https://doi.org/10.2352/ISSN.2470-1173.2018.10.IMAWM-466


# Deep Learning for Moving Object Detection and Tracking from a Single Camera in Unmanned Aerial Vehicles (UAVs)

***Dong Hye Ye***[1] ***, Jing Li***[1] ***, Qiulin Chen***[1] ***, Juan Wachs***[2] ***, and Charles Bouman***[1] ***;***
[1] ***School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA;***
[2] ***School of Industrial Engineering, Purdue University, West Lafayette, IN, USA;***

## Abstract

*Unmanned Aerial Vehicles (UAVs) gain popularity in a wide range of civilian and military applications. Such emerging interest is pushing the development of effective collision avoidance systems which are especially crucial in a crowded airspace setting. Because of cost and weight limitations associated with UAVs' payload, the optical sensors, simply digital cameras, are widely used for collision avoidance systems in UAVs. This requires moving object detection and tracking algorithms from a video, which can be run on board efficiently. In this paper, we present a new approach to detect and track UAVs from a single camera mounted on a different UAV. Initially, we estimate background motions via a perspective transformation model and then identify moving object candidates in the background subtracted image through deep learning classifier trained on manually labeled datasets. For each moving object candidates, we find spatio-temporal traits through optical flow matching and then prune them based on their motion patterns compared with the background. Kalman filter is applied on pruned moving objects to improve temporal consistency among the candidate detections. The algorithm was validated on video datasets taken from a UAV. Results demonstrate that our algorithm can effectively detect and track small UAVs with limited computing resources.*

## Introduction

Thanks to recent development in drone technology, unmanned aerial vehicles (UAVs) will be pervasive in the sky for commercial and individual needs [1, 2, 3]. One of the most important issues facing UAV's use is collision avoidance capability [4]. Since the size and the energy consumption of the UAV are limited, a optical sensor based avoidance system (e.g., Go-Pro color cameras) has the potential to provide cost and weight advantages against the traffic collision avoidance system (TCAS) currently in use on larger aircraft equipped with LIDAR sensors.

Optical sensor based collision avoidance systems then require the detection and tracking of other UAVs from video feeds [5, 6]. Once other UAVs are detected and tracked, strategies involving a sequence of maneuvers for collision avoidance are followed. For example, the spatio-temporal information extracted from other UAVs can be associated with friendly or austere behavior. These moving object detection and tracking operations must be real-time to run on-board even if the connection between the aircraft and the ground control station is lost, or sensors fail.

In this context, real-time moving object detection and tracking has been investigated in large by the computer vision community [7, 8]. For example, in Viola and Jones al. [9], the authors extract the simple Haar features and apply cascading supervised classifiers to detect and track face in a video real-time. In addition, many pedestrian and car detection algorithms [10, 11, 12] are developed for surveillance monitoring and even used in the commercial products. However, it is not appropriate to extend these computer vision algorithms directly to UAV applications due to unique challenges. First, a video is recorded by a moving camera for UAV, on the contrary to a static camera for many computer vision applications. Therefore, for UAV applications, it is difficult to stabilize the rapidly changing background which are non-planar and complex. Second, given the speed of UAVs, the moving objects need to be detected in a far distance for collision avoidance. Then, our targets appear very small in a video often occluded by clutter (e.g. clouds, trees, and specular light).

There are a few attempts to detect and track moving objects using camera-based systems in UAVs [13]. These approaches extract motion information for moving object detection and tracking. Motion-based approaches can then be divided into two main categories: (1) Background Subtraction and (2) Optical Flow. Background subtraction methods identify groups of pixels which are not changing over time and then subtract those pixels from the image to detect moving objects [7, 12]. These background subtraction methods work best when background motion can be easily compensated, which is not the case for fast moving camera. Optical flow methods find the corresponding image regions between frames and depend on the local motion vectors to detect moving objects [14, 15]. However, it is computationally expensive to extract local motion vectors of all pixels in the video frame for real-time operation.

In our previous publication [16], we combine background subtraction and optical flow methods to have synergistic effects. Even though background motion estimation for a moving camera is approximate, we can subtract most of homogeneous regions in the image to isolate the target objects. Then, our target objects are salient in the background subtracted image enabling to identify good points for optical flow matching. More importantly, by comparing background motion and flow vector, we can extract spatio-temporal features which are useful for moving object detection and tracking. Then, the quality of motion vector is critical for accurate detection, which can be low for the blurred image. To tackle these challenges, we use shape and appearance information to detect and track other small UAVs from a video. These approaches extract the features in each individual frames and apply supervised learning techniques to classify the target objects in the training dataset [17]. The trained classifiers based on deep learning (e.g., Convolutional Neural Networks or CNN) can improve

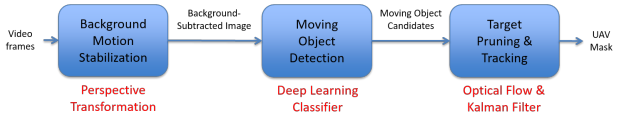


***Figure 1.*** *An overview of our proposed method: We first estimate the background motion between two sequential frames via perspective transformation model. From resulting background-subtracted image, we detect the moving objects by applying deep learning classifier on distinctive patches. Among detected moving object candidates, we prune actual UAVs from spurious noise using the estimated local motion and incorporate the temporal consistency through Kalman filter tracking.*

detection performance even in challenging environments with illumination variations and background clutters. We further apply Kalman filter tracking [18] on our detection to reduce the intermittent miss-detection and false alarms. We tested our proposed algorithm on real videos from UAVs and successfully identify target objects.

## Moving Object Detection and Tracking

As illustrated in Fig. 1, we propose an efficient moving object detection and tracking algorithm for UAVs. We first parse the video into a sequence of frames and estimate the background motion between frames. Our assumption is that other UAVs and the background have very different motion model and thus by compensating the motion of the background, the moving object can be extracted. We estimate the background motion via perspective transform model [19] taking account into globally smooth motion with camera projection. Given background subtracted image, we highlight distinctive patches and then detect the moving object candidates among them by applying deep learning classifier. For each moving object candidates, we find the local motion by applying Lucas-Kanade optical flow algorithm [14] and use spatio-temporal characteristics to identify actual UAVs. We further apply Kalman filter tracking [18] on our detection to reduce the intermittent miss-detection. In following, we describe each component of our algorithm in details.

### *Background Motion Stabilization*

The video is acquired from a moving camera on the UAV, and thus we need to stabilize the rapidly changing background which is often non-planar geometry.

#### *Estimate Background Motion*

For background motion stabilization, we first estimate the background motion via a perspective transformation model [19]. Unlike other global transformation model such as rigid or affine transformation models, the perspective transformation model can take account into projection based on the distance from the camera, which is beneficial to compensate the background motion in a far distance from a camera. To estimate the background motion via a perspective model, we find the correspondence between two consecutive frames on a small set of points and fit them into the perspective transformation model. This is mainly because background motion estimation procedure can be computationally expensive if it is optimized over all pixels in the field of view. We choose the small set of points for correspondence matching based on saliency with appropriately uniform distribution.

We now define the selected point $p_{t-1} \in \mathbb{R}^2$ in the previous frame $X_{t-1}$. We then find the corresponding points $p_t \in \mathbb{R}^2$ in the current frame $X_t$ through simple and efficient block matching [20]. We then estimate the perspective transformation $H_{t-1} \in \mathbb{R}^{3\times3}$ from $X_{t-1}$ to $X_t$, which regularizes local correspondence matching to be smooth in the entire image.

$$H_{t-1} = \arg\min_{H} \sum_{p_t \in \mathbf{P}_t, p_{t-1} \in \mathbf{P}_{t-1}} ||p_t - H \circ p_{t-1}||_2^2, \tag{1}$$

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}, \tag{2}$$

where $\mathbf{P}_t$ and $\mathbf{P}_{t-1}$ represent a set of corresponding points in $X_t$ and $X_{t-1}$, respectively, and $\circ$ is the warping operation. It is worth noting the perspective transformation model is described by only 8 parameters to be optimized very efficiently. Here, $\{h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}\}$ represent affine transformation matrix (e.g., scale, rotation, sheer and translation). Additional parameters $\{h_{31}, h_{32}\}$ allow a perspective projection to the vanishing point.

#### *Compute Background Subtracted Image*

We then subtract the background by using the estimated perspective transformation $H_{t-1}$. Specifically, we take difference between current frame and background motion compensated previous frame. We define the background subtracted image $E_t$ for $X_t$ as following.

$$E_t = |X_t - H_{t-1} \circ X_{t-1}|. \tag{3}$$

It is worth noting that applying $H_{t-1}$ to the entire image may be time-consuming and therefore we only compute the background subtracted image for a certain interval of frames (e.g., every 10 frames).

### *Moving Object Detection*

Given the estimated background subtracted image, we detect the moving object candidates. I first identify salient points in the background subtracted image and extract patch appearance features on those points. We then feed the appearance features to deep neural networks for supervised classification.

#### *Identify Salient Points*

We identify salient points in a background subtracted image $E_t$ using Shi-Tomasi corner detector [21]. We choose Shi-Tomasi corner detector due to efficiency. Shi-Tomasi corner detector finds corners associated with the high local autocorrelation.

Given an image $E_t$, we define the local autocorrelation $C_t$ at the pixel $s$ with a first-order Taylor expansion as following:

$$\begin{aligned} C_t(s) &= \sum_W [E_t(s+\delta s) - E_t(s)]^2, \\ &\approx \delta s^T \sum_W [\nabla E_t(s)^T \nabla E_t(s)] \delta s, \\ &\approx \delta s^T \Lambda_t(s) \delta s, \end{aligned} \tag{4}$$

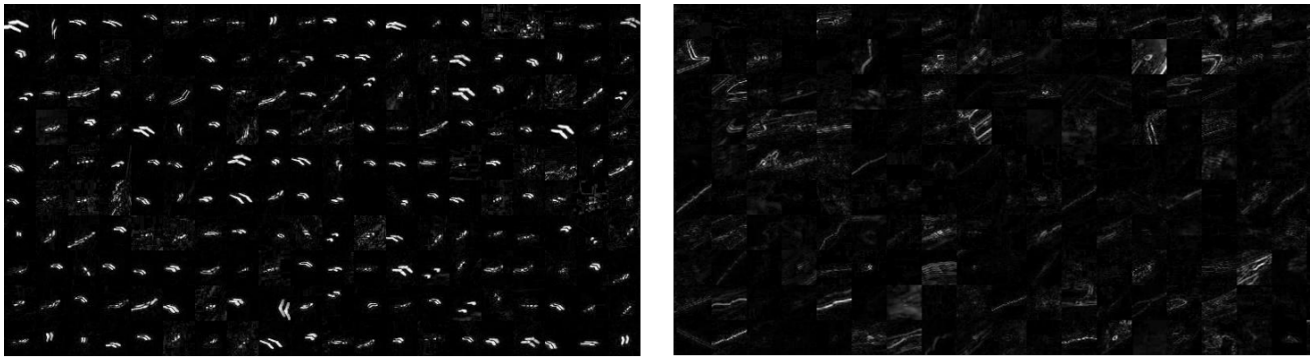


***Figure 2.*** *Example of training patches: [Left] True Moving Objects, [Right] False Alarms: We note that true moving objects have different appearance in the background subtracted images compared with false alarms. True moving objects tend to be distinctly highlighted while false alarms contain blurred edges.*

where $\delta s$ represents a shift, $W$ is a window around $s$, $\nabla$ is the first order derivative, and $\Lambda_t$ is the precision matrix.

We then compute a saliency $Q_t$ for any point in $E_t$ according to eigenvalues of $\Lambda_t$.

$$Q_t(s) = \min\{\lambda_t^1(s), \lambda_t^2(s)\}, \tag{5}$$

where $\lambda_t^1$ and $\lambda_t^2$ are two eigenvalues of $\Lambda_t$. After thresholding on $Q_t$, we find a set of salient points $\{q_t^{(1)}, \cdots, q_t^{(N)}\}$. To ensure sparse distribution, we discard points for which there is a stronger salient points in the neighborhood.

### *Classify Moving Objects*

Given the salient points on the background subtracted image, we perform classification to reject outliers from true moving objects. Toward classification, we extract $40 \times 40$ patch on each salient point $q_t^{(n)}$ in the background subtracted image. Fig.2 shows the example of extracted patches on manually labeled training dataset. It is worth noting that patches of true moving objects look very different from those of false alarms. In the patches, true moving objects tend to have high contrast V-shape while false alarms show the blurred edge. Therefore, appearance information from background subtracted images is powerful to differentiate moving objects from false alarms.

We then train the classifier which separates moving objects from false alarms through deep learning. In deep learning algorithms, the weights of a neural network are trained on large datasets, and then the trained neural network is applied to determine whether the unseen testing object is moving target or not. The network architecture is described in Fig.3. First, we apply 16 filters of $3\times3$ convolution kernel to generate feature maps and then utilize rectified linear units (ReLU) [22] for neuron activation. The batch normalization unit is added between convolution and ReLU to avoid internal covariate shift during mini-batch optimization [23]. Next, we apply max-pooling (spatial downsampling) operation to reduce the size of feature map and increase the number of filters for the following layer. Then, we repeat the convolution (plus batch normalization and neuron activation) with max pooling operation for 2 layers with 32 filters of $3\times3\times16$ kernel and 64 filters of $3\times3\times32$, respectively. Finally, we find the binary classification label by applying fully connected neural network with soft-max function.

By feeding the appearance patches in the unseen testing video frame into the trained neural network, we can find the salient points on the moving object candidates.

## *Target Pruning and Tracking*

While we expect our moving object detection to be effective, we still encounter intermittent miss-detections with false alarms. These false alarms and missed detections can be corrected by observing the temporal characteristics of the detected moving objects. Toward this, we estimate the local motion fields of the detected moving objects through optical-flow matching. Then, we prune the moving objects based on the difference between background and local motion. In addition, we apply Kalman filter to make detected objects correspond to coherent temporal signatures as opposed to spurious intermittent noise.

### *Estimate Local Motion*

Given the detected patch of the moving object at $q_t^{(n)}$ in the background subtracted image $E_t$, we estimate the local motion via optical flow matching. Toward this, we first extract $M$ salient


***Figure 3.*** *Network architecture for deep learning: Given input patches, 3-layer convolutional neural networks are trained to identify moving objects. Note that we use batch normalizaton (BN) and rectified linear unit (ReLU) for efficient training of deep neural networks.*

points $\{r_t^{(n,1)}, \cdots, r_t^{(n,M)}\}$ in corresponding regions of the original frame $X_t$ using Shi-Tomasi corner detector. For each salient point $r_t^{(n,m)}$, we compute the optical flow vector to the corresponding point in the next frame $X_{t+1}$ by applying Lucas-Kanade method.

In Lucas-Kanade method, we assume that all neighbor points around the given pixel have the same motion. So, the local motion can be computed by solving the least square problem.

$$u_t^{(n,m)} = \arg\min_u \sum_{s \in \mathcal{N}(r_t^{(m,n)})} |X_{t+1}(s+u) - X_t(s)|^2, \tag{6}$$

where $\mathcal{N}(r_t^{(m,n)})$ is the neighborhood around $r_t^{(m,n)}$. It is worth noting that local motion estimation via Lucas-Kanade method is efficient because we have a closed-form solution for eq. 6 and we compute the optical flow vector only for small number of salient points in the detected moving object patches.

### *Prune Target Objects*

We prune the detected moving object at $q_t^{(n)}$ based on the difference between the estimated background and local motion. This motion difference represents the actual speed of the moving object relative to rapidly moving camera. So, we prune the target objects if the actual speed of the moving object is too small (e.g., stand still) or too large (e.g., beyond the reasonable speed).

We now define the motion difference $d_t^{(n)}$ for the moving object at $q_t^{(n)}$ between the background and moving object as following:

$$d_t^{(n)} = \frac{1}{M} \sum_{m=1}^{M} (h_t^{(n,m)} - u_t^{(n,m)}), \tag{7}$$

where $h^{(n,m)}$ is interpolated motion vector from the perspective transform $H_t$ between $X_t$ and $X_{t+1}$ at the point $r_t^{(m,n)}$.

We denote $y_t^{(n)}$ as a binary label where the positive value indicates that the moving object at $q_t^{(n)}$ is the target. We then find the pruned target object according to the magnitude of motion difference.

$$y_t^{(n)} = \begin{cases} 1, & \text{if } T_L < ||d_t^{(n)}||_2 < T_H \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $T_L$ and $T_H$ are the empirical low and high threshold for pruning. By applying connected component labeling [24] on the set of salient points on the pruned target object, we generate the bounding box which represents other UAV.

### *Track Target Objects*

Even though our pruning based on motion difference reduces the false alarms, we also need to deal with intermittent miss-detections. To improve the temporal consistency of our target detection, we apply object tracking techniques based on Kalman filter [18].

Kalman filter predicts the current state $b_t$ from previously estimated states $\hat{b}_{t-1}$ with transition model and updates the current measurement $c_t$ with the current state $b_t$ as below:

$$\begin{aligned} b_t &= A\hat{b}_{t-1} + \omega_t, \\ c_t &= Mb_t + \varepsilon_t, \end{aligned} \tag{9}$$

where $A$ is state transition matrix, $\omega_t$ controls the transition modeling error, $M$ is measurement matrix, and $\varepsilon_t$ represents the measurement error. The estimated output $\hat{b}_t$ is then computed with Kalman gain $K$:

$$\begin{aligned} \hat{b}_t &= A\hat{b}_{t-1} + K(c_t - Mb_t), \\ K &= V_\omega M^T (MR_\omega M^T + V_\varepsilon), \end{aligned} \tag{10}$$

where $V_\omega$ and $V_\varepsilon$ are the covariance of $\omega_t$ and $\varepsilon_t$, separately.

Specifically, we assign the size and location of bounding box for the detected target object as state variable $b_t$ and use the constant velocity model to set $A$ and $M$.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{12}$$

.

To initialize the Kalman filter, we find the corresponding objects from optical flow matching in $L$ previous frames and start track if the classification labels $y_{t-1}^{(n)}, \cdots, y_{t-L}^{(n)}$ are positive. Then, we recover the miss-detection for the positive label track based on the Kalman filter output at the current frame. We dismiss the track if we do not have detected objects in the Kalman filter estimation for $L$ frames.

## EXPERIMENTS

We evaluate our moving object detection and tracking method using deep learning on a video data set [1] provided by Naval Postgraduate School. For reference, we also perform our previous moving object detection and tracking [16] which did not incorporate the appearance information with deep learning.

### *Data Set*

The videos are taken in outdoor environment including real-world challenges such as illumination variation, background clutter, and small target objects. The data set comprises 45 video sequences with 30 fps frame rate. Each video is around one minute. They are recorded by a GoPro 3 camera (HD resolution: $1920 \times 1080$ or $1280 \times 960$) mounted on a custom delta-wing airframe. As a preprocessing, we mask out the pitot tube region which is not moving in the videos. For each video, there are multiple target UAVs (up to 8) which have various appearances and shapes. We manually annotate the targets in the videos by using VATIC software [25] to generate ground-truth dataset for training and performance evaluation. We fix 40 videos as a training set for deep learning and assign remaining 5 videos for testing.

### *Parameter Exploration*

There are important parameters in our moving object detection and tracking algorithm. To begin with, we extract Shi-Tomasi corner points in background subtracted image $E_t$ (for moving object detection) and original image $X_t$ (for local motion estimation),

[1] https://engineering.purdue.edu/~bouman/UAV_Dataset/

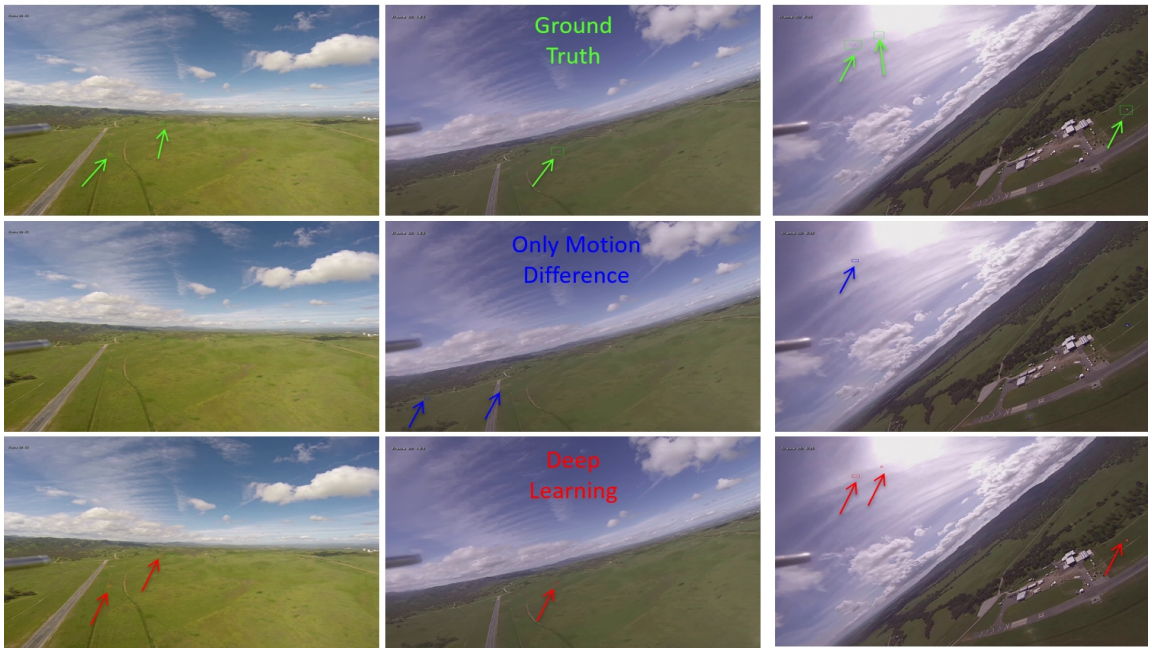


***Figure 4.*** *Results of moving object detection and tracking algorithms for 3 testing videos: [Top-Row] Ground-truth annotation (Green), [Middle-Row] Moving object detection and tracking algorithm purely based on motion difference pruning [16] (Blue), [Bottom-Row] Our deep learning based method toward moving object detection and tracking (Red). Previous method using only motion difference pruning fails to detect some moving objects with false alarms on the complex background (e.g., horizon). By using deep learning based method, we pick the missed detection and reject false alarm by taking advantage of the appearance information. Images are zoomed for better display. See full images in supplementary files.*

respectively. We set higher saliency threshold ($Q_E = 0.01$) than ($Q_X = 0.001$) as we find the sparse set of points in the regions of interest when only moving object should be identified. In addition, we use $15 \times 15$ block size for Lucas-Kanade optical flow matching. Next, we set the threshold $T_L = 1.0$ and $T_H = 10.0$ to prune the moving object with small and large motion difference. Finally, we use $L = 6$ for Kalman filter where we start the track if we detect the object in six previous frames.

## Quantitative Evaluation

The overall goal of this experiment is to measure the detection accuracy of identifying targets in videos. To measure detection accuracy, we report $F$-score which is the harmonic mean of recall and precision rates computed as following:

$$\text{Recall} = \frac{\text{Number of Detected Targets in all Frames}}{\text{Number of Ground-Truth Targets in all Frames}}.$$

$$\text{Precision} = \frac{\text{Number of Detected Targets in all Frames}}{\text{Number of Detected Objects in all Frames}}.$$

$$F = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}.$$

Here, we define the detected target if our detection has overlap with ground truth.

Table 1 summarizes the accuracy scores. By using deep learning method, we achieve higher precision, recall and $F$-score than purely motion based detection. This indicates that appearance information can complement the miss-detection due to the error in motion estimation. In addition, deep learning method can fully take advantage of manually labeled training dataset with over 95% classification accuracy.

**Detection Accuracy**

| | Only Motion Difference | Deep Learning with Appearance |
|---|---|---|
| Precision | 0.630±0.11 | 0.819±0.09 |
| Recall | 0.766±0.15 | 0.798±0.10 |
| F-Score | 0.684±0.10 | 0.806±0.08 |

## Visual Inspection

For qualitative evaluation, we perform visual inspection. Fig. 4 shows exemplar results on 3 different testing videos. First row illustrates the manually labeled ground-truth. Second and third row represent detection results from our previous method [16] only based on motion and proposed deep learning method with appearance information, respectively. We notice that our previous method generates false alarms on the complex backgrounds such as edges. Furthermore, errors in the motion estimation lead to missed detection of moving objects. Our deep learning based method not only removes false alarm by using the appearance patch in the background subtracted image but also preserves moving objects which have relatively small motion difference.

In Fig. 5, we display the dense Shi-Tomasi corner points extracted from detected moving objects and their motion difference vectors. We use different colors for preserved (red) and deleted (green) points by applying thresholding on the magnitude of motion difference. We observe that preserved and deleted points are mostly located around the target UAV and cloud/edges, respectively. This reflects that we supplement appearance based deep learning classifier by pruning the points based on motion difference, improving the detection accuracy.

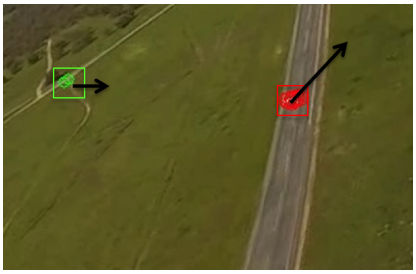

***Figure 5.*** *Motion difference vectors (black) on detected salient points: Red and green dots represent pruned and deleted points based on the magnitude of motion difference vector, respectively. We preserve the points around target UAV (red), while deleting the points located at background (green). This indicates that the magnitude of difference vector between estimated background and local motion is effective to separate the points in the targets from complex backgrounds. Images are cropped for better display.*

## Conclusions

In this paper, we present a deep learning based method for moving objects detection and tracking algorithm. Our method first estimates the background motion from a fast moving camera via perspective transformation model. We then find sparse set of salient points from background subtracted image and use deep learning to classify the patches around the moving objects. Afterwards, dense salients points are extracted only on the positively classified patches and difference between local and background motion is used to prune the actual UAV targets. Kalman Filter is utilized to increase temporal consistency of moving object detection. Experimental results on actual video dataset showed that deep learning method can improve detection accuracy with a help of appearance information.

## References


[1] P. Lin, L. Hagen, K. Valavanis, and H. Zhou. Vision of unmanned aerial vehicle (UAV) based traffic management for incidents and emergencies. In *World Congress on Intelligent Transport Systems*, 2005.

[2] M. Neri, A. Campi, R. Suffritti, F. Grimaccia, P. Sinogas, O. Guye, C. Papin, T. Michalareas, L. Gazdag, and I. Rakkolainen. SkyMedia - UAV-based capturing of HD/3D content with WSN augmentation for immersive media experiences. In *IEEE International Conference on Multimedia and Expo*, 2011.

[3] F. Nex and F. Remondino. UAV for 3D mapping applications: a review. *Applied Geomatics*, 6(1):1–15, 2013.

[4] D. Xing, D. Xu, and F. Liu. Collision Detection for Blocking Cylindrical Objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[5] C. Hane, C. Zach, J. Lim, A. Ranganathan, and M. Pollefeys. Stereo Depth Map Fusion for Robot Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[6] S. Roelofsen, D. Gillet, and A. Martinoli. Reciprocal collision avoidance for quadrotors using on-board visual detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[7] N. Oliver, B. Rosario, and A. Pentland. A Bayesian Computer Vision System for Modeling Human Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[8] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A System for Autonomous Flight Using Onboard Computer Vision. In *IEEE International Conference on Robotics and Automation*, 2011.

[9] P. Viola and M. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[10] Z.Zivkovic and F.van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.

[11] S. Walk, N. Majer, K. Schindler, and B. Schiele. New Features and Insights for Pedestrian Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[12] N. Seungjong and J. Moongu. A New Framework for Background Subtraction Using Multiple Cues. In *Asian Conference on Computer Vision*, 2013.

[13] A. Rozantsev, V. Lepetit, and P. Fua. Flying Objects Detection from a Single Moving Camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[14] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, 1981.

[15] T. Brox and J. Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[16] J. Li, D. Ye, T. Chung, M. Kolsch, J. Wachs, and C. Bouman. Multi-Target Detection and Tracking from a Single Camera in Unmanned Aerial Vehicles (UAVs). In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[17] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *British Machine Vision Conference*, 2009.

[18] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[19] Ingrid Carlbom and Joseph Paciorek. Planar Geometric Projections and Viewing Transformations. *ACM Computing Surveys*, 10(4):465–502, 1978.

[20] K. Hariharakrishnan and D. Schonfeld. Fast object tracking using adaptive block matching. *IEEE Transactions on Multimedia*, 7(5):853–859, 2005.

[21] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *IEEE International Conference on Machine Learning*, pages 807–814, 2010.

[23] S. Ioe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *IEEE International Conference on Machine Learning*, pages 448–456, 2015.

[24] H. Samet and M. Tamminen. Efficient Component Labeling of Images of Arbitrary Dimension Represented by Linear Bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):579–586, 1988.

[25] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently Scaling up Crowd-sourced Video Annotation. *International Journal of Computer Vision*, 101(1):184–204, 2012.