IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 2, 2018, accepted August 4, 2018, date of publication August 10, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2864672

Multi-Object Tracking by Flying Cameras Based on a Forward-Backward Interaction

VINCENZO CARLETTI, ANTONIO GRECO, ALESSIA SAGGESE, AND MARIO VENTO
Department of Information Engineering, Electrical Engineering and Applied Mathematics, University of Salerno, 84084 Fisciano, Italy

Corresponding author: Alessia Saggese (asaggese@unisa.it)

This work was supported by A.I. Tech srl.

ABSTRACT The automatic analysis of images acquired by cameras mounted on board of drones (flying cameras) is attracting many scientists working in the field of computer vision; the interest is related to the increasing need of algorithms able to understand the scenes acquired by flying cameras, by detecting the moving objects, calculating their trajectories, and finally understanding their activities. The problem is made challenging by the fact that, in the most general case, the drone flies without any awareness of the environment; thus, no initial set-up configuration based on the appearance of the area of interest can be used for simplifying the task, as it generally happens when working with fixed cameras. Moreover, the apparent movements of the objects in the images are superimposed to that generated by the camera, associated with the flight of the drone (varying in the altitude, speed, and the angles of yaw and pitch). Finally, it has to be considered that the algorithm should involve simple visual computational models as the drone can only host embedded computers having limited computing resources. This paper proposes a detection and tracking algorithm based on a novel paradigm suitably combining a forward tracking based on local data association with a backward chain, aimed at automatically tuning the operating parameters frame by frame, so as to be totally independent on the visual appearance of the flying area. This also definitively drops any time-consuming manual configuration procedure by a human operator. Although the method is self-configured and requires low-computational resources, its accuracy on a wide data set of real videos demonstrates its applicability in real contexts, even running over embedded platforms. Experimental results are given on a set of 53 videos and more than 60 000 frames.

INDEX TERMS Multi-object tracking, drones, flying cameras, test and set, feedforward tracking.

I. INTRODUCTION

Understanding what is currently done by the objects moving inside a scene can be reconducted to the analysis of their trajectories over time; in fact, trajectories include spatial and temporal information and, once suitably analized with respect to the structure of the environment, it is possible to infer behavioral aspects. For example, if a car is moving in a lane at a speed of 80 mph, we think at a normal situation; viceversa, if the speed is exactly the same but the lane is the one used in case of emergencies, and even more in the wrong direction, we are in presence of a clear alarming situation. Trajectories' analysis of moving objects is currently the paradigm used in many systems for detecting anomalous events [1]–[4].

For several years, the efforts of the scientific community have been mainly addressed at algorithms working with fixed mounted camera [5]–[11]; anyway, the recent availability of low cost drones used for video-surveillance purposes is launching the challenge of addressing the same problem but in the much more complex case of flying cameras. Such new trend adds a great complexity at the problem due to the vibrations and sudden changes of the speed and orientation of the drone during its flight, with consequent huge impacts on the stability and quality of the obtained images.

As a matter of facts, the plenty of algorithms based on background subtraction, currently widely used with fixed cameras, are no more usable; they are in fact based on the hypothesis of a still background (as the camera does not move) and hence on the consideration that, if something appears as moving, it really moves. Viceversa, when dealing with moving cameras, we have two different movements to be taken into account: the one of the objects moving in the scene and the other one, the so called *ego-motion*, generated by the apparent movement of scene due to the fact that the camera itself is moving.

Indeed, the problem of tracking moving objects in videos took by moving cameras has been already faced in the literature, but effective solutions are given only in the special case of tracking a single moving object at a time [12]–[14]: in particular, they are based on the assumption that, in the first frame of the stream, the object of interest is manually indicated by a human operator, so making the overall system not autonomous. Moreover, no any other object can be tracked until the human operator explicitly passes to the identification of a new object to be tracked. It can be noted that these algorithms can be used on systems mounted on drones, but, even after the initial manual indication of the object to be tracked, do not allow to track simultaneously the multitude of moving objects in the scene. In practice, their use is limited to implement the so called *follow me* drone, where the latter is able to chase a target, explicitly selected at a starting time.

Multiple objects tracking aims at a much more general task, i.e. the detection of all the moving objects present in the scene and at the extraction of their trajectories, possibly without using any a priori knowledge about them. This problem requires the two phases of *detection* and *tracking*; the order in which the methods generally proceed is intuitive: the *tracking-by-detection* paradigm works by preliminarily detecting in each frame all the moving objects; then the latter are tracked, so as to obtain their trajectories. The main goal pursued by this class of methods is the so called *drone explorer*: the scene is analysed in terms of the extracted trajectories of the moving objects and these are used to discover potentially abnormal behaviors [15].

It is worth noting that single vs multiple objects tracking are often confused under the common name of *tracking*, but the working assumptions, as said, are significantly different. In the last years some contributions at the multiple object tracking problem have been proposed. Although a strict classification is not trivial, two main strategies for solving this problem emerge: (i) the use of suitably extracted salient points by the sequence of frames, processed so as to separate the ones belonging to moving objects from the ones belonging to the background (the latter appearing as moving as a consequence of the camera motion) and (ii) the generation of a foreground mask obtained by separating background from moving objects pixel by pixel.

The first typology of algorithms is based on the introduction and use of the so called particle trajectories extracted by the optical flow [16] [17] [18] [19]; this approach reveals to be significantly robust as for the characterisation of the movement of the objects (foreground) with respect to the ego motion (background). For instance, in [19], optical flow and particle advection are combined by evaluating multiple frames so as to compute the particle trajectories. Epipolar constraints are thus applied over the particle trajectories so as to isolate those trajectories violating the constraints since corresponding to moving objects. Although the experimentations conducted by the authors highlights that the above methods are very accurate, the main issue lies in the poor quality of the obtained boundaries between the moving object (foreground) and background; indeed, the results are particularly critical especially when the size of the moving objects is very small, since the number of salient points could be not sufficient enough to properly represent the moving object.

The second typology of approaches is based on the extraction of the foreground, starting from the assumption that all the pixels of a moving object have approximately the same value of the motion vector as well as those belonging to the background. Operatively, these methods use clustering criteria for grouping pixels having similar motion, so separating moving objects from the background. For instance, in [20] a *Structure from motion* (SFM) methodology is applied to hand-held cameras and allows to estimate camera parameters, sparse 3D points and depth map. It is important to note that this strategy generates accurate boundaries of the moving objects, but under the assumption that the image is characterised by a large depth difference between foreground and background. In practice, such methods are particularly suited for those cases in which the drones fly at a low altitude, but not otherwise.

In [21] an adaptive background model is built by modelling the camera motion by affine transformation. The main limitation is due to the fact that the model is not general enough for estimating any scene geometry, which is an important and not negligible feature when dealing with flying cameras. More recently, homography has been successfully used in order to estimate the motion of the camera: a frame is transformed in the previous one and the differences between them are considered [22]–[24]. Although being in general very accurate, the moving objects detected were not compact enough in all the environments.

In order to increase the effectiveness of the algorithms, in some cases the two typologies of approaches are combined so as to achieve higher accuracy [25]–[28]. In [28], for instance, keypoints tracking based on pyramidal Lucas-Kanade is combined with foreground segmentation based on local motion history. The temporal interval between two consecutive frames (the so called detection interval) is not fixed a priori, but instead is dynamically adjusted by considering some parameters related to the movement of the drone (its speed and its flying altitude), as well as some parameters related to the algorithm (the computational complexity of the algorithm and the frame rate of acquisition of the video). Thanks to this approach, the algorithm is stable enough to deal with both slow and fast motion of the moving objects. However, except the detection interval, all the configuration parameters are fixed a priori. This property, commonly adopted in the tracking literature, is somehow inherited from the tracking algorithms designed for working with cameras mounted on board of ground platforms, such as robots or vehicles [5], [29]–[33], where the scene is acquired by a frontal view camera, producing a more stable image if compared with top view camera mounted on board of the drones [3], [28], [34], [35].

However, when dealing with flying cameras the configuration parameters can not be fixed a priori, since they need

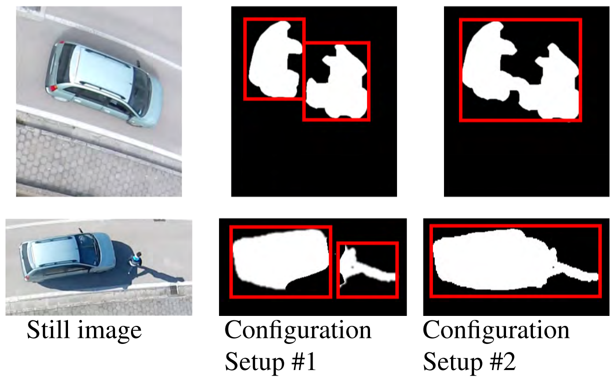

FIGURE 1. **An example of the effects of different configuration parameters (identified as Setup #1 and #2) on the foreground mask extraction and on the detection.**

to be dynamically adjusted during the flight, depending on the movement of the drone (both in terms of altitude and speed) and on the environment where the drone is moving, which determines the size and the typology of the objects to be tracked, as well as the noise to be filtered out. Indeed, a drone is not constrained to move on the road, but instead can fly over a street, over some buildings or over a wood, and with different flying altitudes, thus implying that the properties of the objects may dynamically varies, as well as the ones of the noise. Furthermore, the objects taken by the camera are not a priori known, thus the algorithm needs to be general enough to deal with different and unknown typologies of objects.

As a consequence, the setup of the operating parameters is a very difficult and time consuming task for a human operator: even in the same flight and in the same environment, a configuration setup can be good for a couple of seconds but not for the successive ones. It is important to highlight that a wrong configuration setup of the parameters may be among the main causes of the errors during the detection step, and then during the tracking step as well. An example is shown in Figure 1, where two situations (one per row) are reported: the first column shows the images processed; in the first image there is a vehicle, while in the second image the same vehicle is close to a person. Second and third columns represent the foreground mask obtained by using two different configuration setups (identified in the figure as Configuration Setup #1 and #2, respectively). As we can note, the same setup (for instance, in terms of the radius values for erosion and dilation or other morphological operators, as well as on minimum and maximum size for the filtering) is not good for processing both the frames, even if acquired in the same environment and within a couple of seconds between each other.

In general, the following errors may arise: *false negatives*, corresponding to undetected objects; *false positives*, corresponding to spurious objects, namely something in the background which is wrongly considered as an object moving in the scene; *splits*, corresponding to an object erroneously divided into multiple parts (see an example in Figure 1, where a car is partitioned into two parts in the first row, second image); *merges*, corresponding to multiple object wrongly joined in a single part (see an example in Figure 1, where the car and the person, close each other, are merged in a single box in the second row, third image). Concerning the merge errors, it is also important to note that one of the typical causes is related to the occlusions, namely to some objects which are partially or totally hidden behind other objects (or behind background objects). Considering that we will focus on cameras mounted on board of a drone, this situation should not happen, thus the only reason of the merge is related to the fact that two objects are very close each other and are not well distinguished during the detection step, due to a wrong configuration setup.

Let us remember that the aim of the tracking module is to perform the best possible *association* between the output of the detection at the current frame and the output of the tracking at the previous frame, so as to draw (even in case of errors of the detection step) the trajectory of the object frame by frame and to solve the above mentioned errors of the detection step. In the literature, two main strategies for performing such association can be identified, namely *local* and *global* association. Most of the methods perform the association locally, using a tracking-by-detection approach: given the position of an object in the scene at the current frame, a tracking algorithm properly evaluates the results of the detection step (at the current frame) and the ones of the tracking algorithm (at the previous frame) [28], [36], [37] in order to update the position of an object frame by frame. Anyway, an error during the detection step can not be discovered since the detection, for its nature, only evaluates spatial information and not temporal one, that could help in the configuration parameters adjustment and thus in the results improvement.

Viceversa, for those algorithms based on *global data association*, the association is not performed locally, namely frame by frame, but instead all the objects detected in all the frames are taken into account and the association problem is formulated as an optimization problem [38], [39]. The advantage is evident: a large amount of information is available for taking the decision, thus the results are more accurate. Anyway, the main limitation deriving from these approaches is that they can not work in real time but only off-line, in the sense that the whole sequence of images needs to be collected and then processed to generate the objects trajectories.

A somehow hybrid solution has been exploited in those methods based on the so called D&T (Detection and Tracking) [40]; the main idea is to jointly perform detection and tracking by using a convolutional network fed with multiple frames (instead than a single one, typically employed in objects detectors): indeed, the idea is to compute the convolutional cross-correlation between the features responses of adjacent frames, used to feed both a RoI-pooling layer (which evaluates the spatial information) and a RoI-tracking layer (which evaluates the temporal information). Thus, the RoI-tracking layer combines such responses with the ones of the RoI-pooling layer so as to extract the tracks of the considered frames. The set of the tracks are finally linked to extract the whole trajectories.

A strict cooperation between detection and tracking module has been also explored in [41], where a cooperative detection and tracking (CDT) algorithm has been designed: the main idea is that the detector is configured to have a low sensitivity, and then a very low number of false positives, but also a potentially high number of misses. In order to recover such misses, a backward tracking module is added with respect to the traditional forward one. In practice, a detection-by-tracking approach (given by the forward module) is combined with a tracking-by-detection one (added by the backward module), where this last module allows to solve most of the errors introduced during the detection step. The main issue deriving from this algorithm is that it is able to track only a single object per time and can work on-line only if the backward module is suppressed.

In conclusion, it is evident that including temporal information in the detection step (indirectly, as it happens in global data association, or directly as in D&T and in CDT methods), allows to make the system more robust and then reliable since being based on a larger amount of information. Anyway, this is typically paid back in terms of impossibility to work in real time. Starting from these considerations, we propose a novel detection and tracking algorithm based on a Forward-Backward Interaction (FBI) between the detection and tracking modules: indeed, detection and tracking are not performed one shot (only with a forward chain, as in most of the algorithms available in the literature) but instead the operating parameters are dynamically and automatically adjusted and optimized by means of a backward chain. In more details, the proposed approach acts in two steps:

- **Forward Chain**: (i) given a frame, the detection is performed by combining salient points and foreground mask based methodologies; (ii) a tracking based on local association is performed; (iii) in case the achieved result is reliable enough, then the next frame can be processed. Otherwise, the backward chain is also put in the loop.
- **Backward Chain**: the confidence about data association is evaluated in order to automatically adjust the operating parameters. Then, the forward chain is performed again with the new configuration setup.

It is important to highlight that the on-line and automatic updating of the operating parameters faces the main issues typically arising in real environments: the algorithm is robust enough with respect to the high variability of the environment and of the objects populating it, since the configuration setup is adjusted online and automatically. The typical errors of the detection step are identified and solved by means of the backward chain. Furthermore, a heavy fine tuning of the human operator is not required, since his task becomes to choose the parameters at the startup, thus making the system particularly suited for being used in real applications. Finally, even being inspired by global data association and hybrid association frameworks, differently from state of the art methodologies our FBI based tracking algorithm is able to run in real time since it takes the decision about association only by evaluating the current frame (and the history of the objects), without requiring the whole set of objects appearing in a sequence of frames.

More generally, another important and not negligible aspect that has been taken into account pertains the efficiency. Indeed, the proposed algorithm has been devised so as to be used over low-cost embedded devices, small and low-energy enough to be mounted on board of a drone, without any external server for the processing.

Finally, a new dataset has been proposed and made publicly available for benchmarking purposes, provided with both videos and ground truth for multiple object tracking. Considering that most of the available datasets focuses on single object tracking (and thus provide the ground truth of a single target object), this can be considered another important contribution of this work. In order to prove the effectiveness of the proposed approach, 53 different videos, belonging to four different datasets publicly available and acquired in different scenarios have been used.

The paper is organised as follows: in Section II the proposed method is detailed; Section III describes the datasets used for the experimentation and the obtained results. Finally, conclusions are drawn in Section IV.

II. THE PROPOSED METHOD

The architecture of the proposed method is shown in Figure 2: two different processing chains can be identified, namely a forward chain and a backward chain.

In more details, the forward chain (in the upper part of the figure) combines a detection stage (in orange) with a tracking based on local data association (in blue). The detection module aims at identifying, for each frame, the objects moving in the scene at the current frame, the so called *blobs* (see Section II-A). In order to separate the ego motion from the movement of the objects, we exploit preliminary camera compensation algorithm, aiming at estimating the direction and the magnitude of the camera movement between two consecutive frames (details in Subsection II-A.1). Thus, the two approaches based respectively on the extraction of the foreground mask and on the extraction of some salient points are combined (details in Subsections II-A.2 and II-A.3). This approach, sometimes adopted in the literature as mentioned before, has been proved to increase the overall accuracy of the detection step.

The tracking is based on a local data association, which performs the best possible association between the set of blobs (output of the detection at the current frame) and the set of objects tracked until the previous frame (output of the tracking at the previous frame). Algorithms available in the literature typically differ each other for the way in which such association is performed. Anyway, differently from state of the art approaches, the proposed method combines this *forward tracking* which a *backward tracking*, which is activated only in those cases in which the local data association

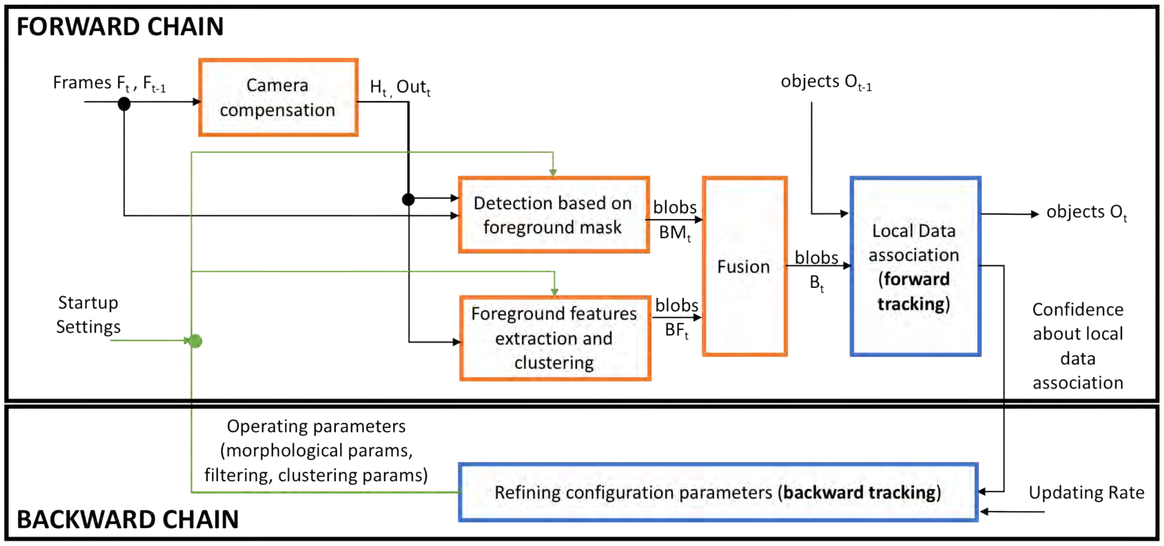
FORWARD CHAIN
Frames F_t, F_{t-1}
Camera compensation
H_t, Out_t
objects O_{t-1}
Detection based on foreground mask
blobs BM_t
Fusion
blobs B_t
Local Data association (forward tracking)
objects O_t
Startup Settings
Foreground features extraction and clustering
blobs BF_t
Confidence about local data association
Operating parameters (morphological params, filtering, clustering params)
Refining configuration parameters (backward tracking)
Updating Rate
BACKWARD CHAIN

FIGURE 2. Overview of the proposed approach: forward and backward chains are in top and in the bottom part of the image, respectively. Detection modules have an orange border, while tracking ones have a blue border.

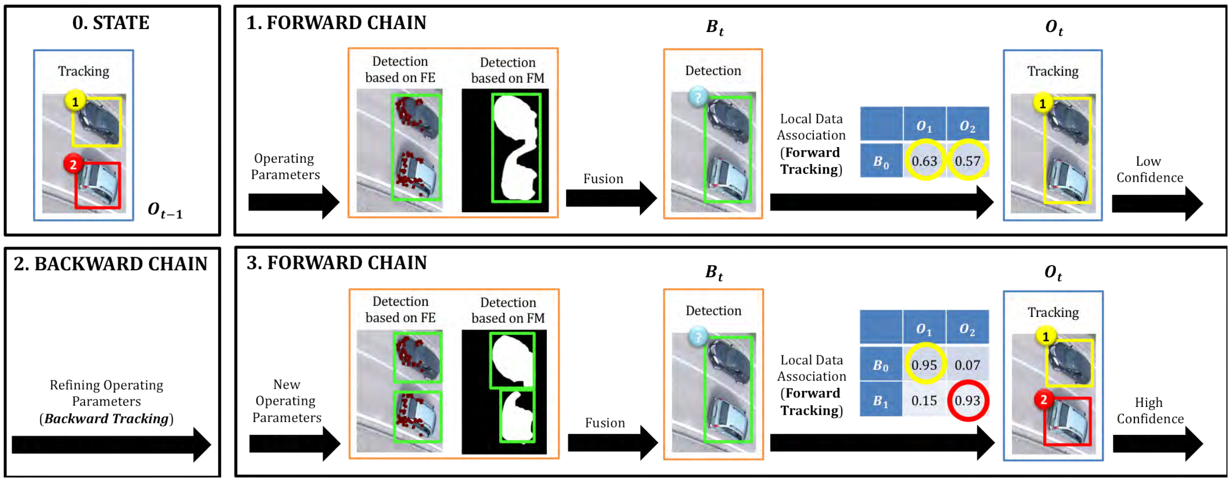

FIGURE 3. An example of the interaction between forward and backward chain: the forward chain is repeated until the local association between blobs and objects reaches a sufficiently high confidence.

is not reliable enough. Indeed, in such a situation, a backward chain (in the lower part in Figure 2) is also put in the loop, aiming at refining the operating parameters to be used for repeating (with these new parameters) the detection at the current frame. In other words, the forward chain is performed starting from a startup settings and is iteratively repeated with refined settings (automatically generated by the backward chain) until the confidence in the local data association is high enough. More details about the tracking module will be presented in Section II-B.

An example showing an interaction between forward and backward chain is shown in Figure 3: starting from the current state, namely from the objects tracked until the previous frame $t-1$ (see box *0. State*), the forward chain is activated with the startup settings. The confidence in the local data association is low (see box *1. Forward Chain*), thus, the backward chain is activated so as to refine configuration parameters, with the aim to improve the detection step and thus increasing the reliability in the local data association (see box *2. Backward Chain*). Given the new set of parameters, the forward chain is activated again and again until the confidence is not sufficient enough (see box *3. Forward Chain*). The main advantage deriving from the introduction of the backward chain is that the tracking module does not need to perform its choice *blindly*, but instead is guided by the detection, which in turns is not *blindly* but instead is guided by the tracking itself. In other words, the introduction of the backward chain allows to merge in our approach a tracking-by-detection with a detection-by-tracking.

A. DETECTION

The detection module can be decomposed in the following four components: 1) *camera compensation*, 2) *detection based on foreground mask*, 3) *foreground features extraction* *and clustering* and 4) *fusion*. More details about each of the components are reported in the following subsections.

1) CAMERA COMPENSATION

The aim of this step is to estimate the movement of the camera in two consecutive frames, by only analysing the two images, namely the current frame and the previous one. Indeed, considering that the movement of the drone can not be predicted, due to the presence of the wind or of other weather factors that can move it independently on its flight planning, we exclude the possibility to exploit any information provided by the drone itself. Furthermore, this choice allows us to be independent on the particular drone to be used for allowing the movement of the camera, as well as on the scenario (outdoor or indoor, where GPS signal works in different way depending on the scenario itself).

Given the two consecutive images, camera compensation module computes the transformation matrix H_t, which maps the frame $t-1$ on the frame t. In the literature there are two main strategies for performing camera compensation: affine transformation and perspective transformation. Affine transformation estimates the displacement of each pixel across successive frames. It has two main drawbacks: first, it is not very general since it only allows to map a rectangle to a parallelogram, thus it is not able to take into account all the movements performed by a drone; furthermore it is very expensive from a computational point of view, since working at pixel level. Thus, we decided to adopt a perspective transformation based on the homography, which is more general since it allows to map a rectangle to any trapezoid. Moreover, the mapping is found in terms of features point instead than pixels, thus it is definitively faster.

In more details, a set of feature points P_t is extracted at the current frame by using *Good Features to Track* based on Shi-Tomasi corner detector [42]. It is important to highlight that the proposed method is general enough to deal with any features points. Anyway, we select the above mentioned typology since it is invariant to rotation and translation and it has been proved to provide a very reliable motion estimation. Before computing the feature points, a gaussian filter is applied on the image. In this way, we avoid to associate feature points to the noise instead than to interest points.

The matching between the feature points detected in two consecutive frames (namely P_{t-1} and P_t) is computed by Lucas-Kanade algorithm [43]: given the two sets of feature points, Lucas-Kanade computes the optical flow vectors of such points; these vectors can belong to two sets of feature points, namely the ones that could be *inliers* and the ones that could be *outliers*. The inliers In_t are the points associated to the background (and then deriving from the ego motion), whose vectors describe the movement of the camera between two consecutive frames. The outliers Out_t represent the movement of the objects; as evident, the outliers vectors should represent a different movement, both in terms of direction and intensity, if compared with inliers.

As evident, the homography should be computed by only considering inliers feature points, so excluding from the analysis the outliers. For achieving this aim, a common approach in the scientific literature is to use the RANdom SAmple Consensus (RANSAC) algorithm (see for instance [28]), able to produce a reliable estimation of the model parameters of the image transformation even in presence of a high number of outlier salient points. RANSAC works in the following two steps that are iteratively repeated: (1) a set of vectors is randomly chosen and the best fitting model is computed by evaluating the minimum squared error; (2) the number of vectors fitting this model (which are considered inliers for this particular iteration) is counted. The set of such vectors is called consensus set. The fitting model corresponding to the highest cardinality of the consensus set is finally selected.

The proposed approach uses a slightly different strategy, based on the Least Median of Squares (LMedS) [44]: the main difference between RANSAC and LMedS is that the latter evaluates each solution in terms of the median Symmetric Epipolar Distances of the data set and chooses the one which minimises the median error e_{med}, while RANSAC aims at maximizing the number of inliers. Once chosen the set, we evaluate the error e for each optical flow matching and we compare it with the median error e_{med}: the points $p_{i,t} \in P_t$ having an error $e_{i,t}$ higher than the median error e_{med} are considered outliers, while the remaining ones are considered as inliers:

$$p_{i,t} \in \begin{cases} Out_t & if\ e_{i,t} \geq e_{med} \\ In_t & otherwise \end{cases} \tag{1}$$

LMedS has been proved to be a better choice than RANSAC in case the number of outliers is lower than 50%. This assumption holds, since in our case most of the motion is related on the ego-motion to be compensated, while the movement of the objects moving in the scene cover only a small portion of the image (surely lower than 50%).

2) DETECTION USING FOREGROUND MASK

An overview of the proposed approach is shown in Figure 4: as already mentioned, two different approaches are fused: the detection based on the foreground mask (in the upper part of the figure) and the detection based on the features points (in the lower part).

As for the first approach, we perform a frame difference between the previous frame F_{t-1} and the current frame F_t, which is transformed in the coordinate system of the previous frame; in more details, the homography H_t is applied to the current frame:

$$\overline{F_t} = H_t(F_t) \tag{2}$$

and the foreground mask FM_t is computed by a frame difference, according to a given threshold M; the generic pixel (x, y) of the foreground mask is computed as follows:

$$FM_t(x,y) = \begin{cases} 1 & if\ \ \overline{F_t}(x,y) - F_{t-1}(x,y) \geq M \\ 0 & otherwise \end{cases} \tag{3}$$

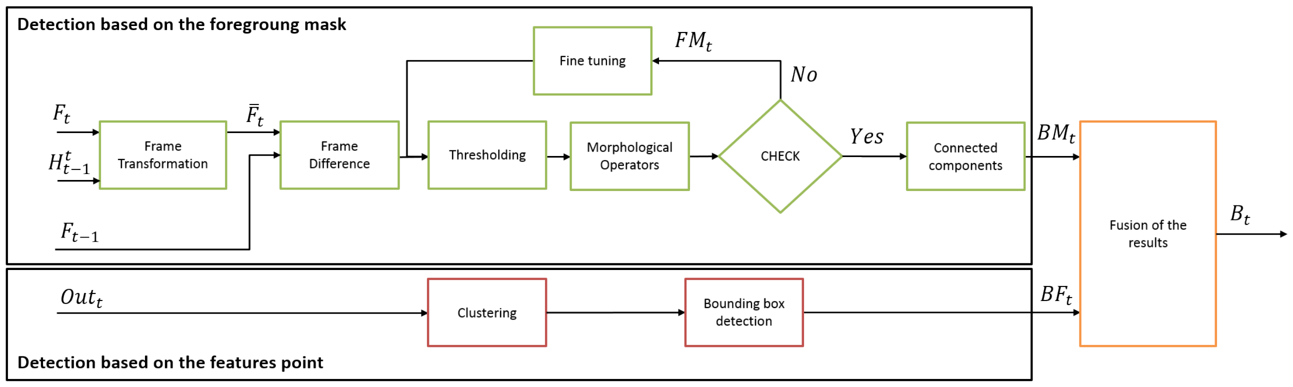

FIGURE 4. Moving objects detection algorithm.

Morphological operators are then applied to the foreground mask. The erosion is used to remove the noise, while the dilation is applied to merge blobs potentially belonging to the same moving object. The operating parameters (namely, the threshold value M and the kernel's size of the morphological operator K) are not fixed a priori, but instead are dynamically and automatically chosen. Indeed, given a starting configuration settings chosen by the human operator at the startup of the system, M and K are updated as follows: in case the backward chain has been activated at the current frame at least once, the new configuration setup is imposed by the backward chain itself. It implies that both the checking condition and the fine tuning modules are not activated. Viceversa, in case the backward chain has not yet been activated at the current frame t, the parameters are updated by a Test and Set procedure, thus including in the processing chain both the checking condition and the fine tuning modules. In more details, the number of moving pixels of the foreground mask (s_t) is computed and is compared with the average value of the last N frames ($\bar{s}_{t-1}$). In case of a small variation (lower than ϵ), the operating parameters are no more updated and the connected components are computed. Otherwise, a fine tuning is applied: if $s_t > \bar{s}_{t-1}$, M increases while K decreases; viceversa, if $s_t < \bar{s}_{t-1}$, M decreases while K increases. The pseudocode of this procedure is shown in Algorithm 1.

Note that Test and Set procedure exploits spatial information related to the whole image; thus, a modification of the parameters during the fine tuning produces the same effect on the whole image. Viceversa, the refining of the operating parameters during the backward tracking acts on a small portion of the image, namely the one where the object lies. Thus, the former is globally applied on the image, while the latter is locally applied, being specialised for a single object.

The configuration setup chosen before going on with the successive frame is used as startup settings for the successive frame. According to our knowledge, there are not any other methods available in the literature able to automatically adjust all the operating parameters. As already mentioned, one of the main advantages deriving from this choice is related to

Algorithm 1 Detection based on the foreground mask: Test and Set procedure.

```
1:  procedure TestAndSetDetection($\bar{F}_t$, $F_{t-1}$, $K$, $M$, backwardOn, $B$)
2:      checkCondition ← false
3:      n ← 0
4:      $FM_t^{original}$ ← FrameDifference($\bar{F}_t$, $F_{t-1}$)
5:      repeat
6:          $FM_t$ ← Thresholding ($FM_t^{original}$, $M$)
7:          $FM_t$ ← MorphologicalOperator ($FM_t$, $K$)
8:                                          ▷ checking condition
9:          $S_t$ ← numberOfMovingPixels($FM_t$)
10:         if $\bar{s}_{t-1} - s_t < \epsilon$ then
11:             checkCondition ← true
12:         else
13:             K,M ← fineTuning($\bar{s}_{t-1}$, $s_t$, $K$, $M$)
14:         update($\bar{s}_{t-1}$, $s_t$)
15:         update(best $K$, $M$)
16:         n ← n+1
17:     until backwardOn = false & checkCondition = false & n < B
```

the fact that a heavy fine tuning from the human operator is no more required, and the proposed approach is able to adapt itself also to different environments. The procedure is repeated until $\bar{s}_{t-1} - s_t < \epsilon$, or, in order to assure the convergence of the algorithm, after a certain number B of attempts. In the latter case, the parameters which assure the minimum difference are used, since they should provide the best possible foreground mask.

Given the foreground mask, the connected components are computed and the corresponding minimum bounding boxes, BM_t, are detected.

3) DETECTION USING FOREGROUND FEATURES

The detection based on the foreground features exploits the intermediate results of the preliminary camera compensation algorithm. In particular, it analyzes only the outlier points found at the current frame, Out_t, and clusterize them, so that each cluster corresponds to a moving object. In particular, we decide to use a density based clustering algorithm. This choice is due to the fact that the number of clusters is not a priori known, since the number of objects is not a priori known; furthermore, in our case the (outlier) points may belong to two different sets, namely (1) the points belonging to moving objects (hereinafter referred to as *foreground points*) and (2) the noise. Thus, some points (corresponding to the noise) have to be excluded. Such constraints prevent the use of both partitional and hierarchical clustering. Viceversa, density based algorithms are able to identify the clusters even when them are irregular or intertwined (as it commonly happens when dealing with non rigid objects such as persons), and when noise points to be excluded are present. The algorithm we decide to use is the well known DBSCAN (Density-Based Spatial Clustering and Application with Noise) algorithm; the basic idea is that for each point of a cluster, the neighborhood of a given radius (eps) has to contain at least a minimum number of points ($MinPts$). The only two configuration parameters thus are $MinPts$ and eps, where $MinPts$ is the minimum number of neighbors within eps radius.

Although being very effective and efficient, its main limitation is due to the fact that the quality of the results strongly depends on the typology of clusters; indeed, it works well only in those cases in which there is not a large difference in the size of the clusters. This is due to the fact that the size of the cluster is univocally determined by the combination $\{MinPts, eps\}$ and can not be properly chosen in a different way for each cluster.

As evident, this assumption does not hold in our context, where the typology of objects populating the scene is not a priori known and it may happen that the same typology of objects (having the same dimension) populates the scene (for instance, some vehicles on the road); but it may also happen that different typologies of objects populate the scene (for instance, persons and vehicles), thus the size of the clusters can not be fixed for all the objects.

This limitation is addressed by the backward chain. Indeed, in those cases in which the local data association is not reliable enough, the clustering parameters are dynamically updated in order to deal with different sized objects.

Finally, the minimum bounding box associated to each cluster is computed so as to produce the list of the bounding boxes BF_t.

4) FUSION OF THE RESULTS

The sets BM_t and BF_t resulting from the detection based respectively on the foreground mask and on the feature points are finally fused. In particular, the i-th box $B_t(i)$ belongs to the final set B_t if it has been detected by both the algorithms:

$$B_t(i) = \begin{cases} BM_t(i) \cup BF_t(i) & \text{if } BM_t(i) \cap BF_t(i) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

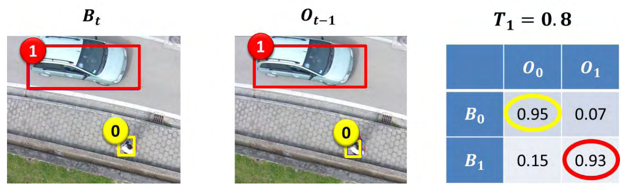

FIGURE 5. An example of 1 : 1 association. B^0 is associated to the object O^0, while B^1 is associated to the object O^1.

The main advantage deriving from this choice lies in the fact that only a reduced number of false positives is detected. On the other hand, the size of the detected object may be overestimated due to the fact that we consider the union of the boxes. Anyway, in case the local data association is not reliable enough, the algorithm activate the backward chain so as to improve the reliability in the association.

B. TRACKING ALGORITHM

The aim of the tracking algorithm is to draw, frame by frame, the trajectories of the objects moving in the scene by performing the best possible association between the set of boxes $B^t = \{B_t^1, ..., B_t^{|B|}\}$ (output of the detection at the current frame) and the set of objects $O_{t-1} = \{O_{t-1}^1, ..., O_{t-1}^{|O|}\}$ tracked until the previous frame $t-1$. For the sake of readability, in the following we will omit the subscripts t and $t-1$ containing temporal information. The tracking is performed in two steps, by combining a forward tracking with a backward one.

1) FORWARD TRACKING

Forward tracking performs a local data association; in more details, the algorithm computes a similarity matrix S, where the generic element $s(i,j)$ is a measure of the similarity between the ith box B^i and the jth object O^j. The computation of $s(i,j)$ is detailed in Subsection II-B.3.

The maximum value of the matrix S:

$$s_{MAX} = s(i_{MAX}, j_{MAX}) \tag{5}$$

is iteratively searched for. In case s_{MAX} is higher than a given threshold T_1, then the object $O^{j_{MAX}}$ is associated to the blob $B^{i_{MAX}}$. The lifecycle of an object after the association to a blob is detailed in Subsection II-B.4. Note that this condition ($s_{MAX} > T_1$) only arises when the detection is very accurate, there are not any errors of split and merge and thus the confidence is sufficiently high. An example of 1:1 association is shown in Figure 5: fixed T_1 to 0.8, the algorithm compute S and finds the maximum value, namely $s(0,0) = 0.95$; thus, it associates the box B^0 to the object O^0. B^0 and O^0 are now excluded from the analysis and can not be further associated to any object and box, respectively. Then, the next maximum value if found, namely $s(1,1) = 0.93$ and the box B^1 is associated to the object O^1. In this situation, no backward chain needs to be activated since the confidence about the local data association is sufficiently high in both the associations.

In case the confidence is not sufficiently high (namely, $s_{MAX} < T_1$), no decision is taken and the backward chain is instead activated. More details about the backward chain are reported in the following Subsection II-B.2.

2) BACKWARD TRACKING

The backward chain is activated mainly in the following three situations: (1) a split occurs, namely a situation in which a single object is partitioned in two or more boxes, due to an error in the detection step; (2) a merge occurs, namely a situation in which two or more objects are merged in a single box, due to an error in the detection step; (3) the association is 1:1, but the confidence is not sufficiently high.

The activation of the forward chain impacts on the following operating parameters: the parameters of the detection using the foreground mask M and K; the parameters of the detection using the foreground features (eps and $MinPts$). The refining of such parameters depends of course on the typology of situation detected. Let us enter in more details of each situation and on the related refining procedure.

A split is detected in case:

- $s_{MAX} = s(i_{MAX}, j_{MAX})$ is lower than T_1
- there is at least another couple $B^{i_{2MAX}}, O^{j_{MAX}}$ such that $s(i_{2MAX}, j_{MAX}) < T_1$

In other words, two boxes (namely $B^{i_{MAX}}$ and $B^{i_{2MAX}}$) are both similar to a same object $O^{j_{MAX}}$, but none of them *enough* similar (that is with a similarity score higher than T_1). The backward chain is activated trying to merge such boxes. In order to achieve this aim, the following actions are performed: (1) the threshold M is decreased, trying to include in the foreground also some pixels between the boxes that have been wrongly included in the background; (2) the radius of the dilation kernel K is increased, trying to merge the blobs; (3) both the radius eps and the minimum number of neighbours $MinPts$ increases, so as to increase the size of the cluster to be considered.

In a more formal way, the refining module acts as following (*Refining1*); the updating rates of the i-th variable μ_i is expressed as a percentage of the variable itself ($\mu_i = \mu * i$, being μ the overall updating rate).

$$M_{NEW} = M - \mu_M \tag{6}$$
$$K_{NEW} = K + \mu_K \tag{7}$$
$$eps_{NEW} = eps + \mu_{eps} \tag{8}$$
$$MinPts_{NEW} = MinPts + \mu_{MinPts} \tag{9}$$

An example is shown in Figure 6: the first row shows the output of both detection and tracking after the first forward chain; the similarity matrix on the right show that a split has been detected: the two boxes B^0 and B^1 are both similar to the object O^1. Thus, the algorithm backtracks and the backward chain is activated; operating parameters are modified and the new forward chain is activated with the new parameters. The results of the detection with the new operating configuration are reported in the second row. Only one box is found, and the similarity value between such box (B^0) and the object O^1 is 0.95, higher than $T_1 = 0.8$. Thus, the association is performed.

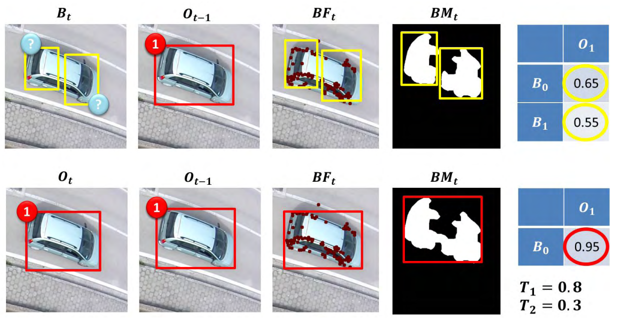

FIGURE 6. Resolution of a split error. The first row shows the output of the detection and of the local data association after the first forward chain. The second row shows the results of the forward chain after the first backtrack, with the new operating setup.

A merge is detected in case:

- $s_{MAX} = s(i_{MAX}, j_{MAX})$ is lower than T_1
- there is at least another couple $B^{i_{MAX}}, O^{j_{2MAX}}$ such that $s(i_{MAX}, j_{2MAX}) < T_1$

In practice, two objects (namely $O^{i_{MAX}}$ and $O^{i_{2MAX}}$) are both similar to a same box $B^{j_{MAX}}$, but none of them *enough* similar (that is with a similarity score higher than T_1). Differently from split problem, the backward chain is activated trying to partition such boxes. Thus, in this case, the following refining are performed: (1) the threshold M is increased, so as to try excluding from the foreground those pixels between the boxes, that should be included in the background; (2) the radius of the dilation kernel K is decreased; (3) both the radius eps and the minimum number of neighbours $MinPts$ decrease, so as to reduce the size of the cluster to be considered.

More formally, the refining module (*Refining2*) in this case acts as following:

$$M_{NEW} = M + \mu_M \tag{10}$$
$$K_{NEW} = K - \mu_K \tag{11}$$
$$eps_{NEW} = eps - \mu_{eps} \tag{12}$$
$$MinPts_{NEW} = MinPts - \mu_{MinPts} \tag{13}$$

An example is shown in Figure 7: after the first forward chain, a merge is detected since B^0 is similar to both O^1 and O^2. Thus, the algorithm backtracks, the operating parameters are refined and the new setup is fed into the forward chain; the results are reported in the second row of Figure 7: two different boxes are identified, namely B^0 and B^1, and are associated to the objects O^1 and O^2, respectively. It is important to highlight that our approach is based on the assumption that the merge is not caused by an occlusion (one object hidden behind another one), but instead by the fact that two objects are very close each other, but can be still visible separated. Anyway, this assumption holds in our scenario, since the images are acquired by a drone, with a head-view mounted camera.

The third situation the proposed approach is able to deal with is the improvement in the 1:1 association. Differently from the previous situations, the algorithm may not know

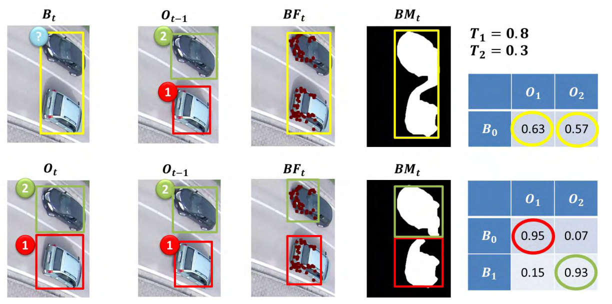

FIGURE 7. Resolution of a merge error. The first row shows the output of the detection and of the local data association after the first forward chain. The second row shows the results of the forward chain after the first backtrack, with the new operating setup.

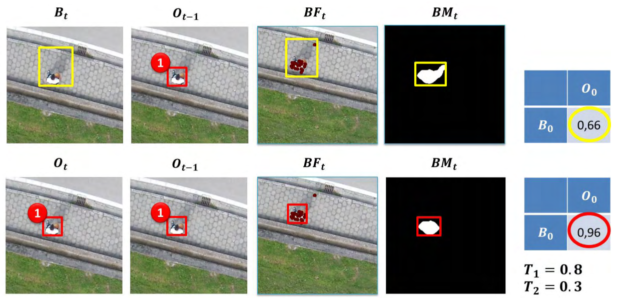

FIGURE 8. An example of 1:1 association, whose similarity value is improved from 0.66 to 0.96 thanks to the activation of the backward chain.

a priori how to modify the configuration parameters, namely if modifying according the rules sets *Refining1* or *Refining2*. Thus, the algorithm acts as follows: (1) if the area of the box is lower than the one of the object, then the rules set *Refining1* (adopted for solving the split) is applied; otherwise, the rules set *Refining2* (adopted for solving the merge) is employed; (2) considering that during the step (1) the wrong decision could have been taken, we evaluate the new similarity and we compare it with respect to the previous one; indeed, in case a new iteration is required (since the similarity is lower than T_1) two situations may arise: (i) the similarity value increases, thus the same refining rule is applied again; (ii) the similarity value decreases, thus a different refining rule is applied. An example is shown in Figure 8, where the size of the object is overestimation during the detection step; the backward chain is activated and the Refining2 rule is applied, so increasing the similarity $s(0, 0)$ from 0.66 to 0.96.

Independently on the refining rules to be activated, the backward procedure is repeated until the similarity value becomes greater than T_1, or, in order to assure the convergence of the algorithm, after a certain number A of attempts. In the latter case, the association with the highest similarity is carried out, since it is anyway the best possible matching.

We also introduced a second threshold, namely T_2. Indeed, in case the similarity is lower than T_2, or there are some blobs that are not associated to any objects, then a new object is created; as we will show in Section II-B.4, the state of this new object will be *Unstable*. Viceversa, in case an object is not associated to any blob, then its state is updated to *Ghost* or *Deleted*, depending on its past state. More information about the state transitions of the objects is detailed in Section II-B.4.

3) SIMILARITY SCORE

The similarity between the i-th box B^i and the j-th object O^j is computed by combining three complementary information, respectively based on the position (s_p), on the shape (s_f) and on the appearance (s_a):

$$s = \frac{\alpha_p \cdot s_p + \alpha_f \cdot s_f + \alpha_a \cdot s_a}{\sqrt{\alpha_p^2 + \alpha_f^2 + \alpha_a^2}} \tag{14}$$

The values of α_p, α_f and α_a have been set to 1, so as to give the same weight to all the contributions.

In more details, s_p evaluates the distance $d(i, j)$ between the B^i and the O^j, normalized with respect to the maximum displacement D of the object in two consecutive frames:

$$s_p = \begin{cases} 1 - \frac{d(i,j)}{D} & \text{if } d(i,j) \leq D \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Note that in our experiments D has been set to the diagonal size of the image, which is the maximum displacement of an object in two consecutive frames.

The second contribution s_f takes into account the areas (in terms of number of pixels) of B^i and of O^j, as follows:

$$s_f = \begin{cases} \dfrac{area(B^i)}{area(O^j)} & \text{if } area(B^i) \leq area(O^j) \\ \dfrac{area(O^j)}{area(B^i)} & \text{otherwise} \end{cases} \tag{16}$$

Finally, s_a evaluates the appearance, in terms of correlation between the histogram of the box B^i, namely h_B, and the histogram of the object O^j, namely h_O:

$$s_a = \frac{\sum_b (h_B(b) - \bar{h}_B)(h_O(b) - \bar{h}_O)}{\sqrt{\sum_b (h_B(b) - \bar{h}_B)^2 \sum_b (h_O(b) - \bar{h}_O)^2}}. \tag{17}$$

where

$$\bar{h}_B = \frac{1}{N} \sum_b h_B(b); \quad \bar{h}_O = \frac{1}{N} \sum_b h_O(b) \tag{18}$$

N is the number of the total bins and b represents the bth bin of the histogram.

As shown in [37], the above described features, have been proved to be as simple as effective even when applied in very challenging scenarios.

4) LIFE CYCLE OF AN OBJECT

The life cycle of the objects is modelled by the finite-state machine shown in Figure 9. The first time each object appears in the scene (namely when a blob is not associated to any object in the similarity matrix), it is considered *unstable*. Such precautionary choice allows to discard all the objects

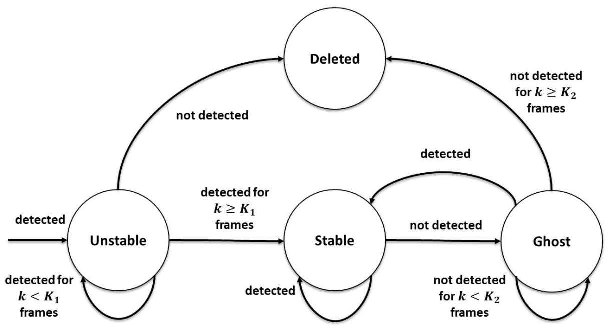
Deleted
not detected
not detected for $k \geq K_2$ frames
detected
detected for $k \geq K_1$ frames
detected
Unstable
Stable
not detected
Ghost
detected for $k < K_1$ frames
detected
not detected for $k < K_2$ frames

FIGURE 9. Finite-state machine which determines the life of the objects tracked by the proposed approach.

FIGURE 10. DJI F-450 drone equipped with a Nilox F60 camera, used for acquiring the MIVIA Drone dataset.

appearing in the scene just for a few frames due to spurious detection errors.

If an *unstable* object is tracked for at least K_1 consecutive frames, it becomes *stable*; otherwise, the algorithm recognises the error and deletes the object from the list. Differently from *unstable* objects, if a *stable* object disappears from the scene, it is not immediately discarded since it may happen that is not detected just for a couple of frames, due to the fact that it stops for a while (and thus is not detected) or it is more simply missed due to an error in the detection step. Thus, its state is updated to *ghost*. Such intermediate state allows to reduce the number of false negatives due to objects that disappear only for a few frames. If a *ghost* object does not appear within K_2 frames, then it becomes *deleted* and it is removed from the list; otherwise, in case it reappears in K_2 frames, it is considered *stable* again.

III. EXPERIMENTAL EVALUATION

In this section we provide an extensive analysis of the proposed approach: the dataset used in the experimentation is detailed in Section III-A, the experimental protocol is presented in Section III-B and the results are reported and discussed in Section III-C.

A. DATASETS

In order to carry out an extensive analysis of the proposed approach, we use four different datasets. Three of them are publicly available and have been widely used by the scientific community for the evaluation of single object tracking with moving camera: the Visual Object Tracking (hereinafter *VOT*) [14], the *UAV123* [13] and the one made accessible by the Communication Systems Group of Technische Universität Berlin (hereinafter *Berlin* dataset) [45]. The fourth dataset (hereinafter *MIVIA Drone* dataset) has been acquired by the authors of this paper at the University of Salerno and has been made publicly available for benchmarking purposes. Considering that the ground truth available for VOT, UAV123 and Berlin is devoted to single object tracking, we have manually produced a new ground truth (in terms of bounding boxes and identifiers) for multi-object tracking. The ground truth of all the videos, together with the new dataset, is available at the following link: http://mivia.unisa.it.

In more details, the VOT dataset [14] is composed by 60 image sequences, acquired with both fixed and moving camera; we used in our experiments only the subset of videos recorded with moving cameras.

The Berlin dataset [45] consists of six videos where the camera, frontal with respect to the targets, follows them in the scene. It is important to highlight that our method is not optimized to deal with videos recorded with frontal view (and then with mobile platforms different by drones, allowing an head view of the objects acquired by the camera like the ones available in the VOT and in the Berlin dataset). However, we decided to use them so as to also verify the behavior of the proposed approach with a generic scenario of moving camera as well as to have the possibility to compare it with state of the art approaches.

The UAV123 [13] is a recent dataset, which consists of 123 videos recorded with flying cameras equipped by unmanned aerial vehicles. We selected 33 videos, chosen so as to cover all the scenarios of the dataset. For such videos, we manually performed the ground truth, being available only for single object tracking and not for multi-objects tracking.

Finally, the MIVIA Drone dataset is composed of four videos recorded with a Nilox F60 camera equipped by a DJI F-450 drone (see Figure 10). During the acquisition, the drone flies at a height of about 20 meters, with fast rotational and translational movements. We collected more than 6, 000 frames divided in 4 videos, considering different challenging scenarios, in order to test the algorithm in presence of shadows, merge and splits. The most critical point is the fact that, differently from the videos available in the UAV123 dataset, the drone does not follow the targets in the scene, so its movement is completely independent of the moving objects.

Table 1 reports the details of the above mentioned datasets in terms of number of frames and videos. Since we are dealing with drones, most of the images belong to the UAV123 and the MIVIA Drone dataset, that have been acquired with flying cameras. In the whole, we used about 61,000 frames partitioned into 53 videos in our experimentation. Moreover it can be noted that the analysis has been carried out on image sequences with significantly different resolutions, ranging from 320×180 to 1920×1080.

TABLE 1. Details about the datasets used for the experimental evaluation. Min. Res. and Max. Res. are the minimum and the maximum resolutions of the videos for each dataset.

Dataset	Frames	Videos	Min. Res.	Max. Res.
VOT	3751	10	320×180	1280×720
Berlin	1470	6	352×192	720×576
UAV123	48770	33	1280×720	1280×720
MIVIA Drone	7518	4	1920×1080	1920×1080
Overall	61509	53	320×180	1920×1080

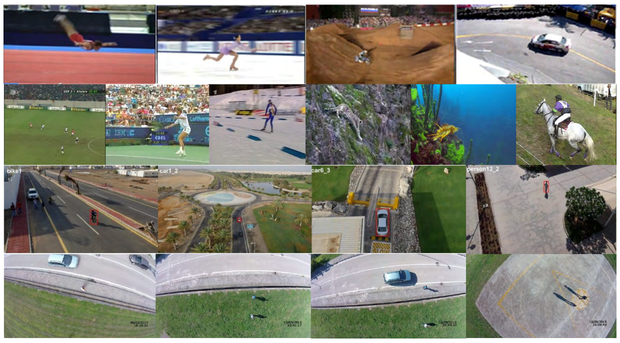

FIGURE 11. Images extracted from the datasets used for the experiments. The first two rows depict some frames belonging to VOT and Berlin dataset, acquired with moving cameras. The last two rows report images from UAV123 and MIVIA Drone dataset, captured with flying cameras.

Figure 11 points out the variety of scenarios considered for the experiments, in terms of objects of interest (people, cars and animals), activities (doing sports, walking, running) and distances from the targets.

B. EXPERIMENTAL PROTOCOL

The experimental protocol has been defined in order to evaluate the quality of the detection and the tracking. The performance of the detection has been evaluated in terms of Precision (P), Recall (R) and F-Score (F):

$$P = \frac{TP}{TP + FP} \tag{19}$$

$$R = \frac{TP}{TP + FN} \tag{20}$$

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{21}$$

where TP, FP and FN represent, respectively, the number of True-Positives (TP), False-Positives (FP) and False-Negatives (FN), computed by evaluating the overlapping between the boxes associated to the object at the ith frame, O_i, and the ground truth GT_i according to the Pascal Criterion [46]. In more details, O_i is a TP if the following condition is satisfied:

$$\frac{area(O_i \cap GT_i)}{area(O_i \cup GT_i)} \geq 0.5 \tag{22}$$

otherwise O_i can be considered a FP.

In order to evaluate the performance of the tracking algorithm, we use a pair of the Classification of Events, Activities and Relationships (*CLEAR*) metrics, namely the Moving Objects Tracking Accuracy (*MOTA*) and the (*MOTP*).

TABLE 2. Detection and tracking results in terms of Precision (P), Recall (R), F-Score (F), MOTA and MOTP for each dataset.

Dataset	P	R	F	MOTA	MOTP
VOT	0.83	0.80	0.81	0.84	0.73
Berlin	0.87	0.88	0.87	0.90	0.69
UAV123	0.48	0.58	0.52	0.32	0.57
MIVIA Drone	0.27	0.18	0.21	0.28	0.38
Overall	0.57	0.62	0.58	0.48	0.60

The MOTA is an accuracy measure for multi-object tracking which takes into account the number of missed detections, false positives and switches for a ground truth object.

$$MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} \left(fn_t + fp_t + log_{10}(im_t) \right)}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}, \tag{23}$$

where fn_t is the number of false negatives, fp_t is the number of false positives, and im_t is the number of ID mismatches in frame t considering the mapping in frame $(t-1)$.

The MOTP is a precision measure for multi-object tracking which considers the spatiotemporal overlap between the objects in the ground truth and the system output tracks:

$$MOTP = \frac{\sum_{i=1}^{N_{mapped}} \sum_{t=1}^{N_{frames}^{(t)}} \frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|}}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}}. \tag{24}$$

where $G_i^{(t)}$ and $D_i^{(t)}$ denote the ith ground truth object and the detected one in frame t.

C. RESULTS

Table 2 reports the results achieved by the proposed algorithm, averaged over all the videos of each dataset.

The average F-score achieved by the algorithm is 0.58, with a good balance between Precision (0.57) and Recall (0.62). The performance of the detection is higher on the VOT and on the Berlin dataset. Such results are not surprising because these datasets are recorded with moving cameras controlled by humans, so the moving targets are followed during the videos and are always in the foreground. Moreover the movement of the camera is controlled and the motion blur is very limited. Viceversa, UAV123 and MIVIA Drone datasets consist of videos acquired using flying cameras, which suffer any kind of sudden movements. Moreover the drone used in the MIVIA Drone dataset does not follow the targets, making the detection even more challenging.

As for the tracking metrics, we can note the same trend. The average MOTP is 0.6, so confirming the performance of the detection algorithm. The performance drop between moving and flying camera in terms of precision is less evident, because the online learning adapts the parameters in order to dynamically improve the detection and preserve the spatiotemporal overlap between the tracked object and the reference one. The average MOTA, equal to 0.48, is a little lower. In this case the accuracy drop is significant (about 0.5) because it is harder for the algorithm to follow multiple targets with sudden translational and rotational movements.

TABLE 3. Comparison of the detection results in terms of F-score on the Berlin dataset. NA (Not Available) indicates the absence of published results for the corresponding video sequence.

	[48]	[25]	[26]	[46]	Our
Allstar	0.52	NA	NA	**0.65**	0.55
BBC Fish	0.80	NA	NA	0.81	**0.96**
Biathlon	0.36	0.34	0.63	0.82	**0.99**
Horse	0.70	NA	NA	0.78	**0.85**
Mountain	0.73	0.39	0.86	0.73	**0.99**
Stefan	0.69	0.28	0.70	0.73	**0.90**

TABLE 4. Analysis of the processing frame rate at different resolutions.

	1280×720	640×360	320×180
FPS	2	16	53

Indeed, the best MOTA is achieved on the Berlin dataset, where in most of the cases the targets are very close to the camera.

Table 3 reports a comparison of the results, in terms of F-score, achieved by the proposed algorithm and the methods described in [25], [26], [45], and [47] on the Berlin dataset. The comparison points out that our approach is able to significantly outperform the others on all the scenarios, excluding the Allstar video. The latter is characterized by football players that are not always in movement, so the algorithm confuses them with the background causing a drop in the Recall. In the other cases the proposed method identifies the moving object with a small number of false negatives and false positives (F-score around 0.9), so confirming its effectiveness also when compared with the state of the art approaches.

In order to evaluate the efficiency of the proposed approach, we carried out an analysis of the processing time. The tests have been performed over a Intel Joule 570X board, equipped with a quad core Intel Atom T5700 CPU@1,7 GHz (max 2,4 GHz) and 4 GB RAM LPDDR4. The idea behind this choice lies in the fact that the board is a system on module, thus can be directly mounted on board of the drone, thus allowing the system to work in real time. Table 4 reports the frame rate obtained by varying the resolution of the images. Note that with a 4CIF resolution (640×360) the proposed algorithm is able to run at 16 fps, thus it is able to run on board in real time and if necessary to send on the ground, to an external server, only the result of the elaboration for further investigation (for instance the detection of events of interest).

D. DISCUSSION

After a quantitative and a qualitative analysis carried out on 53 videos and 61509 frames, a lot of interesting insights on the proposed method can be discussed.

Indeed, the automatic adjusting of the operating parameters is able to adapt the detection and tracking parameters to the specific scenario, making the algorithm able to deal with flying cameras. The experimental results demonstrate the effectiveness of the proposed method on images with various resolutions and completely different environments in terms of objects of interest, activities and distances from the targets.

A qualitative analysis points out that most of the errors are due to objects which stop their movement and are confused with the background. Such detection problems are partially solved in the tracking step by the finite-state machine, which introduces the concept of ghost state and it is able to deal with objects that disappear for a certain number of frames. Moreover, an event recognition module which analyzes the results of the tracking may infer interesting information (e.g. a queue or a car accident on a highway).

Another error identified after the qualitative analysis is related to objects properly detected and tracked by the algorithm, but with artifacts like shadows that warp the bounding boxes and cause a drop in the Precision and in the Recall (the Pascal Criterion is not respected), especially on the Mivia Drone dataset. Such errors are quite hard to solve, but in a real application they should not have a significant impact on the results. Indeed, an event recognition module is designed for processing the trajectories of the objects, so an error in the estimation of the shape should not determine an error in the analysis of the behavior.

A typical problem with flying cameras is the fact that the aerial vehicle may quickly vary the flight altitude, so causing a sharp change of the distance from the objects. Although the proposed algorithm is able to learn the optimal parameters, during the transient there may be errors due to the almost instantaneous change of the appearance (especially in terms of the size of the objects). Moreover, in a real application the aerial vehicle does not follow the movement of the objects (we simulated this situation in the Mivia Drone dataset), so the camera compensation is a very challenging operation liable to errors. A future direction to solve these drawbacks may be the use of the information about the flight. The knowledge of the geographic coordinates of the drone, together with its speed, can give a significant contribution to improve the quality of the camera compensation step. Indeed, the system can take into account the movement of the camera between consecutive frames in order to improve the quality of the foreground mask and the estimation of the background transformation due to the camera movement. In recent years the technology is moving quickly in this direction, so the use of such information to improve the multi-object tracking by flying camera may be not so far.

Before concluding, it is worth to point out that one of the main problems of evaluating this kind of approaches is the lack of publicly available implementations and ground truths for multi-object tracking. We hope that our significant effort of labelling and making publicly available the annotations for such wide datasets will help and encourage the scientific community to use these standard benchmarks for performance comparisons.

IV. CONCLUSION

In this paper we propose a method for tracking moving objects with a flying camera equipped by a drone. The main novelty of the proposed approach is the introduction of a detection and tracking algorithm based on a test and set procedure, where the choice of the optimal parameters is adaptively performed. Such technique allows to adapt the algorithm to each scenario, solving the problem of the configuration for environments not known a priori. The performance, both in terms of accuracy and efficiency, confirms the effectiveness of the proposed approach. The possibility to work in real time over systems on a module demonstrates that the system is suitable for real situation awareness and behavior analysis applications.

REFERENCES

[1] L. Brun, A. Saggese, and M. Vento, "Dynamic scene understanding for behavior analysis based on string kernels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 10, pp. 1669–1681, Oct. 2014.

[2] S. Minaeian, J. Liu, and Y. J. Son, "Vision-based target detection and localization via a team of cooperative UAV and UGVs," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 7, pp. 1005–1016, Jul. 2016.

[3] D. Cavaliere, V. Loia, A. Saggese, S. Senatore, and M. Vento, "Semantically enhanced UAVs to increase the aerial scene understanding," *IEEE Trans. Syst., Man, Cybern. Syst.*, to be published.

[4] A. D'Acierno, M. Leone, A. Saggese, and M. Vento, "A system for storing and retrieving huge amount of trajectory data, allowing spatio-temporal dynamic queries," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 989–994.

[5] K. Jo, M. Lee, J. Kim, and M. Sunwoo, "Tracking and behavior reasoning of moving vehicles based on roadway geometry constraints," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 2, pp. 460–476, Feb. 2017.

[6] J.-Y. Kwak, B. C. Ko, and J. Y. Nam, "Pedestrian tracking using online boosted random ferns learning in far-infrared imagery for safe driving at night," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 69–81, Jan. 2017.

[7] A. D'Acierno, A. Saggese, and M. Vento, "Designing huge repositories of moving vehicles trajectories for efficient extraction of semantic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2038–2049, Aug. 2015.

[8] K.-H. Lee, J.-N. Hwang, G. Okopal, and J. Pitton, "Ground-moving-platform-based human tracking using visual slam and constrained multiple kernels," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3602–3612, Dec. 2016.

[9] T. Chen, R. Wang, B. Dai, D. Liu, and J. Song, "Likelihood-field-model-based dynamic vehicle detection and tracking for self-driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3142–3158, Nov. 2016.

[10] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier, "Tracking all road users at multimodal urban traffic intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3241–3251, Nov. 2016.

[11] S. Gao, Z. Han, C. Li, Q. Ye, and J. Jiao, "Real-time multipedestrian tracking in traffic scenes via an RGB-D-based layered graph model," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2814–2825, Oct. 2015.

[12] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[13] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *Proc. Eur. Conf. Comput. Vis.*. Springer, 2016, pp. 445–461.

[14] M. Kristan *et al.*, "The visual object tracking vot2015 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Dec. 2015, pp. 1–23.

[15] V. P. Malagi, R. Babu, and K. Rangarajan, "Multi-object tracking in aerial image sequences using aerial tracking learning and detection algorithm," *Defence Sci. J.*, vol. 66, no. 2, pp. 122–129, 2016. [Online]. Available: http://publications.drdo.gov.in/ojs/index.php/dsj/article/view/8972

[16] M. Kimura, R. Shibasaki, X. Shao, and M. Nagai, "Automatic extraction of moving objects from uav-borne monocular images using multi-view geometric constraints," in *Proc. Int. Micro Air Vehicle Conf. Competition (IMAV)*, Delft, The Netherlands: Delft Univ. Technology, Aug. 2014.

[17] M. Siam, R. ElSayed, and M. ElHelw, "On-board multiple target detection and tracking on camera-equipped aerial vehicles," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2012, pp. 2399–2405.

[18] W. Yang, G. Gu, and F. Xu, "Motion object detection by three-view constraint using moving camera," *Opt. Rev.*, vol. 22, no. 4, pp. 577–587, 2015, doi: 10.1007/s10043-015-0111-8.

[19] S. Dey, V. Reilly, I. Saleemi, and M. Shah, "Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint," in *Computer Vision—ECCV*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer, 2012, pp. 860–873.

[20] G. Zhang, J. Jia, W. Xiong, T.-T. Wong, P.-A. Heng, and H. Bao, "Moving object extraction with a hand-held camera," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[21] J. Kang, I. Cohen, and G. Medioni, "Continuous tracking within and across camera streams," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2003, pp. I-267–I-272.

[22] J. Kang, S. Kim, T. J. Oh, and M. J. Chung, "Moving region segmentation using sparse motion cue from a moving camera," in *Moving Region Segmentation Using Sparse Motion Cue From a Moving Camera*. Berlin, Heidelberg: Springer, 2013, pp. 257–264, doi: 10.1007/978-3-642-33926-4_24.

[23] S. Bhattacharya, H. Idrees, I. Saleemi, S. Ali, and M. Shah, "Moving object detection and tracking in forward looking infra-red aerial imagery," in *Machine Vision Beyond Visible Spectrum*. Berlin, Germany: Springer, 2011, pp. 221–252.

[24] B. Jung and G. S. Sukhatme, "Real-time motion tracking from a mobile robot," *Int. J. Social Robot.*, vol. 2, no. 1, pp. 63–78, 2010, doi: 10.1007/s12369-009-0038-y.

[25] W.-C. Hu, C.-H. Chen, C.-M. Chen, and T.-Y. Chen, "Effective moving object detection from videos captured by a moving camera," in *Intelligent Data Analysis and Its Applications*, vol. 1. Cham, Switzerland: Springer, 2014, pp. 343–353.

[26] W.-C. Hu, C.-H. Chen, T.-Y. Chen, D.-Y. Huang, and Z.-C. Wu, "Moving object detection and tracking from video captured by moving camera," *J. Vis. Commun. Image Represent.*, vol. 30, pp. 164–180, Jul. 2015, doi: 10.1016/j.jvcir.2015.03.003.

[27] M. Teutsch and W. Krüger, "Detection, segmentation, and tracking of moving objects in UAV videos," in *Proc. IEEE 9th Int. Conf. Adv. Video Signal-Based Surveill. (AVSS)*, Washington, DC, USA: IEEE Computer Society, 2012, pp. 313–318, doi: 10.1109/AVSS.2012.36.

[28] S. Minaeian, J. Liu, and Y.-J. Son, "Effective and efficient detection of moving targets from a UAV's camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 497–506, Feb. 2018.

[29] M. Gupta, S. Kumar, L. Behera, and V. K. Subramanian, "A novel vision-based tracking algorithm for a human-following mobile robot," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 7, pp. 1415–1427, Jul. 2017.

[30] N. Takemura, Y. Nakamura, Y. Matsumoto, and H. Ishiguro, "A path-planning method for human-tracking agents based on long-term prediction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1543–1554, Nov. 2012.

[31] V. Bruni and D. Vitulano, "An improvement of kernel-based object tracking based on human perception," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 44, no. 11, pp. 1474–1485, Nov. 2014.

[32] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M.-M. Meinecke, "Stereo-camera-based urban environment perception using occupancy grid and object tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 154–165, Mar. 2012.

[33] C.-W. Liang and C.-F. Juang, "Moving object classification using a combination of static appearance features and spatial and temporal entropy values of optical flows," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3453–3464, Dec. 2015.

[34] R. O. Chavez-Garcia and O. Aycard, "Multiple sensor fusion and classification for moving object detection and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 525–534, Feb. 2016.

[35] R. Ke, Z. Li, S. Kim, J. Ash, Z. Cui, and Y. Wang, "Real-time bidirectional traffic flow parameter estimation from aerial videos," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 890–901, Apr. 2017.

[36] S. Zhao *et al.*, "A robust real-time vision system for autonomous cargo transfer by an unmanned helicopter," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1210–1219, Feb. 2015.

[37] R. Di Lascio, P. Foggia, G. Percannella, A. Saggese, and M. Vento, "A real time algorithm for people tracking using contextual reasoning," *Comput. Vis. Image Understand.*, vol. 117, no. 8, pp. 892–908, Aug. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1077314213000908

[38] A. Milan, K. Schindler, and S. Roth, "Multi-target tracking by discrete-continuous energy minimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2054–2068, Oct. 2016.
[39] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *Proc. IEEE CVPR*, Jun. 2011, pp. 1201–1208.
[40] C. Feichtenhofer, A. Pinz, and A. Zisserman. (Oct. 2017). "Detect to track and track to detect." [Online]. Available: https://arxiv.org/abs/1710.03958
[41] H.-U. Kim and C.-S. Kim, "CDT: Cooperative detection and tracking for tracing multiple objects in video sequences," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 851–867.
[42] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 1994, pp. 593–600.
[43] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. IJCAI*, vol. 81, 1981, pp. 674–679.
[44] S. Choi, T. Kim, and W. Yu, "Performance evaluation of RANSAC family," in *Proc. Brit. Mach. Vis. Assoc. (BMVC)*, 2009, pp. 1–12. [Online]. Available: http://dblp.uni-trier.de/db/conf/bmvc/bmvc2009.html#ChoiKY09
[45] M. G. Arvanitidou, M. Tok, A. Glantz, A. Krutz, and T. Sikora, "Motion-based object segmentation using hysteresis and bidirectional inter-frame change detection in sequences with moving camera," *Signal Process., Image Commun.*, vol. 28, no. 10, pp. 1420–1434, 2013.
[46] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.
[47] B. Qi, M. Ghazal, and A. Amer, "Robust global motion estimation oriented to video object segmentation," *IEEE Trans. Image Process.*, vol. 17, no. 6, pp. 958–967, Jun. 2008.

VINCENZO CARLETTI received the Ph.D. degree in European label in computer engineering from the University of Salerno, Fisciano, Italy, in 2016. He is currently a Research Fellow with the University of Salerno. His research activity is focused on exact and inexact graph matching for structural pattern recognition, applications in computer vision and pattern recognition.

ANTONIO GRECO received the Ph.D. degree in computer science and computer engineering from the University of Salerno in 2018. His research activity is focused on computer vision and machine learning techniques for retail and video surveillance applications.

ALESSIA SAGGESE received the Ph.D. degree in computer engineering from the University of Salerno, Italy, and the University of Caen, France, in 2014. She is currently an Assistant Professor with the University of Salerno. Her research interests include basic methodologies and applications in computer vision and pattern recognition. She has been a member of the International Association for the Pattern Recognition Technical Committee 15 on Graph-Based Representations in Pattern Recognition since 2012.

MARIO VENTO received the Ph.D. degree in computer engineering from the University of Naples Federico II in 1989. He is currently a Full Professor of artificial vision, machine learning, and cognitive robotics with the University of Salerno, Italy, where he is also the Coordinator of the Artificial Vision Laboratory and the Dean of the Department of Information and Electrical Engineering and Applied Mathematics. His research activities cover real-time video analysis and interpretation for video surveillance applications and affective computing, cognitive robotics, classification techniques, statistical, syntactic and structural, exact and inexact graph matching, multiexpert classification, and learning methodologies for structural descriptions. He is a Fellow Scientist of the International Association Pattern Recognition. He served as the Chairman of the IAPR Technical Committee 15 on Graph-Based Representation in Pattern Recognition from 2002 to 2006. He is currently an Associate Editor of *Pattern Recognition* journal.