fusing the video data with flight telemetry (latitude, longitude, altitude) and the camera pose (yaw, pitch, and roll), we can approximate the density of humans at risk in the area on the ground under the drone mission during a specific time period. We can then suggest safer routes that avoid areas of higher risk such as schools or playgrounds.

The online use cases are more tactical and safety-critical, where our computer vision models can support real-time decision making from onboard autonomy or from a remote pilot. With a real-time pedestrian and vehicle detection capability, we could significantly enhance safety in a populated, dynamic mission area during the landing phase of a regular package delivery, or an emergency landing. We could also use these computer vision models to autonomously avoid walking pedestrians and locate a safe landing space. Additionally, we could create real-time safety tools that alert the pilot when the UAS is approaching a higher risk area with many pedestrians or cars.

This work proposes a deep-learning-based computer vision model for pedestrian detection and tracking. This work is more challenging than street-level pedestrian detection algorithms used by self-driving cars as we are dealing with a variety of camera angles (pitch, in particular), as well as heights. That is, pedestrians need to be detected from a greater distance to the UAV than from a self-driving car, and viewed from above, they often appear in a very small part of a high-resolution image.

*Graduate Research Assistant, Department of Aerospace Engineering, xuxiyang@iastate.edu. Student Member AIAA.
†Graduate Research Assistant, Department of Aerospace Engineering, mwb@iastate.edu. Student Member AIAA.
‡Assistant Professor, Department of Aerospace Engineering, pwei@iastate.edu. Senior Member AIAA.
§Senior Computer Scientist, michael.murphy@airmap.com

## II. Related Work

### A. Object Detection Algorithms

*1. Classical Object Detectors*

Before deep-learning techniques were applied to object detection, the sliding-window paradigm was state-of-the-art. One of such first object detectors is the Viola Jones Object Detector [1], which was primarily used for facial detection. With the introduction of Histogram of Oriented Gradients (HOG) [2], SVM-based classifiers became an effective method for pedestrian detection. Afterwards, the Deformable Part Model(DPM), also based on HOG [3], reigned as the state-of-the-art algorithm for many years.

*2. Two-Stage Detectors*

Another object-detection paradigm is the two-stage detector. The first stage is a region-proposal model that generates a set of candidate proposals from the image. These candidates should have a high probability of being objects and a low probability of being background. The second-stage classifier labels these candidates as different categories (either object classes or background).

The R-CNN Model [4] is known as the first object detector that proposed the two-stage approach. R-CNN applies a non deep learning model, Selective Search [5], as the region proposal model and a convolutional neural network (CNN) is applied to the generated candidates independently to output the classes. With the introduction of Region of Interest pooling (also known as RoI pooling), Fast R-CNN [6] further reduces the computation time of R-CNN by only performing the CNN forward computation on the image as a whole. Faster R-CNN [7] replaces selective search with a region-proposal network (RPN), which reduces the number of proposed regions generated, while ensuring precise object detection. Mask R-CNN [8] uses the same basic structure as Faster R-CNN, but adds a RoI alignment layer to help locate objects at the pixel level and further improve the precision of object detection.

With the region-proposal model, two-stage detectors achieve good detection accuracy. However, the inference of two-stage detectors with these region proposals requires prohibitive computation time, making detection slow, and thus loses appeal for real-time detection.

*3. One-Stage Detectors*

One-Stage detectors are based on global regression and classification and work directly from image pixels to yield bounding-box coordinates and class probabilities of objects in the image. This can reduce computation time, making them more favorable than two-stage detectors.

YOLO [9] makes use of the whole topmost feature map to detect the objects, but has difficulty detecting small objects and unusual aspect ratios. To fix this, SSD [10] was then proposed; it has good performance on multi-scale detection. RetinaNet [11] is a single-step object detector which achieves state-of-the-art accuracy by introducing a novel loss function called Focal Loss to deal with the class imbalance issue. This model represents the first instance where one-stage detectors have surpassed two-step detectors in accuracy while retaining superior speed.

In this paper, we will use RetinaNet as our algorithm to detect pedestrians from drone video.

### B. Aircraft Perception Algorithms

Interest in Unmanned Aerial Vehicles (UAVs) has grown tremendously in recent years. Nowadays, more and more powerful and agile UAVs are recruited for civilian applications in terms of surveillance and infrastructure inspection. The major challenge today is the development of autonomously operating aerial agents capable of completing missions independently of human interaction. To this extent, visual sensing techniques have been integrated into the control pipeline of the UAVs in order to enhance their navigation and guidance skills [12].

Previous research works propose traditional computer vision algorithms that run onboard aircraft to detect road, aircraft, moving targets [13–15]. For target tracking, the R-RANSAC multiple moving target tracker [16–20] is specifically well suited to track moving targets from a rapidly moving camera, and has excellent track continuity. However, these traditional computer vision algorithms face scalability challenges to be applied in the UAM environment, where there are multiple moving targets with variable shapes (pedestrians and cars) in dynamic complex environments. Thus we propose a learning-based perception algorithm that is able to detect and track multiple moving targets in the urban area through training with large-scale datasets. The learning-based perception system is expected to be faster

(more computationally efficient) and more scalable (able to track many moving targets with variable shapes) during a landing event. In addition, we expect this framework can be extended to aircraft detection during en route flight, which is helpful for aircraft separation assurance systems and sense-and-avoid systems.

For aircraft in the UAM environment, there are also several other restrictions to deploy the onboard perception algorithms. First, due to the limited memory and computing power of embedded onboard devices, the trained model needs to be pruned to a smaller size for real-time object detection [21, 22]. Also, the resolution increase in visual sources and relatively small scale of pedestrians makes the problem even harder by raising the expectations to leverage all the details in images [23].

## III. RetinaNet Model

### A. Class Imbalance Problem of One-Stage Detector

In two-stage detectors, the region proposal model at the first stage will narrow the number of candidate object locations to a small number (e.g., 1,000 or 2,000), while filtering out most background samples. At the second stage, classification is performed for each candidate object location. Sampling heuristics using a fixed foreground-to-background ratio (1:3), or online hard example mining (OHEM) [24] to select a small set of anchors (e.g., 256) for each mini-batch. Thus, there is a manageable class balance between foreground and background [11].

For one-stage detectors, a much larger set of candidate object locations is regularly sampled across an image ( 100k locations), which densely cover spatial positions, scales, and aspect ratios. The training procedure is still dominated by easily classified background examples. To resolve this class imbalance problem, a new loss function called Focal Loss [11] was proposed which is a more effective alternative to previous approaches, which has the following form

$$FL(p_t) = -(1 - p_t)^{\gamma} \log p_t \tag{1}$$

where $p_t$ is the model's estimated probability and $\gamma$ is the tuneable focusing parameter.

As shown in Fig. 1, when $\gamma$ is set to a positive number, the easy well-classified examples (background) will contribute less to the total loss function and the hard misclassified examples (foreground objects) will contribute more to the total loss function, thus making the optimization algorithm concentrate more on the hard examples.

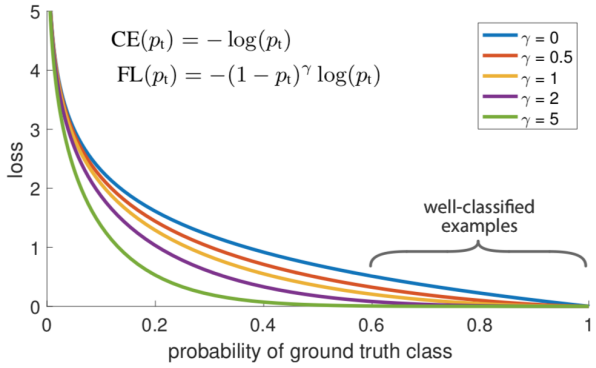



**Fig. 1 Focal Loss adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy loss function. Setting $\gamma > 0$ reduces the relative loss for well-classified examples, putting more focus on hard, misclassified examples [11].**

### B. Retina Detector Architecture

RetinaNet is a single unified network consisting of two parts: a 50-layer ResNet [25] backbone network which is used for deep feature extraction and two task-specific subnetworks which perform convolutional object classification and convolutional bounding box regression. Also, the Feature Pyramid Network (FPN) [26] is used on the top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution input image. More specifically, the FPN layer is built on top of the ResNet architecture [25], from which pyramid levels $P_3$ through $P_7$ are constructed where the number indicated pyramid level ($P_l$ has resolution $2^l$ lower than the input).

In real-world object detection, objects from the same class may be in a wide range of scales in images. A traditional convolution neural network is not good at detecting objects in small scale since feature maps from higher levels are spatially coarser. By combining higher level features and low level features, FPN can help detect objects in various scales, which is helpful in our case since pedestrians in the aerial images are usually of small size.

### C. Anchor Parameter

One of the most important design considerations in the one-stage detector is how densely it covers the space of possible image boxes. Since one-stage detectors use a fixed sampling grid, one popular approach for achieving high coverage of boxes for various sizes (scales and aspect ratios) of objects is to use multiple "anchors" [7] at each spatial position. In the RetinaNet [11] algorithm, the authors use 9 anchors per location spanning 3 scales ($2^{0/3}, 2^{1/3}, 2^{2/3}$) and 3 aspect ratios [1:2, 1:1, 2:1] for the bounding boxes, and the anchor sizes are of 32, 64, 128, 256, 512, according to the 5 different pyramid levels. While this anchor configuration has decent performance on the COCO dataset [27], it is not suitable for the images taken from drones (e.g., in the dataset we are using in this paper, pedestrians are usually smaller than $32 \times 32$ pixel, which is the size of the smallest anchors). In this paper, we drop the biggest one of 512 and instead add a small anchor of size 16, which can cover the scale of 16 - 407 pixels with respect to the network's input image.

### D. Loss Function

The loss function of the RetinaNet model contains two terms: the localization loss $L_{loc}$ and the classification loss $L_{cls}$.

$$L = \lambda L_{loc} + L_{cls} \tag{2}$$

where $\lambda$ is a hyper-parameter that controls the balance between the two losses.

*1. Localization Loss*

Let $(A^i, G^i)_{i=1,..,N}$ represent the matching pairs of anchors and ground truth, with $N$ defined as the number of matches. For each anchor $A^i$ with a match, the regression network predicts four values, denoted as $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$, where $(P^i_x, P^i_y)$ represents the offset between the center of $A^i$ and $G^i$. $(P^i_w, P^i_h)$ represents the offset of the width and height of $A^i$ and $G^i$. We can then define the regression target $T^i$ as follows:

$$\begin{aligned} T^i_x &= \left(G^i_x - A^i_x\right) / A^i_w \\ T^i_y &= \left(G^i_y - A^i_y\right) / A^i_h \\ T^i_w &= \log\left(G^i_w / A^i_w\right) \\ T^i_h &= \log\left(G^i_h / A^i_h\right) \end{aligned} \tag{3}$$

With the notions defined above, the localization loss is then defined as:

$$L_{loc} = \frac{1}{N} \sum_{i}^{N} \sum_{j \in \{x,y,w,h\}} smooth_{\mathrm{L1}}(P^i_j - T^i_j) \tag{4}$$

with $smooth_{\mathrm{L1}}$ defined as:

$$smooth_{\mathrm{L1}}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & |x| \geq 1 \end{cases} \tag{5}$$

*2. Classification Loss*

The classification loss for each anchor in RetinaNet can be defined as:

$$L_{cls} = -\sum_{i=1}^{K} (y_i \log(p_i)(1-p_i)^{\gamma}\alpha_i + (1-y_i)\log(1-p_i)p_i^{\gamma}(1-\alpha_i) \tag{6}$$

where $K$ is the number classes; $y_i$ equals 1 if the ground truth belongs to the $i$-th class and 0 otherwise; $p_i$ represents the predicted probability for class $i$; $\gamma \in (0, +\infty)$ represents the a focusing parameter that reduces the contribution of the loss for easily classified examples (backgrounds), and $\alpha_i \in [0, 1]$ represents a weighting parameter for class $i$.

Both $\gamma$ and $\alpha$ are introduced to help with the class imbalance problem as most locations in an image can be easily classified as the background. This, in turn, results in a disproportionate number of useful learning examples and can reduce the performance of the classifier in classes with a smaller number of training examples.

When making predictions for an image, the RetinaNet model will select at most 1k anchor boxes that has the highest confidence score output from the classification network. Note that an object in the image may be predicted by multiple overlapping anchor boxes. To remove redundancy, non-maximum-suppression (NMS) is applied to each class, which chooses an anchor box with the highest confidence score and removes any overlapping anchor boxes in the same class with an IoU score greater than 0.5. Finally, for each remaining anchor, the output from the regression network will be used to refine the anchor to get the final bounding box prediction.

## IV. Experiments

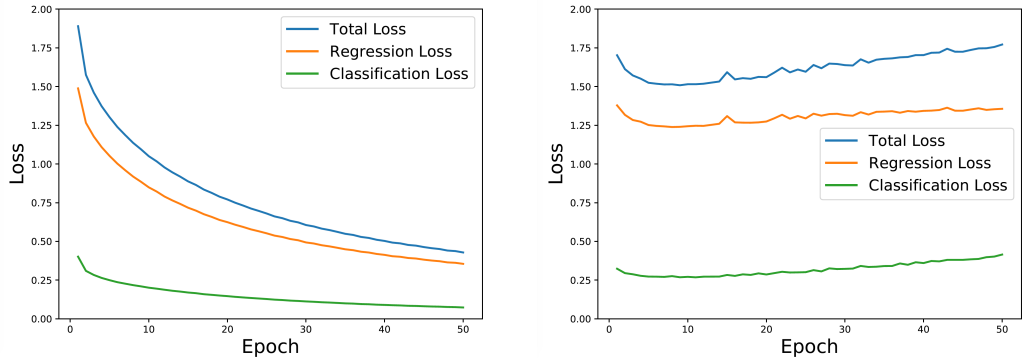
We present experimental results on the *VisDrone2019-Det* benchmark dataset [28]. We use a Linux workstation running Ubuntu 18.04 with an Intel(R) Xeon(R) E5-1650 v2 CPU @ 4.00GHz (12 CPUs), 256GB RAM, and one NVIDIA Titan Xp GPU (12GB) to train and evaluate the models in our experiments.

### A. Dataset

*VisDrone2019-Det* dataset consists of 7,019 static images captured by various drone-mounted cameras, covering a wide range of aspects including location (taken from 14 different cities separated by thousands of kilometers in China), environment (urban and country), objects (pedestrian, vehicles, bicycles, etc.), and density (sparse and crowded scenes). Also, the dataset was collected using various drone platforms (i.e., drones with different models), in different scenarios, and under various weather and lighting conditions. These frames are manually annotated with more than 2.6 million bounding boxes of 10 target objects (i.e., pedestrian, person, car, van, bus, truck, motor, bicycle, awning-tricycle, and tricycle). The RetinaNet model is trained on the training set and evaluated on the validation set.

### B. Model Training

We did experiments using RetinaNet with ResNet-50-FPN backbones. The base RetinaNet is pre-trained on the COCO dataset [27], then trained using the adam algorithm with a learning rate of $10^{-5}$. We use horizontal image flipping as the only form of data augmentation. The training loss is smooth $L_1$ loss for box regression [6] and focal loss for the object classification [11]. Figure 2 shows the loss during training, from which we can see the model begins to overfit the training data after 20 epochs. Figure 3 plots the weighted mAP of the 10 classes on the validation dataset. At epoch 17, the RetinaNet algorithm achieves the highest mAP of 30.21% on the validation set.




**(a) training dataset** **(b) validation dataset**

**Fig. 2 The loss during training on training dataset and validation dataset.**

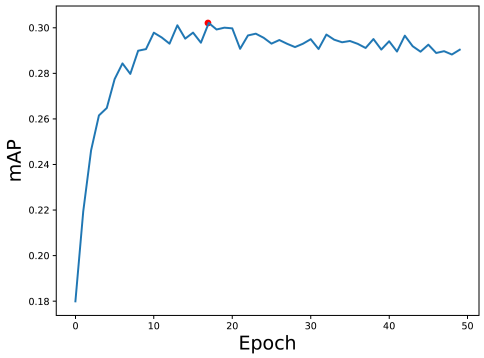



**Fig. 3 The mAP score on the validation dataset during training.**

**Table 1 Detection performance for each category on validation set.**

| Class | Images | Instances | mAP |
|---|---|---|---|
| pedestrian | 548 | 8844 | 36.78 |
| people | 548 | 5125 | 27.33 |
| bicycle | 548 | 1287 | 11.22 |
| car | 548 | 14064 | 71.30 |
| van | 548 | 1975 | 30.37 |
| truck | 548 | 750 | 27.71 |
| tricycle | 548 | 1045 | 16.56 |
| awning-tricycle | 548 | 532 | 6.88 |
| bus | 548 | 251 | 39.09 |
| motor | 548 | 4886 | 34.82 |
| overall | 548 | 38759 | 30.21 |

*VisDrone2019-Det* is a very challenging dataset with a high category imbalance. As shown in Table 1, the category with a large number of instances dominates the loss function, hence this category obtains a higher mAP score (e.g., car and pedestrian). The mAP for the car is higher than the pedestrian is because the relative scale of the car is larger and hence easier for the algorithm to detect. In future work, we are planning to use data augmentation and some oversampling techniques to deal with the class imbalance issue.

## V. Results

We test the performance on images and videos taken by our drone using the model trained above with the highest mAP. All of our drone images and videos were captured with a DJI Mavic Pro 2 drone. The Mavic Pro 2 can shoot video at 4k resolution but it is stored on a memory card on the drone. These videos can be retrieved from the drone after the flight and analyzed offline. The Mavic Pro 2 can also live-stream video at 1080p resolution. For testing in real time, we used DJI's Microsoft Windows SDK to stream the drone video to our models in real time on our GPU-equipped laptop. To use this in the field, we needed to bring the laptop with us. Ideally, we would like to be able to use a smartphone for object detection, but they do not have the computing power to recognize objects in real-time with our model. We are working on speeding up inference to the point where this is possible.

Figure 4 shows two sample detection results of the trained model for the images taken by our drone, where the blue rectangle represents detected pedestrian and the orange rectangle denotes detected cars. From this figure we can see if the angle of the camera is closer to bird eye view, the performance will be worse since the dataset the model trained on doesn't contain many bird eye view images. A sample detected video is also available on Youtube*.

*https://www.youtube.com/watch?v=XFyoZfURRlM

**Fig. 4 Visualized detection results of RetinaNet trained model.**

## VI. Conclusion

In this paper, we propose an efficient learning-based pedestrian algorithm that can be deployed on a drone. By tuning the network architecture and anchor parameter, the proposed algorithm can be used to detect small pedestrians and cars from aerial images and videos. After training the model on a drone image dataset, the proposed algorithm achieves high mAP for detecting cars and pedestrians. Real-world cases also show this algorithm is promising for detecting pedestrians and cars, which can help the drone estimate the risk level through the detection results.

This work proposes a deep-learning-based computer vision model for pedestrian detection and tracking. This work is more challenging than street-level pedestrian detection algorithms used by self-driving cars as we are dealing with a variety of camera angles (pitch, in particular), as well as heights. That is, pedestrians and cars need to be detected from a greater distance to the UAV than from a self-driving car, and viewed from above, they often appear in a very small part of a high-resolution image. In contrast with existing research with ground-level pedestrian detection, the developed algorithm achieves highly accurate multiple pedestrian detection from a bird-eye view, when both the targets and the aircraft platform are moving.

## References


[1] Viola, P., Jones, M., et al., "Rapid object detection using a boosted cascade of simple features," *CVPR (1)*, Vol. 1, No. 511-518, 2001, p. 3.

[2] Dalal, N., and Triggs, B., "Histograms of oriented gradients for human detection," 2005.

[3] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D., "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 32, No. 9, 2009, pp. 1627–1645.

[4] Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[5] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W., "Selective search for object recognition," *International journal of computer vision*, Vol. 104, No. 2, 2013, pp. 154–171.

[6] Girshick, R., "Fast r-cnn," *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[7] Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, 2015, pp. 91–99.

[8] He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask r-cnn," *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[9] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[10] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C., "Ssd: Single shot multibox detector," *European conference on computer vision*, Springer, 2016, pp. 21–37.

[11] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P., "Focal loss for dense object detection," *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[12] Kanellakis, C., and Nikolakopoulos, G., "Survey on computer vision for UAVs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, Vol. 87, No. 1, 2017, pp. 141–168.

[13] Frew, E., McGee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., and Sengupta, R., "Vision-based road-following using a small autonomous aircraft," *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, Vol. 5, IEEE, 2004, pp. 3006–3015.

[14] Molloy, T. L., Ford, J. J., and Mejias, L., "Detection of aircraft below the horizon for vision-based detect and avoid in unmanned aircraft systems," *Journal of Field Robotics*, Vol. 34, No. 7, 2017, pp. 1378–1391.

[15] Saripalli, S., "Vision-based autonomous landing of an helicopter on a moving target," *AIAA guidance, navigation, and control conference*, 2009, p. 5660.

[16] Niedfeldt, P. C., and Beard, R. W., "Multiple target tracking using recursive RANSAC," *2014 American Control Conference*, IEEE, 2014, pp. 3393–3398.

[17] Niedfeldt, P. C., and Beard, R. W., "Convergence and complexity analysis of recursive-RANSAC: A new multiple target tracking algorithm," *IEEE Transactions on Automatic Control*, Vol. 61, No. 2, 2015, pp. 456–461.

[18] Niedfeldt, P. C., Ingersoll, K., and Beard, R. W., "Comparison and analysis of recursive-RANSAC for multiple target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 53, No. 1, 2017, pp. 461–476.

[19] Sakamaki, J. Y., Beard, R. W., and Rice, M., "Tracking multiple ground objects using a team of unmanned air vehicles," *Sensing and Control for Autonomous Vehicles*, Springer, 2017, pp. 249–268.

[20] Ingersoll, K., Niedfeldt, P. C., and Beard, R. W., "Multiple target tracking and stationary object detection in video with Recursive-RANSAC and tracker-sensor feedback," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 1320–1329.

[21] Zhang, P., Zhong, Y., and Li, X., "SlimYOLOv3: Narrower, Faster and Better for Real-Time UAV Applications," *arXiv preprint arXiv:1907.11093*, 2019.

[22] Ringwald, T., Sommer, L., Schumann, A., Beyerer, J., and Stiefelhagen, R., "UAV-Net: A Fast Aerial Vehicle Detector for Mobile Platforms," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[23] Ozge Unel, F., Ozkalayci, B. O., and Cigla, C., "The Power of Tiling for Small Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[24] Shrivastava, A., Gupta, A., and Girshick, R., "Training region-based object detectors with online hard example mining," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.

[25] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S., "Feature pyramid networks for object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[27] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., "Microsoft coco: Common objects in context," *European conference on computer vision*, Springer, 2014, pp. 740–755.

[28] Zhu, P., Wen, L., Bian, X., Haibin, L., and Hu, Q., "Vision Meets Drones: A Challenge," *arXiv preprint arXiv:1804.07437*, 2018.