Fast and Robust UAV to UAV Detection and Tracking from Video

Jing Li, *Member, IEEE,* Dong Hye Ye, *Member, IEEE,* Mathias Kolsch, *Member, IEEE,* Juan Wachs, *Senior Member, IEEE,* and Charles A. Bouman, *Fellow, IEEE.*

***Abstract*—Unmanned Aerial Vehicle (UAV) technology is being increasingly used in a wide variety of applications ranging from remote sensing, to delivery and 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, autonomous "see-and-avoid" systems based on low-cost high-resolution video cameras offer important advantages in terms of light weight and low power consumption. However, in order to be effective, camera based "see-and-avoid" systems require sensitive, robust, and computationally efficient algorithms for autonomous detection and tracking of UAVs from a moving camera.**

In this paper, we propose a general architecture for a highly accurate and computationally efficient UAV to UAV detection and tracking (U2U-D&T) algorithm from a camera mounted on a moving UAV platform. The system is based on a computationally efficient pipeline consisting of a moving target detector, followed by a target tracker. The algorithm is validated using video data collected from multiple fixed-wing UAVs that is manually ground-truthed and is publicly available. Results indicate that the proposed algorithm can be implemented on commodity hardware and robustly achieves highly accurate detection and tracking of even distant and faint UAVs. Open source code for the U2U-D&T algorithm is available at: *https://github.com/jingliinpurdue/Fast-and-Robust-UAV-to-UAV-Detection-and-Tracking.git* .

I. Introduction

Unmanned Aerial Vehicle (UAV) technology is being increasingly used in a wide variety of applications ranging from remote sensing, to delivery and security [1]–[3]. 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 [4], [5]. While active sensors such as Lidar or Radar can provide relatively accurate 3D point clouds, active sensors are typically not practical for small UAVs due to their heavy weight and high power requirements [6].

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 [7]–[9]. 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 [10]. While early object detection methods were based on simple feature extraction [11], more recent algorithms such as deep convolutional neural networks have been broadly accepted as having the best accuracy for the problem of detecting general objects in cluttered scenes [12]. 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 surveillance monitoring and even used in commercial products, in which video is captured from a camera on a fixed platform [13]–[15]. More recently, there has been a variety of research on object detection algorithms for video taken from cameras mounted on moving platforms [16].

The design of U2U-D&T systems presents many unique challenges that necessitate specialized object detection and tracking solutions [7]–[9]. First, in U2U-D&T detection, both the camera platform and the object being detected are rapidly moving and usually with different motion profiles. 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 [17], [18]. 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. 1 illustrates how distant UAVs may only occupy a small number of pixels within the image, 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 [19]. Finally, training data for the problem of U2U-D&T detection is scarce and its collection is difficult since it requires

Jing Li was with the School of Electrical and Computer Engineering, Purdue University. She is now with Texas Instrument Inc., Dallas, TX 75234. Email {lijinginxjtu}@gmail.com.

Dong Hye Ye is with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI 53233, USA. Email: {donghye.ye}@marquette.edu.

Mathias Kolsch are with Naval Postgraduate School, Monterey, CA 93943.

Juan Wachs is with the Department of Industrial Engineering, Purdue University, West Lafayette, IN 47907. Email: {jpwachs}@purdue.edu.

Charles A. Bouman is with the School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Ave., West Lafayette, IN 47907. Email: {bouman}@purdue.edu.

[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. 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.

the simultaneous coordination, flight, and video capture of multiple UAVs [20].

The problem of detecting small ground objects from wide angle aerial images taken from UAVs has been recently explored with exciting results [21]. For example, in [7] Meier et. al used computer vision methods to detect static markers on the ground to assist in UAV localization. Other research has focused on the problem of tracking and estimating the geo-location of ground targets from moving UAVs using computer vision [22]–[24]. In particular, Khanapuri et. al estimated the geo-localization of multiple ground targets with multiple UAVs using neural networks and extended Kalman filters [25]. Ammour et. al proposed an algorithm for detecting and counting cars from UAV imagery [26], and Teutsch et. al detected moving vehicles in videos taken from a UAV by using frame differencing together with appearance classification. Finally, in [26], 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 due to the larger variation in both the motion and the appearance than typically occurs for objects on the ground.

There have been a number of works that specifically treat the problem of detection and tracking of moving objects using UAV mounted cameras [27], [28]. 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 [27], [28]. 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. Saribas et. al [29] proposed a UAV detection and tracking algorithm which combines an appearanced based detector, YOLOv3 [30], and a kernelized correlation filter (KCF) [31] based tracker. However, this algorithms was primarily designed for the detection of UAVs that fill a sufficiently large region of the image so that appearance can be used as a significant discriminator. In addition, the YOLOv3 algorithm requires substantial computational resources.

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 [32] 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. DOTA [33] 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 [34] is a data set primarily consisting of ground-based moving objects taken from cameras mounted on flying UAVs. UAV123 [35] and UAVDT [34] include some videos of other moving UAVs. However, the number of UAVs in an image is typically small, and the UAV videos are taken at close distance.

In this paper, 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 paper builds on our previous publications of [36], [37]. Key innovations in our approach are that we use a modular structure formed by a moving target detector, 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 [20] 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 to be detected by human viewers.

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

The overall architecture of our U2U-D&T algorithm is illustrated in Fig. 2. The algorithm has two stages: the Target Detector and the Target Tracker. The Target Detector generates candidate points that correspond to possible targets by detecting regions with local motion and appearance that is consistent with flying UAVs. Next the Target Tracker uses optical flow together with Kalman tracking to recover accurate continuous tracks of detected targets even in the presence of clutter and missed detections. This structure greatly reduces computation since the computationally intensive classifiers

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

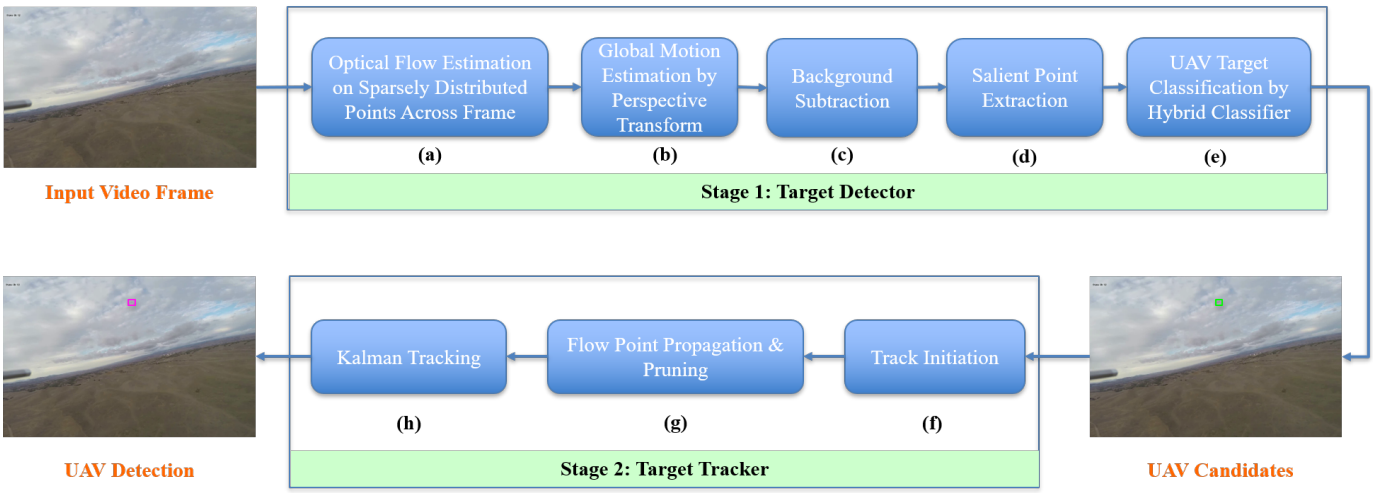


Fig. 2. An overview of the U2U-D&T algorithm: In the first stage, the Moving Target Detector a) estimates optical flow at sparsely distributed points, b) estimates global motion by fitting the points into the background motion model using a perspective transform, c) computes a background subtracted image, d) extracts salient points, and then e) the Hybrid Classifier I uses both appearance and motion information to select candidate points from the salient points. In the second stage, the Target Tracker f) initiates tracks from the candidate points, g) propagates flow points using optical flow and prunes inaccurate propagation using the Hybrid Classifier II, and then h) uses a Kalman Tracking to improve temporal consistency of the detected UAVs (shown with magenta boxes).

need only be applied to sparse regions of the image. In the following sections, we describe each stage of our algorithm in detail.

A. Target Detector

The function of the Target Detector stage is to generate candidate points corresponding to objects that are moving relative to the background. To do this, we first estimate the background motion using a global motion model, and then we subtract subsequent frames after aligning with the estimated global motion parameters. We then detect salient points in this background subtracted image, and use a hybrid detector to classify these points as candidate points or false alarms. Importantly, generating this background subtracted image dominates computation because it requires every pixel to be processed. So to reduce overall computation, the Target Detector is only applied to every L_0^{th} frame.

The Target Detector is implemented in the following five steps detailed below: optical flow estimation on sparsely distributed points, global motion estimation, background subtraction, salient point extraction, and UAV Target classification.

1) Optical flow estimation on sparsely distributed points: In this step, we estimate the optical flow at a sparse set of feature points distributed across the image. To do this, we start by selecting K feature points at uniformly distributed random points across the previous image frame X_{n-1}. For each of the selected points, we compute its saliency using the Shi-Tomasi corner detector's criteria [38] in frame X_{n-1}. We then discard points for which there is a more salient point within a specified distance D_0. This results in a set of K_n feature points from the previous frame X_{n-1}. These feature points are denoted by $p_{n,i} \in \Re^2$ for $i = 1, \cdots, K_n$, where each $p_{n,i}$ is a 2D vector that determines a discrete pixel location in the frame.

Next, we estimate both the forward and backward optical flow vectors at each feature point. More specifically, we use the Lucas-Kanade method [39] to estimate the forward optical flow vector to subpixel accuracy by solving the least squares problem

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

where $\mathcal{N}(p_{n,i})$ is a neighborhood around the point $p_{n,i}$. From this, we can compute the corresponding feature points in the frame X_n given by

$$\tilde{p}_{n,i} = p_{n,i} + u_{n,i} \ .$$

Notice, that in this case, $\tilde{p}_{n,i}$ is a continuous 2D position specified to subpixel accuracy. Using this feature point, we can compute the backward optical flow vector given by

$$v_{n,i} = \arg\min_{v} \sum_{s \in \mathcal{N}(\tilde{p}_{n,i})} \|X_n(s) - X_{n-1}(s-v)\|^2 \qquad (2)$$

In principle, the forward and backward optical flow vectors should be the same as they represent local motion of the rigid body, so we will discard any point such that $||u_{n,i} - v_{n,i}||_2 > M_d$ for some fixed threshold parameter M_d.

2) Global motion estimation by perspective transform: Our next step is to fit optical flow vectors into the global background motion using a 2D perspective transform [40]. We use the perspective transform because it exactly models the 2D background motion in the special case when the background results from the relative motion of a 2D plane in the 3D world and the camera image is formed by a perspective projection (See appendix for proof). In practice, this is often a reasonable approximation since the ground can often be approximated as planar. However we note that high buildings or towers can significantly violate this approximation and generate false targets.

The 2D perspective transform $y = T(x; H)$ transforms a point $x \in \Re^2$ to a point $y \in \Re^2$ and is parameterized by the 3×3 matrix H. The transform can be computed as

$$T(x; H) = \begin{bmatrix} \frac{h_{11} \cdot x_1 + h_{12} \cdot x_2 + h_{13}}{h_{31} \cdot x_1 + h_{32} \cdot x_2 + h_{33}} \\ \frac{h_{21} \cdot x_1 + h_{22} \cdot x_2 + h_{23}}{h_{31} \cdot x_1 + h_{32} \cdot x_2 + h_{33}} \end{bmatrix} , \quad (3)$$

where h_{ij} is the $(i,j)^{th}$ element of the matrix H, and x_k is the k^{th} component of x. Typically, the matrix H can be normalized so that $h_{33} = 1$; so there are only 8 degrees of freedom to the parameter space.

We estimate, H_n, the parameters of the perspective transform from frame $n-1$ to n by solving the following optimization problem.

$$H_n = \arg\min_H \sum_{i \in \mathcal{S}_n} \|\tilde{p}_{n,i} - T(p_{n,i}; H)\|_2^2 , \quad (4)$$

where the optimization is performed using an iterative least squares method [41], and $\mathcal{S}_n$ denotes a subset of original K_n detected points designed to remove outliers. The subset, $\mathcal{S}_n$, is constructed by first removing points using the bi-directional pruning method described above. Then iterative RANSAC [42] processing is then used to remove any additional outliers from the set.

3) Background subtraction: In this step, we compute the background subtracted image by subtracting the subsequent frames after global background motion correction. This highlights objects that are not moving along with the background clutter. To do this, we first estimate the previous frame using the current frame along with the estimated global transformation parameters. This is given by

$$\hat{X}_{n-1}(s) = X_n(T(s; H_n)) , \quad (5)$$

where $\hat{X}_{n-1}$ is the estimated previous frame and $s \in \Re^2$ denotes a point on the discrete 2D grid. When components of $T(s; H_n)$ are not integer, we use bi-linear interpolation to compute the estimated pixel value. Then the background subtracted image is then computed as

$$E_{n-1} = |X_{n-1} - \hat{X}_{n-1}| . \quad (6)$$

4) Salient point extraction: In this step, we identify a sparse set of salient patches in the background subtracted image, E_{n-1}, that are passed on for further processing. Salient points are detected using the Shi-Tomasi corner detector [38] algorithm. The three parameters that control the corner point detection are 1) quality level, λ^c, 2) minimum distance between two points, d^c, and 3) maximum number of detected points, m^c. We denote the Q_n selected salient points as $q_{n,*} = \{q_{n,i}\}_{i=1}^{Q_n}$. For each salient point, we extract a 40×40 patch in both the background subtracted image and original image to form a salient patch.

5) UAV target classification by the hybrid classifier: Hybrid Classifier I is used to classify the salient points as either "targets" or false alarms based on a combination of local motion and appearance features. Fig. 3 shows the structure of the Hybrid Classifier which consists of a Appearance Classifier, a Motion Classifier and an AdaBoost algorithm used to fuse the outputs into a single hard decision.

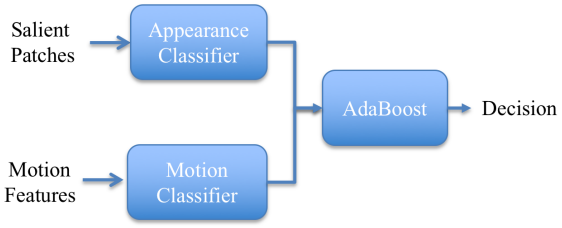


Fig. 3. The Hybrid Classifier uses AdaBoost to compute a classification decision based on the combined results of an appearance and motion classifier.

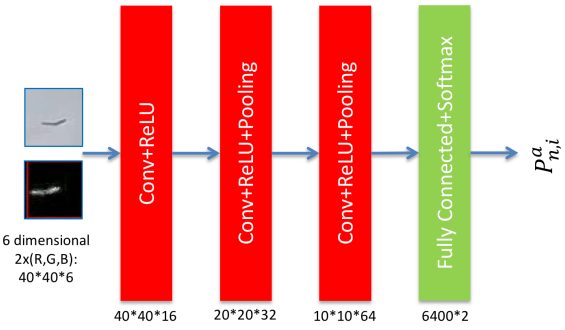


Fig. 4. Appearance classifier architecture: The input to the network is a $40 \times 40 \times 6$ salient input patch corresponding to a two color images. The output represents the probability that the patch is a moving target.

The Appearance Classifier is a convolutional neural network as shown in Fig. 4. The input to the classifier is a $40 \times 40 \times 6$ tensor corresponding to (r, g, b) color patches extracted around the salient point from both the image, X_n, and the background subtracted image, E_{n-1}. We use 3×3 convolution filters with 16-channels for layer 1, 32-channels for layer 2, and 64-channels for layer 3. Each convolution layer uses ReLU activation followed by max pooling with 2×2 receptive fields to reduce the image size. We also use dropout and batch normalization to stabilize the training process. The final layer uses a fully connected network with a softmax activation into two classes corresponding to target and no-target. In the end, we compute the probability $P^a_{n,i}$ that the patch of each salient point, $q_{n,i}$, corresponds to a moving target.

The Motion Classifier is a dense neural network that is trained to detect targets based on their motion features. To do this, we first compute forward and backward optical flow vectors for each salient point in a manner similar to what was done in Section II-A1. For the salient point, $q_{n,i}$, we denote the forward motion estimate as $\mu_{n,i}$ and the backward motion estimate as $\nu_{n,i}$. Both estimates are computed using equations and methods as were used for the feature points specified in equations (1) and (2). We also define the local background motion for the salient point to be

$$h_{n,i} = T(q_{n,i}; H_n) - q_{n,i} ,$$

where H_n is the estimated global background motion parameter matrix. Intuitively, $h_{n,i}$ represents the local motion that would have occurred for any object in the background at that salient point location. The target motion is then defined by

$$d_{n,i} = \mu_{n,i} - h_{n,i} , \quad (7)$$

TABLE I
TABLE OF MOTION FEATURES

	Equation	Description
1	$d_{n,i} = \mu_{n,i} - h_{n,i}$	Target motion
2	$l_{n,i} = \|d_{n,i}\|$	Magnitude of target motion
3	$\alpha_{n,i} = \arctan d_{n,i}$	Angle of target motion
4	$\epsilon_{n,i} = \|\mu_{n,i} - \nu_{n,i}\|$	Bi-directional verification distance
5	$\eta_{n,i} = \arctan h_{n,i} - \arctan \mu_{n,i}$	Angle difference between global and local motion
6	$\delta_{n,i} = \|\|h_{n,i}\| - \|\mu_{n,i}\|\|$	Magnitude difference between global and local motion

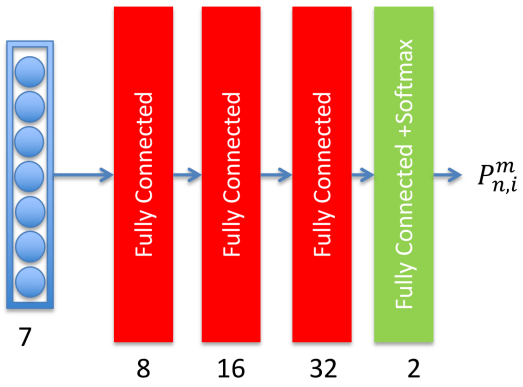


Fig. 5. Motion classifier architecture: The network takes a 7D motion feature vector as input. Given the input feature, a 3-layer neural network followed by fully connected layer are trained to identify moving objects. A softmax layer is utilized at the output to compute $P^m_{n,i}$.

is then the velocity of the moving target relative to the background motion for the i^{th} salient point. Typically, the motion of a target UAV will not be the same as the background motion, so then the resulting non-zero value of $d_{n,i}$ can be used to detect moving targets. Table I lists out the 6 motion features we use to describe each salient point. Since the first feature, $d_{n,i}$, is a 2D vector, this leads to a 7D motion feature vector that we will use as input to the motion classifier.

Fig. 5 shows the network architecture for the motion classifier. It consists of three fully connected layers of size 8, 16, and 32. Like the appearance classifier, we use dropout and batch normalization in training. The final output is generated using a softmax activation with two output classes corresponding to target and non-target. The final output is the probability $P^m_{n,i}$ that the patch corresponding to a salient point $q_{n,i}$ is a moving target.

Finally, we classify each salient point as "target" or false alarm by integrating the outputs of the appearance and motion classifier. To do this, we used the AdaBoost-SAMME [43] algorithm with decision tree [44] classifiers implemented in the sklearn package. The input to the AdaBoost classifier is the 2D vector $(P^m_{n,i}, P^a_{n,i})$, and the parameters d_m and M_0 were used to specify the maximum tree depth and the maximum number of decision trees, respectively. We discard any salient points that are classified as false alarms. The remaining points are passed on as candidate points denoted by $c_{n,i} \in \Re^2$ for $i = 1, \cdots, C_n$. We also note that since the Target Detector is only run every L_0 frames, candidate points can only be generated every L_0 frames.

B. Target Tracker

The function of the Target Tracker is to take candidate points, $c_{n,i}$, that are produced every L_0^{th} frame and to accurately track the target through the intermediate $L_0 - 1$ frames. The target tracker can also add new tracks starting with new candidate points, or remove tracks where targets are no longer detected. We refer to every L_0^{th} frame in which the Target Detector generates a candidate point as a "detection frame", and we refer to the $L_0^{th} - 1$ intermediate frames as "tracking frames".

The Target Tracker is implemented in the following three steps detailed below: track initiation, flow point propagation & pruning and Kalman tracking.

1) Track initiation: This operation of track initiation is only performed on the detection frames that occur every L_0^{th} frames. In this operation, candidate points are used to initiate new tracks whenever they are determined to be distinct from established tracks.

The set of established tracks are denoted by $t_{n,j} \in \Re^2$, where $t_{n,j}$ denotes the 2D position of the j^{th} track in the n^{th} frame. The relative motion estimated by the Kalman tracker for the j^{th} target in the n^{th} frame is denoted by $k_{n,j}$. The details of how $k_{n,j}$ is estimated are given below.

A candidate point, $c_{n,i}$ is only used to start a new track if it represents a new target (i.e. there is no existing track) or it does not match with any existing track. To determine this, we first check if the following conditions hold for all tracks j,

$$\text{New Track if } \begin{cases} \|c_{n,i} - t_{n,j}\| > T_d \\ \text{or} \\ \|d_{n,i} - k_{n,j}\| > T_v \end{cases},$$

where T_d and T_v are position and velocity thresholds, and $d_{n,i}$ is the relative velocity of the candidate point computed in equation (7). These conditions ensure that the new candidate point has a position or velocity that is distinct from existing tracks.

Once the decision is made to start a new track, then a set of points are extracted about the target that we call *flow points*. Intuitively, the flow points identify distinct features of the object that can be tracked individually.

We denote the flow points for the j^{th} target in the n^{th} frame by $r_{n,j,l}$ where l indexes the particular flow point. The goal of using flow points is to improve robustness by tracking a set of feature points in the target, rather than tracking a single centroid point [45].

To do this, we first extract an $N_c \times N_c$ patch about the candidate point, and from this patch a set of flow points

TABLE II
TABLE OF PARAMETERS

Notaion	Description	Value
K	Number of selected feature points	600
D_0	Minimum distance between feature point	25
M_d	Bidirectional check threshold for local motion	1
λ^c	Quality level of salient patch	0.01
d^c	Minimum distance between salient patches	50
m^c	Maximum number of salient patches	600
L_0	Detection interval	6
d_m	Maximum depth of decision tree	2
N_c	Size of candidate patch	40
M_0	Number of estimator in AdaBoost	20
λ^t	Quality level of flow points	0.001
d_t	Minimum distance between flow points	1
m_t	Maximum number of flow points	50
σ_ω	Transition modelling error level	0.1
σ_ϵ	Measure error level	1.0
T_d	Minimum distance to initiate new track	20
T_v	Minimum motion difference to initiate new track	0.5
L	Maximum unseen frames for Kalman tracking	6

are extracted using the Shi-Tomashi algorithm as was done in Section II-A4. However, since we require multiple flow points for each candidate point, we use different values of the 3 tunable parameters of the Shi-Tomasi algorithm to get more points as shown in Table II. The three parameters are, quality level λ^t, minimum distance between two points d^t and maximum number of detected points m^t.

2) Flow point propagation and Pruning: Flow point propagation and Pruning is run for every frame. Its function is to propagate the set of all flow points at time n to the next frame at time $n+1$ and to remove any flow points that do not correspond to true target features. Notice, that the set of flow points includes any points generated in track initiation as well as flow points that have been propagated from previous frames.

For each flow point, $r_{n,j,l}$, we compute a corresponding local motion vector, $u_{n,j,l}$, using the Lucas-Kanade method (Section II-A1). Then we propagate $r_{n,j,l}$ to the next frame using,

$$r_{n+1,j,l} = r_{n,j,l} + u_{n,j,l} \ . \tag{8}$$

Next, we use a second hybrid classifier to prune the flow points based on appearance and motion information with the goal of removing inaccurately propagated flow points. We refer to this second hybrid classifier as the Hybrid Classifier II.

For each flow point, $r_{n,j,l}$, we extract a $N_c \times N_c$ patch centered at $r_{n,j,l}$ from X_n. Then, for each flow point we compute a motion feature vector $f_{n,j,l} \in \Re^7$ listed in Table I using the same method we used to extract motion feature vectors for salient points in Section II-A5.

Next, we use the Hybrid classifier II to classify the flow points as either "target" (class = 1) or "no target" (class = 0). The inputs to the classifier are the motion feature vector, $f_{n,j,l}$, and an $N_c \times N_c \times 3$ sized patch centered on the flow point that is extracted from the (r, g, b) color image X_n. It is important to note that, in this case we do not have access to the background subtracted image as we did in the Hybrid classifier I, and therefore, cannot use it as an input to the classifier.

Any flow point that is classified as "no target" is pruned from the set of available flow points and is no longer propagated forward or used for tracking. If all flow points are deleted from a track, then the track is designated as "empty". If a track remains empty for L frames, then the track is removed and is no longer tracked by the Kalman filter.

3) Kalman tracking: The function of the Kalman tracker is to better estimate the position of the target, $t_{n,i}$, and to track the target through the tracking frames. Importantly, we use a Kalman Filter to track the relative velocity of the target denoted rather than tracking the relative position of the target. We do this because in experiments we found it to be both simpler and more robust to target motion.

To do this we start by computing a preliminary track location, $\tilde{t}_{n+1,j}$, for the j^{th} track using the previously selected valid flow points,

$$\tilde{t}_{n+1,j} = \frac{1}{|\mathcal{T}_{n,j}|} \sum_{l \in \mathcal{T}_{n,j}} r_{n+1,j,l} \ , \tag{9}$$

where $\mathcal{T}_{n,j}$ is the set of current flow points for j^{th} track in frame n.

Next, we compute the global motion at the track location as

$$v^g_{n,j} = T(t_{n,j}; H_n) - t_{n,j} \ , \tag{10}$$

and local motion of the candidate point as

$$v^l_{n,j} = \tilde{t}_{n+1,j} - t_{n,j} \ . \tag{11}$$

Then, we compute the measured velocity for $t_{n,i}$ relative to the background motion is given by,

$$k_{n,j} = v^l_{n,j} - v^g_{n,j} \ . \tag{12}$$

We use the Kalman filter to track the $k_{n,j}$ and $\dot{k}_{n,j}$ the derivative of $k_{n,j}$ with respect to time, using the following model. We use the simple constant velocity model to obtain the estimated motion of the target to have smooth motion and to try to recover from any intermittent missed detections.

$$\begin{aligned} b_{n,j} &= A b_{n-1,j} + \omega_{n,j}, \\ k_{n,j} &= M b_{n,j} + \epsilon_{n,j}, \end{aligned} \tag{13}$$

where A is state transition matrix, M is the observation matrix, $b_n = \begin{bmatrix} k_{n,j} & \dot{k}_{n,j} \end{bmatrix}^T$ is the state variable, $\omega_{n,j} \sim N(0, \sigma^2_\omega I)$ is the transition modeling error, and $\epsilon_{n,j} \sim N(0, \sigma^2_\epsilon I)$ represents the measurement error. Here I is the identity matrix.

For the choice of A and M, we use the "constant velocity model", where

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{14}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ . \tag{15}$$

It is important to note that, we are in fact tracking a velocity, $k_{n,i}$, and therefore the "constant velocity" refers to the velocity of $k_{n,i}$ and corresponds to constant relative acceleration of the target.

Fig. 6. Vatic Annotation interface: first let annotator initiate object to be annotated and give label and index to the object. Then the software will provide detected object in consequential frames. The Annotator need manually correct the detection.

Finally, we estimate the location of candidate point at time $n+1$ as,

$$t_{n+1,j} = t_{n,j} + v^g_{n,j} + k_{n,j}. \tag{16}$$

We fix the size of detection as N_c.

III. Dataset for UAV to UAV Detection and Tracking

We created a dataset referred as U2U-D&TD which contains 50 video sequences with up to 8 UAVs in one frame, chosen from approximately 100 hours of videos. All the videos was taken from a camera a GoPro camera mounted on a single UAV flying with a larger group of multiple UAVs flying simultaneously. The videos were taken outdoor with 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 approximately one minute in duration. 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 step, we masked out the pitot tube which is mounted on the aircraft and in the cameras field of view. For each video, there are multiple target UAVs (up to 8) which have various appearances and shapes. We manually annotated the targets in the videos by using VATIC software [46] (see Fig.6) to generate a ground-truth dataset for training and performance evaluation.

The data set contained some challenging detection cases that are very hard to detect visually (see Fig.7). In some cases, our algorithm detected UAV's that were not detected as ground truth, but could be visually detected after they were identified. In these cases, we refined the ground truth annotation by adding these additional tracks once identified (see Fig.7).

IV. Experimental Results

In this section, we investigate the accuracy and robustness of our algorithm and compare results to existing state-of-the-art methods. The accuracy comparisons are based primarily on Recall, Precision and F-scores, and computation times are measured on two platforms: a GPU platform and and Odroid board based on an Arm processor.The GPU platform consists of two Sky Lake CPUs (2.60GHz) with one Tesla P100 GPU and 96 GB of RAM; while the Odroid board consists of a Samsung Exynos5 Octa ARM Cortex-A15 Quad 2GHz, along with a Cortex-A7 Quad 1.3GHz CPU and 2 GB of LPDDR3 RAM clocked at 933MHz. We implemented our algorithm in Python without custom low-level optimization using a combination of OpenCV libraries for general purpose computer vision operations and Keras/TensorFlow libraries for neural networks.

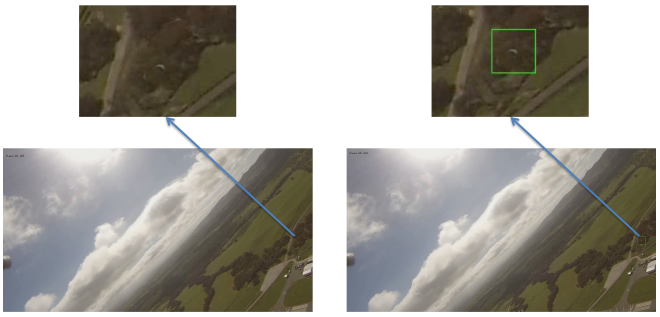


Fig. 7. Missed annotation in the first round of annotations. Left is original ground truth, Right is U2U-D&T's detection results (Green box is detection). U2U-D&T detects UAVs missed by human eye.

In our experiments, we report the recall, precision and F-score as defined below

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

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

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

We define a "detected target" as one for which $IoU \geq 0.5$ where the Intersection over Union (IoU) is defined by

$$IoU = \frac{A_o}{A_u}\ , \tag{17}$$

A_o is the area of intersection of detected bounding box and ground-truth, and A_u is the area of union of detected bounding box and ground-truth. All parameters for our U2U-D&T algorithm are listed in Table II.

We bench-marked our results against a state-of-the-art method which we denote as "EPFL" [28] using the the optimized parameter setting provided in [28]. We note that the EPFL algorithm was implemented with Matlab and used Caffe for the deep learning portion.

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

The training patches in Hybrid Classifier I are the $N_c \times N_c$ patch centered at the salient points in both background subtracted image and the original image. The training patches in Hybrid Classifier II are the $N_c \times N_c$ patch centered at the flow

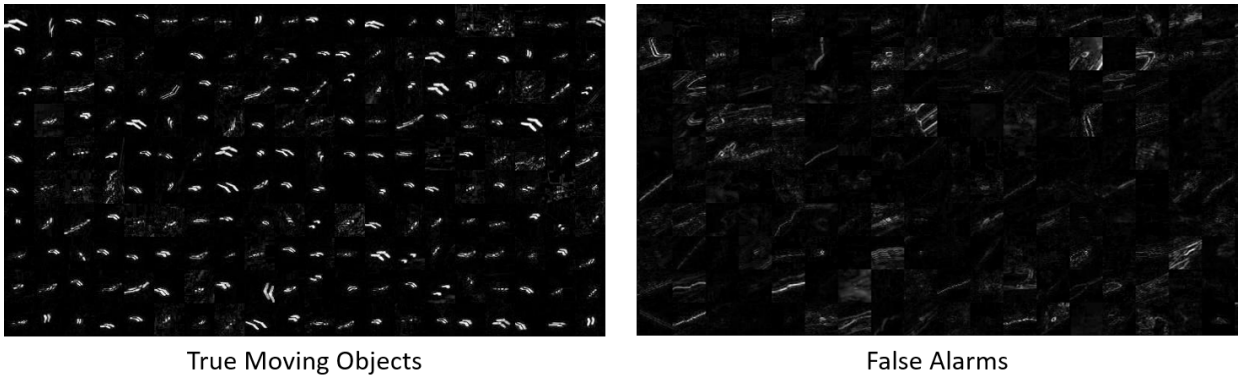


Fig. 8. Example of training patches (Background subtracted image): [*Left*] True Moving Targets, [*Right*] False Alarms: We note that true moving targets have different appearance in the background subtracted images compared with false alarms. True moving objects tend to be distinctly highlighted while false alarms contain blurred edges.

TABLE III
COMPARISON OF PRECISION, RECALL AND F-SCORE ON GPU

	EPFL	U2U-D&T
Precision	0.648±0.16	**0.887**±0.10
Recall	0.427±0.15	**0.892**±0.06
F-Score	0.515±0.14	**0.890**±0.07

points in the original image. The labels of the training patch are obtained by using the same criteria as computing detection by computing the IoU with ground-truth using Equation 17.

Fig. 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. True moving objects tend to have the shape of a high contrast doublet V-shape, while false alarms have an indistinct form. The V-shape is likely due to the delta wing shape of the UAVs, and the doublet form is caused by the motion of the UAVs between frames. Therefore, appearance information is a powerful feature for the differentiation of moving UAVs from false alarms.

For both neural networks used in the appearance and motion classifiers, we followed the following procedure. We used categorical cross-entropy as the loss function, and we divided the training data into a training set (9/10) and validation set (1/10). During training we minimize the validation loss, used the Adam optimizer, and perform the optimization on the Tesla P100 GPU. For the appearance classifier, the learning rate was set to 0.001 for 150 epochs with a batch size of 256; and for the motion classifier, the learning rate was set to 0.001 for 1500 epochs with a batch size of 256. The training took approximately 5 hours for 40 videos for the appearance classifier; and it took approximately 2 hours for 40 videos for the motion classifier.

A. Comparison with Existing Methods

For comparison, we bench-marked our method, U2U-D&T, against the EPFL algorithm as described above [28] tested on 5 videos randomly selected from our data set. Both algorithms were run on the GPU platform.

TABLE IV
COMPARISON OF COMPUTATION TIME (MS) ON GPU

	EPFL	U2U-D&T
Time per Frame	167580.93	29.09

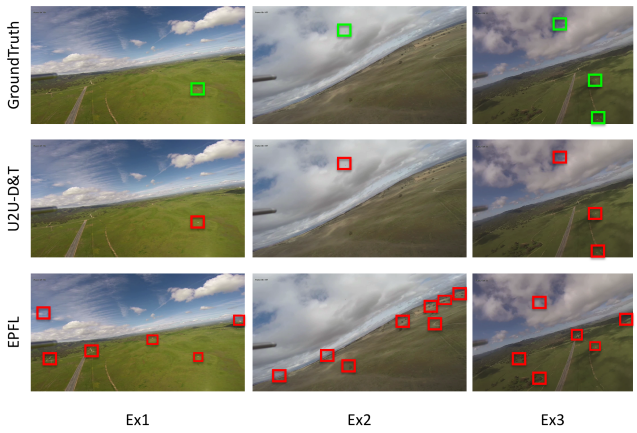


Fig. 9. UAV detection results of U2U-D&T and "EPFL": [*Top-Row*] Ground-truth annotation (Green), [*Mid-Row*] "EPFL" (Red), [*Bottom-Row*] Our proposed U2U-D&T (Red). Green boxes represent the groundtruth annotations, red boxes denote the detection results. "EPFL" turns to detect a lot false alarms and misses the true moving targets.

Figure 9 shows the detection results for some typical frames. Notice that our algorithm has many fewer false alarms but detects all the ground-truthed targets compared with EPFL. This observation is consistent with the classification accuracy results of Table III which show that the U2U-D&T algorithm has much higher precision and recall. Table IV shows that the computation time for the U2U-D&T algorithm is much lower than the EPFL algorithm and is also consistent with real-time operation at 30 fps.

The U2U-D&T algorithm has two major advantages over the EPFL algorithm in this application. First, the EPFL algorithm is designed to both detect targets and their motion based on appearance. However, distant UAV may be very small, in this case, appearance-based processing may be unreliable. Second, the EPFL algorithm detects targets using a sliding window. This requires that the appearance classifier to be

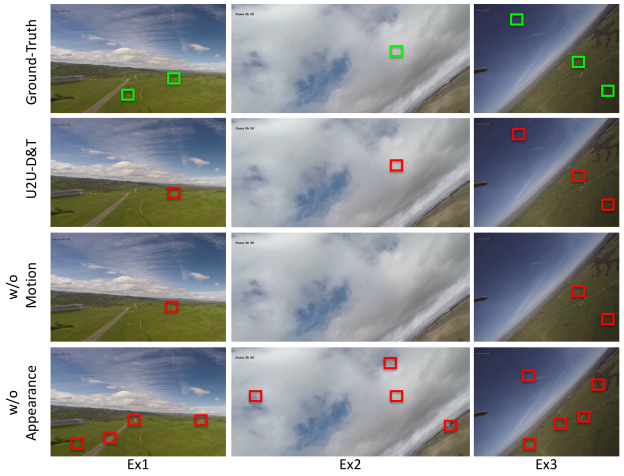


Fig. 10. Ground-Truth and detection results from U2U-D&T, w/o Motion, w/o Appearance. [*Top-Row*] Ground-truth annotations in Green boxes, [*Second-Row*] U2U-D&T detection results, [*Third-Row*] w/o Motion and [*Bottom-Row*] w/o Appearance. Three examples of detection results: when use appearance to 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.)

TABLE V
PRECISION, RECALL AND F-SCORE FOR ROBUSTNESS INVESTIGATION

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

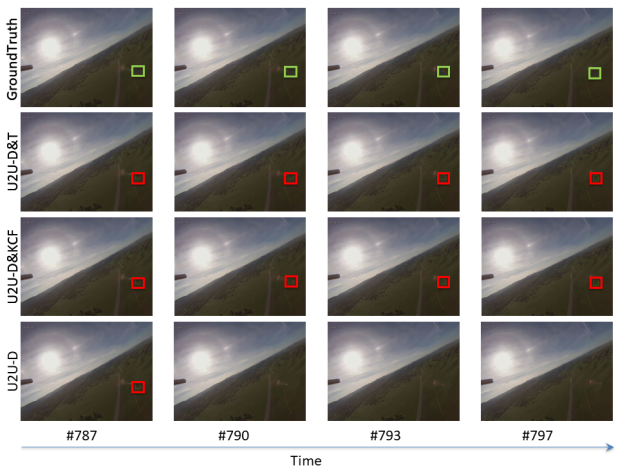


Fig. 11. Results of moving object detection and tracking algorithms sequential frames in a testing video: [*Top-Row*] Ground-truth annotation (Green), [*Second-Row*] U2U-D&T detection, [*Third-Row*] U2U-D&KCF detection, [*Bottom-Row*] w/o Kalman results. Horizontal line denotes time. Without Kalman tracking, there is intermitent missed detections while U2U-D&T recovers the missed detections. Kalman Tracking helps recover intermittent missed detection. Kalman Tracking performs very close to KCF Tracking [31]

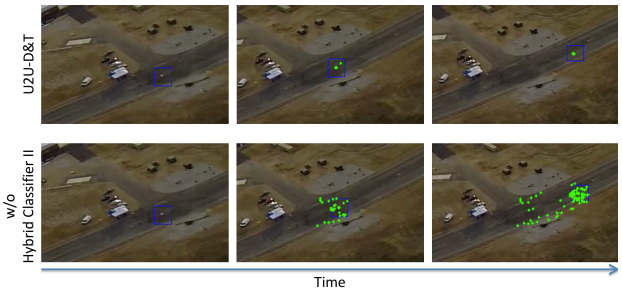


Fig. 12. Results of using Hybrid classifier II during flow point propagation. Blue boxes are the ground truth annotation, green dots are the tracked flow point. First row is the results having Hybrid classifier II to prune flow points and second is the result using optical flow matching directly to propagate the flow points.

run at every location, which is much more computationally intensive than sparse application of the classifiers used in the U2U-D&T algorithm.

B. Classification & Tracking Robustness

In this section, we analyze the importance of motion classification, appearance classification, Kalman Tracking, and tracked-point pruning to the accuracy and robustness of our U2U-D&T algorithm.

To do this, we compare our based line method with the following modified algorithms:

w/o Motion: U2U-D&T without motion classifier
w/o Appearance: U2U-D&T without appearance classifier
w/o Kalman: U2U-D&T without Kalman tracking
w/o Hybrid classifier II: U2U-D&T without Hybrid classifier II

Fig. 10 compares the results of the baseline U2U-D&T algorithm to the modified algorithms without motion classification and without appearance classifier for three representative example frames in the video. Table V lists the performance increases compared to motion based classifier.

w/o Motion*:* The third row of Fig. 10 compares the results of the baseline U2U-D&T algorithm to the modified algorithm without motion classification for three representative example frames in the video. Notice that this algorithm fails when the moving target is far away and only occupies a few pixels. Table V supports this conclusion by showing a significant reduction in recall when the motion classifier is removed. This is important since detecting distant targets can be critical in collision avoidance applications.

w/o Appearance*:* The fourth row of Fig. 10 compares the results of the baseline U2U-D&T algorithm to the modified algorithm without appearance classification for the same examples. Notice that the number of false alarms increases dramatically in this case perhaps due to the inaccuracy of Lucas-Kanade optical flow matching. Again, Table V supports this conclusion with a corresponding dramatic reduction in precision for the "w/o Appearance" algorithm. This demonstrates that purely motion-based classification is not sufficient to robustly detect moving targets while rejecting spurious noise.

w/o Kalman*:* Fig. 11 illustrates the effect of removing Kalman filtering on a representative sequence of frames, and Table V shows the corresponding effect on the accuracy for the full set of test videos. The reduction in accuracy "w/o Kalman" is not as dramatic as resulted from the removal of the appearance or motion classifiers, but it still is significant and primarily effects the recall. Typically, the errors occur in sequences of

TABLE VI
GPU COMPUTATION TIME

		Computation Time (ms)	Update Rate	Time per Frame (ms)
Target Detector	Local & Global Motion Estimation	6.50	6	1.08
	Background Subtraction	87.23	6	14.54
	Salient Point Extraction	48.76	6	8.13
	Hybrid Classifier I	20.2	6	3.37
Target Tracker	Flow Point Extraction	6.90	6	1.15
	Flow Point Propagation	0.15	1	0.15
	Hybrid Classifier II	0.41	1	0.41
	Kalman Tracking	0.12	1	0.12
Total Time per Frame				**28.95**

TABLE VII
ODROID COMPUTATION TIME

		Computation Time (ms)	Update Rate	Time per Frame (ms)
Target Detector	Local & Global Motion Estimation	286	6	47.67
	Background Subtraction	4627	6	771.17
	Salient Point Extraction	1491	6	248.50
	Hybrid Classifier I	/	/	/
Target Tracker	Flow Point Extraction	200	6	33.33
	Flow Point Propagation	60	1	60
	Hybrid Classifier II	/	/	/
	Kalman Tracking	4	1	4
Total Time per Frame				**1164.67**

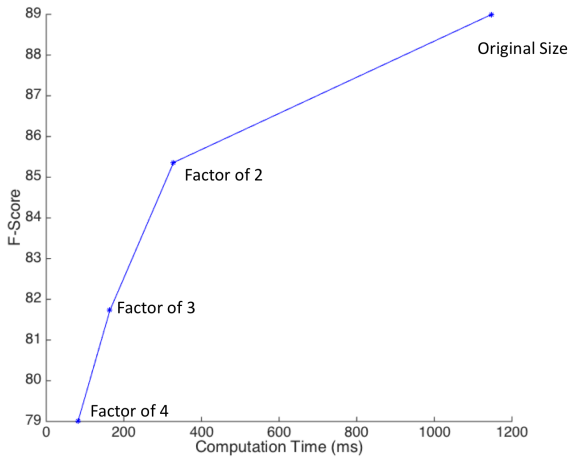


Fig. 13. Trade off between accuracy and computation time by downsampling the video.

missed detections when the target is obscured by clutter and the track is lost. Here we also compare against a more recent tracking algorithm, kernelized correlation filter (KCF) [31], but the results indicate that the simpler Kalman filtering based method can achieve the goal of recovering intermediate missed detections.

w/o Hybrid classifier II*:* Fig. 12 illustrates that if we propagate the flow points using optical flow matching only, then there are an excessive number of residual points in the background. This is caused by the inaccuracy of optical flow matching. To solve this, we add the Hybrid classifier II to prune the flow points, then the propagation is improved by deleting spurious noise from the background clutter.

C. Computation Time

Table VI breaks out the measured computation time for each component of the U2U-D&T algorithm on the GPU platform. Notice that the moving target proposer is the most time consuming component. In order to achieve real-time performance at 30 fps, we run this component every 6 frames, so that the time per frame is less than the required 30 ms.

We then analyze computation time for each component which boosts accuracy in our algorithm. Notice that the Hybrid Classifier takes about 12% of the total computation time. So the additional computation for both appearance and motion classifiers is not large. The Target Tracker requires less than 2ms per frame, which is also very time efficient given the increased accuracy it provides.

Table VII breaks out the computation time required to run on the Odroid board using the ARM processors. The Odroid board has very limited computational power and memory compared to the GPU and implementing the deep neural network is difficult since the associated software packages are not available for the ARM architecture on our OdroidXU4 board. Therefore, we implemented the algorithm without the neural network classifiers and listed the computation time for the remaining operations. The results show that, even without the neural network classifiers, the algorithm cannot run at real-time rates with the full resolution images using the Odroid board. However, Fig.13 shows that near real-time processing can be achieved by reducing the spatial resolution of the video; however, this is at the price of a decreased F-Score. It is worth mentioning that the computational complexity of the EPFL algorithm is much higher compared to our algorithm. So EPFL algorihtm is not feasible to be run on Odroid board.

V. CONCLUSION

In this paper, we present a framework for a low complexity U2U-D&T algorithm for autonomous see-and-avoid systems. The U2U-D&T algorithm is designed for robustly detecting and tracking target UAVs from cameras mounted on a flying UAV platform. Our U2U-D&T algorithm is based on two

phases: A Target Detector and a Target Tracker. The moving target detector accurately subtracts the background from subsequent frames by using a sparsely estimated global perspective transform. We find that the recall and precision can be greatly improved with approximately a 10% increase in computation through the inclusion of a Hybrid Classifier which uses Adaboost to combine both appearance and motion classifiers. Kalman tracking is shown to further improve accuracy with little increase in computation.

Experimental results using a publicly available ground-truthed video data set with multiple fixed wing UAVs shows that the U2U-D&T algorithm is both robust and effective. Importantly, the algorithm is efficient enough to be run in near real time on a low power arm board, albeit with reduced resolution video. In future work, the algorithm can be extended for efficient unmanned ground vehicles (UGA) detection and tracking.

ACKNOWLEDGMENTS

The authors acknowledge support from the Naval Post Graduate School (NPS). Thanks to Dr. Mathias Kolsch, Dr. Timothy Chung and Dr. Oleg Yakimenko for their assistance collecting the video data. Especially thanks Dr. Timothy Chung for all his suggestions and input to this research during his work at the NPS.

REFERENCES

[1] 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.

[2] G. Loianno, V. Spurny, J. Thomas, T. Baca, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho, S. Martin, and V. Kumar, "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.

[3] 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.

[4] 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.

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

[6] 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.

[7] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A system for autonomous flight using onboard computer vision," in *International Conference on Robotics and Automation*. IEEE, 2011.

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

[9] 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.

[10] 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.

[11] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[12] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[13] 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.

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

[15] N. Seungjong and J. Moongu, "A new framework for background subtraction using multiple cues," in *Asian Conference on Computer Vision*, 2013.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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*, 2016, pp. 1255–1262.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] A. Rozantsev, V. Lepetit and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 879–892, 2017.

[29] H. Saribas, B. Uzun, B. Benligiray, O. Eker, and H. Cevikalp, "A hybrid method for tracking of objects by uavs," in *Proceedings of Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2019.

[30] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[31] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[32] 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.

[33] 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.

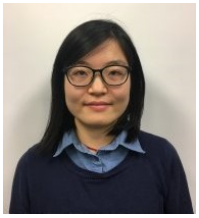
[34] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: object detection and tracking," *arXiv preprint arXiv:1804.00518*, 2018.

[35] 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.

[36] 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 *International Conference on Intelligent Robots and Systems*. IEEE, 2016.

[37] 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.

[38] J. Shi and C. Tomasi, "Good features to track," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 1994.
[39] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981.
[40] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.
[41] H. Goldstein, "Multilevel mixed linear model analysis using iterative generalized least squares," *Biometrika*, vol. 73, no. 1, pp. 43–56, 1986.
[42] 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.
[43] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.
[44] D. M. Magerman, "Statistical decision-tree models for parsing," in *Proceedings of annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1995, pp. 276–283.
[45] P. Tissainayagam and D. Suter, "Object tracking in image sequences using point features," *Pattern Recognition*, vol. 38, no. 1, pp. 105–113, 2005.
[46] 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.
[47] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *British Machine Vision Conference*, 2009.
[48] M. Teutsch and M. Grinberg, "Robust detection of moving vehicles in wide area motion imagery," in *Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2016, pp. 27–35.



Jing Li received a B.A. degree in 2010 and a MS degree in 2013 from Xi'an Jiaotong University. In 2019 she received a Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, IN, USA, in 2019. She joined Texas Instruments where she is currently a Researcher with the Perception and Analytic Lab in Texas Instruments, TX, USA. Jing's research interests include Computer Vision, Deep Learning and Image Processing.

Dong Hye Ye received his PhD in bioengineering from the University of Pennsylvania in 2013. He also received Bachelors degree from Seoul National University in 2007 and Master's degree from Georgia Institute of Technology in 2008. His graduate research focused on medical image analysis such as image registration, segmentation, and classification using machine learning. He currently works in Electrical and Computer Engineering at Purdue University as a Research Assistant Professor. His research interests include sparse sampling for microscopic imaging, model based iterative tomographic reconstruction, and UAV sensing via machine learning.

Mathias Kolsch is an assistant professor of computer science at the Naval Postgraduate School in Monterey, California. He has a PhD in computer science from the University of California, Santa Barbara. His research interests include computer vision, human-computer interaction, computer graphics, and AR.

Juan P. Wachs is a University Scholar, Full Professor in the Industrial Engineering School at Purdue University, Professor of Biomedical Engineering (by courtesy) and an Adjunct Associate Professor of Surgery at IU School of Medicine. He is the director of the Intelligent Systems and Assistive Technologies (ISAT) Lab at Purdue, and he is affiliated with the Regenstrief Center for Healthcare Engineering. Dr. Wachs received his B.Ed.Tech in Electrical Education in ORT Academic College, at the Hebrew University of Jerusalem campus. His M.Sc and Ph.D in Industrial Engineering and Management from the Ben-Gurion University of the Negev, Israel.

Charles A. Bouman received a B.S.E.E. degree from the University of Pennsylvania in 1981 and a MS degree from the University of California at Berkeley in 1982. From 1982 to 1985, he was a full staff member at MIT Lincoln Laboratory and in 1989 he received a Ph.D. in electrical engineering from Princeton University. He joined the faculty of Purdue University in 1989 where he is currently the Showalter Professor of Electrical and Computer Engineering and Biomedical Engineering.

Professor Boumans research is in statistical signal and image processing in applications ranging from medical to scientific and consumer imaging. Professor Bouman is a member of the National Academy of Inventors, a Fellow of the IEEE, a Fellow of the American Institute for Medical and Biological Engineering (AIMBE), a Fellow of the society for Imaging Science and Technology (IS&T), a Fellow of the SPIE professional society.