Moving Object Detection and Tracking in Forward Looking Infra-Red Aerial Imagery

Subhabrata Bhattacharya, Haroon Idrees, Imran Saleemi, Saad Ali and Mubarak Shah

Abstract This chapter discusses the challenges of automating surveillance and reconnaissance tasks for infra-red visual data obtained from aerial platforms. These problems have gained significant importance over the years, especially with the advent of lightweight and reliable imaging devices. Detection and tracking of objects of interest has traditionally been an area of interest in the computer vision literature. These tasks are rendered especially challenging in aerial sequences of infra red modality. The chapter gives an overview of these problems, and the associated limitations of some of the conventional techniques typically employed for these applications. We begin with a study of various image registration techniques that are required to eliminate motion induced by the motion of the aerial sensor. Next, we present a technique for detecting moving objects from the ego-motion compensated input sequence. Finally, we describe a methodology for tracking already detected objects using their motion history. We substantiate our claims with results on a wide range of aerial video sequences.

Keywords Aerial image registration · Object detection · Tracking

S. Bhattacharya (✉) · H. Idrees · I. Saleemi · M. Shah
University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32826, USA
e-mail: subh@cs.ucf.edu

H. Idrees
e-mail: haroon@cs.ucf.edu

I. Saleemi
e-mail: imran@cs.ucf.edu

M. Shah
e-mail: shah@cs.ucf.edu

S. Ali
Sarnoff Corporation, 201 Washington Road, Princeton, NJ 08540, USA
e-mail: sali@sarnoff.com

© Springer-Verlag Berlin Heidelberg 2011

1 Introduction

Detection and tracking of interesting objects has been a very important area of research in classical computer vision where objects are observed in various sensor modalities, including EO and IR, with static, hand-held and aerial platforms [34]. Many algorithms have been proposed in the past that differ in problem scenarios especially in camera dynamics and object dynamics [14, 15]. Tracking of a large, variable number of moving targets has been a challenging problem due to the sources of uncertainty in object locations, like, dynamic backgrounds, clutter and occlusions, and especially in the scenario of aerial platforms, measurement noise. In recent years, a significant amount of published literature has attempted to deal with these problems, and novel approaches like tracking-by-detection have been increasingly popular [17]. Such approaches involve the process of continuously applying a detection algorithm on single frames and associating detections across frames. Several recent multi-target tracking algorithms address the resulting data association problem by optimizing detection assignments over a large temporal window [2, 5, 17, 24].

Aerial tracking of multiple moving objects is however much more challenging because of the small object sizes, lack of resolution, and low quality imaging. Appearance based detection methods [10] are therefore, readily ruled out in such scenarios. The motion based object detection approaches rely on camera motion stabilization using parametric models [20], but in addition to parallax, cases of abrupt illumination changes, registration errors, and occlusions severely affect detection and tracking in airborne videos. Many algorithms have been proposed to overcome these problems of detection and tracking on frame to frame and pixel to pixel bases, including global illumination compensation [32], parallax filtering [37], and employing contextual information for detection [13, 29]. Some existing algorithms have performed well in planar scenes where adequate motion based foreground–background segmentations are achievable [36]. Most of the existing methods however have concentrated on medium and low altitude aerial platform sequences. Although such sequences suffer from the problem of strong parallax induced by structures perpendicular to the ground plane, like trees, towers, they do offer more pixels per target.

Effective use of visual data generated by UAVs requires design and development of algorithms and systems that can exhaustively explore, analyze, archive, index, and search this data in a meaningful way. In today's UAV video exploitation process, a ground station controls the on-board sensors and makes decisions about where the camera mounted on the bottom of the UAV should be looking. Video is relayed back to the intelligence center or some standard facility for assessment by the analysts. Analysts watch the video for targets of interest and important events which are communicated back to soldiers and commanders in the battle zone. Any post collection review of the video normally takes several hours for analysts to inspect a single video. The inherent inefficiency of this process and sheer magnitude of the data leads to an inability to process reconnaissance information as fast as it becomes available. The solution to this problem lies in augmenting the manual video exploitation process with computer vision based systems that can automatically manage and process ever increasing volume of aerial surveillance information without or with minimal involvement of human analyst. Such systems should handle all tasks from video reception to video registration, region of interest (ROI) detection to target tracking and event detection to video indexing. It should also be able to derive higher level semantic information from the videos which can be used to search and retrieve a variety of videos. Unfortunately, however there is still a gap between the operational requirements and the available capabilities in today's system for dealing with the UAV video stream.

A system capable of performing the above mentioned tasks for UAV videos will have to grapple with significantly higher levels of complexity as compared to the static camera scenario, as both the camera and the target objects are mobile in a dynamic environment. A significant amount of literature in the computer vision community has attempted to deal with some of these problems individually. We present a brief overview of these methods individually, along with the challenges and limitations involved.

1.1 Ego-Motion Compensation

Tracking of moving objects from a static camera platform is a relatively easier task than those from mobile platforms and is efficiently accomplished with sophisticated background subtraction algorithms. For a detailed study of these tracking techniques, the interested reader is requested to refer to [25]. Cameras mounted on mobile platforms, as observed in most aerial surveillance or reconnaissance, tend to capture unwanted vibrations induced by mechanical parts of the platform coupled with directed translation or rotation of the whole platform in 3-dimensional space. All the aforementioned forms of motion render even the most robust of the background subtraction algorithms ineffective in scenarios that involve tracking from aerial imagery.

A straightforward approach to overcome this problem is to eliminate the motion induced in the camera through the aerial platform, which is also known as *ego-motion compensation* in computer vision literature [12, 33, 35]. The efficacy of almost all image-based ego-motion compensation techniques depends on the underlying image registration algorithms they employ.

This step is also known as video alignment [9, 26] where objective is to determine the spatial displacement of pixels between two consecutive frames. The benefit of performing this step comes from the fact that after aligning the video, the intensity of only those pixels will be changing that correspond to moving objects on the ground. A detailed survey of various image alignment and registration techniques is available in [26]. Ideally an alignment algorithm should be insensitive to platform motion, image quality, terrain features and sensor modality. However, in practice these algorithms come across several problems:

- Large camera motion significantly reduces the overlap between consecutive frames which does not provide sufficient information to reliably compute the spatial transformation between the frames.
- Most of the alignment algorithms assume presence of dominant plane which is defined as a planar surface covering majority of pixels in an image. This assumption does not remain valid when a UAV views a non-planar terrain or takes a close up view of the object, which results in presence of multiple dominant planes. This causes parallax which often is hard to detect and remove.
- Sudden illumination changes result in drastic pixel intensity variations and make it difficult to establish feature correspondences across different frames. Gradient based methods for registration are more robust to an illumination change, rather than the feature based methods. Motion blur in the images can also throw off the alignment algorithm.

1.2 Regions of Interest Detection

Once the motion of the moving platform is compensated the next task is to identify 'regions of interest' (ROIs) from the video, the definition of which varies with application. In the domain of wide area surveillance employing UAVs, all the moving objects fall under the umbrella of ROI. Reliable detection of foreground regions in videos taken by UAVs poses a number of challenges, some of which are summarized below:

- UAVs often fly at a moderate to high altitude thus gathering the global context of the area under surveillance. Therefore, sizes of the potential target objects often appear very small in the range of 20–30 pixels. Small number of pixels on a target makes it difficult to distinguish it from the background and noise.
- As a UAV flies around the scene, the direction of illumination source (Sun) is continuously changing. If the background model is not constantly updated that may results in spurious foreground regions.
- Sometimes there are uninteresting moving objects present in the scene e.g., waving fags, flowing water, or moving leaves of a tree. If a background subtraction method falsely classifies such a region as a foreground region, then this region will be falsely processed as a potential target object.

1.3 Target Tracking

The goal of tracking is to track all the detected foreground regions as long as they remain visible in the field of view of the camera. The output of this module consists of trajectories that depict the motion of the target objects. In case of UAV videos several tracking options are available. One can perform tracking in a global mosaic or opt for tracking using geographical locations of the objects. Tracking in geographical locations is often called geo-spatial tracking and requires sensor modeling. Tracking algorithms also have to deal with number of challenges:

- Due to the unconstrained motion of the camera it is hard to impose constraints of constant size, shape, intensity, etc., on the tracked objects. An update mechanism needs to be incorporated to handle the dynamic changes in appearance, size and shape models.
- Occlusion is another factor that needs to be taken into account. Occlusions can be inter-object or caused by the terrain features e.g trees, buildings, bridges, etc.
- Restricted field of view of the camera adds to the complexity of the tracking problem. Detected objects are often geographically scattered. Restricted field of view of the camera allows UAV to track only certain number of objects at a time. It either has to move back and forth between all previously detected object or has to prioritize which target to pursue based upon the operational requirement.
- Tracking algorithms also have to deal with the imperfections of the object detection stage.

While designing a computer vision system that is capable of performing all the above mentioned tasks effectively in infra-red sequences, we need to consider the following additional issues:

- FLIR images are captured in significantly lower resolution compared to their EO counterparts as for a given resolution, infra-red sensor equipments are comparatively more expensive to install and maintain.
- FLIR sensing produces noisier images than regular EO imaging systems.
- As FLIR images tend to have lower contrast, they require further processing to improve the performance of algorithms used in ego-motion compensation, ROI detection and tracking.

The rest of this chapter is organized as follows: in Sect. 2 we discuss some of the prominent advances in the field of automatic target detection and tracking from aerial imagery. Section 3 provides a detailed description of our system that we have developed for tracking of objects in aerial EO/FLIR sequences. This section is followed by experimental results on 38 sequences from the VIVID-3 and AP-HILL datasets, obtained under permission from the Army Research Lab and US Govt.'s DARPA programs, respectively. We conclude the chapter with some of the limitations that we intend to address in future.

2 Related Work

Tracking moving objects from an aerial platform has seen numerous advances [1, 3, 16, 18, 30, 31, 38] in recent years. We confine our discussion to only a subset of the literature that has strong relevance with the context of this chapter. The authors of [16] present a framework that involves separating aerial videos into the static and dynamic scene components using 2-D/3-D frame-to-frame alignment followed by scene change detection. Initially, local tracks are generated for detected moving objects which are then converted to global tracks using geo-registration with a controlled reference imagery, elevation maps and site models. The framework is also capable of generating mosaics for enhanced visualization.

Zhang and Yuan [38] address the problem of tracking vehicles from a single moving airborne camera under occluded and congested circumstances using a tracker that is initialized from point features extracted from selected region of interest. In order to eliminate outliers that are introduced due to partial occlusion, an edge feature based voting scheme is used. In case of total occlusion, a Kalman predictor is employed. Finally, an appearance based matching technique is used to ensure that the tracker correctly re-associates objects on their re-entry into the field of view.

In [3], the authors use a video processor that has embedded firmware for object detection and feature extraction and site modeling. A multiple hypothesis tracker is then initialized using the positions, velocities and features to generate tracks of current moving objects along with their history.

The authors of [18] address the issue of urban traffic surveillance from an aerial platform employing a coarse-to-fine technique consisting of two stages. First, candidate regions of moving vehicle are obtained using sophisticated road detection algorithms followed by elimination of non-vehicle regions. In the next stage, candidate regions are refined using a cascade classifier that reduces the false alarm rate for vehicle detection.

Yalcin et al. [31] propose a Bayesian framework to model dense optical flow over time which is used to explicitly estimate the appearance of pixels corresponding to the background. A new frame is segregated into background and foreground object using an EM-based motion segmentation which is initialized by the background appearance model generated from previous frames. Vehicles on ground can be eventually segmented by building a mosaic of the background layer.

Xiao et al. [30] in their paper on moving vehicle and person tracking in aerial videos present a combination of motion layer segmentation with background stabilization, for efficient detection of objects. A hierarchy of gradient based vehicle versus person classifier is used on the detected objects prior to the generation of tracks.

The COCOA system [1] presented by Ali et al. is a 3-staged framework built using MATLAB, capable of performing motion compensation, moving object detection and tracking on aerial videos. Motion compensation is achieved using direct frame to frame registration which is followed by an object detection algorithm that relies on frame differencing and background modeling. Finally, moving blobs are tracked as long as the objects remain in the field of view of the aerial camera. The system has demonstrated its usability in both FLIR and EO scenarios.

The COCOALIGHT system is built from scratch keeping speed and portability into consideration while supporting the core functionalities of [1]. A detailed analysis of the algorithms employed for motion compensation, object detection and tracking with the justification behind their selection is provided in this chapter. We intend to disburse the technical insight while developing a practical system that is targeted to solve some of the predominant problems encountered while tracking in aerial imagery both within and beyond visible spectrum.

3 COCOALIGHT System Overview

The COCOALIGHT system shares the concept of modularity from its predecessor COCOA with complete change in design and implementation to facilitate tracking with near real-time latency. The software makes use of a widely popular open-source computer vision library which helps in seamlessly building the application both in 32 and 64-bit Windows and Linux PC platforms. Since the system is compiled natively, it is inherently much faster than interpreted MATLAB instructions present in COCOA. Furthermore, the software is packaged as an easy to use command-line console application eliminating memory intensive user interfaces from COCOA, rendering it a program with a low memory footprint, justifying the name COCOALIGHT. The design also exploits computational benefits from multi-threading during important fundamental image processing operations, e.g., gradient computation, feature extraction, computation of image pyramids.

Similar to COCOA, this system also consists of three independent components. However, unlike the COCOA system, which only supports processing in batch mode (an entire sequence needs to be processed to generate tracks), COCOALIGHT has capability for both batch and online processing. In the online mode, the tracking algorithm can be initialized with as few as only first ten frames from the video sequence. In addition, the software can leverage FFMPEG library support to process encoded videos without decompressing the video into image frames which is a significant improvement in usability over its MATLAB counterpart.

Having provided some knowledge about the implementation, we proceed towards a detailed discussion of the individual modules of the system.

3.1 Motion Compensation

The *motion compensation* module is the first and foremost module of the COCOALIGHT software framework. Any errors incurred in this module while eliminating camera motion get propagated to the subsequent modules namely the object detection and tracking modules. Due to this fact, the motion compensation stage necessitates employing highly accurate image alignment algorithms. Motivated solely by this objective we investigated several image alignment algorithms to suit our requirement. All our experiments are performed on sequences from VIVID dataset and from three other datasets, each collected using different aerial platforms flying over different geographical locations under different illumination conditions.

A study of the image registration techniques [9, 26] reveals that a registration algorithm must address the following issues which need careful consideration:

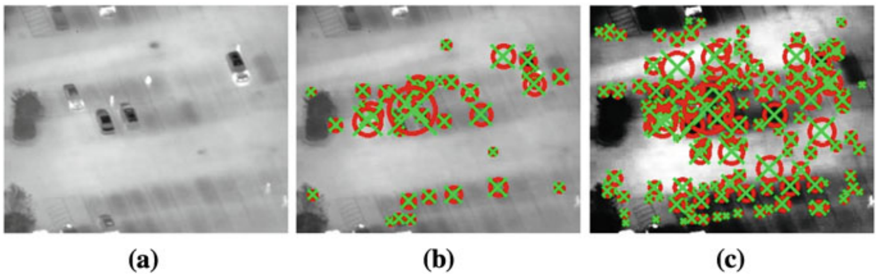


Fig. 1 Effect of histogram equalization on the detection of SURF interest points on a low contrast FLIR image. **a** Original FLIR image, **b** has a total of 43 SURF interest points whereas, **c** has a total number of 174 interest points after histogram equalization

- detecting candidate features also known as control points, from image pair to be registered,
- establishing correspondence between pairwise candidate features,
- estimating transformation model from point correspondence, and
- mapping image pair using the computed transformation model.

From our collection of video sequences, we observe that most of the frames demonstrate perspective projection artifacts. For this reason, we set our registration algorithm to estimate projective transformation parameters, also known as homography. Once homography parameters are obtained, a standard technique is available to perform the mapping operation between image pairs. In this section, we concentrate on the steps that involve proper selection of candidate features and establishing correspondence between the feature pairs.

In order to enhance feature detection in FLIR imagery, all the frames are subjected to a pre-processing stage. Histogram equalization is a widely popular technique to improve contrasts of IR images that are usually blurry. The effect of histogram equalization is clearly evident in the images shown in Fig. 1 with the histogram equalized image producing more interest points denoted by red–green circular cross-hairs.

3.1.1 Gradient-Based Method

Featureless spatio-temporal gradient-based methods are widely popular in image registration literature [9, 26] because of their ease of implementation. We use the unweighted projective flow algorithm proposed by Mann and Piccard in [20] to compute the homography parameters.

A homography $H = \{h_{ij}\}$, is a 3×3, 8 DOF projective transformation that models the relationship between the location of a feature at (x, y) in one frame, and the location (x', y') of the same feature in the next frame with eight parameters, such that,

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}, \quad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}. \tag{1}$$

The brightness constancy constraint results in a non-linear system of equations involving all pixels in the overlap range (region where the source and target images overlap). Using the method of [20], this system can be linearized for a least squares solution, such that, given two images, $I(x, y)$ and $I'(x, y)$, each pixel $i \in [1, N_p]$, then contributes an equation to the following system,

$$\begin{bmatrix} & x_i I_x(x_i, y_i) & \\ & y_i I_x(x_i, y_i) & \\ & I_x(x_i, y_i) & \\ & x_i I_y(x_i, y_i) & \\ \cdots & y_i I_y(x_i, y_i) & \cdots \\ & I_y(x_i, y_i) & \\ & x_i I_t(x_i, y_i) - x_i^2 I_x(x_i, y_i) - x_i y_i I_y(x_i, y_i) & \\ & y_i I_t(x_i, y_i) - x_i y_i I_x(x_i, y_i) - y_i^2 I_y(x_i, y_i) & \end{bmatrix}^\top \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i I_x(x_i, y_i) + y_i I_y(x_i, y_i) \\ -I_t(x_i, y_i) \\ \vdots \end{bmatrix}, \tag{2}$$

$$\mathbf{A}_{N_p \times 8} \mathbf{x}_{8 \times 1} = \mathbf{B}_{N_p \times 1}, \tag{3}$$

where $I_t(x_i, y_i) = I(x_i, y_i) - I'(x_i, y_i)$, $I_x(x_i, y_i) = \frac{\partial I(x_i, y_i)}{\partial x}$, and $I_y(x_i, y_i) = \frac{\partial I(x_i, y_i)}{\partial y}$, while h_{33} is 1. The least squares solution to this over-constrained system can be obtained with a singular value decomposition or pseudo-inverse. A coarse to fine estimation is achieved using three levels of Gaussian Pyramids. The spatial and temporal derivatives are also computed after smoothing using a Gaussian kernel of fixed variance. This process is fairly computation intensive as it involves solving a linear system of N_p equations where N_p is the number of pixels in each layer of the Gaussian pyramid. We used this technique as a baseline for comparison with our feature based registration algorithm in terms of speed and accuracy.

3.1.2 Feature-Based Method

As alternative to featureless gradient based methods, we study the performance of some feature-based alignment algorithms. We use two different algorithms to estimate homography with several types of feature detector algorithms. These two algorithms differ in the way they obtain correspondence between candidate feature-pairs of source and target images. Here is a detailed description of both the algorithms:

Flow based feature correspondence. In this algorithm, we extract invariant features from source image by applying one of the following methods:

- KLT [27] features. We obtain interest points in the image with significantly large eigenvalues by computing minimal eigenvalue for every source image pixel followed by non-maxima suppression in a local $d \times d$ neighborhood patch. Interest points with minimal value less than an experimentally determined threshold are eliminated prior to a final filtering based on spatial proximity of the features in order to extract only strong candidate interest points.

- SIFT [19] features. These features are extracted by computing the maxima and minima after applying difference of Gaussians at different scales. Feature points that lie along edges and points with low contrast are eliminated from the list of potential interest points. The dominant orientations are assigned to localized feature points. A 128-dimensional feature descriptor is obtained at each interest point extracted in this manner. We modify an open-source implementation of the SIFT algorithm[1] for extracting interest points.
- SURF [4] features. Speeded Up Robust Features are computed based on sums of responses obtained after applying a series of predefined 2-dimensional Haar wavelet responses on 5×5 image patches. The computation efficiency is enhanced up using integral images. A 128-dimensional vector is finally generated for each interest point.
- Random MSER [21] contour features. As the name suggests, we extract random points from contours returned after determining Maximally Stable Extremal Regions from an image. The MSERs are determined by first sorting image pixels according to their intensity, followed by a morphologically connected region merging algorithm. The area of each connected component is stored as a function of intensity. A larger connected component engulfs a smaller component until a maximally stable criterion is satisfied. Thus, MSERs are those parts of the image where local binarization is stable over a large range of thresholds.

Pixel locations corresponding to the features extracted using one of the above algorithms are stored in an $N \times 2$ matrix. These pixel locations are iteratively refined to find the interest point locations accurate to subpixel level. Using these sparse set of points from the source image, we compute respective optical flows in the target image. A pyramidal implementation [8] of Lucas Kanade's method is employed for this task which returns us corresponding points in the subsequent frame from the video sequence. A block diagram describing the important steps of this algorithm is shown in Fig. 2.

Descriptor similarity based feature correspondence. This algorithm works for those feature extraction methods that yield well defined descriptors for all detected interest points for e.g., SIFT and SURF, in a given source image. We first compute interest points in both source and destination images using either of the two methods. Thus we obtain two sets, which may not have equal number of interest points. Since each interest point is described in high-dimensional space, correspondences could be estimated using an approximate nearest neighbor search. We use a fast, freely available implementation [22] for this purpose. A block diagram is provided in Fig. 3 which explains this process. This technique is more robust as compared to the flow based mapping technique since it considers several attributes of the extracted feature points while generating the correspondence. However, it is computationally

[1] http://web.engr.oregonstate.edu/hess/downloads/sift/sift-latest.tar.gz

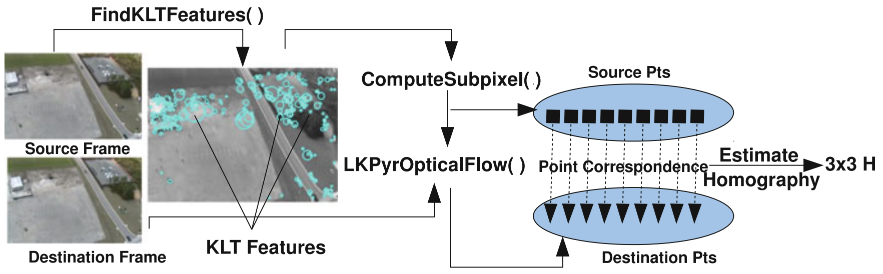


Fig. 2 Schematic diagram of the optical flow based correspondence mapping algorithm used for the motion compensation stage

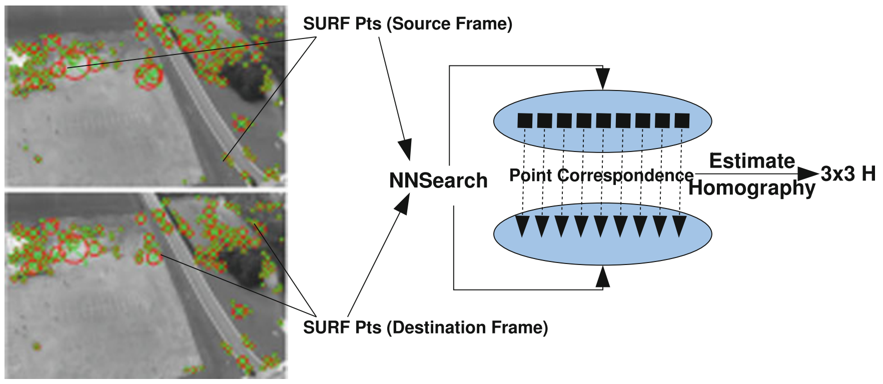


Fig. 3 Demonstration of the corresponding mapping algorithm based on descriptor similarity. This is used as an error correction mechanism in the cummulative homography computation step, within the motion compensation technique proposed here

more expensive than the former. We determine the accuracy of the registration algorithm by measuring the frame difference (FD) score. Formally, the FD score between a pair of consecutive intensity images I_t and I_{t+1} can be defined as:

$$FD = \frac{1}{N_p} \sum_{j=1}^{N_p} |I_t^j \times M(I_{t+1}^j) - W(I_{t+1}^j)|, \tag{4}$$

where $M(I_{t+1})$, $W(I_{t+1})$ are the outlier mask and the warped output of I_{t+1} with respect to I_t, respectively and N_p being the total number of pixels in a frame.

From the point correspondences established using either of the two methods discussed, we obtain respective pixel locations that are used to compute homography with the help of the following set of equations:

$$H = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^T, \tag{5}$$

$$a_x = [-x_i, -y_i, -1, 0, 0, 0, x_i'x_i, x_i'y_i, x_i']^T, \tag{6}$$

$$a_y = [0, 0, 0, -x_i, -y_i, -1, y_i' x_i, y_i' y_i, y_i']^T. \tag{7}$$

For a given set of N corresponding point pairs $\{(x_i, y_i), (x_i', y_i')\}$ for $1 \leq i \leq N$, the following linear system of equations hold good:

Given a set of corresponding points, we can form the following linear system of equations:

$$[{a_{x_1}}^T, {a_{y_1}}^T, {a_{x_2}}^T, {a_{y_2}}^T, \ldots, {a_{x_N}}^T, {a_{y_N}}^T]^T H = 0, \tag{8}$$

which is usually solved using random sampling technique [11] that iteratively minimizes the back-projection error, defined as:

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2, \tag{9}$$

where x_i, y_i and x_i', y_i' are the actual and estimated 2D pixel locations and $h_{11} \ldots h_{33}$ are the nine elements of the homography matrix. It is interesting to note that the homography computation time in this case is significantly smaller than that observed in the feature-less method because the linear system formed here has significantly lesser number of equations than the former method.

The homography computed using the above methods reflects the transformation parameter from one frame to other and are only relative to a pair of subsequent frames. In order to have an understanding of the global camera motion, it is desired to obtain the transformation parameters of all subsequent frames with respect to the initial frame in the sequence. Therefore, we need to perform a cumulative multiplication of the homography matrices. Thus, the relative homography between image frame I_0 and I_n is

$$H_{0,n} = H_{0,1} \times H_{1,2} \times H_{2,3} \times \cdots \times H_{n-1,n}, \tag{10}$$

where, corresponding sets of points $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ in homogenous coordinates, for two frames I_t and I_{t+1}, can be related by

$$\mathbf{x}_{t+1} \approx H_{t,t+1}\mathbf{x}_t. \tag{11}$$

Now, for each of the cumulative homography matrix computed as above, we measure the curl and deformation metrics [28], using the following equations:

$$\text{curl} = |h_{12} - h_{21}|, \tag{12}$$

$$\text{deformation} = |h_{11} - h_{22}|. \tag{13}$$

These metrics are an approximate measure of the change in camera viewpoint in terms of camera orientation and translation. If either of these metrics are larger

```
Procedure CompensateMotion ($(V, k)$)
Input: Video Sequence ($V$)
Input: Number of frames to invoke error correction ($k$)
Output: Array of Cumulative Homographies $H[]$
count ← 0;
$H[0] \leftarrow I$;
src ← init frame;
src = EqualizeHistogram (src);
while not End of Sequence do
    dst ← next frame in sequence;
    dst = EqualizeHistogram (dst);
    if count is multiple of k then
        surfsrc = computeSURF (src);
        surfdst = computeSURF (dst);
        [srckpts, dstkpts] ← findNearestNeighbor (surfsrc, surfdst);
    else
        kpts = findKLTFeatures (src);
        srcpts = computeSubpixel (kpts);
        dstpts = LKPyrOpticalFLow (src, dst, srcpts);
    h ← RANSACFitHomography (srckpts, dstkpts);
    $H[count+1] = h * H[count]$;
    curl ← computeCurl(H[count+1]);
    def ← computeDef(H[count+1]);
    if $curl > CURL\_THRES$ or $def > DEF\_THRES$ then
        $H[count+1] \leftarrow I$;
    result ← warpProjective (dst, H[count]);
    src ← dst;
    count ← count +1;
```

Algorithm 1 Pseudo-code describing the motion compensation algorithm used by COCOA-LIGHT on FLIR imagery with KLT features for establishing flow based correspondence and SURF used to regulate cumulative homography drift

than an empirical threshold, the consecutive frames indicate a significant change in view-point, and therefore a higher likelihood of erroneous alignment. Under these circumstances we reset the relative homography matrix to identity and frames from here on are treated as a new sub-sequence.

However, the cumulative homography computation as discussed is not robust to errors. Slight noise in the estimation of homography in one pair of frames can be easily propagated through the cumulative homography matrix resulting in errors that could affect the overall accuracy of the motion compensation, thereby causing errors in the object detection stage. In order to alleviate the effect of such erroneous calculations, we introduce a small error correction measure after every K frames, where the cumulative homography is replaced with homography estimated directly from descriptor mapping. This enhances the overall accuracy with the cost of a slight computation overhead. The results of applying motion compensation on an example three vehicle sequence are shown in Fig. 4. Each image in the figure is generated by allocating the first two channels of an RGB image matrix with reference frame

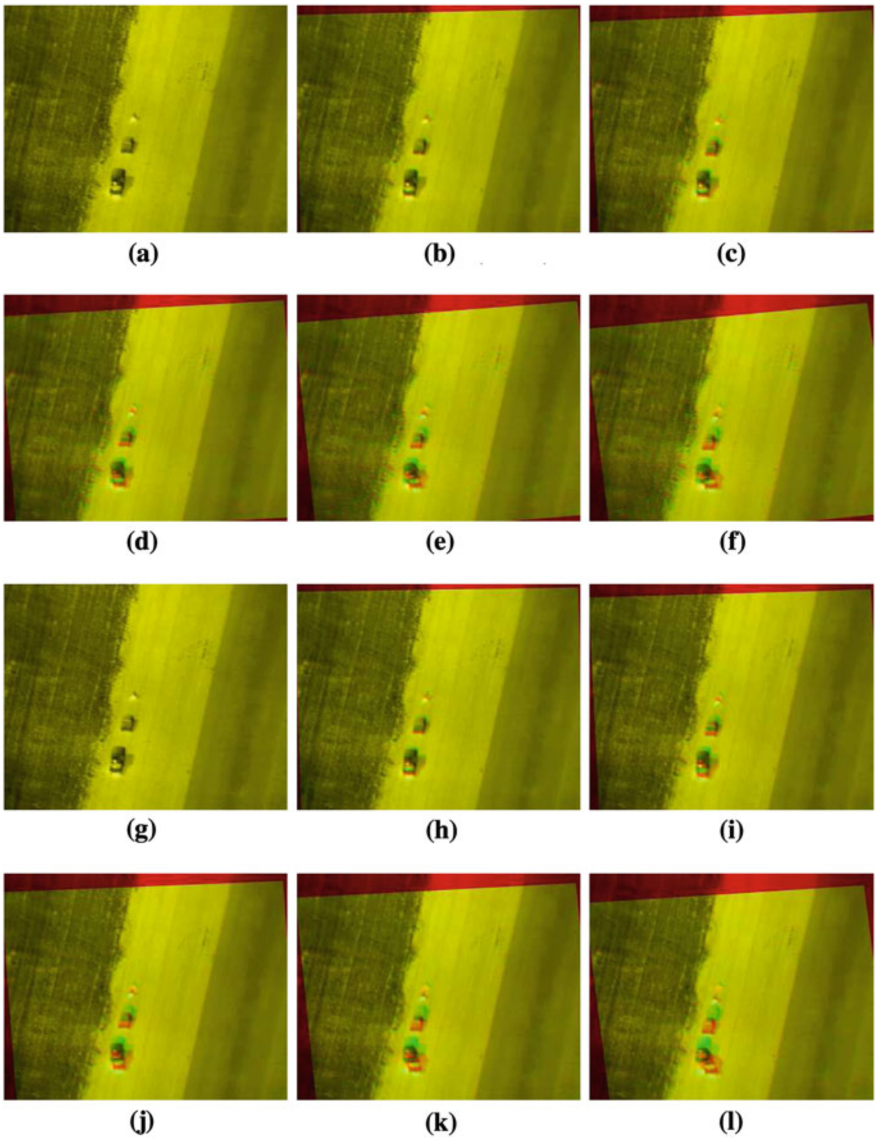


Fig. 4 Comparing alignment using cummulative homography computed using gradient based and KLT-feature based methods: images labeled **a**–**f** are aligned using the gradient feature based registration algorithm while images from **g**–**l** are aligned using the KLT-feature based algorithm. The real-valued number in parentheses corresponding to each image is its normalized frame difference scores obtained by subtracting aligned destination frame from the initial frame in the sequence. A smaller score indicates a better accuracy in alignment. **a** Frame 0 (0.0000), **b** Frame 0–10 (1.0110), **c** Frame 0–20 (1.1445), **d** Frame 0–30 (1.6321), **e** Frame 0–40 (1.9821), **f** Frame 0–50 (2.3324), **g** Frame 0 (0.0000), **h** Frame 0–10 (0.9121), **i** Frame 0–20 (1.1342), **j** Frame 0–30 (1.5662), **k** Frame 0–40 (1.8995), **l** Frame 0–50 (2.3432)

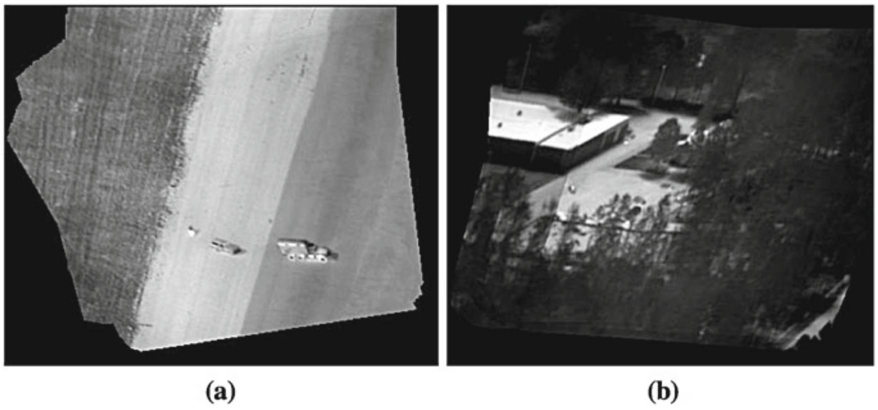


Fig. 5 Global mosaics generated after image alignment are shown for **a** the three vehicle sequence, and **b** the distant view sequence

and its subsequent aligned counterpart in grayscale, respectively. Regions that do not align properly are visible as green patches. Hence, in a set of correctly motion compensated frames, the green patches correspond to the moving objects as evident in Fig. 4. Global mosaics corresponding to the two different sequences discussed in this paper are shown in Fig. 5a and b. The complete motion compensation algorithm is listed in Algorithm 1.

With this knowledge, we proceed to our next section that discussed the methods we have employed to detect moving objects from a set of ego-motion compensated frames.

3.2 Object Detection

Given a sequence of frames, the goal of object detection is to obtain blobs for foreground objects. Background subtraction is a popular approach for static cameras where the background at each pixel can be modeled using mean, median, Gaussian, or a mixture of Gaussians. In aerial videos, background modeling is hindered due to camera motion. Although the aligned frames seem visually similar to a sequence of frames from a static camera, there are marked differences at the pixel level where errors in alignment cause small drifts in the pixel values. Such drifts are more pronounced near sharp edges. Furthermore, these drifts can be in different directions in different parts of the scene for each frame.

The most significant amongst the issues that pose challenge to background modeling in aerial videos are the errors due to parallax. Since we use features-based alignment, there are many features which come from out-of-plane objects such as buildings and trees. These features affect the computation of homography which is computed using all feature correspondences between a pair of consecutive frames.

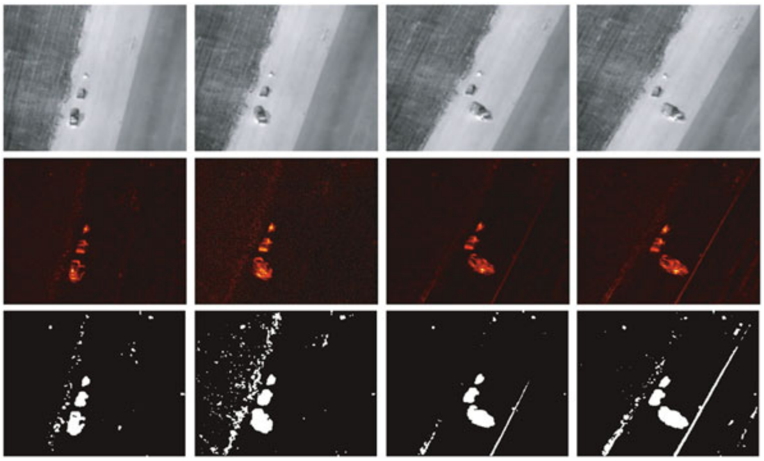


Fig. 6 The *first row* shows the original frames (10, 50, 100 and 150) from Sequence 1 while the *second* and *third rows* show accumulative frame difference (AFD) and AFD after thresholding respectively

This is the inherent source of error whose effect is visible near high gradients in a frame. Since all the homographies are computed between consecutive frames, the error in alignment accumulates with time. Even if we choose a small yet reasonable number of frames for constructing the background, the drift in the scene due to accumulated errors hampers the computation of background. (See discussion for Fig. 18).

Another reason is the limitation on the number of frames available for modeling the background. A region has to be visible for a reasonable number of frames to be learned as background. In the case of a moving camera, the field-of-view changes at every frame which puts restraints on the time available for learning. If the learning time is too short, some pixels from foreground are modeled as background. A constant change in field-of-view is also the reason that it is not possible to choose a single reference frame when performing alignment doing which can allow us to get rid of accumulated errors. After a few frames, the field-of-view of the new frame might not overlap with that of the reference frame and will thus disallow the computation of homography.

In addition to the two issues mentioned above, background modeling is computationally expensive for registered images which are usually greater in size than the original frames, and is thus prohibitive for longer sequences. In order to make foreground detection close to real time and cater for the non-availability of color information in FLIR imagery, we use a more feasible alternative of accumulative frame differencing (AFD), which takes as input only a neighborhood of n frames for detection at each time step (Fig. 6).

For each frame, the algorithm is initialized using a constant number of frames called temporal window of size of $2n + 1$ with n frames on both sides of the

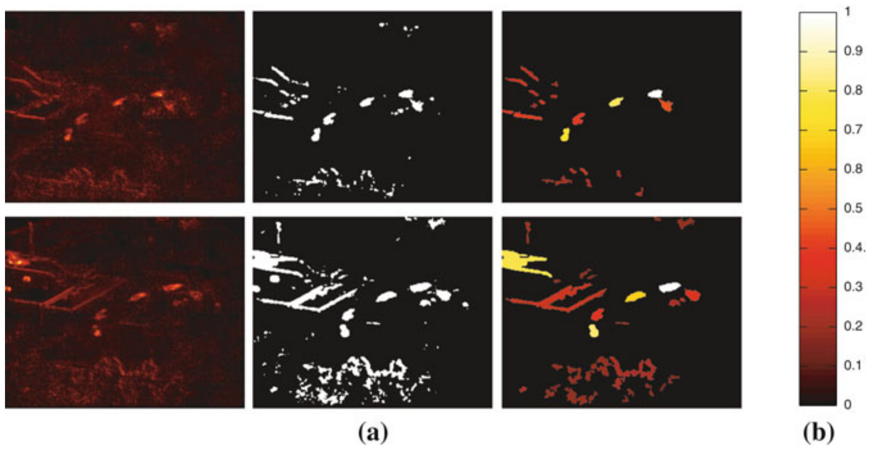


Fig. 7 **a** The *first column* shows AFD for two frames from Distant view Sequence whereas *second column* shows AFD after thresholding. As can be seen from the *third column*, mean gray area (normalized between 0 and 1) of blobs corresponding to moving objects is high which can be used to separate moving objects from noisy blobs. **b** shows the *gray-map* used for all the figures in this section

current frame. This means the detection procedure will have a lag of n frames. The accumulative frame difference for ith frame (I_i) for temporal window from $-n$ to n is given by

$$\mathrm{AFD}(I_i, n) = \sum_{k=i-n}^{i+n} |I_i - W(I_i, I_k)|, \tag{14}$$

where $W(I_i, I_k)$ is a function to warp kth frame to the ith frame.

We experimented with different size of temporal window with the conclusion that $n = 10$ is empirically the most suitable value. If n is close to 2 , the blobs are small, incomplete and missing. If we go beyond 10, the blobs start to merge and sharp edges of the background begin to appear as false positives.

The grayscale image obtained after accumulative frame differencing is normalized between 0 and 1 followed by thresholding (with discardThreshold T). Blobs are obtained using connected-component labeling. Since pixels belonging to moving objects have higher values in accumulative frame difference than noise (see Fig. 7), mean gray area of such blobs is correspondingly high. Moreover, it can be observed that blobs corresponding to moving objects are compact and regular in shape when compared against irregular shaped blobs due to noise (see Fig. 8). However, an exception to this are the noisy blobs that come from regions of high gradients some of which might not be irregular in shape. Instead, they have a prominent characteristic of being elongated with higher eccentricity. Figure 9 explains the use of eccentricity as a measure to cater for such blobs.

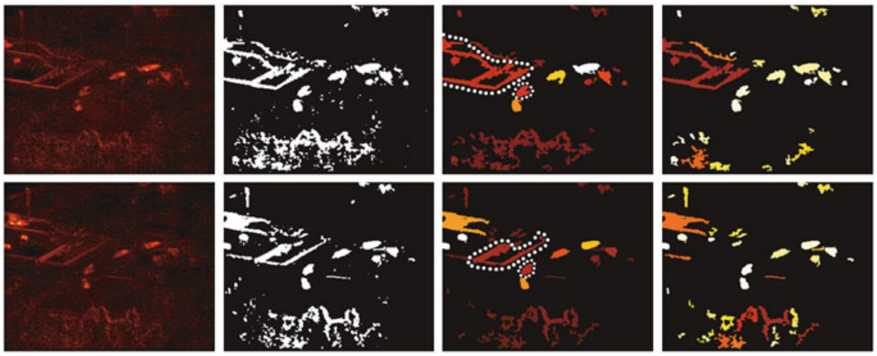


Fig. 8 This figure illustrates the advantage of using compactness for removing false positives. From *left* to *right*: AFD, AFD > T, MGA, and compactness. In the *third column*, notice that both the highlighted irregular shaped blobs due to parallax error and the nearby moving object have similar MGA. However, blobs due to moving objects are more compact (*fourth column*) and will therefore get higher weight

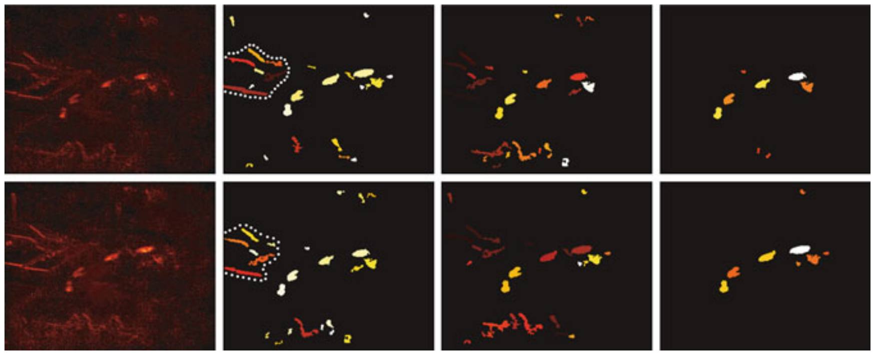


Fig. 9 From *left* to *right*: AFD, compactness, eccentricity and weights of final blobs. The highlighted elongated blobs due to noise do not get suppressed using compactness in the *second column* but do get lower eccentricity weight as shown in the *third column*

We will now give definitions for the three measures. If $b_t^i \in \mathbf{B}_t$ denotes the ith blob at frame t, then its mean gray area, compactness and eccentricity are computed using the following formula:

$$\text{Mean Gray Area}^i = \frac{\sum_{\forall p(x,y)\in b_t^i} \text{AFD}(x,y)}{|b_t^i|}, \tag{15}$$

$$\text{Compactness}^i = \frac{|P(b_t^i)|}{2\pi\sqrt{|b_t^i|/\pi}}, \tag{16}$$

$$\text{Eccentricity}^i = \sqrt{\frac{2C_{xy}}{u_{xx} + u_{yy} + C_{xy}}}, \tag{17}$$

where P gives perimeter of the blob. u_{xx}, u_{yy} and C_{xy} are given by

$$u_{xx} = \frac{\sum_{\forall p(x,y)\in b_t^i} (x - \bar{x})^2}{|b_t^i|} + \frac{1}{12}, \quad u_{yy} = \frac{\sum_{\forall p(x,y)\in b_t^i} (y - \bar{y})^2}{|b_t^i|} + \frac{1}{12} \tag{18}$$

$$C_{xy} = \sqrt{(u_{xx} - u_{yy})^2 + 4u_{xy}} \quad \text{where } u_{xy} = \frac{\sum_{\forall p(x,y)\in b_t^i} (x - \bar{x})(y - \bar{y})}{|b_t^i|} \tag{19}$$

where 1/12 is the normalized second central moment of a pixel with unit length.

The following equation describes the scheme to combine weights from mean gray area, compactness and eccentricity:

$$W^i = \alpha_1 \times \text{MGA}^i + \alpha_2 \times (2 - \text{Compactness}^i) + \alpha_3 \times (1 - \text{Eccentricity}^i), \tag{20}$$

where α_1, α_2, and α_3 are empirically determined constants with relatively higher weight given to MGA. The blobs are sorted according to their weights W^i and normalized between 0 and 1 and only min(maxObjects, $|b_t^i| \; |W^i > T$) are returned where maxObjects is a hard limit on the maximum number of output objects. The reason AFD and W^i are normalized by their respective maximum values is to keep T constant across different sequences. The empirical value of discardThreshold T is .005 or .5% of maximum value. If T is too low for some frame, it can cause blobs from moving objects to merge with those from the noise (see Fig. 10). Since pixels from high motion objects will have higher values in AFD, all such pixels should be output as foreground. If the detection procedure discards pixels that should have been included in output, T is progressively increased till all high motion objects are included in the output.

Though the proposed approach gives reasonable results across a wide variety of sequences without changing any weights and the threshold T, information regarding minimum and maximum blob size can be incorporated in Eq. 20 to fine tune the results for a particular configuration of camera altitude and scene clutter. Figure 19 provides intermediate for the detection in three frames from Distant View Sequence.

We evaluate the performance of our detection algorithm, using Multiple Object Detection Precision (MODP) [6] scores in addition to the standard Probability of Detection (PD) and False Alarm Rate (FAR) metrics from Automatic Target Recognition literature [23]. The MODP is calculated on a per frame basis and is given as:

$$\text{MODP}_t = \frac{1}{N_t} \sum_{i=1}^{N_t} \frac{|G_t^i \cap B_t^i|}{|G_t^i \cup B_t^i|}, \tag{21}$$

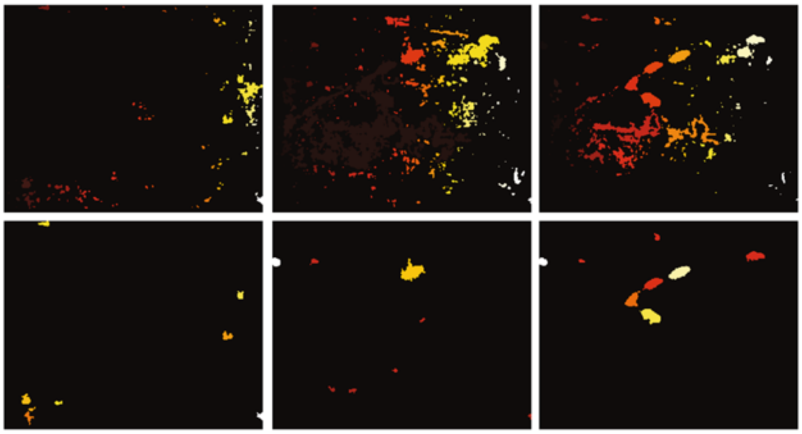


Fig. 10 Progressive thresholding: *top row* shows the connected component labels obtained with discardThreshold = .5%, .8% and 1%. The invisible *top-left* region corresponds to blobs with smaller labels (close to zero). *Bottom row* depicts the corresponding detections. discardThreshold is progressively increased from .5% to 1% till the objects and noise form separate blobs

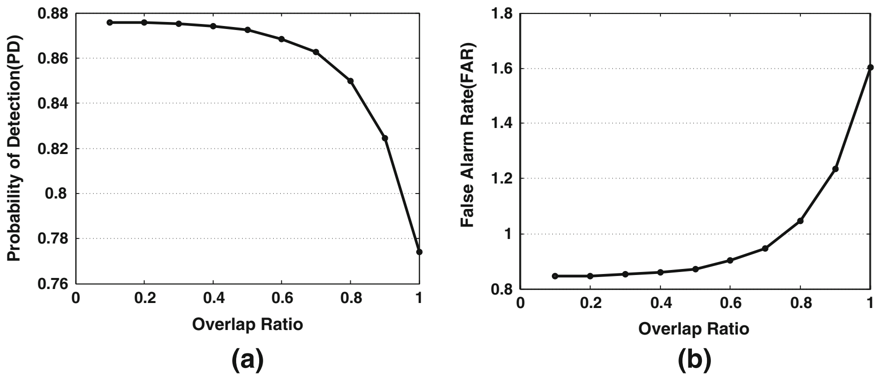


Fig. 11 Detection evaluation obtained on Distant View Sequence with varying overlap ratio from 0.1 to 1. **a** Probability of detection scores, **b** false alarm rate

where B_t and G_t are the respective set of corresponding objects output by the detection stage and that present in Ground Truth, at frame t, N_t being the cardinality of the correspondence. The fractional term in Eq. 21 is also known as the spatial overlap ratio between a corresponding pair of bounding boxes of ground-truthed and detected objects. Figure 11 reports the PD and FAR scores for Distant View Sequence, obtained by varying the bounding box overlap ratio.

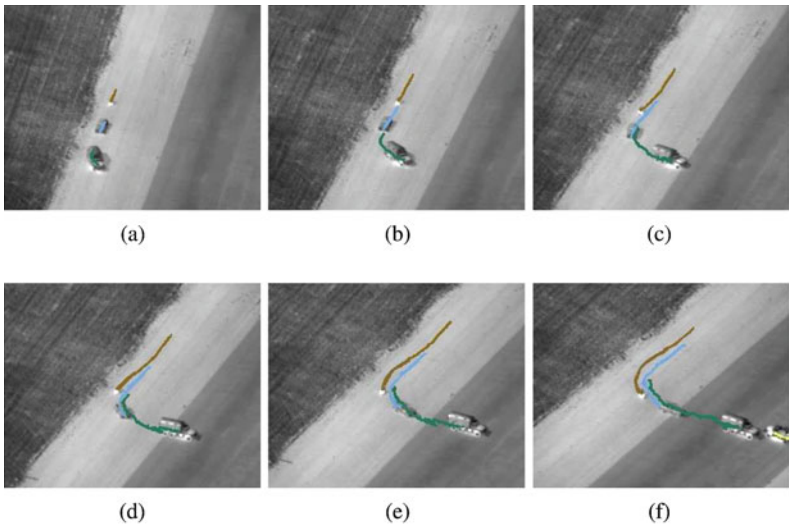


Fig. 12 Tracking results for three vehicle sequence. Tracks of multiple objects are overlaid on every 50th frame. All three visible objects are tracked correctly for the duration of the sequence. A fourth object just entering the camera's field of view is visible in frame 300. **a** Frame 50, **b** Frame 100, **c** Frame 150, **d** Frame 200, **e** Frame 250, **f** Frame 300

3.3 Tracking

The process of object detection provides a set of unique labels assigned to mutually exclusive groups of pixels for each image, where each label ideally corresponds to a single moving object. Given that the set of observed objects is denoted by $\mathbf{B}_t = \{b^i\}$, where $1 \leq i \leq O_t$, and O_t is the number of objects detected in frame t, the problem of tracking is defined as computation of a set of correspondences that establishes a 1–1 relationship between $b^i \in \mathbf{B}_t$ for all i, with an object $b^j \in \mathbf{B}_{t+1}$. In addition to problems like occlusions, non-linear motion dynamics, and clutter, that are traditionally encountered in object tracking in static, surveillance cameras, tracking in aerial FLIR imagery is made much harder because of low image resolution and contrast, small object sizes, and artifacts introduced in images during the platform motion compensation phase. Even small errors in image stabilization can result in a significant number of spurious object detections, especially in regions with high intensity gradients, further complicating the computation of the optimal object correspondence across frames (Fig. 12).

3.3.1 Kinematic Constraint

Our tracking algorithm employs a constant velocity motion model, and various cues for object correspondence, including appearance, shape and size. Furthermore, due to severe splitting and merging of objects owing to potential errors in detection, as well as apparent entry and exit events due to object to object, and object to background occlusions, our tracking method handles blob splitting and merging, and occlusions explicitly. At any given frame t, the state X_t^i of an object $b^i \in \mathbf{B}_t$ being tracked, can be represented by its location and motion history. We write the state as,

$$X_t^i = [x_t^i, y_t^i, \rho_t^i, \theta_t^i], \tag{22}$$

where (x^i, y^i) represents the 2d location of the object on the image plane at time (frame) t, and (ρ^i, θ^i) are the magnitude and orientation of the mean velocity vector of the object. The state vector for object i at frame $t+1$, X_{t+1}^i is predicted as follows:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} \rho_t \cos\theta_t \\ \rho_t \sin\theta_t \end{bmatrix} + \begin{bmatrix} \gamma_x \\ \gamma_y \end{bmatrix}, \tag{23}$$

where (γ_x, γ_y) depict Gaussian process noise with zero mean and standard deviations σ_x and σ_y in x and y directions, which are derived from the variation in (ρ, θ) over time (the correlation is assumed to be zero). Assuming the magnitude and orientation of the velocity vector between an object's location at time frame t and $t-1$ to be $\hat{\rho}_t$ and $\hat{\theta}_t$ respectively, the velocity history in the state vector is updated by computing the weighted means of object's velocity magnitude and orientation in the current and previous frames, i.e., ρ_t and $\hat{\rho}_t$. The orientation of the object's velocity is similarly updated, by phase change invariant addition, subtraction and mean functions.

The motion model based probability of observing a particular object with state X_t^i in frame t, as object $b^j \in \mathbf{B}_{t+1}$ with centroid (x_{t+1}^j, y_{t+1}^j) in frame $t+1$ can then be written as

$$P_m(b_{t+1}^j | X_t^i) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left\{ -\frac{1}{2} \left[\frac{(x_{t+1}^i - x_{t+1}^j)^2}{\sigma_x^2} + \frac{(y_{t+1}^i - y_{t+1}^j)^2}{\sigma_y^2} \right] \right\}. \tag{24}$$

Notice that we can compute (x_{t+1}^i, y_{t+1}^i) from the constant velocity motion model as described before.

3.3.2 Observation Likelihood

In addition to the motion model described above, the key constituent of correspondence likelihood between two observation in consecutive frames is the observation model. Various measurements can be made from the scene to be employed for use in observation model, which combined with the kinematics based prediction defines the cost of association between two object detections. As described earlier, we used appearance, shape and size of objects as measurements. These observations for an object denoted by b^i are denoted by δ_c^i, δ_g^i, δ_s^i, and δ_a^i, for intensity histogram, mean gray area (from frame difference), shape of blob, and pixel area of the blob respectively. The probability of association between two blobs using these characteristics can then be computed as follows. $P_c(b_{t+1}^j|X_t^i)$ denotes the histogram intersection between histograms of pixels in object's bounding box in the previous frame, and the detection under consideration, i.e., b_{t+1}^j.

The probability $P_g(b_{t+1}^j|X_t^i)$ can simply be computed using the difference in the mean gray values of each blob after frame differencing, normalized by maximum difference possible. The shape based likelihood is computed by aligning the centroids of blobs b_t^i and b_{t+1}^j, and computing the ratio of blob intersection and blob union cardinalities, and is represented by $P_s(b_{t+1}^j|X_t^i)$. Finally the pixel areas for the blobs can be compared directly using the variance in an object's area over time which is denoted by σ_a^i. The probability of size similarity is then written as, $P_a(b_{t+1}^j|X_t^i) = \mathcal{N}(\delta_a^j|\delta_a^i, \sigma_a^i)$, where $\mathcal{N}$ represents the Normal distribution.

Assuming the mutual independence of motion, appearance, shape and size, we can write the probability of a specific next object state (the blob detection b_{t+1}^j), given all the observations, to be,

$$\begin{aligned}P(b_{t+1}^j|X_t^i, \delta_c^i, \delta_g^i, \delta_s^i, \delta_a^i)X = {} & P_m(b_{t+1}^j|X_t^i)P_c(b_{t+1}^j|X_t^i)P_g(b_{t+1}^j|X_t^i)\\ & \times P_s(b_{t+1}^j|X_t^i)P_a(b_{t+1}^j|X_t^i),\end{aligned} \tag{25}$$

which gives the aggregate likelihood of correspondence between the blob $b_t^i \in \mathbf{B}_t$ in frame t represented by state X_t^i, and the blob $b_{t+1}^j \in \mathbf{B}_{t+1}$ in frame $t + 1$.

3.3.3 Occlusion Handling

Tracking in traditional surveillance scenarios and especially in aerial FLIR imagery suffers from the problems of severe object to object and object to background occlusions. Furthermore, the low resolution and low contrast of these videos often induce high similarity between objects of interest and their background, thus resulting in mis-detections. Consequently, a simple tracker is likely to initialize a new track for an object undergoing occlusion every time it reappears. To overcome this problem, our tracking algorithm continues the track of occluded object by adding hypothetical points to the track using its motion history. In actuality, the track of every object in the current frame, that does not find a suitable correspondence in the next frame, within an ellipse defined by five times the standard deviations σ_x and σ_y, is propagated using this method. In particular, it is assumed that the occluded object will maintain persistence of appearance, and thus have the same intensity histogram, size, and shape. Obviously, according to the aggregate correspondence likelihood, such a hypothetical object will have nearly a 100% chance of association. It should be noted however that an implicit penalty is associated with such occlusion reasoning that arises from the probability term $P_g(\cdot)$, which in fact can be computed regardless of detection. In other words, the mean gray area of the hypothetical blob (deduced using motion history) is computed for the frame in question, which reduces the overall likelihood of association as compared to an actual detected blob which would have a relatively low likelihood otherwise. This aggregate probability is denoted by $P_o(b^{\hat{k}}_{t+1}|X^k_t)$, where $b^{\hat{i}}_{t+1}$ is the hypothetical blob in frame $t+1$, resulting from motion history based propagation of the blob b^i_t described by the state vector X^i_t. The track of an object that has exited the camera view can be discontinued by either explicitly testing for boundary conditions, or by stopping track propagation after a fixed number of frames.

3.3.4 Data Association

Given blobs in consecutive frames t and $t+1$ as $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$, their state and measurement vectors, probability of association between every possible pair of blobs is computed. The goal of the tracking module then is to establish 1–1 correspondence between the elements of the sets $\mathbf{B}_t$ and $\mathbf{B}_{t+1}$. Numerous data association techniques have been proposed in the computer vision literature, including methods for single, few, or a large number of moving targets. Many of these methods (e.g., bipartite graph matching) explicitly enforce the 1–1 correspondence constraint, which may not be ideal in the FLIR sequences scenario, since a non-negligible number of false positive and false negative detections can be expected.

We, therefore, employ an object centric local association approach, rather than a global association likelihood maximization. This technique amounts to finding the nearest measurement for every existing track, where ‘nearest’ is defined in the observation and motion likelihood spaces (not the image space). This approach is also known as the greedy nearest neighbor (GNN) data association [7]. Formally, for the trajectory i, containing the measurement $b^i_t \in \mathbf{B}_t$, described by the current state X^i_t, the next associated measurement can be computed as

$$b^i_{t+1} = \underset{j \in [1, O_{t+1}]}{\operatorname{argmax}} \, P(b^j_{t+1}|X^i_t, \delta^i_c, \delta^i_g, \delta^i_s, \delta^i_a). \tag{26}$$

The objects in the set $\mathbf{B}_{t+1}$, that are not associated with any existing track can be initialized as new trajectories, while existing tracks not able to find a suitable correspondence are associated with a hypothetical measurement as described earlier. If a track cannot find real measurements after addition of a predetermined number of hypothetical blobs, the track is discontinued.

The performance of the tracking algorithm discussed here is evaluated using a metric similar to the one shown in Eq. 21. Multiple Object Tracking Precision (MOTP) is given by

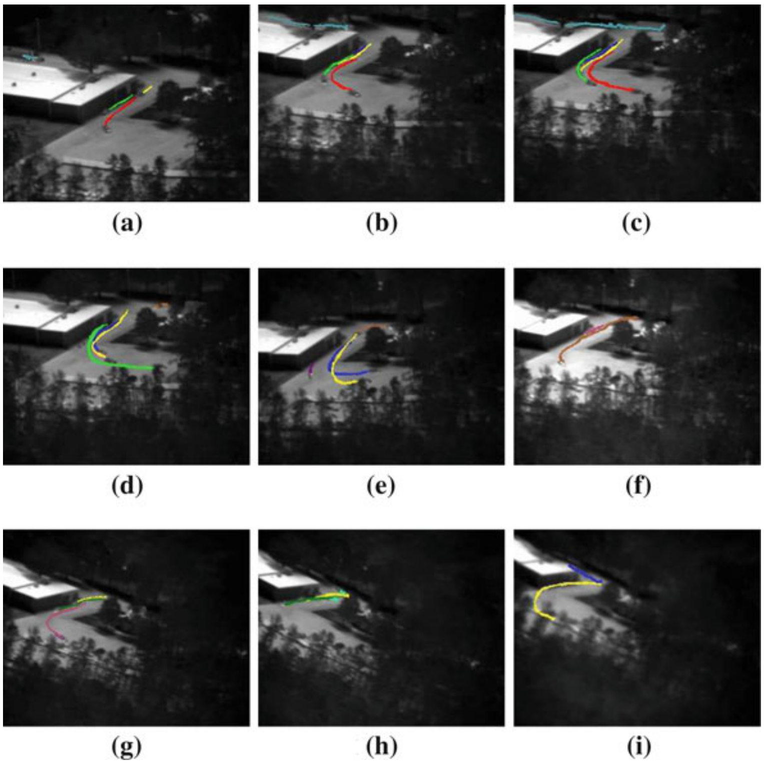


Fig. 13 Tracking of vehicles in distant field of view. Tracks of multiple objects are overlaid on frames of the sequence at regular intervals. The same *gray-scale* indicates consistent labeling of the object. Most of the objects are tracked throughout their observation in the camera's field of view. Notice the low resolution and contrast. **a** Frame 56, **b** Frame 139, **c** Frame 223, **d** Frame 272, **e** Frame 356, **f** Frame 422, **g** 500, **h** 561, **i** 662

$$\text{MOTP}_t = \frac{\sum_{i=1}^{N_t} \sum_{t=1}^{N_f} \left[\frac{|G_t^i \cap B_t^i|}{|G_t^i \cup B_t^i|} \right]}{\sum_{j=1}^{N_t} N_t^j} \tag{27}$$

where N_t refers to the mapped objects over an entire trajectory as opposed to a single frame. The MOTP scores for a subset of 12 sequences are shown in Fig. 13, in the following section.

4 Discussion

In this section we provide an in-depth analysis of the various algorithms that are used in cocoalight in terms of their individual performance followed by an overall execution summary of the system. All the following experiments are conducted on a

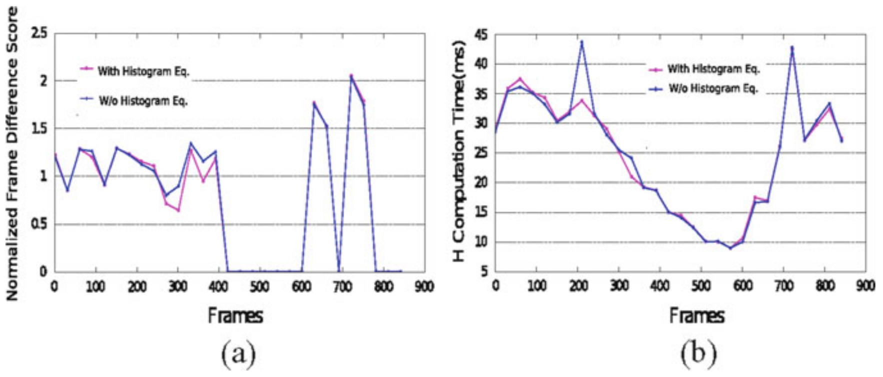


Fig. 14 Effect of histogram equalization on the accuracy of alignment and computation time. **a** Accuracy achieved in alignment after histogram equalization three vehicle sequence. The results shown here indicate that histogram equalization is beneficial for feature extraction in FLIR imagery. **b** Although the histogram equalization stage increases some computation overhead, overall we notice negligible change in alignment speed as with more number of KLT features extracted, the homography estimation routine takes fewer RANSAC iterations to generate optimal solution

desktop computing environment with a 1.6 GHz Intel x86 dual core CPU and 2 GB physical memory. The two sequences containing vehicular traffic, shown earlier in this paper are acquired from the VIVID 3 dataset. In addition, we use a more challenging AP-HILL dataset, containing both pedestrian and vehicular traffic, acquired by Electro-optic and FLIR cameras, to test our system.

A quantitative improvement in alignment accuracy and computational performance due to contrast enhancement is shown in Fig. 14. It can be noted that the total frame difference per frame is reduced after alignment using histogram equalization, due to an increased number of relevant feature points in regions of previously low contrast. On the other hand, this process is not a computational burden on the system, and in some cases can even improve the transformation computation time. In Fig. 15, we analyze the drift or error in estimation that is introduced in the cumulative homography computation stage. For the sake of simplicity, we only show the results corresponding to the parameters that only determine translation across frames in a sequence. We observe that curves corresponding to either parameters, have similar slopes which indicates that the proposed algorithm 1 achieves results closer to the gradient based method. It is worthwhile to note that our algorithm is more robust to change in background than the gradient based method as it has lesser number of homography reset points (where the curves touch the *x*-axis).

Figure 16 summarizes the impact of increasing the number of KLT features in the motion compensation stage. As the number of features are increased, we observe a drop in the computation speed in Fig. 16b. The accuracy in alignment, which is measured in terms of normalized frame difference scores, however shows marginal improvement beyond 512 features. In a slightly different setting, we evaluate different types of feature extraction strategies against the gradient based method. In

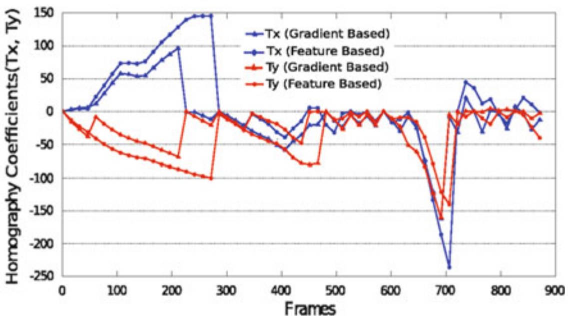


Fig. 15 Comparing homography parameters estimated using KLT feature based method against the gradient-based method. Parameters corresponding to the translation along *x* and *y* axes, represented by *curves of different gray-scale values*. It is interesting to observe the frame locations along *x*-axis where the parameter curves touch the *x*-axis. These locations indicate the positions when the homography is reset to identity because of large frame motion

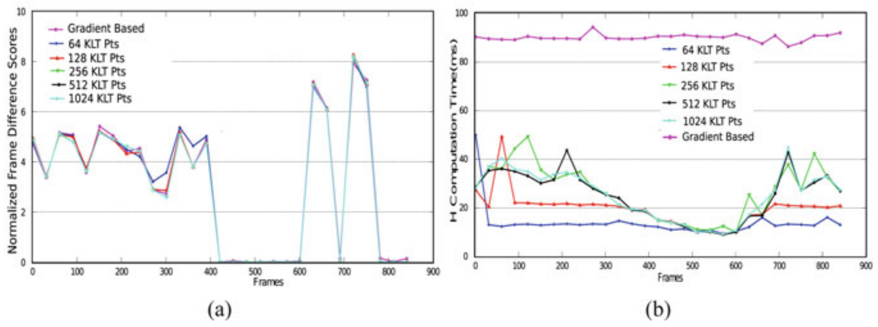
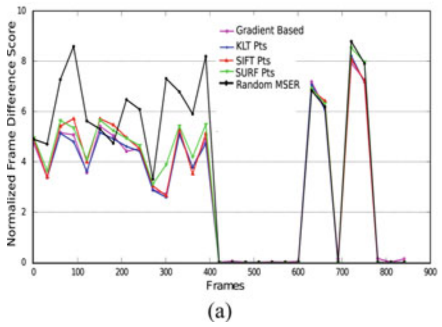


Fig. 16 Effect of increasing KLT features on alignment: **a** accuracy achieved in alignment on three vehicle sequence with different number of KLT features. As number of features are increased, the alignment accuracy reaches close to that achieved using gradient based method. **b** Computation time of homography is maximum with gradient based method and reduces significantly with decrease in number of KLT features

Fig. 17a, we notice that both KLT and SIFT feature based methods achieve accuracies comparable to the gradient based scheme with the KLT feature based method being twice as computationally efficient as the gradient and SIFT feature based methods.

The alignment algorithm used by Cocoalight makes a planarity assumption on the input scene sequences. This implies that pixels from ground plane, that contribute to the linear system of equations for computing homography, should outnumber those from outside the ground plane. If this criterion is not satisfied, homography between two frames cannot be computed accurately. This is usually observed in typical urban scenarios that consist of tall buildings imaged by low flying UAVs. We demonstrate this issue in Fig. 18. The alignment error is largely visible as we proceed towards the end of the sequence in Fig. 18c.



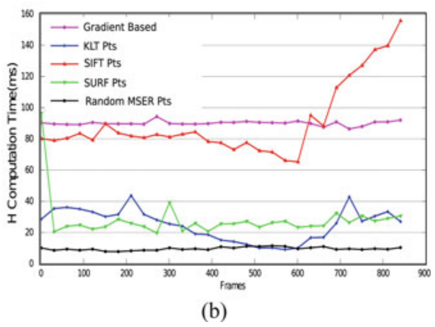


Fig. 17 Effect of different types of features on alignment: **a** accuracy achieved with different feature extraction algorithms (KLT, SIFT, SURF, MSER) in comparison to the gradient based method, and **b** their respective homography computation time

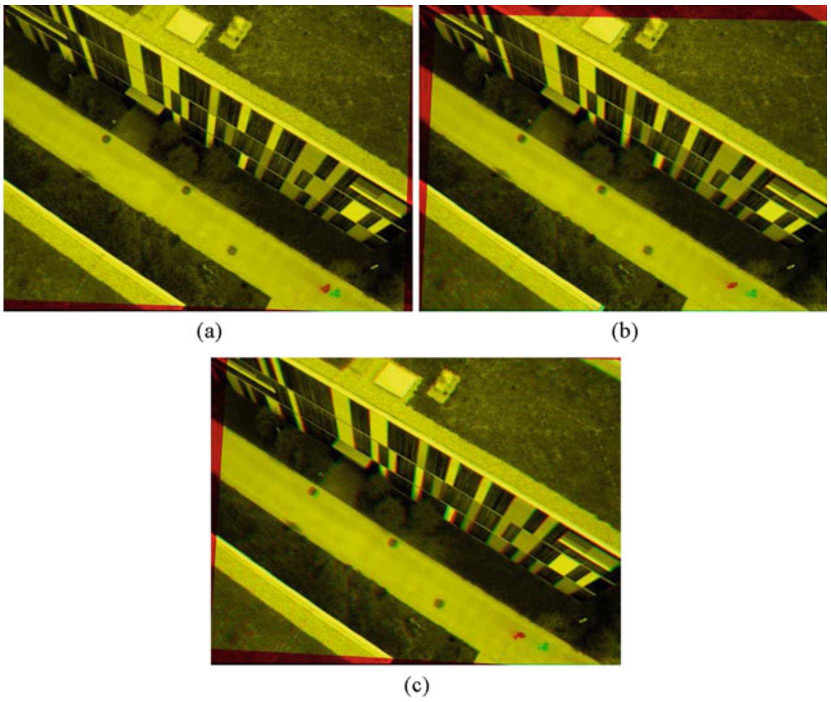


Fig. 18 Erroneous alignment due to pixels outside the ground plane contributing in homography estimation. **c** *Green* visible patches near the circular drainage holes did not align properly. **a** Frame 0/20, **b** Frame 0/40, **c** Frame 0/60

In Table 1, we report the performance of our detection and tracking setup against different evaluation metrics, namely PD, MODP, MOTP and FAR for a subset of 12 sequences from our datasets. These sequences are characterized by

Table 1 Quantitative evaluation of runtime for individual modules, namely Motion compensation (alignment), ROI detection and Tracking for 12 FLIR aerial sequences from the AP-HILL dataset containing moving vehicles and human beings

Sequence	Frames	Alignment	Detection	Tracking	FDA	PD	FAR	MOTP	MOTA
Seq. 01	742	23.3	8.3	36.1	4.89	0.81	0.12	0.67	0.74
Seq. 02	994	21.6	7.9	39.6	6.77	0.89	0.08	0.69	0.71
Seq. 03	1138	24.0	6.1	38.1	10.89	0.88	0.09	0.65	0.76
Seq. 04	1165	22.2	6.5	40.6	11.32	0.78	0.05	0.69	0.81
Seq. 05	1240	24.3	9.4	40.2	4.22	0.83	0.13	0.75	0.82
Seq. 06	1437	25.1	6.2	41.0	7.95	0.91	0.06	0.63	0.69
Seq. 07	1522	21.4	8.3	36.7	6.83	0.87	0.04	0.61	0.78
Seq. 08	1598	25.6	7.9	38.2	5.39	0.76	0.06	0.64	0.75
Seq. 09	1671	24.8	6.1	36.1	7.94	0.73	0.11	0.61	0.74
Seq. 10	1884	22.8	6.1	42.1	8.83	0.75	0.09	0.59	0.78
Seq. 11	1892	23.6	6.7	39.4	12.56	0.82	0.12	0.66	0.69
Seq. 12	1902	21.7	8.4	41.5	10.21	0.89	0.06	0.72	0.73

Each video sequence has a spatial resolution of 320×240 and are arranged in ascending order of the number of frames contained in them for better readability. The Frame Difference Score averaged over the total number of frames in a given sequence serves as the performance metric for the alignment module. Probability of Detection (PD) and False Alarm Rate (FAR) measures provide vital insights on the performance of the detection module. Finally, Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA) scores are presented for each of these sequences to measure the performance of the Tracking module

the following: (a) small and large camera motion, (b) near and distant field of views, (c) varying object sizes (person, motorbike, cars, pick-ups, trucks and tanks), (d) background clutter.

Some qualitative tracking results for near and far field sequences are shown in Figs. 12 and 13 respectively. Object tracks are represented as trajectories, which are lines connecting the centroids of blobs belonging to the object in all frames. The same color of a track depicts consistent labeling and thus correct tracks. Notice the extremely small object sizes and the low contrast relative to the background. Background subtraction based methods fail in such scenarios where the lack of intensity difference between object and background result in a large number of false negatives (Fig. 19).

5 Conclusion

The chapter has presented a detailed analysis of the various steps in the aerial video tracking pipeline. In addition to providing an overview of the related work in the vision literature, it lists the major challenges associated with tracking in aerial videos, as opposed to static camera sequences, and elaborates as to why the majority of algorithms proposed for static camera scenarios are not directly applicable to the

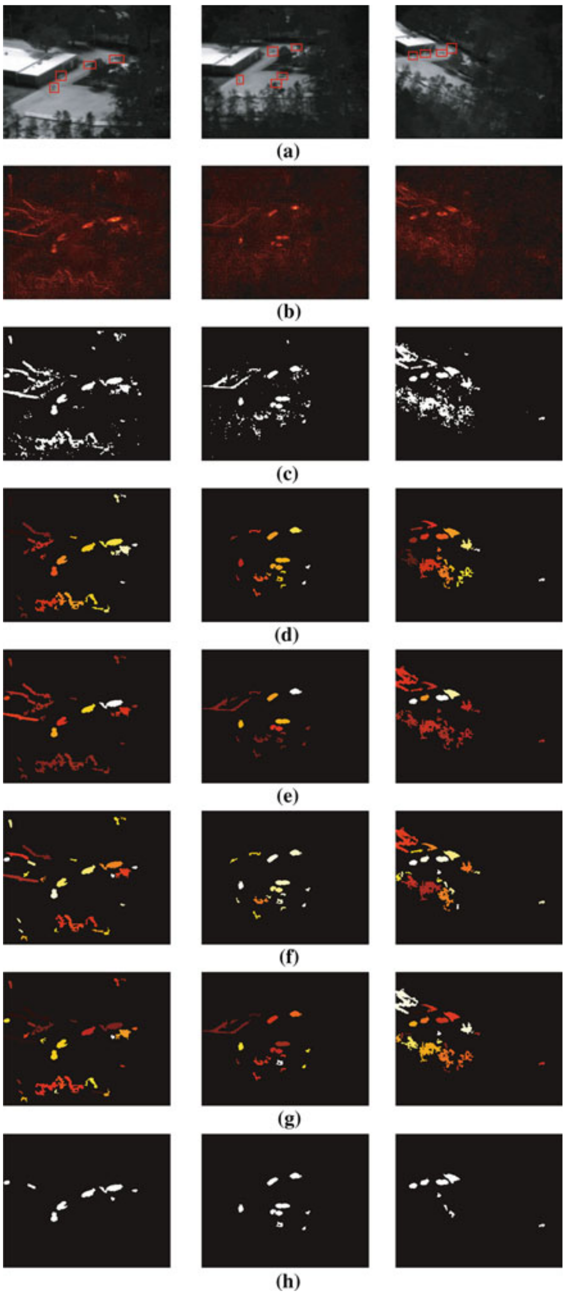


Fig. 19 Intermediate results for three frames from Distance View Sequence. *Bounding rectangles* in the original frames show the positions of groundtruth. **a** Original frames, **b** accumulative frame difference, **c** AFD > T, **d** connected components (30, 17 and 23), **e** mean gray area, **f** compactness, **g** eccentricity, **h** output blobs

aerial video domain. We have presented both the theoretical and practical aspects of a tracking system, that has been validated using a variety of infrared sequences.

References

1. Ali, S., Shah, M.: Cocoa—tracking in aerial imagery. In: SPIE Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications (2006)
2. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: CVPR (2008)
3. Arambel, P., Antone, M., Landau, R.H.: A multiple-hypothesis tracking of multiple ground targets from aerial video with dynamic sensor control. In: Proceedings of SPIE, Signal Processing, Sensor Fusion, and Target Recognition XIII, vol. 5429, pp. 23–32 (2004)
4. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: speeded up robust features. In: ECCV (2006)
5. Berclaz, J., Fleuret, F., Fua, P.: Robust people tracking with global trajectory optimization. In: CVPR (2006)
6. Bernardin, K., Elbs, A., Stiefelhagen, R.: Multiple object tracking performance metrics and evaluation in a smart room environment (2006)
7. Blackman, S., Popoli, R.: Design and Analysis of Modern Tracking Systems. Artech House, Boston (1999)
8. Bouguet, J.: Pyramidal implementation of the Lucas–Kanade feature tracker: description of the algorithm. TR, Intel Microprocessor Research Labs (2000)
9. Brown, L.G.: A survey of image registration techniques. ACM Comput. Surv. **24**(4), 325–376 (1992)
10. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
11. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
12. Gandhi, T., Devadiga, S., Kasturi, R., Camps, O.: Detection of obstacles on runway using ego-motion compensation and tracking of significant features. In: Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision, p. 168
13. Heitz, G., Koller, D.: Learning spatial context: using stuff to find things. In: ECCV (2008)
14. Isard, M., Blake, A.: Condensation: conditional density propagation for visual tracking. In: IJCV (1998)
15. Jepson, A., Fleet, D., El-Maraghi, T.: Robust online appearance models for visual tracking. In: IEEE TPAMI (2003)
16. Kumar, R., Sawhney, H., Samarasekera, S., Hsu, S., Tao, H., Guo, Y., Hanna, K., Pope, A., Wildes, R., Hirvonen, D., Hansen, M., Burt, P.: Aerial video surveillance and exploitation. IEEE Proc. **89**, 1518–1539 (2001)
17. Leibe, B., Schindler, K., Gool, L.V.: Coupled detection and trajectory estimation for multi-object tracking. In: ICCV (2007)
18. Lin, R., Cao, X., Xu, Y., Wu, C., Qiao, H.: Airborne moving vehicle detection for video surveillance of urban traffic. In: IEEE Intelligent Vehicles Symposium, pp. 203–208 (2009)
19. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
20. Mann, S., Picard, R.W.: Video orbits of the projective group: a simple approach to featureless estimation of parameters. IEEE Trans. Image Process. **6**, 1281–1295 (1997)
21. Matas, J., Chum, O., Martin, U., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: BMVC (2002)
22. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (2009)

23. Olson, C.F., Huttenlocher, D.P.: Automatic target recognition by matching oriented edge pixels. IEEE Trans. Image Process. **6**, 103–113 (1997)
24. Perera, A., Srinivas, C., Hoogs, A., Brooksby, G., Hu, W.: Multi-object tracking through simultaneous long occlusions and split–merge conditions. In: CVPR (2006)
25. Piccardi, M.: Background subtraction techniques: a review. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 3099–3104 (2004)
26. Shah, M., Kumar, R.: Video Registration. Kluwer Academic Publishers, Dordrecht (2003)
27. Shi, J., Tomasi, C.: Good features to track. In: CVPR, pp. 593–600 (1994)
28. Spencer, L., Shah, M.: Temporal synchronization from camera motion. In: ACCV (2004)
29. Xiao, J., Cheng, H., Han, F., Sawhney, H.: Geo-spatial aerial video processing for scene understanding. In: CVPR (2008)
30. Xiao, J., Yang, C., Han, F., Cheng, H.: Vehicle and person tracking in aerial videos. In: Multimodal Technologies for Perception of Humans: International Evaluation Workshops CLEAR 2007 and RT 2007, pp. 203–214 (2008)
31. Yalcin, H., Collins, R., Black, M., Hebert, M.: A flow-based approach to vehicle detection and background mosaicking in airborne video. In: CVPR, p. 1202 (2005)
32. Yalcin, H., Collins, R., Hebert, M.: Background estimation under rapid gain change in thermal imagery. In: OTCBVS (2005)
33. Yilmaz, A.: Target tracking in airborne forward looking infrared imagery. Image Vis. Comput. **21**(7), 623–635 (2003)
34. Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. **38**(4), 1–45 (2006)
35. Yilmaz, A., Shafique, K., Lobo, N., Li, X., Olson, T., Shah, M.A.: Target-tracking in flir imagery using mean-shift and global motion compensation. In: Workshop on Computer Vision Beyond the Visible Spectrum, pp. 54–58 (2001)
36. Yin, Z., Collins, R.: Moving object localization in thermal imagery by forward–backward mhi. In: OTCBVS (2006)
37. Yuan, C., Medioni, G., Kang, J., Cohen, I.: Detecting motion regions in presence of strong parallax from a moving camera by multi-view geometric constraints. IEEE TPAMI **29**, 1627–1641 (2007)
38. Zhang, H., Yuan, F.: Vehicle tracking based on image alignment in aerial videos. In: Energy Minimization Methods in Computer Vision and Pattern Recognition, vol. 4679, pp. 295–302 (2007)