FAST AND ROBUST UAV TO UAV DETECTION AND TRACKING ALGORITHM

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Jing Li

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Charles A. Bouman, Chair

School of Electrical and Computer Engineering

Dr. Juan P. Wachs, Co-Chair

School of Industrial Engineering

Dr. Jan P. Allebach

School of Electrical and Computer Engineering

Dr. Michael D. Zoltowski

School of Electrical and Computer Engineering

Approved by:

Dr. Pedro Irazoqui

Head of the School Graduate Program

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Charles A. Bouman, who always gives me support and advise. His enthusiasm and persistence toward research have always been a great inspiration to me.

I would like to thank my co-advisor Prof. Juan P. Wachs and Prof. Jan P. Allebach for always giving me guidance and support. Their wisdom, exceptional vision and great personality have a great influence in me.

I also would like to thank my committee members, Prof. Dong Hye Ye and Prof. Michael D. Zoltowski, for their time and input to my research.

I thank all my labmates in the Integrated Imaging Lab at Purdue University, especially Dilshan Godaliyadda, Zeeshan Nadir, Yandong Guo, Venkatesh Sridhar, Soumendu Majee and Hani Almansouri, for all the help given to me. All of us make a strong team.

I would like to thank my friends Jieqiong Zhao, Di Wang, Yufang Sun, Yunlan Zhang for providing food when I forgot to eat and acting as family of mine in Purdue. I also thank Qing Li, Xingchi Kong for providing all kinds of support.

Last but not the least, I want to thank my family, Mr. Zhaoping Li, Ms. Zhenrong Wang, Mr. Yongjun Li and Ms. Lanying Li for their unwavering support, encouragement, understanding and love.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

ABSTRACT

Li, Jing Ph.D., Purdue University, May 2019. Fast and Robust UAV to UAV Detection and Tracking Algorithm. Major Professors: Charles A. Bouman, Juan P. Wachs.

Unmanned Aerial Vehicle (UAV) technology is being increasingly used in a wide variety of applications ranging from remote sensing, to delivery, to security. As the number of UAVs increases, there is a growing need for UAV to UAV detection and tracking systems for both collision avoidance and coordination. Among possible solutions, autonamous "see-and-avoid" systems based on low-cost high-resolution video cameras offer the important advantages of light-weight and low power sensors. However, in order to be effective, camera based "see-and-avoid" systems will require sensitive, robust, and computationally efficient algorithms for UAV to UAV detect and tracking (U2U-D&T) from a moving camera.

In this thesis, we propose a general architecture for a highly accurate and computationally efficient U2U-D&T algorithms for detecting UAVs from a camera mounted on a moving UAV platform. The thesis contains three studies of U2U-D&T algorithms.

In the first study, we present a new approach to detect and track other UAVs from a single camera in our own UAV. Given the sequence of video frames, we estimate the background motion via perspective transformation model and then identify distinctive points in the background subtracted image to detect moving objects. We find spatio-temporal characteristics of each moving object through optical flow matching and then classify our targets which have very different motion compared with background. We also perform tracking based on Kalman filter to enforce the temporal consistency on our detection. The algorithm is tested on real videos from UAVs and results show that it is effective to detect and track small UAVs with limited computing resources.

In the second study, 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.

The system in the third study is based on a computationally efficient pipeline consisting of moving object detection from a motion stabilized image, classification with a hybrid neural network, followed by Kalmann tracking. The algorithm is validated using video data collected from multiple fixed-wing UAVs that is manually ground-truthed and publicly available. Results indicate that the proposed algorithm can be implemented on practical hardware and robustly achieves highly accurate detection and tracking of even distant and faint UAVs.

1. MULTI-TARGET DETECTION AND TRACKING FROM A SINGLE CAMERA IN UNMANNED AERIAL VEHICLES (UAVS)

1.1 Introduction

Thanks to recent development in commercial drone use, the sky will be packed with unmanned aerial vehicles (UAVs) [1–3]. One of the most important issues facing UAV use is then collision avoidance or sense-and-avoid capability [4]. Since the size and the energy consumption of the UAV are limited, an inexpensive light sensor such as camera has the great potential to provide cost and weight advantageous avoidance system against the systems currently in use on larger aircraft, like traffic collision avoidance system (TCAS).

The camera based collision avoidance systems then require detection and tracking of other UAVs (targets) from a video [5, 6]. Once other UAVs are detected and tracked, policies and maneuvers for collision avoidance can be adopted. These detection and tracking operations must be computationally efficient to run on-board even if the connection between the aircraft and the control station is lost or some of the on-board sensors fail. A large volume of work in the moving object detection and tracking methods has been developed in 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 [10–12] are developed for surveillance monitoring and even used in the commercial products.

However, it is not appropriate to extend these moving object detection and tracking 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

Fig. 1.1.: Challenge in detecting other UAVs: [*Left*] Original Video, [*Right*] Video with our detection and tracking; Other UAVs are very small and occluded by complex backgrounds (i.e. cloud) and thus not even recognizable by human eyes. Our proposed method detects and tracks multiple small UAVs successfully as highlighted in red boxes.

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 complex backgrounds like clouds and buildings (See Fig. 1.1 *Left*).

To tackle these challenges, we propose a new approach to detect and track other small UAVs from a video filmed by a rapidly moving camera. Specifically, we 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 [13] taking account into globally smooth motion with camera projection. We highlight distinctive points of the moving objects from background subtracted image and then find the local motion of the moving object by applying Lucas-Kanade optical flow algorithm [14]. Using the spatio-temporal features from the estimated local motion of the moving object, we efficiently classify the small target objects. We further apply Kalman filter tracking [15] 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 that are not even visible due to small size and background occlusion (See Fig. 1.1 *Right*).

1.2 Related Work

There are a few attempts to detect and track moving objects using camera-based systems in UAVs [16]. These approaches extract the features in each individual frames and use machine learning techniques to learn the shape and appearance of the target objects in the training dataset [17]. The trained classifiers including Convolutional Neural Networks (CNN) [18] and Random Forests (RF) [19] showed powerful detection performance even in challenging environments with lightning variations and background clutters. However, in order to extract shape and appearance features, they assume sufficiently large and clearly visible moving objects which are not the cases in our UAV applications.

Another class of methods uses motion information for moving object detection and tracking. Since motions can be estimated in local regions between frames, motion-based approaches are suitable for characterizing small moving objects. 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, 20]. Then, the quality of local motion vector is critical for accurate detection, which can be low for the blurred image.

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. Then, our target objects are sharper in the background subtracted image helping identifying good points for flow estimation. More importantly, by comparing background motion and flow vector, we can extract spatio-temporal features which are useful for moving object detection and tracking.

Fig. 1.2.: An overview of our proposed method: We first estimate the background motion between two sequential frames. From resulting background-subtracted image, we detect the moving objects by pruning spurious noise. Among detected objects, we differentiate UAVs from false alarms using spatio-temporal characteristics and track them for temporal consistency.

1.3 Multi-Target Detection and Tracking

As illustrated in Fig. 1.2, we propose an efficient multi-target detection and tracking algorithm for UAVs. We start from the video taken from a moving UAV and aim to detect and track other moving UAVs. In order to achieve that goal, we first estimate the background motion between two sequential video frames and subtract the background to highlight the regions where potential moving objects have occurred. Additional moving object detection operations are performed to differentiate target objects from spurious noise. Finally, we use spatio-temporal characteristics of each detected object to identify actual UAV and incorporate the temporal consistency of detected objects through tracking. In following, we describe each component of our algorithm in details.

1.4 Background Motion Estimation

For background motion estimation, we assume that the background moves smoothly, not allowing local warping. From a sequence of video frames,the background motion is estimated. First we extract a set of points and estimate local motion fields on those selected points. The local motion estimation procedure can be computationally expensive, so it is only performed on a sparse set of selected points based on saliency with appropriately uni-

form distribution. The computed local motion fields are then fit into a global transformation which represents the background motion.

1.4.1 Identify Salient Points

First, we identify salient points in a video frame. Here, we use two different corner detector: Harris corner detector [21] and Shi-Tomasi corner detector [22] . While many other detectors exist, these two are selected based on their computational efficiency. Both corner detectors are based on the assumption that corners are associated with the local autocorrelation function.

Given an image X, we define the local autocorrelation function C at the pixel s as following:

$$C(s) = \sum_{W} [X(s+\delta s) - X(s)]^2 \tag{1.1}$$

where δs represents a shift and W is a window around s.

The shifted image $X(s+\delta s)$ is approximated by a first-order Taylor expansion and then eq. (1.1) can be rewritten as following:

$$\begin{aligned} C(s) &= \sum_{W} [\nabla X(s) \cdot \delta s]^2 \\ &= \delta s^T \sum_{W} [\nabla X(s)^T \nabla X(s)] \delta s \\ &= \delta s^T \Lambda \delta s \end{aligned} \tag{1.2}$$

where ∇X is the first order derivative of the image and Λ is the precision matrix.

In Harris corner detection, a saliency Q is defined according to determinate and trace of Λ.

$$Q(s) = det(\Lambda) - k(trace(\Lambda))^2 \tag{1.3}$$

where $det(\Lambda)$ and $trace(\Lambda)$ are determinate and trace of Λ, k is Harris corner free parameter.

In Shi-Tomasi corner detection, a saliency Q is computed according to eigenvalues of Λ.

$$Q(s) = \min\{\lambda_1, \lambda_2\} \tag{1.4}$$

where λ_1 and λ_2 are eigenvalues of Λ.

After thresholding on Q, we find a set of salient points. To ensure appropriately uniform spatial distribution, we discard points for which there is a stronger corner points at a certain distance.

1.4.2 Find Local Motion Fields on Salient Points

We now find the local motion fields from the previous frame X_t-1 to the current frame X_t on identified salient points. We denote p_{t-1} as one of corner points in X_{t-1}. Then, we compute the motion vector u_t from the point p_{t-1} using Lucas-Kanade method [14] assuming that our local motion is optical flow.

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

$$u_t = \arg\min_{u} \sum_{s \in \mathcal{N}(p_{t-1})} |X_t(s+u) - X_{t-1}(s)|^2 \tag{1.5}$$

where $\mathcal{N}(p_{t-1})$ is the neighborhood around p_{t-1}. It is worth noting that eq. 1.5 is easy to solve with a closed-form solution.

Furthermore, as illustrated in Fig. 1.3, we use bi-directional verification to obtain accurate motion vector such that $||u_t + (u_t)^{-1}||_2$ has small value.

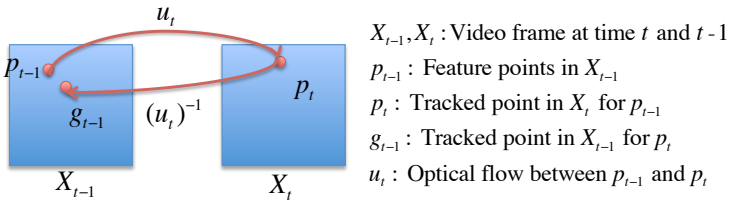

Fig. 1.3.: Bi-directional verification for optical flow. We delete feature points when $||u_t + (u_t)^{-1}||_2$ has large value.

1.4.3 Fit Local Motion Fields to a Global Transformation

After finding a set of local motion fields u_t, we fit them into a global transformation. We now denote $p_t = p_{t-1} + u_t$ as the corresponding point in the current frame X_t through optical-flow matching.

We then find the global transformation H_t which regularizes local motion fields to be smooth in the entire image.

$$H_t = \arg\min_{H} \sum_{p_t \in \mathbf{P}_t, p_{t-1} \in \mathbf{P}_{t-1}} ||p_t - H \circ p_{t-1}||_2^2 \tag{1.6}$$

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.

Then, there are many widely-used global transformation models such as rigid or affine transformation model. Here, we choose the perspective transformation model [13] reflecting the fact that a UAV occupy a small portion of the field of view. The perspective transformation model is efficient to compute because it requires only 9 parameters to describe and it can take account into projection based on the distance from the camera. We assign the resulting perspective transformation in eq. 1.6 as the background motion between two consecutive frames. In consider of the robustness, RANSAC [23] is used for the global transform estimation.

1.5 Moving Object Detection

Given the estimated background motion, we compute the background subtracted image to highlight moving objects which have more complex motion. Then, we identify the salient points in the background subtracted image and use appearance information to find the local motion vector on those points. Then additional test is performed to prune spurious noise assuming that motion of target objects is largely different from the background motion.

1.5.1 Compute Background Subtracted Image

We can subtract the background by taking difference between original image and background motion compensated image. However, the estimated background motion may not be accurate as single plane assumption on the perspective model can be violated in a video. Therefore, we use the background motions estimated from multiple previous frames to obtain more accurate background subtracted image.

We now denote H_{t-1} as the perspective transform between two previous frames from X_{t-2} to X_{t-1}. Then, we compute the background subtracted image E_{t-1} for X_{t-1} by taking average of forward and backward tracing.

$$E_{t-1} = \frac{1}{2}|X_{t-1} - H_{t-1} \circ X_{t-2}| + \frac{1}{2}|X_{t-1} - (H_t)^{-1} \circ X_t| \tag{1.7}$$

where $(H_t)^{-1}$ is the inverse transform of H_t. It is worth noting that we compute the background subtracted image for the previous frame X_{t-1} for symmetry.

1.5.2 Find Salient Points on Moving Objects

The moving objects are highlighted in the background subtracted image E_{t-1}. Then, we need to find the corresponding regions in X_t to detect moving objects in the current frame. Toward this, we first extract Shi-Tomasi corner points in E_{t-1} (refer Section 1.4.1) and propagate them to appearance image X_{t-1}. For each propagated corner point from X_{t-1}, we find the corresponding point in X_t by applying Lucas-Kanade method (refer Section 1.4.2).

We now denote q_{t-1} as the corner point in X_{t-1} propagated from E_{t-1}. Then, the local motion field v_t is computed as following:

$$v_t = \arg\min_{v} \sum_{s \in \mathcal{N}(q_{t-1})} |X_t(s+v) - X_{t-1}(s)|^2 \tag{1.8}$$

where $\mathcal{N}(q_{t-1})$ is the neighborhood around q_{t-1}. It is worth noting that we do not use the background subtracted image but the appearance image to estimate the local motion.

1.5.3 Prune Salient Points based on Motion Difference

We now have the corresponding points in X_t from E_{t-1}, which can be used to detect moving objects. However, we may have points on spurious noise (i.e. edge of background) due to incorrect background motion estimation.

Therefore, we prune the points according to the difference between the estimated background and local motion. This is based on the assumption that target object has very different motion compared with background. We now define the motion difference d_t between the background and moving object as following:

$$d_t = h_t - v_t \tag{1.9}$$

where h_t is interpolated motion vector from the perspective transform H_t at the point q_{t-1}. We then find the pruned point r_t according to the magnitude of motion difference.

$$r_t = q_{t-1} + v_t \qquad \text{if } ||d_t||_2 > T \tag{1.10}$$

where T is the empirical threshold for pruning.

By applying connected component labeling [24] on the set of pruned points, we can cluster them according to spatial proximity. We generate the bounding box for each cluster of points which represents our detection for a single moving object.

1.5.4 Target Classification and Tracking

While we expect our moving object detection to be effective, we still encounter false alarms among the detected objects. Therefore, we obtain a set of spatio-temporal features for each detected object and determine whether the object is our target or not. In addition, in order to prevent intermittent miss-detection and false alarm, we apply a tracking technique enforcing coherent temporal signatures of detection.

1.5.5 Classify Target Objects

Given the detected objects, we perform classification to reject outliers from true targets. We now denote $\mathbf{R}_t^{(n)}$ as a cluster of points to represent the n^{th} object in X_t. Then, we compute the two features which encode spatio-temporal characteristics of the object.

The first feature characterizes the coherency of motion difference vectors in $\mathbf{R}_t^{(n)}$. Here, we assume that the target is non-deformable object, and thus the motion vectors on the target object are consistent. We now define the feature $f_t^{(n)}$ as the angle variance of motion difference vectors.

$$\begin{aligned} f_t^{(n)} &= \frac{\sum_{d_t \in \mathbf{D}_t^{(n)}} |\arctan d_t - \mu_\theta^{(n)}|^2}{S_t^{(n)}} \\ \mu_\theta^{(n)} &= \frac{\sum_{d_t \in \mathbf{D}_t^{(n)}} \arctan d_t}{S_t^{(n)}}, \end{aligned} \tag{1.11}$$

where $\mathbf{D}_t^{(n)}$ is the set of motion difference vectors for $\mathbf{R}_t^{(n)}$ and $S_t^{(n)}$ is the number of points in $\mathbf{R}_t^{(n)}$.

The second feature characterizes the spatial distributions of points in the object. Here, we assume that there are densely distributed salient points for the target object. The feature $g_t^{(n)}$ is then defined as the point density in $\mathbf{R}_t^{(n)}$.

$$g_t^{(n)} = \frac{S_t^{(n)}}{B_t^{(n)}} \tag{1.12}$$

where $B_t^{(n)}$ is the area of minimum bounding box that encloses all points in $\mathbf{R}_t^{(n)}$.

Given these features, we build a classifier to identify target objects. We denote $y_t^{(n)}$ as a classification label where the positive value indicates that n^{th} object is the target. Then, the target classifier is defined as following:

$$y_t^{(n)} = \begin{cases} 1, & \text{if } f_t^{(n)} < T_1 \text{ and } g_t^{(n)} > T_2 \\ -1, & \text{otherwise} \end{cases} \tag{1.13}$$

where T_1 and T_2 are the empirical thresholds for angle variance and point density in $\mathbf{R}_t^{(n)}$.

1.5.6 Track Target Objects

Even though our target classifier reduces the false alarms, we also expect to have intermittent miss-detections and false alarms. These intermittent miss-detections and false alarms can be corrected by observing the temporal characteristics of the detected objects. Therefore, we apply tracking techniques to enforce coherent temporal signatures of detected objects. Specifically, we use the Kalman filter [15] for object tracking.

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:

$$\begin{aligned} b_t &= A\hat{b}_{t-1} + \omega_t \\ c_t &= Mb_t + \varepsilon_t \end{aligned} \tag{1.14}$$

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

$$\begin{aligned} \hat{b}_t &= A\hat{b}_{t-1} + K(c_t - Mb_t) \\ K &= V_\omega M^T (MR_\omega M^T + V_\varepsilon) \end{aligned} \tag{1.15}$$

where V_ω and V_ε are the covariance of ω_t and ε_t, separately.

In our application, we assign the size and location of bounding box for the detected object as state variable b_t and use the constant velocity model to set A and M. 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 consistent. Then, we recover the miss-detection for the positive label track and delete the false alarm of the negative 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.

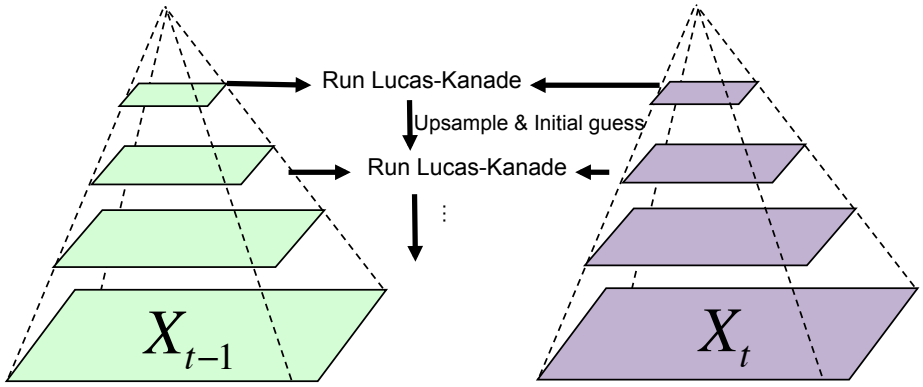

Fig. 1.4.: Use pyramid Lucas-Kanade optical flow in order to satisfy the assumption of small local motion.

1.6 Towards Real Time: Odroid Board Implementation

1.6.1 Parallelization for Lucas-Kanade Feature Tracking

Lucas-Kanade method has the assumption that all neighbor points around given pixel should have same motion and it only works when local motion is relatively small. In our case, the motion of feature point is not always small, so we make use of the pyramid Lucas-Kanade optical flow [25] as illustrated in Fig. 1.4.

The computation of Lucas-Kanade at each layer for feature points can be parallelized. In order to make our algorithm run near real-time on board, we use OpenMP library to implement the parallelization as illustrated in Fig. 1.5.

1.6.2 Mono-Directional Background Subtracted Image

While bi-directional background subtracted image help to suppress noise, it involves more computation then mono-directional background subtracted image. Considering the computational capacity of Odroid board, here we introduce the method of computing mono-directional background subtracted image.

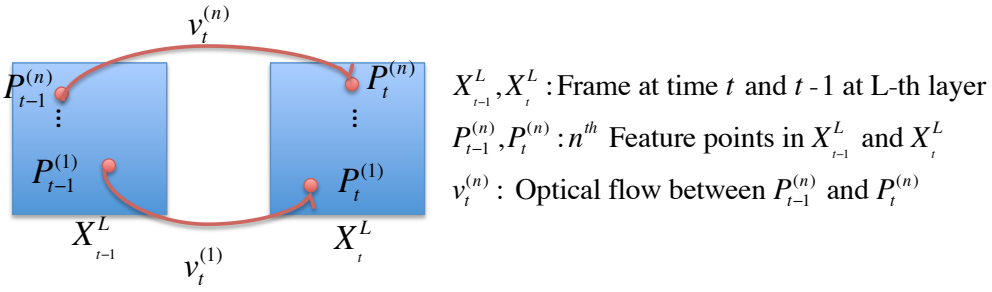

Fig. 1.5.: Use OpenMP to parallel Lucas-Kanade for corner points in each layer.

We now denote H_{t-1} as the perspective transform between two previous frames from X_{t-1} to X_t. Then, we compute the background subtracted image E_{t-1} for X_{t-1} by subtracting the backward motion compensated image.

$$E_{t-1} = |X_{t-1} - (H_{t-1})^{-1} \circ X_t| \tag{1.16}$$

where $(H_t)^{-1}$ is the inverse transform of H_t. Here we only take the mono-directional background subtracted image for computational efficiency. Since applying transformation to the entire image is time-consuming, we only compute E_{t-1} for a certain interval of frames.

1.6.3 Multi-resolution Investigation

The video taken from the UAV has a resolution of 720P or 1080p, both of which is high resolution. Considering the computational capacity of Odroid board, here we down-sample the video in to different resolution(i.e. factor of 2-4).

1.7 Experiment

We evaluate our multi-target detection and tracking algorithm for UAVs on a video data set provided by Naval Postgraduate School. The videos are taken in outdoor environment including real-world challenges such as illumination variation, background clutter, and small target objects.

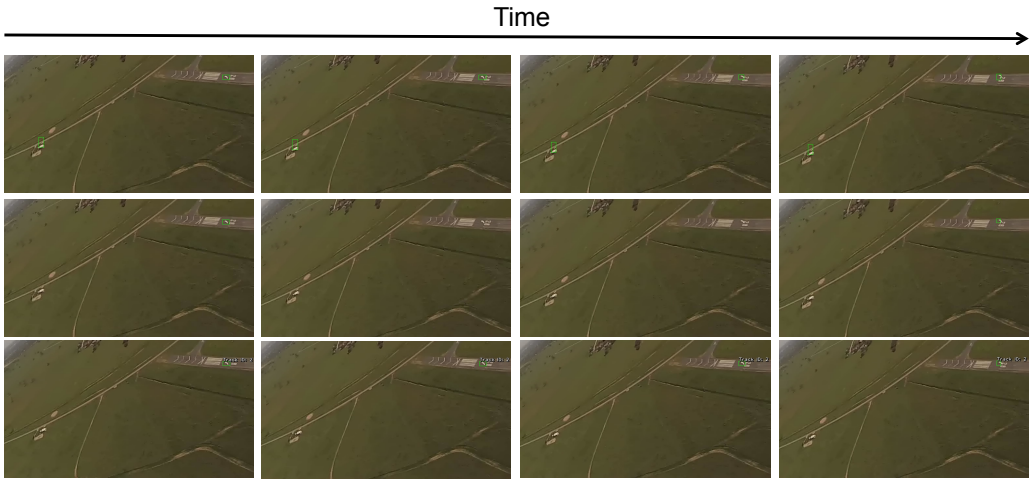

Fig. 1.6.: Results of multi-target detection and tracking algorithms for four consecutive frames: [*Top-Row*] Background subtraction method [10], [*Middle-Row*] Our method with target classification only, [*Bottom-Row*] Our method with target classification and tracking; Green boxes represent the detected objects. Background subtraction method (*Top-Row*) detects false alarms on the complex background (i.e. building). By using the target classifier (*Middle-Row*), we reject false alarms but miss the detection on target UAVs occluded by background. Target tracking (*Bottom-Row*) enforces temporal consistency of our detection recovering intermittent miss-detection. Images are zoomed for better display. See full images in supplementary files.

1.7.1 Experimental Setup

Data Set

The data set comprises 5 video sequences of 1829 frames with 30 fps frame rate. They are recorded by a GoPro 3 camera (HD resolution: 1920×1080 or 1280×960) 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 4) which have various appearances and shapes. We manually annotate the targets in the videos by using VATIC software [26] to generate ground-truth dataset for performance evaluation.

Parameter Exploration

There are important parameters in our multi-target detection and tracking algorithm. To begin with, we extract Shi-Tomasi corner points in original image X_{t-1} (for background motion estimation in 1.4.1) and background subtracted image E_{t-1} (for moving object detection in 1.5.2), respectively. We set higher saliency threshold ($Q_E = 0.15$) for E_{t-1} than that ($Q_X = 0.001$) for X_{t-1} as we find the sparser set of points in the background subtracted image where only moving objects should be identified. In addition, we use 15×15 block size for Lucas-Kanade optical flow matching (Section 1.4.2 and 1.5.2). Next, we set the threshold $T = 1.8$ to prune the points with large motion difference (Section 1.5.3) and the thresholds $T_1 = 5$ and $T_2 = 0.02$ for target classifier with angle variance and point density features (Section 1.5.5). Finally, we use $L = 6$ for Kalman filter where we start the track if we detect the object in six previous frames (Section 1.5.6).

1.7.2 Quantitative Evaluation

The overall goal of this experiment is to measure the detection accuracy of identifying targets in videos. We also analyze the computational time as our algorithm needs to be run on board efficiently.

Table 1.1.: Detection Accuracy

	F
Background Subtraction [10]	0.552
Target Classification Only (Ours)	0.777
Target Classification / Tracking (Ours)	0.866

Detection Accuracy

To measure detection accuracy, we report F-score which is the harmonic mean of recall and precision rates computed as following:

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

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

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

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

We compare our proposed method with the state-of-the art background subtraction method [10] which was developed for pedestrian detection with a static camera. We also report detection accuracy only with our target classification to highlight the importance of tracking. Table 1.7.2 summarizes the accuracy scores. The background subtraction method shows low F-score. This is because fast moving background in the video from UAV causes many false alarms. Our method significantly improves F-score indicating that our target classifier based on background and motion difference correctly identify target objects. By using our target tracking, F-score was further improved due to reduced intermittent miss-detections and false alarms.

Computational Time

We run our algorithm on a standard 3.5GHz clock rate Intel processor desktop with 8GB memory. We implement single-threaded Python codes with OpenCV library. The average computational time for each frame is $112.06 \pm 23.36ms$. The main computational bottleneck is to compute the background subtracted image since applying a global transformation to the large HD image (1920×1080 or 1280×960) is a heavy computational burden. By down-sampling the video by factor of 2 and using multi-thread implementation, our algorithm is efficient enough to run on board near real-time.

1.7.3 Odroid Board Results

Since our ultimate goal is to have our algorithm run on board in real time, we investigate computational time versus detection accuracy with different strategies towards the goal of real time implementation.

(1) We run our algorithm on ODROID XU4 board

(2) Use TBB to parallel the Lucas-Kanade optical flow

(3) We implement Lucas-Kanade optical flow with OpenMP library

(4) Use mono-directional background subtraction instead of bi-directional background subtraction

(5) Increase interval of background motion estimation from 3 to 10

Fig. 1.7 shows the trade off between accuracy and computational time for our algorithm. We display how we move towards our ultimate goal of real time processing step by step. The yellow line in Fig. 1.7 represents the accuracy and computational time curve of single threaded implementation on Odroid board in different resolution. In order to achieve our objective of near real time we need to down-sample the original video by factor of 4. But the accuracy decreases since the moving objects are usually pretty small.

By taking advantage of multi-threads of the Odroid board, the computational time decreases as shown in the blue line in Fig. 1.7. And our implementation of Optial Flow using OpenMP(black line) further decreases the computational time. Until now, the accuracy keeps the same, since we only optimize our implementation without changing our algorithm. But in order to make it run on the board near real time, we need still to push the line much further.

So the next change is that we change the bi-directional background subtracted image to mono-directional background subtracted image computation, aiming to save half of the computation on background subtracted image computation which is one of the most time consuming module in our algorithm. Results show that the computational time drops but with a slight loss of accuracy(red line). We try to increase the interval of background

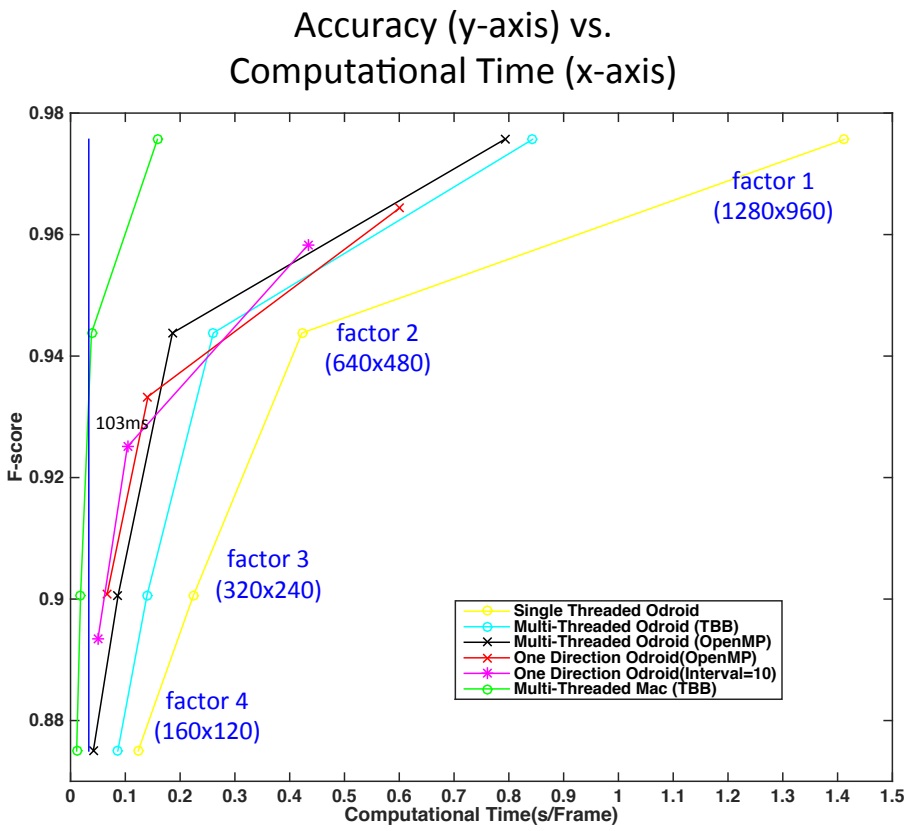

Fig. 1.7.: Time accuracy trade off. Lines with different colors represent accuracy vs. computational time of different methods respectively. Down-sampling the video decreases the computational time with acceptable loss of accuracy.

motion estimation with further saving computational time and slightly decreasing in accuracy(magenta line).

By down-sampling the video by factor of 2 and using OpenMP implementation, our algorithm run on the board near real-time(103ms). Further down-sampling the video can achieve less computational time, but it leads to lose of accuracy since our moving objects are normally small.

1.7.4 Visual Inspection

We complement the quantitative evaluation above with qualitative visual inspection. Fig. 1.6 shows exemplar results from our method with target classifier only and with tracking. For reference, we also illustrate the result with background subtraction method [10]. We notice that background subtraction method generates false alarms on complex backgrounds such as buildings. Our method rejects most false alarms thanks to target classifier based on motion difference but misses the detection on targets due to background clutter. Our Kalman filter tracking significantly improves the detection on targets by enforcing temporal consistency of the detection.

In Fig. 1.8, we display the Shi-Tomasi corner points detected from background subtracted images. 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 building structures, respectively. This reflects that we supplement the background subtraction by pruning the points based on motion difference, improving the detection accuracy.

Fig. 1.8.: Shi-Tomasi corner points detected from background subtracted images in two videos: 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 corner of building structures (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.

2. DEEP LEARNING FOR MOVING OBJECT DETECTION AND TRACKING

2.1 Introduction

Thanks to recent development in drone technology, unmanned aerial vehicles (UAVs) will be pervasive in the sky for commercial and individual needs [1–3]. 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 [10–12] 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 [16]. 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, 20]. 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 [27], 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 detection performance even in challenging environments with illumination variations and background clutters. We further apply Kalman filter tracking [15]

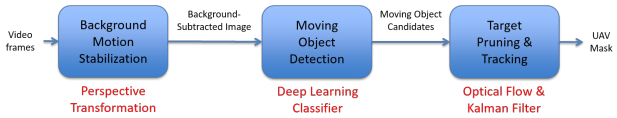

Fig. 2.1.: An overview of our proposed method: We first estimate the background motion between two sequential frames via perspective transformation model. From resulting background-subtracted image, we detect the moving objects by applying deep learning classifier on distinctive patches. Among detected moving object candidates, we prune actual UAVs from spurious noise using the estimated local motion and incorporate the temporal consistency through Kalman filter tracking.

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.

2.2 Moving Object Detection and Tracking

As illustrated in Fig. 2.1, 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 [13] 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 [15] on our detection to reduce the intermittent miss-detection. In following, we describe each component of our algorithm in details.

2.2.1 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 [13]. 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 [28]. We then estimate the perspective transformation $H_{t-1} \in \mathbb{R}^{3\times 3}$ from X_{t-1} to X_t, which regularizes local correspondence matching to be smooth in the entire image.

$$H_{t-1} = \arg\min_{H} \sum_{p_t \in \mathbf{P}_t, p_{t-1} \in \mathbf{P}_{t-1}} ||p_t - H \circ p_{t-1}||_2^2, \tag{2.1}$$

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}, \tag{2.2}$$

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.

$$E_t = |X_t - H_{t-1} \circ X_{t-1}|. \tag{2.3}$$

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).

2.2.2 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 [22]. We choose Shi-Tomasi corner detector due to efficiency. Shi-Tomasi corner detector finds corners associated with the high local autocorrelation.

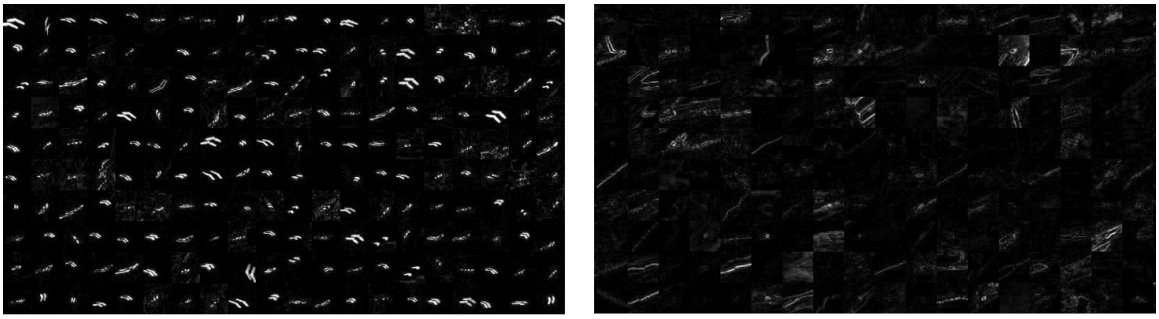

Fig. 2.2.: Example of training patches: [*Left*] True Moving Objects, [*Right*] False Alarms: We note that true moving objects have different appearance in the background subtracted images compared with false alarms. True moving objects tend to be distinctly highlighted while false alarms contain blurred edges.

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

$$\begin{aligned} C_t(s) &= \sum_W [E_t(s+\delta s) - E_t(s)]^2, \\ &\approx \delta s^T \sum_W [\nabla E_t(s)^T \nabla E_t(s)] \delta s, \\ &\approx \delta s^T \Lambda_t(s) \delta s, \end{aligned} \tag{2.4}$$

where δs represents a shift, W is a window around s, ∇ is the first order derivative, and Λ_t is the precision matrix.

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

$$Q_t(s) = \min\{\lambda_t^1(s), \lambda_t^2(s)\}, \tag{2.5}$$

where λ_t^1 and λ_t^2 are two eigenvalues of Λ_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 40×40 patch on each salient point $q_t^{(n)}$ in the background subtracted image. Fig.2.2 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.2.3. First, we apply 16 filters of 3×3 convolution kernel to generate feature maps and then utilize rectified linear units (ReLU) [29] for neuron activation. The batch normalization unit is added between convolution and ReLU to avoid internal covariate shift during

Fig. 2.3.: 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.

mini-batch optimization [30]. Next, we apply max-pooling (spatial down-sampling) 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.

2.2.3 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 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.

$$u_t^{(n,m)} = \arg\min_u \sum_{s \in \mathcal{N}(r_t^{(m,n)})} |X_{t+1}(s+u) - X_t(s)|^2, \tag{2.6}$$

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. 2.6 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:

$$d_t^{(n)} = \frac{1}{M} \sum_{m=1}^{M} (h_t^{(n,m)} - u_t^{(n,m)}), \tag{2.7}$$

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.

$$y_t^{(n)} = \begin{cases} 1, & \text{if } T_L < ||d_t^{(n)}||_2 < T_H \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

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 [15].

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:

$$\begin{aligned} b_t &= A\hat{b}_{t-1} + \omega_t, \\ c_t &= Mb_t + \varepsilon_t, \end{aligned} \tag{2.9}$$

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

$$\begin{aligned} \hat{b}_t &= A\hat{b}_{t-1} + K(c_t - Mb_t), \\ K &= V_\omega M^T (MR_\omega M^T + V_\varepsilon), \end{aligned} \tag{2.10}$$

where V_ω and V_ε are the covariance of ω_t and ε_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.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.12}$$

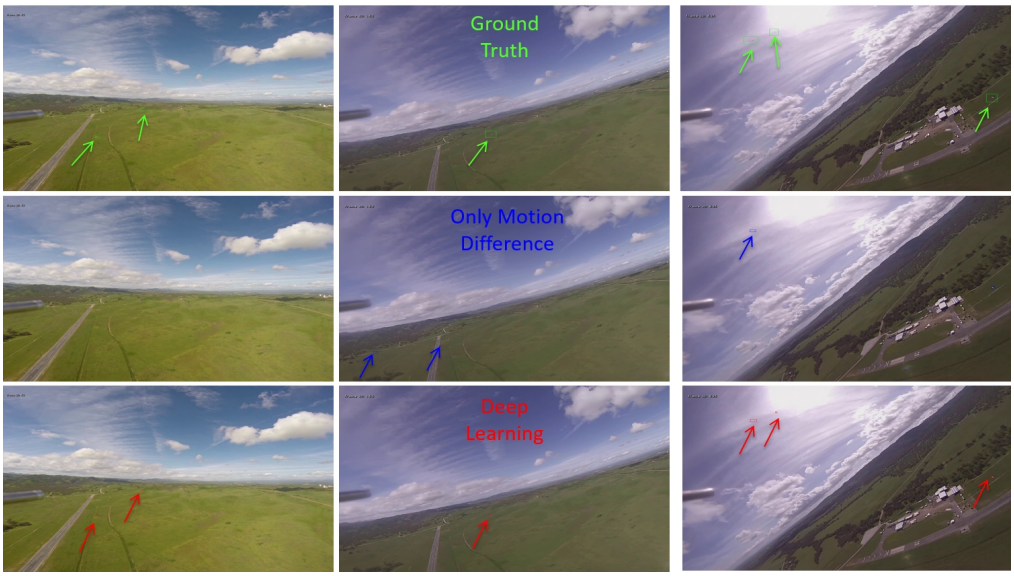

Fig. 2.4.: Results of moving object detection and tracking algorithms for 3 testing videos: [*Top-Row*] Ground-truth annotation (Green), [*Middle-Row*] Moving object detection and tracking algorithm purely based on motion difference pruning [27] (Blue), [*Bottom-Row*] Our deep learning based method toward 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.

.

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.

2.3 Experiment

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 [27] which did not incorporate the appearance information with deep learning.

2.3.1 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: 1920×1080 or 1280×960) 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 [26] 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.

2.3.2 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), 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 15×15 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

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

motion difference. Finally, we use $L = 6$ for Kalman filter where we start the track if we detect the object in six previous frames.

2.3.3 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:

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

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

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

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

Table 2.1 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.

2.3.4 Visual Inspection

For qualitative evaluation, we perform visual inspection. Fig. 2.4 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 [27] 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

Table 2.1.: 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

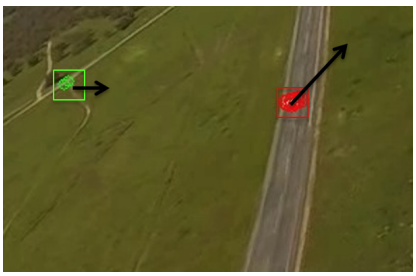

Fig. 2.5.: Motion difference vectors (black) on detected salient points: Red and green dots represent pruned and deleted points based on the magnitude of motion difference vector, respectively. We preserve the points around target UAV (red), while deleting the points located at background (green). This 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.

the appearance patch in the background subtracted image but also preserves moving objects which have relatively small motion difference.

In Fig. 2.5, 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.

3. FAST AND ROBUST UAV TO UAV DETECTION AND TRACKING FROM VIDEO

3.1 Introduction

Unmanned Aerial Vehicle (UAV) technology is being increasingly used in a wide variety of applications ranging from remote sensing, to delivery and security [31–34]. As UAVs have become more popular, there is a growing need for UAV to UAV detection and tracking systems for both collision avoidance and coordination of multiple UAVs [35, 36]. While active sensors such as Lidar or Radar can provide a relatively accurate 3D point cloud, active sensors are typically not practical for small UAVs due to their high weight and power requirements [37].

Alternatively, passive optical sensors such as high-definition digital cameras are light weight and low power sensors that can be used to implement a more traditional "see-and-avoid" system [6, 8, 38]. We will refer to such automated UAV to UAV see-and-avoid systems as autonomous see and avoid. However, in order to be effective, autonomous see and avoid systems will require sensitive, robust, and computationally efficient algorithms for detection and tracking of UAVs from a camera mounted on a moving UAV platform. We will refer to such systems as UAV to UAV detection and tracking (U2U-D&T).

Real-time moving object detection and tracking has been widely studied in the computer vision community [39, 40]. While early object detection methods were based on simple feature extraction [9, 19, 41–43], more recent algorithms such as deep convolutional neural networks have been broadly accepted has having the best accuracy for the problem of detecting general objects in cluttered scenes [44–48].

More recently, there has been particularly high interest in the detection of pedestrians and cars for applications ranging from intelligent highways to the safety of autonomous vehicles. Many pedestrian and car detection algorithms have been developed for surveil-

lance monitoring and even used in commercial products, in which video is captured from a camera on a fixed platform [10–12]. More recently, there has been a variety of research on object detection algorithms for video taken from cameras mounted on a moving vehicle [49–51].

The design of U2U-D&T systems presents many unique challenges that necessitate specialized object detection and tracking solutions [6, 8, 38]. First, in U2U-D&T detection, both the camera platform and the object being detected are rapidly moving. This is because both the UAV with the camera and the UAV to be detected are typically moving independently.[1] Consequently, object detection and tracking algorithms must be robust to background motion that is non-planar and complex [52, 53]. Second, robust detection and tracking requires that algorithms work reliably for both near and distant UAVs. This means that in some cases distant targets may be very small and often obscured by the background. For example, Fig. 3.1 illustrates how distant UAVs may only occupy a small number of pixels, and can be occluded by the clouds or other background clutter. However, in other cases, nearer targets may occupy a much larger field of view. Therefore, appearance by itself may not be a reliable feature for robust detection [54]. Finally, training data for the problem of U2U-D&T detection is scarce and its collection is difficult since it requires the simultaneous coordination, flight, and video capture of multiple UAVs [55].

A number of researchers have studied the problem of detecting small ground objects from wide angle aerial images taken from UAVs [56]. For example, in [8] Meier et. al used computer vision methods to detect static markers on the ground to assist in UAV localization. Other researchers have studied the problem of tracking and estimating the geo-location of ground targets from moving UAVs using computer vision [57–59]. In particular, Khanapuri et. al estimated the geo-localization of multiple ground targets with multiple UAVs using neural network and extended Kalman filters [60]. In other research, Ammour et. al proposed an algorithm for detecting and counting cars from UAV imagery [61], and Teutsch et. al detected moving vehicles in videos taken from a UAV by using frame

[1] We note that fixed wing UAVs, which will be the primary focus of this research, must maintain a minimum airspeed above their stall speed. So they also typically maintain a non-zero ground velocity.

Fig. 3.1.: Challenges in detecting other UAVs: In some cases distant UAVs may be very small and occluded by similar or cluttered backgrounds. In this case, UAVs may not be easily recognizable to human observers.

differencing together with an appearance based classifier. Finally, in [61], LaLonda et. al presented a spatio-temporal algorithm to detect small objects in videos taken from UAVs flying high above targets. This method can detect both moving and stationary objects that are on the ground by using a two-stage neural network in which the first stage detects a region of interest, and the second stage detects the specific location. However, detection of objects flying in the sky is more challenging since there can be greater variation in both the motion and the appearance than typically occurs for objects on the ground.

There have been a number of papers that specifically treat the problem of detection and tracking of moving objects using UAV mounted cameras [62,63]. Rozantsev et. al proposed an algorithm for detecting other flying UAVs by first performing motion compensation to center the moving object, and then using a deep neural network (DNN) to detect the flying target [62, 63]. While this method was effective, it assumed that the UAVs were relatively close so that the target UAV occupied a large number of pixels in the field-of-view (FOV). This allowed the DNN to accurately detect the appearance of the target UAV. Moreover, they used a sliding window to detect the moving target, so that the computation required was large compared to what would likely be available on a small UAV for real-time detection and tracking.

Since many modern detection algorithms depend on training, high quality video training data is also crucial to successful U2U-D&T design. There are a number of datasets taken from cameras mounted on UAVs looking down at ground-based objects. For example, Campus [64] is a dataset taken from a UAV looking at a campus with objects such as cars, pedestrains, bicycles and other objects on a university campus. CARPK [65] is a similar dataset for parking lots containing static cars. DOTA [66] is a more general data set containing videos of public areas in multiple cities. In the videos, there are objects, such as cars, ships, and helicopters, all of which are static. UAVDT [67] is a data set primarily consisting of ground-based moving objects taken from cameras mounted on flying UAVs. UAV123 [68] and UAVDT [67] include some videos of other moving UAVs. However, the number of UAVs in a image is typically small, and the UAVs videos are taken at close distance.

In this chapter, we present a low complexity algorithm for U2U-D&T that is capable of robustly detecting and tracking target UAVs from cameras mounted on a flying UAV platform. This research builds on our previous publications of [27, 69]. Key innovations in our approach are that we use a modular structure formed by a moving target proposer, hybrid classifier, and target tracker in order to minimize computation while achieving high accuracy. Moreover, by integrating our motion and appearance into the classification process, we are able to increase detection probability while reducing false alarms. Finally, we utilize tracking to increase accuracy of detecting faint, distant, and obscured UAVs with intermittent single frame detection.

We also present a publicly available data set of UAV to UAV video suitable for training of U2U-D&T algorithms [2]. The data is unique in that it was taken with multiple UAVs flying simultaneously [55] and is available with associated ground-truth. When tested with this real video data, we find that our proposed U2U-D&T algorithm has a high probability of detection and low false alarm rate even in complex and clustered environments and can detect distant and faint UAVs even when they are difficult for humans to visually detect.

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

3.2 UAV to UAV Detection and Tracking Algorithm (U2U-D&T)

The overall architecture of our algorithm for U2U-D&T is illustrated in Fig. 3.2. The U2U-D&T system has three stages: Moving target proposer, hybrid classifier, and target tracker. In the first stage, target proposals are generated by estimating the background motion between two sequential video frames and highlighting regions where changes have occurred. In the next stage, proposed targets are classified as either true or false by combining deep learning classifier outputs based on motion and appearance information. Finally, we use optical flow together with Kalman tracking to recover accurate continuous tracks even in the presence of missed detections. This structure greatly reduces computation since computationally intensive classifiers need only be applied to regions of interest that are generated by the moving target proposer.

In the following sections, we describe each stage of our algorithm in detail.

3.2.1 Moving Target Proposer

The function of the first stage is to generate proposals corresponding to objects that are moving in front of the background. To do this, we must distinguish between the background motion and that of moving foreground objects. We first estimate the background motion using a global motion model. We then use the global motion to align two sequential image frames, and subtract them to remove background from the image. After removing the background, we generate moving target proposals by extracting patches from the background subtracted image, and the patch locations are determined by identifying salient points.

We implement our moving target proposer in five steps: feature point selection, local motion estimation, global motion estimation, background subtraction, salient patch extraction. In the following, we provide more details about each of these steps.

Feature point selection: In this step, we select a set of points used to estimate global motion.

We first randomly select K points in the original image. For each point, we compute its saliency using Shi-Tomasi corner detector's criteria [22]. Then we discard points for which

there is a more salient point at certain distance D_0. Here $p_{n,*} = \{p_{n,i}\}_{i=1}^{K_n}$ denotes the K_n selected points in the frame X_n.

Local motion estimation: We now find the local motion for $p_{n-1,*}$ points selected in previous step from the previous frame X_{n-1} to the current frame X_n.

We compute the motion vectors $u_{n,i}$ for each point $p_{n-1,i}$ using the Lucas-Kanade method [14] assuming that our local motion is an optical flow. In Lucas-Kanade method, the local motion is computed by solving the least square problem

$$u_{n,i} = \arg\min_{u} \sum_{s \in \mathcal{N}(p_{n-1,i})} |X_n(s+u) - X_{n-1}(s)|^2 \tag{3.1}$$

where $\mathcal{N}(p_{n-1,i})$ is a neighborhood around the point $p_{n-1,i}$. Furthermore, bi-directional verification is utilized. Toward this, we compute the reverse motion v_n using Lucas-Kanade method

$$v_{n,i} = \arg\min_{u} \sum_{s \in \mathcal{N}(\widetilde{p}_n)} |X_n(s) - X_{n-1}(s-u)|^2 \tag{3.2}$$

where $\widetilde{p}_{n,i}$ is the corresponding point in current frame computed as $\widetilde{p}_{n,i} = p_{n-1,i} + u_{n,i}$. Then we discard points when $||u_{n,i} - v_{n,i}||_2 > M_d$. Now the number of remaining points is $K_n^{'}$.

Global Motion Estimation: Our next step is to fit the global background motion using a perspective transform model. We use perspective transform because it is a non-liner transformation which is perfect when the camera has a perfect perspective projection and the world is planar [70].

The perspective transform $y = T(H,x)$ parametrized by matrix H is computed as

$$y = \begin{bmatrix} \frac{h_{11}\cdot x^{(1)} + h_{12}\cdot x^{(2)} + h_{13}}{h_{31}\cdot x^{(1)} + h_{32}\cdot x^{(2)} + h_{33}} \\ \\ \frac{h_{21}\cdot x^{(1)} + h_{22}\cdot x^{(2)} + h_{23}}{h_{31}\cdot x^{(1)} + h_{32}\cdot x^{(2)} + h_{33}} \end{bmatrix}, \tag{3.3}$$

where h_{ij} is the i-th row and j-th column of matrix H, $x^{(k)}$ is k-th component of x. H is a 8 parameter matrix with h_{33} always equals to 1.

Then H_n is computed by fitting $\{p_{n,i}\}_{i=1}^{K_n^{'}}$ and $\{\widetilde{p}_{n,i}\}_{i=1}^{K_n^{'}}$ into global perspective tranform. First we use RANSAC [71] to compute a good set of inliers. Here $p_{n,*}{}' = \{p_{n,i}{}'\}_{i=1}^{K_n^{''}}$ and

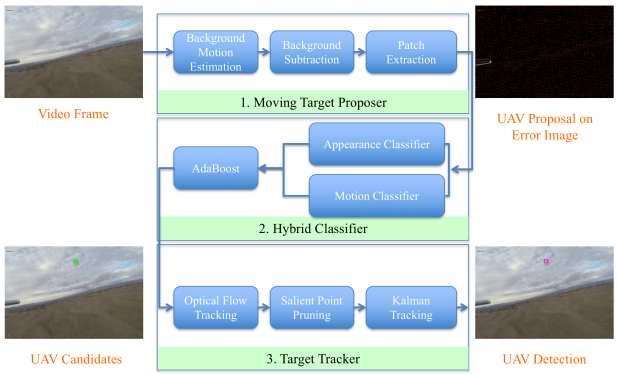

Fig. 3.2.: An overview of U2U-D&T algorithms: We first get the moving UAVs proposals using traditional computer vision method by first estimating background motion between two sequential frames via perspective transform model and then identifying the proposals (red boxes in the upper right image) in the background subtracted image. Then we apply deep network based classifier for the proposals (green box is the moving object candidate). Among detected moving object candidates, we extract salient points and use Optical Flow tracking to track, and increase temporal consistency through Kalman Tracking (magenta box is the final detection).

$\widetilde{p}'_{n,*} = \{\widetilde{p}'_{n,i}\}_{i=1}^{K''_n}$ denotes the K''_n point pairs in the set of inliers. Then H_n is computed by solving the least square problem,

$$H_n = \arg\min_{H} \sum_{i=1}^{K''_n} ||\widetilde{p}'_{n,i} - T(H, p_{n,i}{}')||_2^2, \tag{3.4}$$

using iterative least squares method [72]. We assign the resulting perspective transformation in eq. 3.4 as the background motion between two consecutive frames.

Background subtraction: In this step, we subtract the background to highlight regions where changes have occurred. We compute the background subtracted image because motion of moving targets is different from background motion.

In order to compute background subtracted image, we first estimate previous frame using current frame

$$\hat{X}_{n-1}(s) = X_n(T(H_n, s)), \tag{3.5}$$

where $\hat{X}_{n-1}$ is the estimated previous frame and s denote point location. When components of $T(H_n, s)$ are not integer, we use bi-linear interpolation to get the estimated pixel value. Then the background subtracted image E_{n-1} for X_{n-1} is computed as

$$E_{n-1} = |X_{n-1} - \hat{X}_{n-1}|. \tag{3.6}$$

We only compute E_n for every L_0 frames.

Salient patch extraction: In this step, we identify salient point in the background subtracted image and extract salient patches on original image and background subtracted image located around the salient points. By subtracting estimated background, the moving targets remain salient on background subtracted image. So we determine the location of patches by identifying salient point in background subtracted image.

We use Shi-Tomasi corner detector [22] to obtain locations of moving target proposal's location. When we detect corner points, we have three parameters to control the quality of corner points: more salient than quality level λ^c, minimum distance between two points is greater than d^c, and maximum number of detected points are m^c. Here $q_{n,*} = \{q_{n,i}\}_{i=1}^{Q_n}$ denotes the Q_n selected points for moving target proposals. Then extract 40×40 patch on each salient point $q_{n,i}$ in both the background subtracted image and original image. Each moving object proposal contains two components: one from background subtracted image and one from original image. For the following procedures, we only deal with the proposals instead of the whole image domain.

3.2.2 Hybrid Classifier

In this stage, we use hybrid classifier to prune moving target proposals generated in previous step. Due to camera distortion and the violation of planar assumption in real case, the perspective transform model is not perfect to describe the global motion. In addition, Lucas-Kanade optical flow is also not accurate to estimate the local motion of points. We encounter false alarms in the moving object, so we propose hybrid classifiers to remove spurious noise while keeping real moving targets.

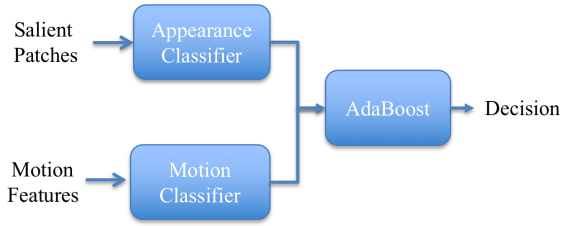

Fig. 3.3.: Hybird classifier uses AdaBoost [73] to combine appearance classifier and motion classifier to classifier the moving target proposals. Motion classifiers take motion features as input and appearance classifier takes salient patches as input.

As shown in Fig. 3.3 our classifier contains three parts: appearance classifier, motion classifier and AdaBoost. We provide more details for each part.

Appearance classifier: We train an appearance classifier in Fig. 3.4 to separate moving objects from false alarms through convolutional neural network. In the convolutional neural network 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. More details about training are provided in Section 3.4.

First, we apply 16 filters of 3×3 convolution kernel followed by ReLUto the input moving target proposals. Then we apply 32 filters of 3×3 convolution kernel followed by ReLU [74] and max pooling layer. Similarly we apply 64 filters of 3×3 convolution kernel followed by ReLU and max pooling layer. In the end, we find probability $p^a_{n,i}$ that the patch on each salient point $q_{n,i}$ belongs to moving object by applying fully connected neural network with soft-max function.

When we do the training, we use dropout [75] to avoid over-fitting. We also use batch normalization to save training time.

Motion classifier: We define the motion difference $d_{n,i}$ for the moving object at $q_{n-1,i}$ between the perspective motion and local motion as following:

$$d_{n,i} = T(H_n, q_{n-1,i}) - \widetilde{q}_{n,i}, \tag{3.7}$$

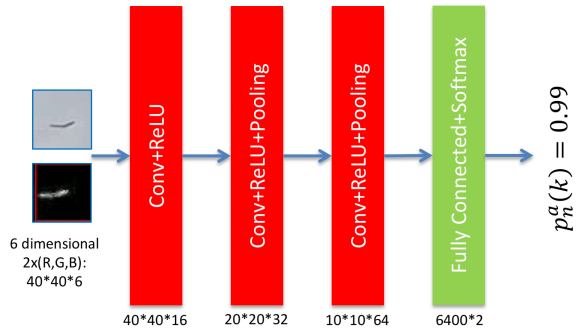

Fig. 3.4.: Network architecture for appearance classifier: The network take 6 channels input, 3 channels for original image and 3 channels for the background subtracted image. Given input patches, 3-layer convolutional neural networks and one-layer of Dense layers are trained to identify moving objects.

where $\widetilde{q}_{n,i}$ is corresponding point in current frame using Lucas-Kanade optical flow matching. $v_{n,i}$ is backward local motion which can be computed using eq.3.2. In addition, we denote $u_{n,i} = \widetilde{q}_{n,i} - q_{n-1,i}$ and $h_{n,i} = T(H_n, q_{n-1,i}) - q_{n-1,i}$.

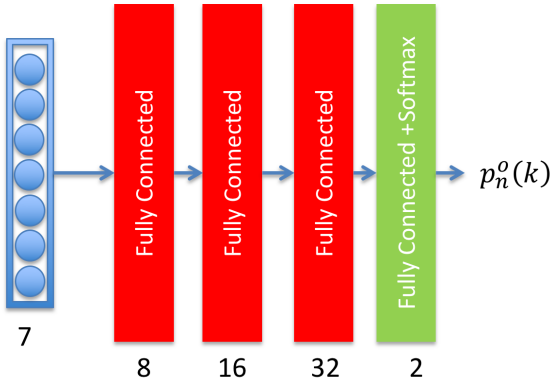

Fig. 3.5.: Network architecture for motion classifier: The network take 7-D motion features. Given input feature, 3-layer neural networks followed by fully connected layer are trained to identify moving objects . In the end, softmax layer is utilized to compute $p_n^o(k)$.

This motion difference represents the actual speed of the moving target relative to the background motion.

Table 3.1.: Table of Motion Features

	Equation	Description
1	$l_{n,i} = \|\|d_{n,i}\|\|$	magnitude of motion difference
2	$\alpha_{n,i} = \arctan d_{n,i}$	angle of motion difference
3	$\varepsilon_{n,i} = \|\|u_{n,i} - v_{n,i}\|\|$	bi-directional verification distance
4	$\theta_{n,i} = \arctan h_{n,i} - \arctan u_{n,i}$	perspective-local motion angle difference
5	$\delta_{n,i} = \|\|\|h_{n,i}\|\| - \|\|u_{n,i}\|\|\|$	perspective-local motion magnitude difference

Combined motion features in Tabel 3.1 with motion difference we use 7-D motion features for motion based target objects classification. We train a classifier to separate moving objects from false alarms through neural network.

The network architecture is described in Fig. 3.5. First, we apply 8 filters neural network followed by 16 filters and then 32 neurons to generate features. In the end, we get probability $p^o_{n,i}$ that the patch on each salient point $q_{n,i}$ belongs to moving object by applying fully connected neural network with soft-max function.

Similarly with appearance classifier, during training, between each layer, dropout layer [75] is used to avoid over-fitting.

AdaBoost: In this step, we use AdaBoost [73] algorithm to combine appearance and motion classification results together to provide a more robust detection results. There is case when moving target has low speed, which makes motion information not reliable in detecting moving target. On the other hand, when moving target is so high that there is only few pixel in the image, appearance information is not enough to identify the target. To solve this challenge, we propose to use AdaBoost to merge appearance and motion information together to make the final decision.

An AdaBoost classifier begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but adaptively adjust the weights of incorrectly classified instances such that difficult cases are given more emphasize in classification.

Here we take $p^a_{n,i}$ and $p^o_{n,i}$ as input for AdaBoost algorithm. For the week classifiers in AdaBoost, we use decision tree [76] for simplicity, maximum depth of the decision tree is 2. We obtain optimal weight of each week classifier by training the training dataset using AdaBoost-SAMME [77]. $w_* = \{w_i\}_{i=1}^{M_0}$ denotes weight for M_0 weight classifiers.

$$\beta_{n,i} = \sum_{m=1}^{M_0} w_m * g_m(p^o_{n,i}, p^a_{n,i}), \tag{3.8}$$

where g_m is decision tree classifier, $p^o_{n,i}$ and $p^a_{n,i}$ are probability computed from motion and appearance separately .

We denote $y_{n,i}$ as a binary label computed as

$$y_{n,i} = \begin{cases} 1, & \text{if } \beta_{n,i} > 0.5 \\ 0, & \text{otherwise} \end{cases}, \tag{3.9}$$

where 1 indicates that the moving object at $q_{n,i}$ is the target and 0 is noise. We only discard proposals which are predicted as noise in AdaBoost classifier. We keep the positive patches as our moving UAV candidates.

3.2.3 Target Tracker

In this stage, we use target tracker to obtain detections for frames between two detection interval and to recover intermittent missed detection. Due to limited computation power, we only add new detection for every L_0 frame. To obtain detection for frames in between we use optical flow tracking. And we also use Kalman tracking to increase temporal consistency.

Target tracker is implemented in three steps: Optical flow tracking, tracked points pruning and Kalman tracking. We provide more details about each step in the following.

Optical flow matching: In this step, we obtain detection in next frame given detection in current frame.

Toward this, we first extract Shi-Tomasi corners for positive patches located at $q_{n,i}$ on E_n. Here parameters to control the quality of corner points: more salient than quality level

λ^t, minimum distance between two points is greater than d^t, and maximum number of detected points are m^t. Here $r^*_{n,i} = \{r^j_{n,i}\}^J_{j=1}$ denotes J corner points extracted on patch located at $q_{n,i}$. We then track each point $r^j_{n,i}$ from X_n to obtain corresponding points $r^j_{n+1,i}$ in the next frame using optical flow matching by applying Lucas-Kanade method (Section3.2.1).

Tracked point pruning: In this step, we prune the incorrectly matched tracked points. Due to inaccuracy of Lucas-Kanade optical flow matching, points have miss match. We use hybrid classifier here to discard mismatched points.

For each corner point $r^j_{n,i}$, we extract 40×40 patch on X_n, and we compute the same 7-D motion features as in Section 3.2.2. We train a new Hybrid classifier for tracked points using the same structure as in Section 3.2.2, except for the appearance classifier, here we only have original frame which is 3 channel input instead of 6. Then we use trained classifier to discard tracked points which are classified as noise. Here $r^{*}_{n,i}{}' = \{r^{j}_{n,i}{}'\}^{J'}_{j=1}$ denotes J' remaining points extracted on patch located at $q_{n,i}$. Final location of detection is computed as average of J' remaining points.

Kalman tracking: We use Kalman Tracking [15] here to make detected objects correspond to coherent temporal signatures as opposed to spurious intermittent noise.

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

$$\begin{aligned} b_n &= A_n \hat{b}_{n-1} + \omega_n, \\ c_n &= M b_n + \varepsilon_n, \end{aligned} \tag{3.10}$$

where A_n is state transition matrix, ω_t controls the transition modeling error, M is measurement matrix, and ε_t represents the measurement error. The estimated output $\hat{b}_n$ is then computed with Kalman gain K:

$$\begin{aligned} \hat{b}_n &= A_n \hat{b}_{n-1} + K(c_n - M b_n), \\ K &= V_\omega M^T (M R_\omega M^T + V_\varepsilon), \end{aligned} \tag{3.11}$$

where V_ω and V_ε are the covariance of ω_t and ε_t, separately.

The motion of the moving object contains two parts, perspective transform and motion difference compared to background motion. So here when we use Kalman tracking, we

also take perspective transform into consideration. However, perspective transform is not linear, so we approximate it using Affine transform. Specifically, we assign the location of bounding box and velocity for the detected target object as state variable, we augment the state variable by setting the 3rd element of b_n as 1.

$$b_n = \begin{bmatrix} b_x \\ b_y \\ 1 \\ b_{vx} \\ b_{vy} \end{bmatrix}, \tag{3.12}$$

where b_x and b_y are x and y location of bounding box, b_{vx} and b_{vy} are horizontal and vertical velocity. And we use the constant velocity model to set A and M.

$$A_n = \begin{bmatrix} h_{11} & h_{12} & h_{13} & 1 & 0 \\ h_{21} & h_{22} & h_{23} & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.13}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{3.14}$$

where h_{ij} is the i-th row and j-th column of matrix H_n which is the matrix we computed from Section 3.2.1.

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 $y_{n-L,i}, \cdots, y_{n-1,i}$ 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.

3.3 UAV to UAV Detection and Tracking Dataset (U2U-D&TD)

U2U-D&TD contains 50 video sequences with up to 8 UAVs in one frame, chosen from around 100 hours of videos taken from a camera mounted on a UAV with multiple UAVs flying in the sky simultaneously.

3.3.1 Data Collection

Our data set is collected from fixed wing UAVs flying high in the sky by Naval Post Graduate School. 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 50 video sequences with 30 fps frame rate. Each video is around one minute. They are recorded by a GoPro 3 camera (HD resolution: 1920×1080 or 1280×960) 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.

3.3.2 Data Annotation

We manually annotate the targets in the videos by using VATIC software [78] (see Fig.3.6) to generate ground-truth data set for training and performance evaluation.

3.3.3 Annotation Refinement Using Detection

There are very challenging cases which is very hard for human eye to annotate (see Fig.3.7).

However our proposed algorithm detected. We then use the detection result to guide annotator to refine the annotation (see Fig.3.7).

Fig. 3.6.: Vatic Annotation interface: first initiate object to be annotated and give it label and index. Then it will track using optical flow, and annotator manually corrected the detection.

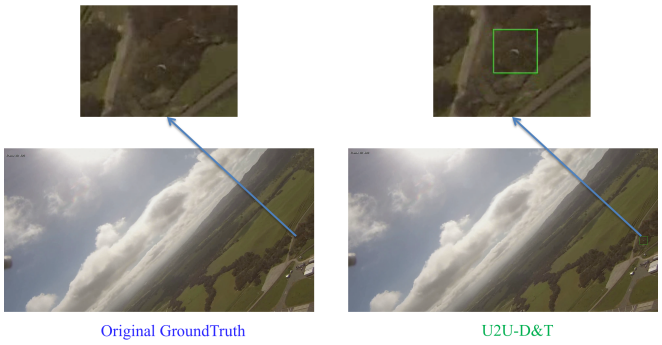

Fig. 3.7.: Missed annotation in first round annotation. Left is original Groundtruth, Right is U2U-D&T's detection results (Green box is detection). U2U-D&T detects UAVs missed by human eye.

3.4 Experiment Results

In this section, we compare U2U-D&T with the results of current state of art method EPFL [63]. In addition, we investigate robustness of our algorithm. The performance comparisons are based primarily on Recall, Precision and F-scores. Computation times were measured on two platforms: a GPU computer and and Odroid board based on an Arm

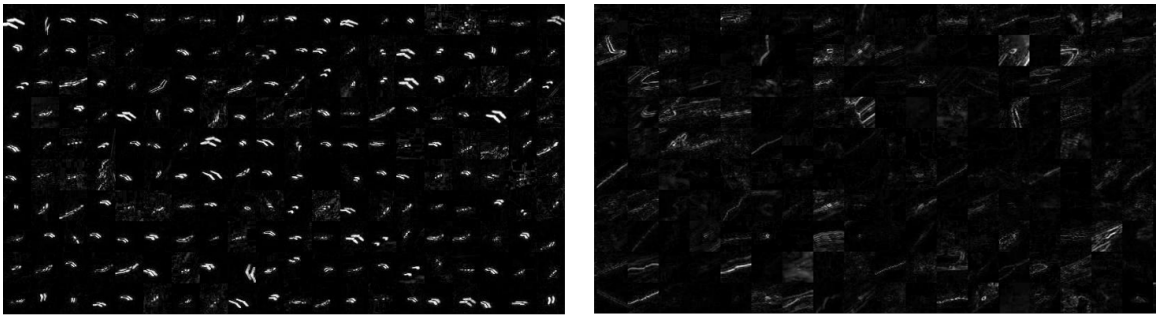

Fig. 3.8.: Example of training patches (Background subtracted image): [*Left*] True Moving Targets, [*Right*] False Alarms: We note that true moving targets have different appearance in the background subtracted images compared with false alarms. True moving objects tend to be distinctly highlighted while false alarms contain blurred edges.

processor shown in Fig.3.9. For GPU computer, it has Two Sky Lake CPUs (2.60GHz) with three Tesla P100 GPUs and 96 GB of RAM. For Odroid board, it has Samsung Exynos5 Octa ARM Cortex-A15 Quad 2GHz, Cortex-A7 Quad 1.3GHz CPUs and 2 GB LPDDR3 RAM at 933MHz. We implemented our algorithm in Python, for neural network, we use Keras and we also use OpenCV library. Our code is not optimized.

Fig. 3.9.: Odroid board utilized in real UAV flying system.

The overall goal of this experiment is to measure the detection accuracy of identifying targets in videos. To measure detection accuracy, we report the recall, precision and F-score as defined below

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

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

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

Here, to determine whether the target is detected using Intersection over Union (IoU) which is defined as

$$IoU = \frac{A_o}{A_u}. \tag{3.15}$$

where A_o is the area of intersection of detected bounding box and ground-truth, A_o is the area of union of detected bounding box and ground-truth. We define the detection of target if IoU is greater than 0.5.

For our algorithm, we list all parameters of U2U-D&T algorithm in Table 3.2.

3.4.1 Training

In all our experiments, we used the U2U-D&TD data set described in Section 3.3. All experiments used 5 fold cross validation where we randomly divided the 50 videos into 5 subsets, trained on 4 subsets and tested on the remaining subset.

Fig. 3.8 shows an example of training patches on the background subtracted image. 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 is powerful to differentiate moving objects from false alarms.

To obtain labels of extracted patches, we use IoU to determine whether the patch is moving target or noise.

Here we provide details of training procedure of our classifiers.

Table 3.2.: Table of Parameterss

Notaion	Description	Value
K	number of points in feature point selection	600
D_0	minimum distance in feature point selection	25
M_d	threshold for bidirectional check in local motion estimation	1
λ^c	quality level of salient patch	0.01
d_c	minimum distance between two salient patches	50
m_c	maximum number of salient patches	600
L_0	detection interval	6
M_0	number of classifier in AdaBoost	20
λ^t	quality level of tracked points	0.001
d_t	minimum distance between two tracked points	1
m_t	maximum number of tracked points	50
L	maximum length of unseen frames for Kalman tracking	6

Appearance classifier: In the neural network, we use categorical cross-entropy as loss function. We divided the training data into training set (9/10) and validation set (1/10). During training we monitor validation loss. We use Adam optimizer. The learning rate is 0.001 for 150 epochs. The batch size was set to 256 for the training on three Tesla P100 GPUs. The training process took about 5 hours for 40 videos.

Motion classifier: In motion classifier, we also use categorical cross-entropy as loss function. We divided the training data into training set (9/10) and validation set (1/10). During training we monitor validation loss. We use Adam optimizer. The learning rate is 0.001 for 1500 epochs. The batch size was set to 1024 for the training on three Tesla P100 GPUs. The training process took about 2 hours for 40 videos.

Table 3.3.: Precision, Recall and F-Score for Robustness Investigation

	U2U-D&T	w/o Kalman	w/o Motion	w/o Appearance
Precision	0.888	0.871	0.829	0.263
Recall	0.890	0.830	0.723	0.674
F-Score	0.889	0.850	0.774	0.379

3.4.2 Comparison with Existing Methods

As a reference, we compare our proposed U2U-D&T algorithm with the state-of-art UAV detection algorithm described in [63], which we will refer to as "EPFL". We use the optimized parameter setting provided in [63], and we randomly selected 5 videos from our data set to compare two methods. Both algorithms ran on the GPU machine. The "EPFL" was implemented with Matlab and used Caffe for the deep learning portion; and our U2U-D&T algorithm was implemented in Python and utilized Keras for the deep learning classifier.

Table 3.4.: Comparison of Precision, Recall and F-Score

	EPFL	U2U-D&T
Precision	0.648	0.887
Recall	0.427	0.892
F-Score	0.515	0.890

Figure 3.10 shows the detection results, from which we can see that our algorithm dramatically improve the detection accuracy. We have less false alarm while have a high

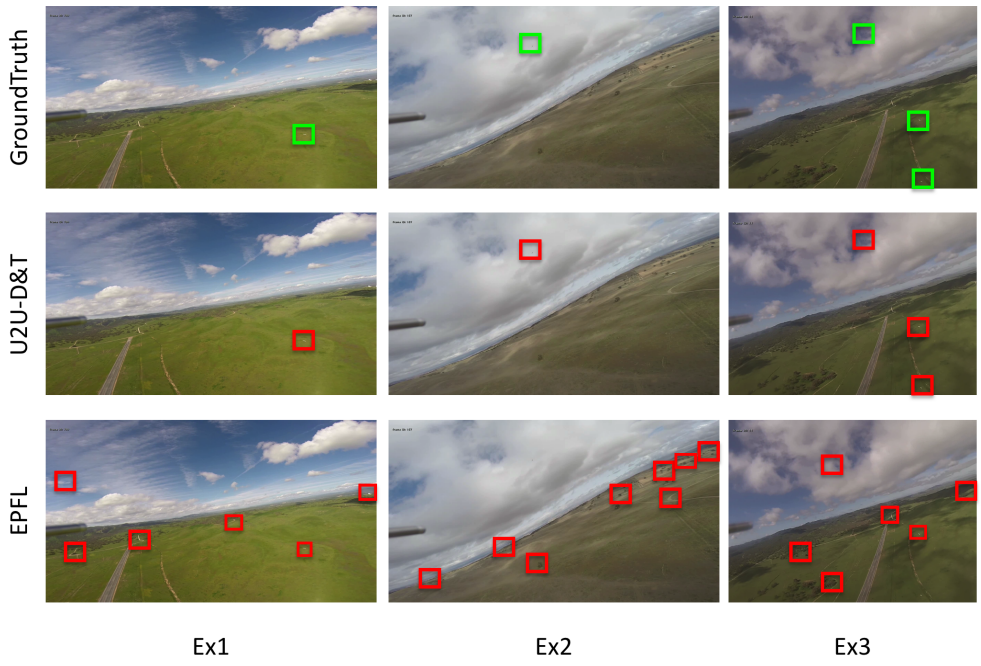

Fig. 3.10.: UAV detection results of U2U-D&T and “EPFL”: [*Top-Row*] Ground-truth annotation (Green), [*Middle-Row*] “EPFL” (Red), [*Bottom-Row*] Our proposed U2U-D&T (Red). Green boxes represent the groundtruth annotations, red box denotes the detection results. ”EPFL” turns to detect a lot false alarms and misses the true moving targets.

Table 3.5.: Computation Time (ms) on GPU

	EPFL	U2U-D&T
Total Time per Frame	167580.93	29.09

recall rate. This is due to that "EPFL" is not optimized for detecting real flying UAVs in real cases. Table 3.4 and Table 3.5 lists out quantitative measures of accuracy and computation speed. The results show that our algorithm can be run in real time on GPU machine.

3.4.3 Classification & Tracking Robustness

In our U2U-D&T algorithm, we identify the moving targets in the background subtracted image. We use hybrid classifiers to combine appearance and motion information to remove false alarms. To save computational time, we only add new detection for certain frame intervals. For frames in between, we use optical flow tracking and Kalman obtain detection.

So our claim is that we use hybrid classifier to increase accuracy and tracking to save computational time while increasing temporal consistency. In order to investigate robustness of our U2U-D&T algorithm, we analyze the importance of motion, appearance, tracked point pruning and the Kalman Tracking.

Towards this, we compare our based line method with the following:

- U2U-D&T without Kalman tracking (w/o Kalman): it removes Kalman tracking at the end of U2U-D&T.

- U2U-D&T without motion classifier (w/o Motion): it uses Appearance based classifier for the detection instead of using AdaBoost to combine Appearance and motion, others are the same as U2U-D&T.

- U2U-D&T without appearance classifier (w/o Appearance): it uses Motion based classifier for the detection instead of using AdaBoost to combine Appearance and motion.

- U2U-D&T without tracked point pruning (w/o Pruning): it removes tracked point pruning in target tracker.

w/o Appearance: From the examples shown in bottom row on Fig. 3.11, we see that motion based classifier detects a lot of false alarms, due to the inaccuracy of Lucas-Kanade optical flow matching. With respect to the accuracy on the whole data set in Table. 3.3, the precision is low while it still has the problem of detecting the real moving targets.
w/o Motion: As shown in third row on Fig. 3.11, appearance based classifier fails when the moving target is far away and has only few pixels. With respect to the accuracy on the whole data set in Table. 3.3, the performance increases compared to motion based classifier. However, the recall rate is still a problem since in real case, moving target can be far away. In collision avoidance application, we need to detect the potential moving target in a very far distance.
w/o Pruning: The examples in Fig. 3.13 visualizes the tracked points for the target tracker part. If we track the patch using optical flow tracking only, there is severe problem having a lot residuals on the background (second row). To solve this, we add hybrid classifier to prune the tracked points, then the tracking is more accurate by deleting spurious noise on the background.
w/o Kalman: Even though we have more accurate optical flow tracking by pruning mismatched points. We encounter missed detection as shown in bottom row on Fig. 3.12. By adding Kalman tracking, we recover the missed detections between two detection intervals (see first row on Fig. 3.12).

Fig. 3.11, Fig. 3.12 and Fig. 3.13 show that our proposed system is most accurate visually. Quantitatively, accuracy is boosted by using AdaBoost to combine appearance and motion information together. In addition, by adding Kalman tracking, both recall rate and precision rate is further increased.

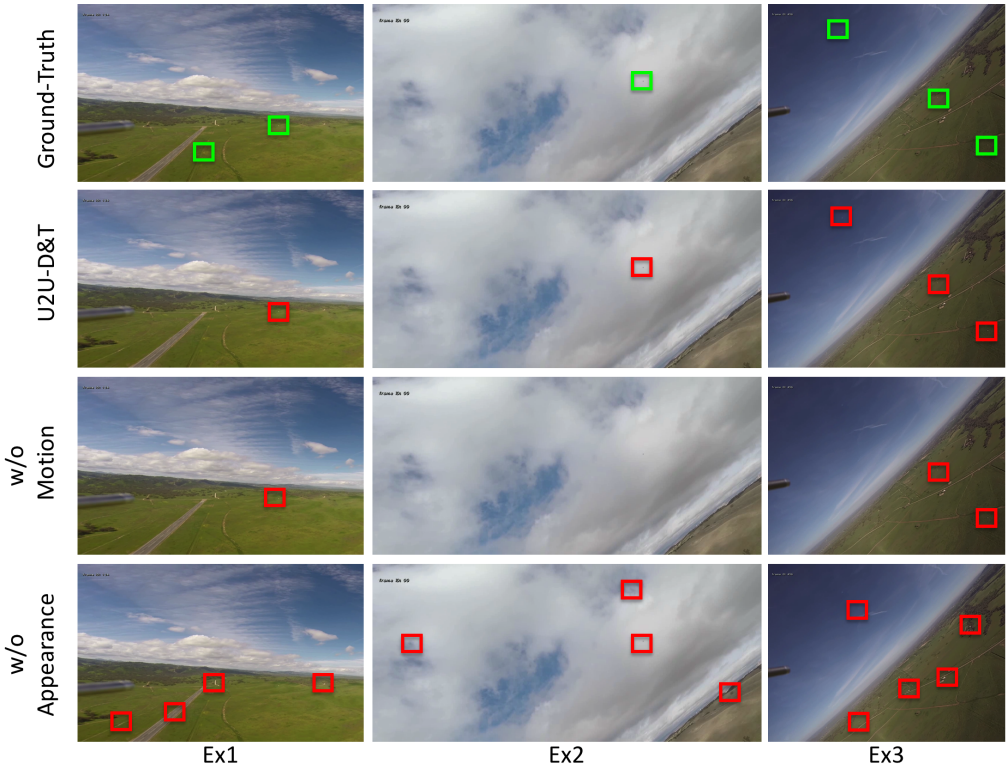

Fig. 3.11.: GroundTruth and detection results from U2U-D&T, w/o Motion, w/o Appearance. [*Top-Row*] Ground-truth annotations in Green boxes, [*Second-Row*] U2U-D&T detection results, [*Third-Row*] w/o Motion and [*Bottom-Row*] w/o Appearance. Three examples of detection results: when use appearance to classification, moving target with few pixels is challenging. For motion based method, too many false alarms due to motion's lack of robustness. By combining motion and appearance, U2U-D&T successfully picked up most of the moving targets. (Note: we cropped the image in order to show moving targets which are too small if we use original frame.)

3.4.4 Computation Time

To investigate the efficiency of our proposed algorithm, here we measure computation time for each part of our algorithm. Moving target proposer is the most time consuming part, which takes around 80% of the total computation time. To save computation time, we

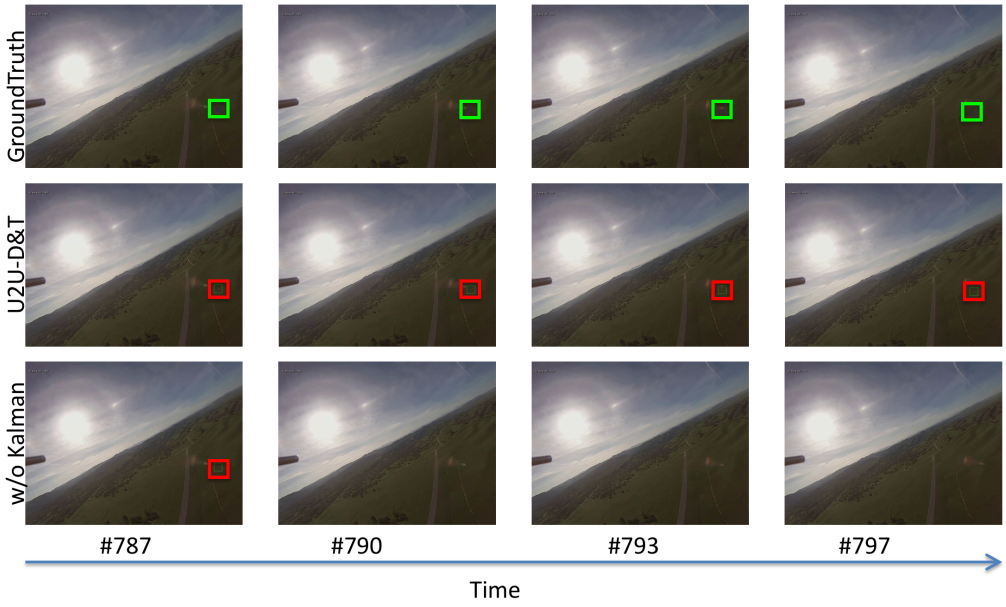

Fig. 3.12.: Results of moving object detection and tracking algorithms sequential frames in a testing video: [*Top-Row*] Ground-truth annotation (Green), [*Middle-Row*] U2U-D&T detection, [*Bottom-Row*] w/o Kalman results. Horizontal line denotes time. Without Kalman tracking, there is intermittent missed detection while U2U-D&T recovers the missed detection. Kalman Tracking helps recover intermittent missed detection.

add new detection every 6 frames, so for each step in "moving target proposer", the update rate is 6. And computation time per frame is divided by the update rate.

We then analyze computation time for each component which boosts accuracy in our algorithm. Hybrid classifier takes about 10% of the total computation time. AdaBoost is also very efficient since it only take a 2 dimensional input.

Target tracker only takes less than 2ms which is also very time efficient while it increases accuracy.

Since our algorithm is designed to be run on the arm board, we measure its computation time on Odroid board. We also implemented the system on Odroid board, which does not have GPU on it and have very limited computational power and memory. Also implementing deep neural network is difficult since the packages are not available for ARM

Table 3.6.: GPU Computation Time

		Time	Rate	Time/Frame
Target Proposor	Background Motion Estimation	6.47ms	6	1.08ms
	Background Subtraction	86.99ms	6	14.50ms
	Patch Extraction	50.13ms	6	8.36ms
Hybrid Classifier	Appearance Classifier	2.28ms	1	2.28ms
	Motion Classifier	1.02ms	1	1.02ms
	AdaBoost	0.02ms	1	0.02ms
Target Tracker	Optical Flow Tracking	1.28ms	1	1.28ms
	Tracked Point Pruning	0.40ms	1	0.40ms
	Kalman Tracking	0.15ms	1	0.15ms
Total Time per Frame				29.09ms

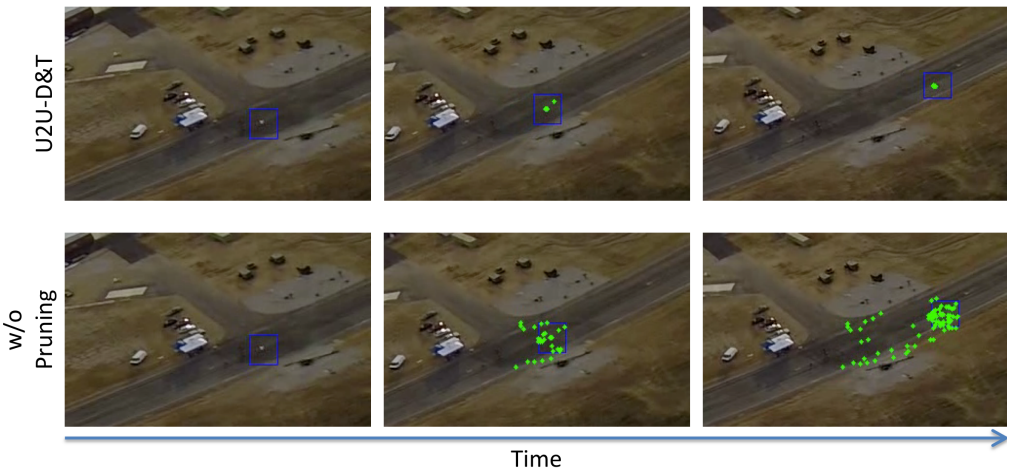

Fig. 3.13.: Results of using Hybird classifier during optical flow tracking. Blue boxes are the ground truth annotation, green dots are the tracked salient point. First row is the results having Hybrid classifier to prune the points and second is the result using optical flow matching directly.

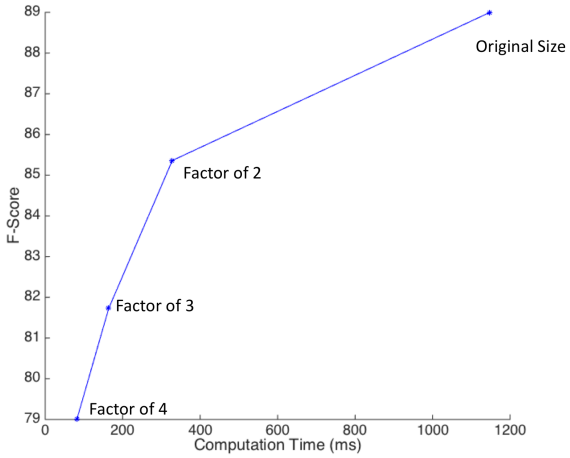

Fig. 3.14.: Trade-off between accuracy and computation time by downsampling the video.

architecture on our OdroidXU4 board. Here, we tested the computational time without the deep learning part.

Table 3.7.: Odroid Computation Time

		Time	Rate	Time/Frame
Target Proposor	Background Motion Estimation	286ms	6	47.67ms
	Background Subtraction	4627ms	6	771.17ms
	Patch Extraction	1491ms	6	248.50ms
Hybrid Classifier	Appearance Classifier	/	/	/
	Motion Classifier	/	/	/
	AdaBoost	/	/	/
Target Tracker	Optical Flow Tracking	84ms	1	84ms
	Tracked Point Pruning	12ms	1	12ms
	Kalman Tracking	4ms	1	4ms
Total Time per Frame				1173.34ms

In order to make it run on the board in real time, we downsampled the video to investigate the relationship between accuracy and computation time.

Fig.3.14 shows the tradeoff between accuracy and computation time. So by downsampling the original video, U2U-D&T is efficient enough to be run on board in near real time.

4. CONCLUSION

In my thesis, we present a framework for constructing large scale data sets for moving objects and deep learning based moving objects detection and tracking algorithm. Our method first use traditional computer vision technique to generate moving object proposals. We then classify the proposals using deep learning classifiers. By combining appearance and motion information, we boost the performance. Afterwards, dense salient points are extracted on the moving object candidates and both motion and appearance information is utilized to prune noise on the background and in the end Kalman Filter is used to increase time coherency. Experiment results on real UAVs data set showed that deep learning classifier can improve detection accuracy. In my thesis, we construct a data set. As future work, we plan to construct data sets with other moving object(cars, pedestrians) and applying our algorithm to different kinds of data sets. We believe our algorithm can be generalized and utilized in automotive driving scenario.

REFERENCES

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*, vol. 6, no. 1, pp. 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*, vol. 22, no. 8, pp. 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*, vol. 57, no. 2, pp. 137–154, 2004.

[10] Z.Zivkovic and F. der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 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] I. Carlbom and J. Paciorek, "Planar Geometric Projections and Viewing Transformations," *ACM Computing Surveys*, vol. 10, no. 4, pp. 465–502, 1978.

[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] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," University of North Carolina at Chapel Hill, Tech. Rep., 1995.

[16] 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.

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

[18] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust Object Recognition with Cortex-Like Mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.

[19] A. Bosch, A. Zisserman, and X. Munoz, "Image Classification Using Random Forests and Ferns," in *International Conference on Computer Vision*, 2007.

[20] T. Brox and J. Malik, "Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2011.

[21] C. Harris and M. Stephens, "A combined corner and edge detector." in *Alvey vision conference*, 1988.

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

[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[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*, vol. 10, no. 4, pp. 579–586, 1988.

[25] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," 2001.

[26] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently Scaling up Crowd-sourced Video Annotation," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 184–204, 2012.

[27] 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.

[28] K. Hariharakrishnan and D. Schonfeld, "Fast object tracking using adaptive block matching," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 853–859, 2005.

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

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

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

[32] M. Bagaram, D. Giuliarelli, G. Chirici, F. Giannetti, and A. Barbati, "UAV remote sensing for biodiversity monitoring: Are forest canopy gaps good covariates?" *Remote Sensing*, vol. 10, no. 9, p. 1397, 2018.

[33] G. Loianno, V. Spurny, J. Thomas, T. Baca, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho *et al.*, "Localization, grasping, and transportation of magnetic objects by a team of MAVs in challenging desert like environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1576–1583, 2018.

[34] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "A hierarchical detection and response system to enhance security against lethal cyber-attacks in UAV networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1594–1606, 2018.

[35] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[36] S. Rabinovich, "Multi-UAV coordination for uncertainty suppression of natural disasters," Ph.D. dissertation, UC Santa Cruz, 2018.

[37] K. May and N. Krouglicof, "Moving target detection for sense and avoid using regional phase correlation," in *International Conference on Robotics and Automation*. IEEE, 2013, pp. 4767–4772.

[38] D. Bratanov, L. Mejias, and J. J. Ford, "A vision-based sense-and-avoid system tested on a scaneagle UAV," in *International Conference on Unmanned Aircraft Systems*. IEEE, 2017, pp. 1134–1142.

[39] K. A. Joshi and D. G. Thakore, "A survey on moving object detection and tracking in video surveillance system," *International Journal of Soft Computing and Engineering*, vol. 2, no. 3, pp. 44–48, 2012.

[40] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2017.

[41] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *International Conference on Computer Vision*. IEEE, 2011, pp. 89–96.

[42] X. Ren and D. Ramanan, "Histograms of sparse codes for object detection," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2013, pp. 3246–3253.

[43] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

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

[45] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[46] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2016, pp. 779–788.

[48] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 3038–3046.

[49] M. Yazdi and T. Bouwmans, "New trends on moving object detection in video images captured by a moving camera: A survey," *Computer Science Review*, vol. 28, pp. 157–177, 2018.

[50] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-aware fast r-cnn for pedestrian detection," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 985–996, 2018.

[51] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Towards reaching human performance in pedestrian detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 973–986, 2018.

[52] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *European Conference on Computer Vision*. Springer, 2012, pp. 228–241.

[53] D. Zamalieva and A. Yilmaz, "Background subtraction for the moving camera: A geometric approach," *Computer Vision and Image Understanding*, vol. 127, pp. 73–85, 2014.

[54] R. LaLonde, D. Zhang, and M. Shah, "Clusternet: Detecting small objects in large scenes by exploiting spatio-temporal information," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2018.

[55] T. H. Chung, M. R. Clement, M. A. Day, K. D. Jones, D. Davis, and M. Jones, "Live-fly, large-scale field experimentation for large numbers of fixed-wing UAVs," in *International Conference on Robotics and Automation*. IEEE, 2016, pp. 1255–1262.

[56] L. W. Sommer, M. Teutsch, T. Schuchert, and J. Beyerer, "A survey on moving object detection for wide area motion imagery," in *Winter Conference on Applications of Computer Vision*. IEEE, 2016, pp. 1–9.

[57] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, "Vision-based target geo-location using a fixed-wing miniature air vehicle," *Journal of Intelligent and Robotic Systems*, vol. 47, no. 4, pp. 361–382, 2006.

[58] N. Farmani, L. Sun, and D. Pack, "Tracking multiple mobile targets using cooperative unmanned aerial vehicles," in *International Conference on Unmanned Aircraft Systems*. IEEE, 2015, pp. 395–400.

[59] L. Zhang, F. Deng, J. Chen, Y. Bi, S. K. Phang, X. Chen, and B. M. Chen, "Vision-based target three-dimensional geolocation using unmanned aerial vehicles," *IEEE Transactions on Industiral Electronics*, vol. 65, no. 10, 2018.

[60] E. M. Khanapuri and R. Sharma, "Uncertainty aware geo-localization of multi-targets with multi-UAV using neural network and extended kalman filter," in *AIAA Scitech Forum*, 2019, p. 1690.

[61] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep learning approach for car detection in UAV imagery," *Remote Sensing*, vol. 9, no. 4, p. 312, 2017.

[62] A. Rozantsev, V. Lepetit, and P. Fua, "Flying objects detection from a single moving camera," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 4128–4136.

[63] ——, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 879–892, 2017.

[64] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*. Springer, 2016, pp. 549–565.

[65] M.-R. Hsieh, Y.-L. Lin, and W. H. Hsu, "Drone-based object counting by spatially regularized regional proposal network," in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 1, 2017.

[66] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2018.

[67] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking," *arXiv preprint arXiv:1804.00518*, 2018.

[68] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European conference on computer vision*. Springer, 2016, pp. 445–461.

[69] D. H. Ye, J. Li, Q. Chen, J. Wachs, and C. Bouman, "Deep learning for moving object detection and tracking from a single camera in unmanned aerial vehicles (uavs)," *Electronic Imaging*, vol. 2018, no. 10, pp. 466–1, 2018.

[70] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.

[71] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[72] H. Goldstein, "Multilevel mixed linear model analysis using iterative generalized least squares," *Biometrika*, vol. 73, no. 1, pp. 43–56, 1986.

[73] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[74] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[75] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[76] D. M. Magerman, "Statistical decision-tree models for parsing," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1995, pp. 276–283.

[77] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[78] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 184–204, 2013.