J Real-Time Image Proc (2016) 11:315–334
DOI 10.1007/s11554-012-0251-z

SPECIAL ISSUE

Stabilized active camera tracking system

Javed Ahmed · Ahmad Ali · Asifullah Khan

Received: 21 August 2011 / Accepted: 7 April 2012 / Published online: 13 May 2012

Abstract We present a robust and real-time stabilized active camera tracking system (ACTS), which consists of three algorithmic modules: visual tracking, pan-tilt control, and digital video stabilization. We propose an efficient correlation-based framework for visual tracking module that is designed to handle the problems which severely deteriorate the performance of a traditional tracker. The problems that it handles are template drift, noise, object fading (obscuration), background clutter, intermittent occlusion, varying illumination in the scene, high computational complexity, and varying shape, scale, and velocity of the manoeuvring object during its motion. The pan-tilt control module is a predictive open-loop car-following control strategy, which moves the camera efficiently and smoothly so that the target being tracked is always at the center of the video frame. Video stabilization module is required to eliminate the vibration in the video, when the system is mounted on a vibratory platform such as truck, helicopter, ship, etc. We present a very efficient video stabilization method that adds no extra computational overhead to the overall system. It exploits the coordinates of the target, computed by the tracker module, to sense the amount of vibration and then filters it out of the video. The proposed system works at full frame rate (30 fps), and has been successfully used in uncontrolled real-world environment. Experimental results show the efficiency, precision, and robustness of the proposed stabilized ACTS.

Keywords Machine vision · Edge-enhancement · Correlation tracking · Car-following control · Video stabilization

J. Ahmed
National University of Sciences and Technology, Islamabad, Pakistan
e-mail: javed@mcs.edu.pk

A. Ali · A. Khan (✉)
Pakistan Institute of Engineering and Applied Sciences, Nilore, Pakistan
e-mail: asif@pieas.edu.pk

A. Ali
e-mail: ahmadali@pieas.edu.pk

1 Introduction

An active camera tracking system (ACTS) tracks an object of interest automatically with a *moving* video camera. The system, in its simplest form, is illustrated as a section of a block diagram shown in Fig. 1. It consists of a video camera, a visual tracking algorithm, a pan-tilt control algorithm, and a pan-tilt unit (PTU). Every frame acquired from the video camera is analyzed by the visual tracking algorithm, which localizes the object of interest inside the image in pixel coordinates. The coordinates are then sent to a pan-tilt control algorithm which rotates a PTU according to the motion of the object. Since the camera is attached to the PTU, it also rotates in sync with the PTU. Thus, the tracked object (target) is always projected at the center of the video frames, regardless of whether the object is moving or stationary.

An ACTS has been proposed in Ref. [1], but there are some limitations in it. Firstly, it is not a general-purpose target tracker; it is designed to track only a human target. Secondly, it requires the camera to be stationary while selecting the target to be tracked. Thirdly, its pan-tilt control algorithm tries to move the PTU so that the target can be inside a large central region instead of the central pixel in the frame. There is also another ACTS proposed in Ref. [2], but it does not work when the target changes its

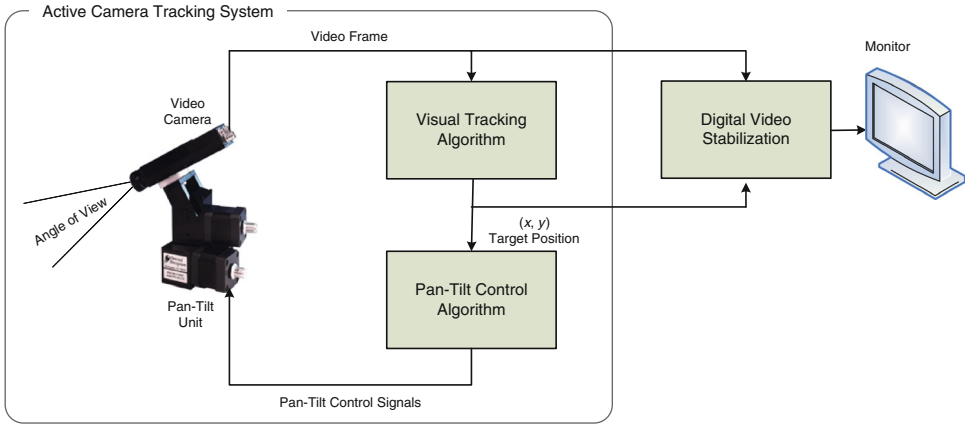


Fig. 1 Simplified block diagram of the proposed stabilized active camera tracking system

appearance abruptly because it updates its model after every 30 video frames.

If the ACTS is mounted on a vibratory platform such as truck, helicopter, ship, etc., we are required to stabilize the video without affecting the efficiency of the system. The purpose of the video stabilization is to filter out the annoying vibration from the video to reduce the unnecessary strain on the eyes of the viewer.

A simplified block diagram of a complete stabilized ACTS is shown in Fig. 1. We provide an introduction to the individual algorithmic components of the system as follows.

1.1 Visual tracking

Visual tracking, in general, can be defined as localizing the object of interest in the consecutive frames of a video. Efficient tracking of the object in complex environments is a challenging task for the computer vision community. Some widely known applications of real-time visual tracking are surveillance and monitoring [3], perceptual user interfaces [4], smart rooms [5, 6], enabling an unmanned ground vehicle (UGV) to handle a road intersection [7], and video compression [8]. The computational complexity of the tracker is critical for most of the applications. Only a small percentage of the system resources can be allocated for tracking while the rest is assigned to pre-processing stages or high-level tasks such as recognition, trajectory interpretation, and reasoning [9].

Several techniques have been proposed by the researchers for target tracking in the consecutive video frames. Most of these are either limited to tracking a specific class of objects [10, 11, 12, 13], or assume that the camera is stationary (and exploit background subtraction) [5, 14]. The trackers based on the particle filter or condensation [15, 16] and active contours [17, 18] do not assume constant background. They are reported to track the whole object instead of only the centroid or a portion of the object [17]. However, they are computationally too expensive to be deployed on a moderate computing machine for a practical real-time tracking application. Real-time vision-based tracking proposed in Ref. [19] requires 3D information about the target scene. Real-time zoom tracking has been reported in Ref. [20], but it works only for digital static cameras. The mean-shift tracker [21, 22] has gained a significant influence in the computer vision community in the recent years, because it is fast, general-purpose, and does not assume static background. Mean shift is a non-parametric density gradient estimator to find the image window that is most similar to the object's color histogram in the current frame. It iteratively carries out a kernel-based search starting at the previous location of the object [23]. There are variants, e.g. [24], to improve its localization using additional modalities, but the original method requires the object kernels in the consecutive frames to have a certain overlap. The success of the mean-shift tracker highly depends on the discriminating power of the histograms that are considered as the object's probability density function [23]. Another issue in the mean-shift tracker is inherent in its use of histogram, which does not carry the spatial information of the pixels [25]. The integral histogram-based tracker [26] matches the color histogram of the target with every possible region in the whole frame; therefore, it can track even a very fast-moving object. It works slower than the mean-shift tracker, because the mean-shift tracker searches for the target in only a small neighbourhood of the previous target position. On a P4 3.2 GHz machine, this tracker works with the speed of about 18 fps (frames per second), and the mean-shift tracker works with the speed of about 66 fps [26]. Since the histogram does not contain the spatial information, and there is a risk of picking up a wrong target having similar histogram (especially when the search is carried out in the whole image), this tracker is not adequately robust. More recently, in the covariance tracking [23], the object is modeled as the covariance matrix of its features, and the region (in the search image) which has minimum covariance distance with the model is considered to be the next

target position. The covariance matching process is carried out on a half-resolution grid in the search image, so the accuracy of the target coordinates found by the algorithm is reduced. The reported results are quite robust, but the computational efficiency of the algorithm is not adequate for a real-time tracking application, because its maximum throughput (as reported in Ref. [23]) is only 7 fps on a P4, 3.2 GHz PC. There is also a widely used tracker based on correlation process [27, 28], in which a template of the object to be tracked is matched with every candidate region in the search image and the best-match region is found by locating the highest peak in the whole correlation surface. The matching can be performed using standard correlation (SC) [27, 28], phase correlation (PC) [29, 30], normalized correlation (NC) [31], or normalized correlation coefficient (NCC) [27, 28, 32]. The NCC is generally considered the most robust of them for matching two grey-level images [32, 33]. Nevertheless, the NCC alone cannot handle template drift, noise, object fading (obscuration), background clutter, intermittent occlusion, high computational complexity, varying illumination in the scene, and varying shape, scale, and velocity of the manoeuvring object. Therefore, we propose a robust and real-time correlation-based visual tracking algorithm that can handle all these problems while working at full frame rate of 30 fps. The template matching in the framework is performed by "NC on edge-enhanced images", because this approach outperforms even the "NCC on actual grey-level images", as discussed in Ref. [33].

1.2 Pan-tilt control

The camera in the stabilized ACTS is mounted on top of a PTU, so it moves in sync with the PTU. The PTU motion is controlled by a control algorithm. If the control is not smooth and precise, the object in the video will oscillate to-and-fro from the center of the frame, and in the worst case the object may get out of the field of view (FOV). One approach is to use a classic proportional-integral-derivative (PID) controller [34]. However, its design requires a mathematical model of the system. Besides, it necessitates a sensitive and rigorous tuning of its three gain parameters (i.e., proportional, differential, and integral) for use with all the zoom levels of the camera. An alternative approach is to use a fuzzy controller [11], [35] that does not require the system model, but choosing a set of right membership functions and fuzzy rules calibrated for every zoom level of the camera is practically very cumbersome. Another alternative is to implement a neural network controller [36], but it is heavily dependent on the quality and the variety of the examples in the training dataset, which can accurately represent the complete behaviour of the controller in all possible scenarios, including the varying zoom levels of the camera. Furthermore, the traditional control algorithms, e.g. [37], are generally implemented based on the difference between the center of the frame and the current target position in the frame. These algorithms do not account for the target velocity. As a result, there will be oscillations (if the object is moving slow), a lag (if it is moving with a mediocre speed), and loss of the object from the frame (if it is moving faster than the maximum pan-tilt velocity generated by the control algorithm). Keeping in view the above-mentioned limitations of the various control algorithms, we use a predictive open-loop car-following control (POL-CFC) algorithm [33]. Its basic idea is borrowed from the car-following control (CFC) strategy [38]. The CFC assumes that the actual velocity of the PTU is observable through a velocity sensor. However, the POL-CFC does not make this assumption and simply considers that the current PTU velocity is the previous velocity command sent to the PTU. Then, it computes the velocity of the target relative to the PTU velocity from the predicted target positions provided by the Kalman filter in the current and the next frame. Finally, it generates precise velocity commands for the PTU to move the camera towards the target accurately in real-time. Thus, the proposed control strategy is very useful for controlling a system, which does not feedback its current velocity, such as stepper-motor PTU. Its performance is tested on real-world scenarios and has proven to be adequately smooth, fast and accurate. The POL-CFC algorithm in the proposed stabilized ACTS offers 0 % overshoot, 0 steady-state tracking error, and 1.7 s rise-time at least for 1–6× zoom levels of the camera.

1.3 Video stabilization

Video stabilization is the process of removing vibration from the video. It has very wide application spectrum ranging from consumer devices (e.g., handy-cams, mobile phone with video camera, etc.) to state-of-the-art military and defence systems, e.g., the payloads for u*nmanned aerial vehicle* (UAV) and *unmanned ground vehicle* (UGV) [39], etc. There are many hardware as well as software solutions available for video stabilization with their own merits and demerits depending upon their applications. There are two types of motion when camera is mounted on a PTU. One is valid or required motion that comes due to the motion of the object to be tracked. The other one is the annoying or unwanted motion that may come due to the mechanical vibration transmitted from vibratory vehicle (on which the PTU is mounted) or environmental factor (such as wind). The aim of video stabilization is to filter out the latter motion [40, 41].

The ideal approach for video stabilization is the hardware solution. Use of mechanical tools to physically avoid camera vibration is one of the hardware solutions. Another

solution may be to exploit optical or electronic devices to influence how the camera sensor receives the input light [42, 43]. These are expensive solutions or need some additional information about camera motion. Therefore, image processing-based video stabilization (also called digital video stabilization) is the approach of common choice [43].

Optical flow based method is opted by Chand et al. [44] for digital video stabilization. However, it has inherent aperture problem [45]. Fuzzy logic modeling is used in Ref. [43] for video stabilization, but, it is time consuming to select the membership functions and tune their parameters to achieve the satisfactory results. Image based rendering technique is used in Ref. [46]. However, it works well only in case of slow camera-motion. Techniques in Refs. [47, 48, 49] emphasise on block-matching methods. These algorithms do not track blocks in consecutive frames, so they may be misled by large moving objects [43]. In order to handle these problems, we propose a stabilization algorithm which estimates the vibratory motion between the frames by taking inputs from our visual tracking module. Thus, it does not add any extra computational overhead to the system for estimating the instantaneous vibration. The vibratory motion in the video is then filtered using a simple low-pass filter. Thus, the stabilization algorithm works at the full frame rate of a standard video (i.e., 30 fps).

The rest of the paper is organized as follows. Section 2 discusses the proposed visual tracking framework, Sect. 3 describes the proposed pan-tilt control strategy, Sect. 4 explains the proposed digital video stabilization technique, and Sect. 5 presents the experimental results of individual modules as well as the complete system. Finally, Sect. 6 concludes the paper.

2 Proposed visual tracking algorithm

We use Sony FCB-EX 780BP analog camera, and acquire video frames each of size 320 × 240 pixels at the rate of 30 fps using Dazzle DVC-90 video digitizer. Once the object of interest appears in the video, we initialize the template of the object from its current (x, y) position in the video frame. Thus, we have the template as a matrix, t, of size $K \times L$, with K and L being odd integers to have a proper center. The complete flow chart of the proposed algorithm is shown in Fig. 2 and the different steps involved in the algorithm are explained as follows.

2.1 Dynamic search window

Since we know the exact position of the target in the initial frame, we do not search for it. However, from the second frame onward, we look for the target in a small search window (where the probability of finding the target is high) instead of the whole frame, so that the false alarm rate due to the background clutter may be reduced and the tracking algorithm may be computationally efficient. If the search window is of constant size throughout the tracking session as in Refs. [50, 51], the fast manoeuvring object may get out of the window (if the window is inadequately small), or the computation complexity of the normalized correlation

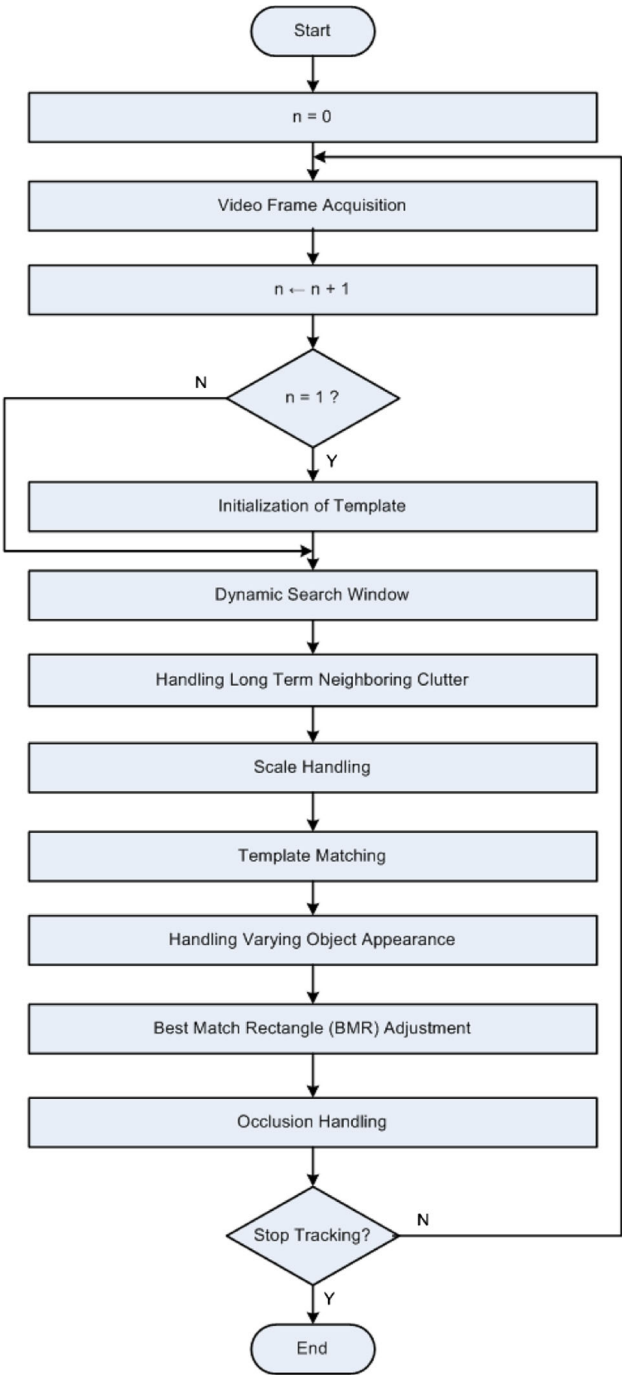


Fig. 2 Flow chart of the proposed visual tracking algorithm

process between the search window and the template may be increased (if the window is redundantly large). Therefore, we create a dynamic search window of an appropriate size automatically, as described in Eq. (1), where $(x_{\text{tl}}, y_{\text{tl}})$ and $(x_{\text{br}}, y_{\text{br}})$, respectively, are the coordinates of the top-left and bottom-right vertices of the search window in the frame, and $\left(x^*_{n+1|n}, y^*_{n+1|n}\right)$ are the next target-coordinates predicted by Kalman filter [52]. It may be noted that we use a three-state 2D Kalman filter and "constant acceleration with random walk model" for the target motion.

$$\begin{aligned} x_{\text{tl}} &= x^*_{n+1|n} - \left(\frac{L-1}{2} + \kappa + a_{\text{tx}}|\varepsilon_x|\right) \\ y_{\text{tl}} &= y^*_{n+1|n} - \left(\frac{K-1}{2} + \kappa + a_{\text{ty}}|\varepsilon_y|\right) \\ x_{\text{br}} &= x^*_{n+1|n} + \left(\frac{L-1}{2} + \kappa + a_{\text{bx}}|\varepsilon_x|\right) \\ y_{\text{br}} &= y^*_{n+1|n} + \left(\frac{K-1}{2} + \kappa + a_{\text{by}}|\varepsilon_y|\right) \end{aligned} \tag{1}$$

The three states of the filter are position, velocity, and acceleration of the target in the image in both axes. The first three terms in case of every coordinate in Eq. (1) make a *minimum-size search window* of size $(K + 2\kappa) \times (L + 2\kappa)$, where κ is the minimum width of the border around $K \times L$ area, centered at the predicted position. Experimentally, $\kappa = 19$ provides us with the desired results even in complex object-motion scenarios. Furthermore, ε_x and ε_y are the *prediction errors*, defined as $\varepsilon_x = x_n - x^*_{n|n-1}$ and $\varepsilon_y = y_n - y^*_{n|n-1}$, where (x_n, y_n) are the current target coordinates and $(x^*_{n|n-1}, y^*_{n|n-1})$ are the target coordinates predicted by the Kalman filter in the previous iteration. The a_{tx}, a_{ty}, a_{bx}, and a_{by} parameters in Eq. (1) are the scaling factors, which compensate for the possible prediction errors in case of a sudden manoeuvring of the object. If any of the scaling factors is positive, the *minimum-size search window* will be expanded further in the direction of the object motion proportional to the corresponding prediction error. On the other hand, if it is negative, the *minimum-size search window* will be contracted from opposite direction of the object motion proportional to the corresponding prediction error. We set the values of these scaling factors as:

$$(a_{\text{tx}}, a_{\text{bx}}) = \begin{cases} (a_1, a_2) & \text{if } \varepsilon_x \geq 0 \\ (a_2, a_1) & \text{otherwise} \end{cases} \tag{2}$$

and

$$(a_{\text{ty}}, a_{\text{by}}) = \begin{cases} (a_1, a_2) & \text{if } \varepsilon_y \geq 0 \\ (a_2, a_1) & \text{otherwise} \end{cases} \tag{3}$$

where we have $a_1 = -0.25$ and $a_2 = +1.25$ after a few experiments and they are fixed for all scenarios. Let the resulting search window be a matrix, s, of size $M \times N$ for easy reference. Figure 3 compares the results of the traditional fixed-size search window and the proposed dynamic search window on the video *sequence_fast.avi* obtained from [53]. The search window is shown in blue and the best-match rectangle in yellow. When a $(60 + K) \times (60 + L)$ fixed-size search window centered at the predicted position is used, the fast to-and-fro motion of the target (i.e., face) causes it to get out of the search window as shown in Frame 21. When the proposed dynamic search window is used, the target is always inside the search window. Furthermore, it is also important to mention that the mean-shift [21, 22] and the condensation [16, 54, 55] trackers could not track the fast-moving face (see Fig. 6 in Ref. [15]).

2.2 Handling long-term neighbouring clutter

The search window handles the *background clutter*, which is far from the object. However, it does not handle the *neighbouring clutter* which is present immediately around the object. There are two kinds of neighbouring clutter: *short-term* and *long-term*. The effect of the short-term neighbouring clutter is efficiently diminished by a smoothing temporal filter described in Sect. 2.5 . However, we decrease the effect of the long-term neighbouring clutter by assuming that the object is at the center of the template (due to a *best-match rectangle adjustment* algorithm described in Sect. 2.6) and applying a weight on every pixel in the template. The farther the pixel from the center of the template, the lower the weight it gets. Specifically, we use a template-size 2D Gaussian weighting function with its two means (which are basically the xy-coordinates of the center of the template) and two standard deviation parameters (σ_w) computed as [56]:

$$\sigma_w = 0.3\left(\frac{w}{2} - 1\right) + 0.8 \tag{4}$$

where w represents a dimension (i.e., width or height) of the template.

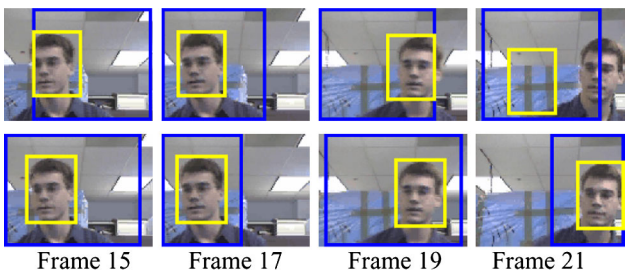


Fig. 3 Comparison of the traditional fixed-size search window (*upper row*) and the proposed dynamic search window (*lower row*)

2.3 Scale handling

The moving object in the video can get larger (or smaller) with time when the distance between the object and the camera is varied or the zoom level of the camera is changed. If the template size is kept constant throughout the tracking session, it may create two problems: (1) when the object gets smaller, the background pixels will invade into the template and the template will become a representative of the background instead of the object, and (2) when the object gets larger, the fixed template will be representing only a very small part of the object, and it will not contain adequate number of pixels of the object to make it distinct from the background clutter. Due to these problems, the template can match with clutter more than it does with the actual object. As a result, the false alarm rate will be high, resulting in the failure of the tracking session. In order to handle the varying scale of the object, we exploit a fact about the peak value in the correlation response. The correlation peak is high, if the scale of the object in the *template* is same as the scale of the object in the *search window*; otherwise, it is low. Thus, we handle the varying size of the object by cross-correlating the search window with three sizes of the template: 110, 100, and 90 %. The 100 % size of the template is the original one which comes from the previous iteration. As a result, we get three correlation surfaces with the corresponding three peaks. We select the best-match corresponding to the template size which produces the highest of the three peaks, i.e., $c_{\max}$. Thus, the template for the next iteration is scaled according to the current scale of the object.

2.4 Template matching

As far as the template matching process is concerned, we perform it using the edge-enhanced back-propagation neural network (BPNN)-controlled fast normalized correlation (EE-BCFNC) technique [33], which makes the images edge-enhanced before they are correlated. It is faster and more robust to varying illumination conditions, neighbouring clutter, and intermittent object fading than NCC. In this approach, the edge-enhancement stage consists of Gaussian smoothing, fast gradient magnitude, normalization, and thresholding [33]. The NC in the spatial domain is computed as:

$$c(m,n)=\frac{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1}s(m+i,n+j)t(i,j)}{\sqrt{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1}s^2(m+i,n+j)}\sqrt{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1}t^2(i,j)}} \quad (5)$$

where $c(m, n)$ is the element of the correlation surface (i.e., matrix) at row m and column n, where $m = 0, 1, 2,\ldots,M - K$, and $n = 0, 1, 2,\ldots,N - L$. The BPNN in the EE-BCFNC technique controls whether the correlation is computed directly using Eq. (5) or fast Fourier transform–summed-area table (FFT-SAT) method to get the desired efficiency. In FFT-SAT method, the numerator of NC is computed using FFT and its denominator is efficiently computed using the concept of summed-area-table (or integral image), as discussed in Refs. [32, 33, 57]. At the end of the template matching process, we have the best-match rectangle (BMR) with its center at (m_c, n_c) in the search window and (x, y) in the frame, and a normalized confidence level (global peak in the correlation surface) in the range [0.0, 1.0].

2.5 Handling varying object appearance

The shape and orientation of the object may change during its motion in the video. Therefore, a constant template cannot work for a long and good tracking session. It must be adapted with time according to the change in the appearance of the object. If the current template is replaced with the current best-match region to generate the new template, it causes template drift problem and the tracker loses the target when the object is faded intermittently. If the template is updated with traditional α-tracker updating method [50], the robustness of the method mainly depends on the value of α. If it is set to a higher value, there will be more template drift, and less robustness to object fading. On the other hand, if it is set to a very low value, the template will not be updated adequately in case of fast manoeuvring object [31]. Therefore, we use a temporal IIR filter with adaptive coefficients, given as:

$$t[k+1]=\begin{cases}\beta b[k]+(1-\beta)t[k] & \text{if } c_{\max}>\tau_t\\ t[k] & \text{otherwise}\end{cases} \quad (6)$$

where $\beta = 0.2c_{\max}$, $b[k]$ is the best-match region in the edge-enhanced search image in the current iteration, $t[k]$ and $t[k + 1]$ are the current and the updated template, respectively, and τ_t is a threshold, such that $0 < \tau_t < 1$. Typically, we get satisfactory results by setting $\tau_t = 0.84$ in all test scenarios. It may be noted that, due to adaptive coefficients, the amount of change to be introduced in the updated template is determined by the quality of the correlated object. A stronger match is a good candidate for being the next template, so it introduces a larger weight to the best-match when the peak correlation value, $c_{\max}$, is large.

2.6 Best-match rectangle (BMR) adjustment

There are two main concerns that should be addressed properly for precise correlation-based tracking. Firstly, the human operator is usually unable to initialize (or extract) a good template of the object of interest while the object is

moving and manoeuvring in the video. As a result, the extracted template is usually larger or smaller than the object, or the object captured inside it is significantly deviated from the center of the template. Secondly, the object drifts away from the center of the template slowly with time in a typical correlation tracking session, especially when the object is rotating in the video. The proposed template-updating method discussed above reduces the template drift significantly, but the technique does not completely eliminate the problem. The *incorrect template initialization* and the *template drift* problems severely deteriorate the performance of the correlation tracker in two ways. Firstly, they make the background pixels invade into the template and the tracker starts assuming that it has to track the background clutter instead of the desired object, resulting in a total failure of the tracking session. Secondly, the object remains deviated from the center of the frames in the resulting video, even when the pan-tilt control algorithm itself is very efficient and precise. In order to solve the two problems, we use *BMR-adjustment* algorithm. The algorithm analyzes the content of the temporally filtered (i.e., updated) template and resizes and/or relocates the BMR so that it can have more of the object and less of the background inside it. As a result, the object remains always at the center of the template as well as the frame (if the camera is tracking the object as in our case). The detail and mathematical formulation of BMR adjustment is given in detail in Ref. [58]. The adjusted BMR is then exploited to generate a good template for the next iteration of the tracker.

2.7 Occlusion handling

A target is said to be (partially or completely) occluded when it is (partially or completely) hidden due to the appearance of another object between the camera and the target. Before handling the occlusion, we need to know exactly when an occlusion has occurred. We exploit the fact that when the object being tracked is suddenly occluded by another object, the peak correlation value is dropped below the threshold (τ_t) (it may be noted that this threshold is the same as the one in Eq. (6)). When it happens, we:

1. Assume that the target coordinates provided by the correlation process are not correct and that the target is at the coordinates predicted by the Kalman filter in the previous iteration.
2. Update the Kalman filter in the current iteration with its own prediction made in the previous iteration.
3. Stop updating the template, because we do not want our template to be distorted due to the shape of the other object which has occluded our target. Thus, during the occlusion, we use the template which we had before the occlusion.
4. Reduce the threshold iteratively at nth iteration, as:

$$\tau_t[n] = \tau_t[n-1] - 0.0005 \tag{7}$$

if $\tau_t[n] \geq \tau_{tmin}$, where $\tau_{tmin} = 0.65$ in our case. We reduce the threshold as described, because if the object is again visible after some frames, it may look a little different from how it looked before the occlusion; thus, the correlation peak obtained after the occlusion will not necessarily come out to be above the initial threshold value.
5. Make the dynamic search window larger and larger with time by iteratively varying the value of the border parameter κ in Eq. (1), as:

$$\kappa[n] = \kappa[n-1] + 2 \tag{8}$$

We do so in order to compensate for the uncertainty in the speed and direction of the object during occlusion. This approach, effectively, increases the search window by four rows and four columns in every iteration n during the occlusion.
6. After some frames, if the correlation peak reaches above the current threshold value, we assume that the object has come out of the occlusion, and that the coordinates provided by the correlation process are now correct. Thus, we re-initialize the values of τ_t and κ to their initial default values, i.e., 0.84 and 19, respectively, for normal correlation tracking.

3 Proposed pan-tilt control algorithm

The pixel coordinates of the object obtained from the visual tracking algorithm discussed in the previous section are sent to the pan-tilt control algorithm which rotates the PTU according to the motion of the object. Since the camera is attached to the PTU, it also rotates in sync with the PTU, so that the object (target) can be projected always at the center of the video frame, regardless of whether the object is moving or stationary.

Our pan-tilt control algorithm has been derived from the basic CFC law [38]. The CFC law can be used only for closed-loop system in which the current velocity of the PTU is fed back to the control algorithm. We have modified the CFC law so that it can be used for an open-loop system. We call the modified control algorithm POL-CFC strategy. The algorithm generates the pan-tilt velocity commands in accordance with the Kalman predicted [33] target velocity components in the video frames. The use of predicted velocity (instead of the current velocity) is helpful in compensating for the inertia of the pan-tilt mechanism and hence following the target without any lag or inaccuracy. The POL-CFC strategy is described mathematically as:

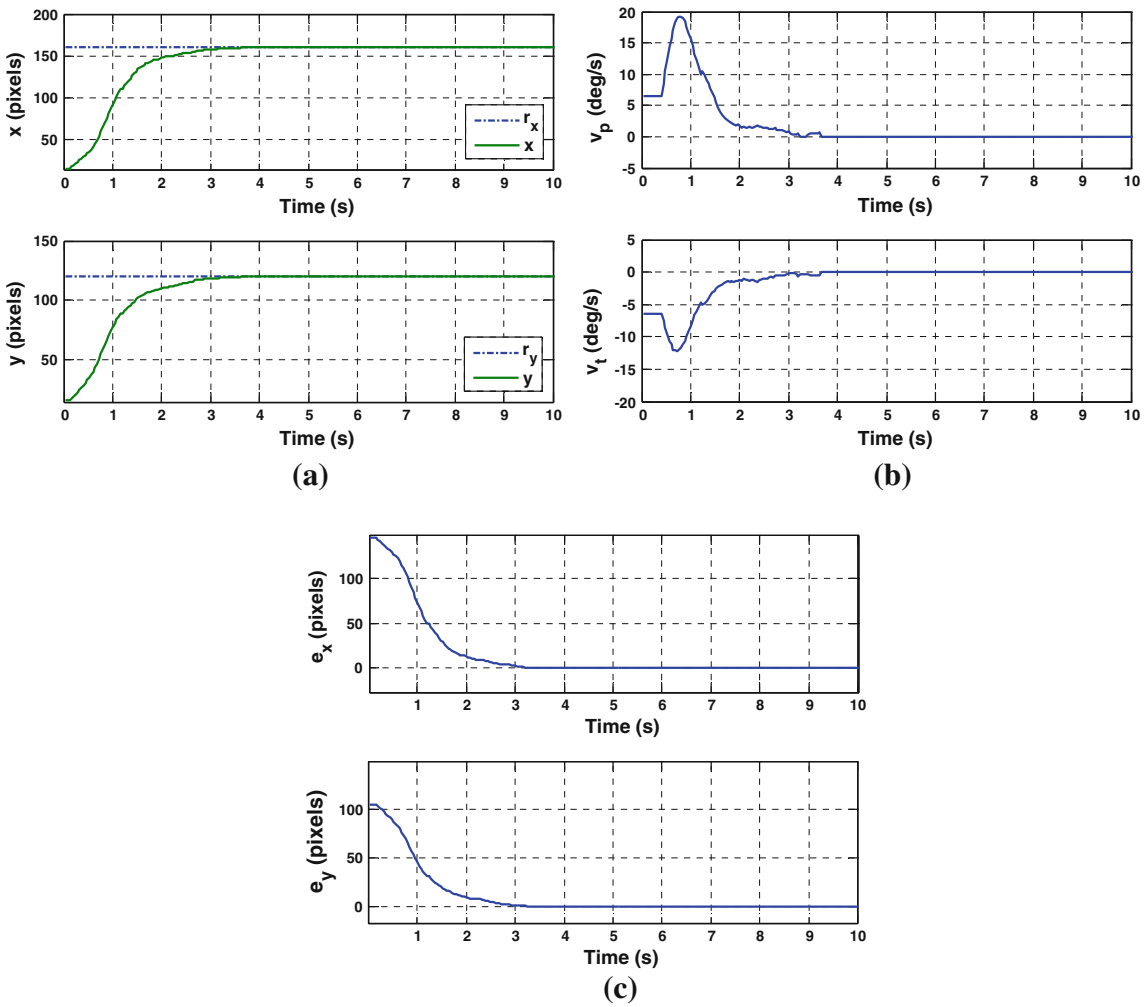


Fig. 4 Target trajectory, generated velocity, and tracking error curves in both axes, when a stationary object was being centralized in the video frames by the proposed tracking system. **a** Target trajectory in the video frames. **b** Velocity generated by the POL-CFC. **c** Tracking error with respect to center of the frame

$$\begin{aligned} v_{\mathrm{p}}[n+1] &= v_{\mathrm{p}}[n] + \eta\left(Ke_x^*[n+1|n] - v_{\mathrm{rp}}^*[n+1|n]\right) \\ v_{\mathrm{t}}[n+1] &= v_{\mathrm{t}}[n] + \eta\left(v_{\mathrm{rt}}^*[n+1|n] - Ke_y^*[n+1|n]\right) \end{aligned} \tag{9}$$

where $v_p[n]$ and $v_t[n]$ are the current pan and tilt velocities of the PTU (which were generated by Eq. (9) in the previous iteration), η is a small positive constant in the range [0.0, 1.0] which controls the amount of the velocity added to the *previous* velocity, K is the proportional gain parameter (which is the *only* parameter to be tuned for every zoom level of the camera), $e_x^*[n+1|n]$ and $e_y^*[n+1|n]$ are the predicted errors in both the axes defined as:

$$\begin{aligned} e_x^*[n+1|n] &= r_x - x^*[n+1|n] \\ e_y^*[n+1|n] &= r_y - y^*[n+1|n] \end{aligned} \tag{10}$$

Furthermore, the $v_{rp}^*[n+1|n]$ and $v_{rt}^*[n+1|n]$ in Eq. (9) are the predicted relative velocities of the target in terms of pan-tilt degrees per second, defined as:

$$\begin{aligned} v_{\mathrm{rp}}^*[n+1|n] &= C_{\mathrm{dpp}}\left(\frac{x^*[n+1|n] - x^*[n|n-1]}{T}\right) \\ v_{\mathrm{rt}}^*[n+1|n] &= C_{\mathrm{dpp}}\left(\frac{y^*[n+1|n] - y^*[n|n-1]}{T}\right) \end{aligned} \tag{11}$$

where C_{dpp} is a conversion ratio in degrees per pixel determined by a simple camera calibration procedure for all the zoom levels of the camera, T is the sampling time (which is inverse of the video frame rate), and $(x^*[n+1|n],\ y^*[n+1|n])$ and $(x^*[n|n-1],\ y^*[n|n-1])$ are the target coordinates in the video frame predicted by Kalman filter in the current and the previous iterations, respectively.

In order to evaluate the performance of the POL-CFC algorithm, a stationary object was selected from the top-left section of the live video from the camera and the algorithm was let to generate the appropriate pan-tilt velocities to move the camera to centralize the object in the video frames. Figure 4 shows the instantaneous position of the object in the video frame, the instantaneous

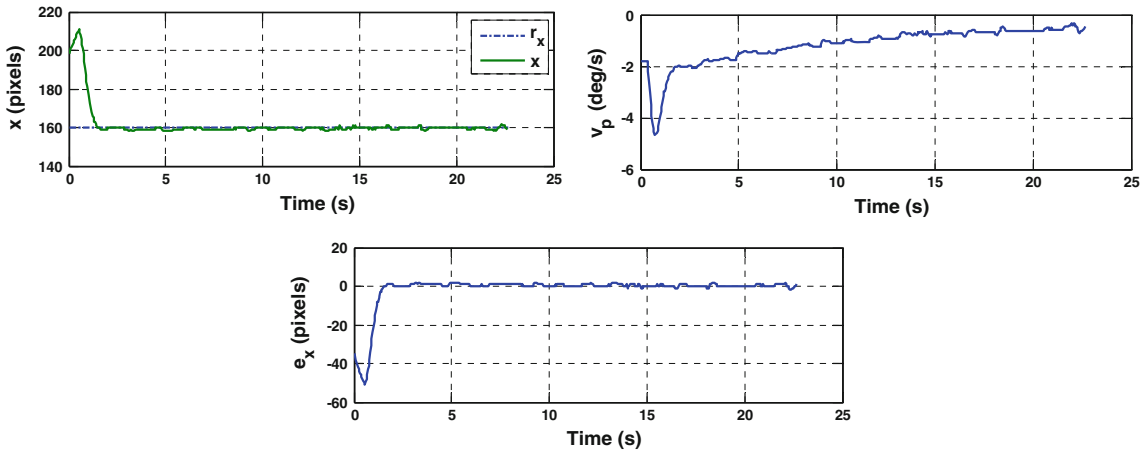


Fig. 5 Target trajectory, velocity and tracking error curves for pan motion, when a flying helicopter was being tracked

control action (velocity) generated by the proposed controller and the instantaneous positional error in both the axes, while the object was being centralized. The curves illustrate that initially the controller generates a constant velocity to start the motors from rest. When the controller senses that the positional error is not reduced adequately, it starts increasing the speed after about 0.4 s. As a result, the object starts coming closer and closer to the center of the frame, i.e., (160, 120), efficiently, within the time span between 0.5 and 1.5 s, as shown in Fig. 4a. When the error is reduced significantly and the current PTU velocity is greater than necessary for the current position of the object, the control algorithm reduces the velocity smoothly until the object approaches the center of the video frame. It can be observed that there is 0 % overshoot, 1.7 s rise-time, and 0 steady-state error. The definitions of these terms related to control systems can be found in Ref. [34].

The same experimental setup was also used for various moving objects such as helicopters, airplanes, vehicles, walking and running persons, etc. in real-world scenarios to test the performance of the system. The target trajectory in x-axis, the pan velocity, and the tracking error in x-axis for the case of a flying helicopter are shown in Fig. 5. In this scenario, the helicopter was captured from the position $(x, y) = (192, 118)$ in the frame as shown by the initial point in the trajectory curve. Since the helicopter was moving fast towards right and the camera was initially stationary, the helicopter in the video moved further towards right. This is shown by the initial upward bulge above $x = 192$ in the trajectory curve. The PTU motion catches up with the helicopter motion after about 0.4 s. Then, the control algorithm starts centralizing the helicopter very efficiently. It can be seen in the curves that the helicopter is finally centralized by about 1.4 s and it remains at the center of the frames afterwards with the maximum steady-state error of only ± 1 pixel. This experiment was performed, when the camera was operating at 5× zoom level, and some frames from the resulting video are shown in Fig. 6. After doing such rigorous experimentation on different moving objects, we have found that the proposed POL-CFC have achieved 0 % *overshoot*, 1.47 s *rise-time*, and *maximum steady-sate error* of only ± 4 pixels at 25× zoom level. Table 1 shows the *maximum steady-state error* of the proposed tracking system at different zoom levels.

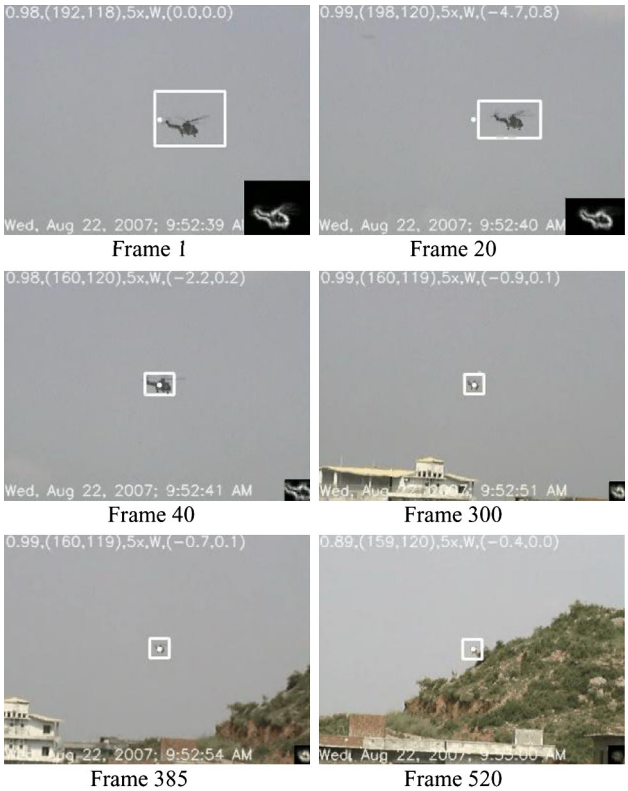


Fig. 6 A helicopter is being tracked persistently and precisely with the proposed tracking system even when the user has initialized the template inaccurately, and the size, the appearance, and the velocity of the helicopter is continuously varying

Table 1 Maximum steady-state error of the tracker

Zoom	Maximum steady-state error (pixels)
1–6×	±1
7–15×	±2
16–19×	±3
20–25×	±4

4 Proposed video stabilization algorithm

Our video stabilization algorithm takes two inputs: the current video frame and the current image coordinates (x, y) of the target (estimated by the tracker described in Sect. 2). The algorithm outputs the stabilized video frame, which can be seen on a monitor, as illustrated in the block diagram shown in Fig. 1. Software approach for video stabilization (which is also called digital video stabilization) requires a foreground object with respect to which the stabilization process is performed. In our algorithm, the target is taken as foreground object. The inter-frame motion is estimated with the help of the image coordinates (x, y) of the target. This motion contains low-frequency components (i.e., valid motion) as well as high-frequency components (i.e., vibration). In order to filter out the latter components, a simple low-pass filter is used both in x and y axes given as:

$$\begin{bmatrix} \hat{x}_n \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \alpha & (1-\alpha) & 0 & 0 \\ 0 & 0 & \alpha & (1-\alpha) \end{bmatrix} \begin{bmatrix} x_n \\ \hat{x}_{n-1} \\ y_n \\ \hat{y}_{n-1} \end{bmatrix} \tag{12}$$

where x_n and y_n are the xy-coordinates of the target in the unstabilized current frame estimated by the tracking module, $\hat{x}_n$ and $\hat{y}_n$ are the filtered xy-coordinates of the target in the current frame, $\hat{x}_{n-1}$ and $\hat{y}_{n-1}$ are the filtered xy-coordinates of the target in the previous frame, and α is the coefficient of the filter having value from the range $0 < \alpha < 1$ to meet the filter stability criterion. The lower the value of α, the lower the cut-off frequency of the low-pass filter (as illustrated in Fig. 7). Thus, we can set the value of α according to the vibration frequency involved in the application at hand. For example, the frequency response of the filter, when α is set to 0.11, is shown in Fig. 8. This figure also shows that the cut-off frequency of the filter is 0.5 Hz (at the magnitude of 0.707). It may be observed that the frequency response around cut-off frequency (i.e., roll-off) is not perfectly steep because the filter is real (not ideal) and of first order (i.e., single pole). We use this filter because it is easy to tune its single parameter α in run-time while observing its effects on the stabilized video. However, if one is sure about the optimal cut-off frequency for a specific application, a higher-order filter can be designed and implemented to have as small transition region as possible around the cut-off frequency in its frequency response.

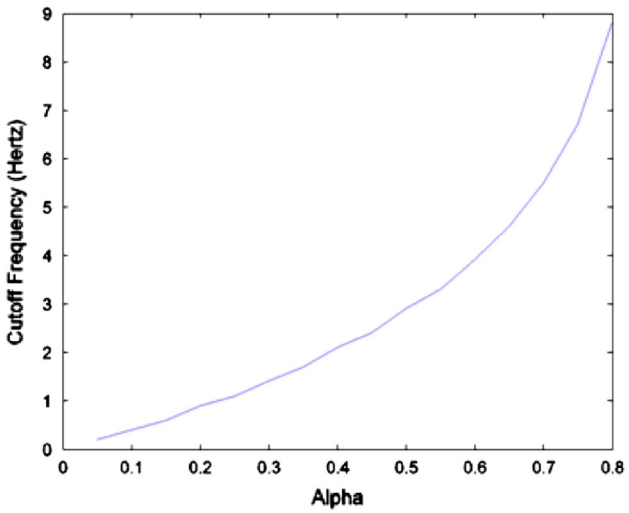


Fig. 7 Relationship between α and cut-off frequency of the low-pass filter

Once the stabilized xy-coordinates of the target in the current frame are obtained, the vibratory motion estimation vector is calculated as below:

$$\begin{bmatrix} M_{x_n} \\ M_{y_n} \end{bmatrix} = \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \end{bmatrix} - \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{13}$$

where M_{x_n} and M_{y_n} are the xy-components of the vibratory motion estimation vector for the current frame. The estimated vibratory motion is then compensated by translating every frame-pixel at (i, j) in the opposite direction of the vibratory motion such that the new coordinates of the pixel become (i', j'), calculated as:

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} i - M_{x_n} \\ j - M_{y_n} \end{bmatrix} \tag{14}$$

It may be noted that i and j are the horizontal and vertical coordinates of the pixel, respectively. If the new position of the pixel is obtained outside the frame boundaries, the corresponding pixel is discarded. As a result, we get the stabilized video frame with respect to the target at the cost of some undefined or vacant regions at the boundaries from where the pixels were translated. There are two widely used strategies to fill these regions. One approach is to fill the vacant regions with black pixels, but it creates unpleasant impact on the viewer because the number of black pixels is continuously changing according to the varying instantaneous magnitude of oscillations due to vibration. Other approach is to fill the vacant regions by the same regions as that of the previous frame, but it creates unpleasant artefacts at the boundaries of the video

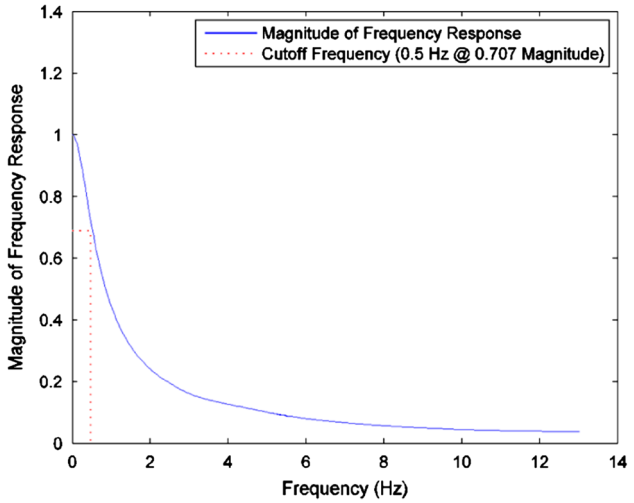


Fig. 8 Magnitude of frequency response of the low-pass filter at $\alpha = 0.11$

frame. Therefore, we use a black boundary of a *fixed width* greater than the anticipated vibration amplitude.

5 Results and discussion

In this section, we present the results of: (a) the proposed visual tracking algorithm tested off-line on some challenging real-world videos, (b) the video stabilization algorithm, (c) the ACTS, and finally (d) the stabilized ACTS.

5.1 Performance of visual tracking algorithm

First, we compare the proposed correlation tracker (PCT) with a traditional correlation tracker (TCT) on two challenging image sequences: S_1 and S_2. The TCT uses NCC for template matching, α-tracker template-updating scheme with $\alpha = c_{\max}$ [31] and a search window of size $3K \times 3L$ centered at the previous target position. In both the trackers, the initial template of size 19×25 is selected from the same position in the initial frame, and the threshold for the correlation peak τ_t is set to 0.84. The image sequence S_1 contains 300 frames and was recorded by us with a handy-cam. During its recording, we continuously and randomly moved the camera very fast to create random and fast motion of the object (helicopter) in the video frames. Since the zoom level of the camera was high at the time of recording, the object was suddenly faded for a short time period as usually happens with almost all the cameras. The image sequence S_2 (available at Ref. [59]) contains 390 frames, in which an aircraft is taking-off, during which its size is varying and the background is cluttered with trees and small buildings. In order to evaluate the tracking accuracy of the algorithms, the ground truth containing true target-coordinates was generated manually for every frame in both the sequences. Figure 9 shows the results of TCT and PCT for S_1 sequence, in which it is visible that TCT is not able to handle the template drift and object fading problems (see frame 274), but PCT tracks the target successfully even in the presence of these problems. The complete trajectory of the S1 sequence provided by TCT is compared with the true trajectory in Fig. 10, which illustrates the failure of the algorithm starting at the Frame 274. Figure 11 shows the robustness of the PCT that keeps a very good track of the target in the same video. It can be observed, that there is no template drift or track-loss even during the severe and sudden fading of the object in the presence of fast and random object motion in the low contrast imagery. Figure 12 shows the results of both the trackers for S_2 image sequence. It can be observed that TCT is distracted due to the clutter (e.g., white roof of a building) at Frame 93, but the PCT is not. The complete trajectory of the airplane provided by TCT is compared with the true trajectory in Fig. 13, which illustrates the failure of the algorithm starting from Frame 93. However, Fig. 14 shows the robustness of the PCT that keeps a good track of the airplane with negligible template drift even during the sudden appearance of the white roof of the building, the surrounding clutter, drastic change in the intensity of the background, and varying scale. Table 2 shows the post regression analysis for comparing the accuracy of TCT and PCT. It provides R-value (the correlation coefficient between the true and the calculated coordinates), and the best-fit linear equation between them consisting of a slope (m) and intercept (C). If the trajectory provided by the tracker and the ground truth trajectory are exactly similar, then $R = 100\ \%$, $m = 1.0$, and $C = 0$. The data in the table illustrates that PCT outperforms TCT in accuracy for the two real-world and complex test sequences.

Next, we present the results of PCT for other real-world image sequences. Figure 15 shows some frames from an image sequence in AV-16.3 dataset [60]. We can see that our tracker is tracking the head of a person robustly even in the presence of similar multiple objects, clutter, and most importantly occlusion. However, when we tested TCT on this video, it could not survive the occlusion. We do not show its results due to space constraint. Figure 16 depicts the result of PCT for an image sequence of a road-traffic. The PCT is tracking a car (of which size is continuously reducing) by adapting the size of the template to the size of the car (the current template is overlaid at the top-right corner of each frame for demonstration). On the contrary, when we tested TCT on this sequence, it got stuck with the background (at frame 50) which had invaded inside its fixed-size template. Again, we do not show the results of TCT due to the space constraint. Figure 17 presents some

Fig. 9 Results of TCT and PCT for S_1 image sequence. *Left column* TCT does not handle template drift and object fading. *Right column* PCT handles the template drift and object fading

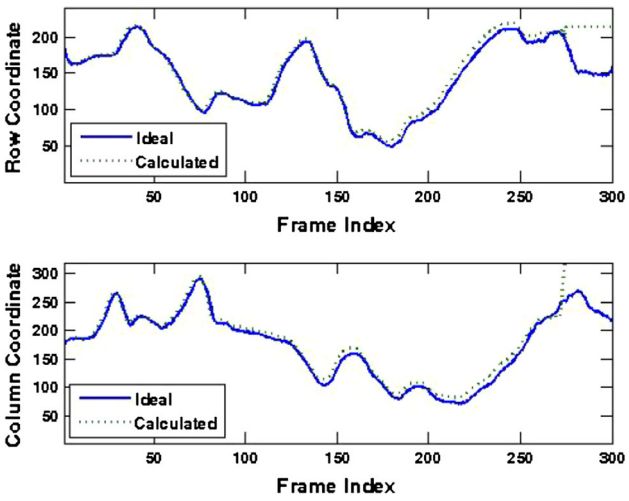


Fig. 10 Target trajectory (*row and column coordinates*) produced by TCT for S_1 sequence showing the failure from frame 274 through the last frame of the image sequence

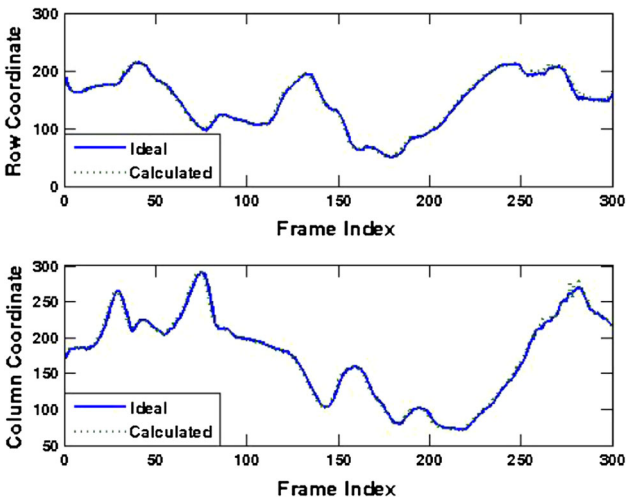


Fig. 11 Target trajectory (*row* and *column coordinates*) for S1 sequence produced by our algorithm. Note that the computed trajectory is perfectly matching the ground truth trajectory for almost all the frames

frames from a video in the PETS dataset [61] showing a car being successfully tracked by PCT in the presence of background clutter and variation in the scale as well as the shape of the car. Figure 18 presents some frames from *seq_mb.avi* video (available at Ref. [53]). PCT tracks the face of the girl even during occlusion by a man. However, the mean-shift [21, 22] and the condensation [15, 54, 55] trackers could not survive the occlusion in this sequence (see Fig. 7 in Ref. [15]). Similarly, these two algorithms fail to track fast and to-and-fro moving head of the person

Fig. 12 Results of TCT and PCT for S_2 image sequence. *Left column* TCT is distracted due to the clutter (e.g., white roof of a building). *Right column* PCT is not distracted due to the clutter

in the video sequence *seq_fast.avi* (available at Ref. [53]) as reported by Fig. 6 in Ref. [15]. Our proposed scheme successfully tracks it as shown in Fig. 3 (lower row).

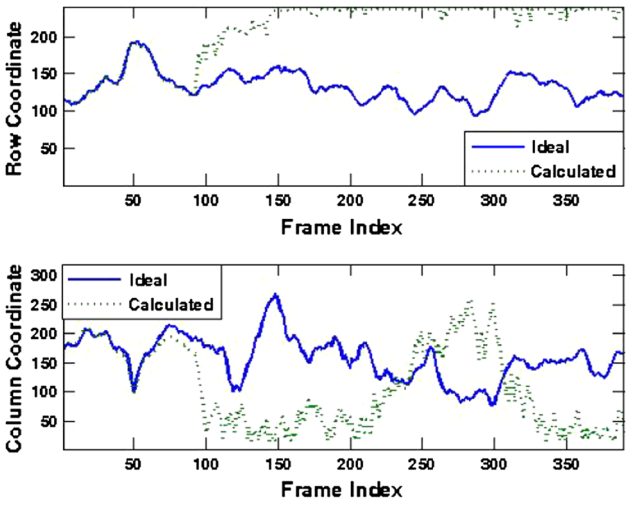


Fig. 13 Target trajectory (*row and column coordinates*) produced by TCT for S_2 sequence, showing its failure starting from frame 93

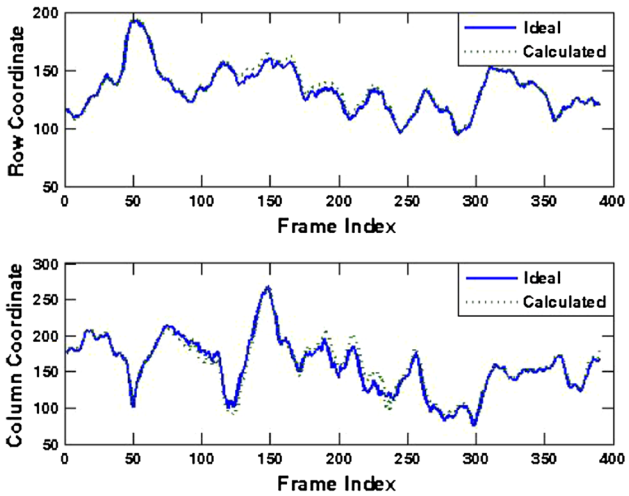


Fig. 14 Target trajectory provided by PCT for S_2 sequence. It accurately follows the ground truth trajectory in almost all the frames

Table 3 shows these results in numerical form. Finally, Fig. 19 shows some frames of a video clip in CAVIAR dataset [62], depicting the robustness of our tracker to the presence of multiple similar objects, varying scene illumination, clutter, object scaling, and partial occlusion.

5.2 Performance of stabilization algorithm

Figure 20 shows original vibratory aerial video frames (left side) versus stabilized video frames (right side) using the proposed algorithm. The un-stabilized video was recorded from a flying helicopter while tracking the truck using the proposed tracking algorithm and it contains jitters due to the helicopter vibration creating unpleasant effect on the eyes of the viewer. The large yellow crosshair in the video frames is overlaid to easily perceive the vibration in the

Table 2 Post-regression analysis for comparing accuracy of TCT and PCT

	S_1 sequence (300 frames)							S_2 sequence (360 frames)						
	x			y			Tracked frames	x			y			Tracked frames
	R (%)	m	C	R (%)	m	C		R (%)	m	C	R (%)	m	C	
TCT	94.8	1	−1.3	94.5	1	7.6	273	34.6	0.6	−0.8	−0.16	−0.4	258	92
PCT	99.8	1	−0.2	99.7	1	0.4	300	97.4	0.95	8.9	99.3	1	0.4	360

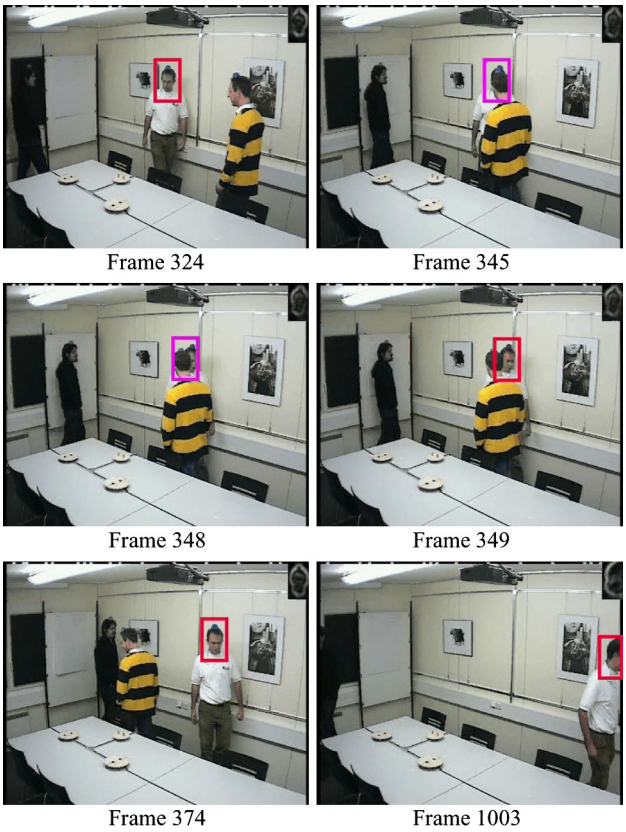


Fig. 15 The PCT is handling occlusion and clutter while tracking a person's face in a long video. The *red rectangle* indicates there is no occlusion and the algorithm is in normal tracking mode. When the algorithm detects and handles the occlusion, the *rectangle* color is changed to *pink* for demonstration

un-stabilized frames and its effective attenuation in the stabilized frames. In order to visualize the effects of stabilization on the whole image sequence, we logged the xy-coordinates of the left truck in the original and the stabilized video frames, because the vibratory motion of the truck can be considered as the vibratory motion of the whole scene. The coordinates are shown in Figs. 21 and 22, where it may be noted that the coordinates of the truck in the stabilized frames are smoother than those in the original vibratory video frames. Another example to show the efficacy of the stabilization algorithm is illustrated in Fig. 23.

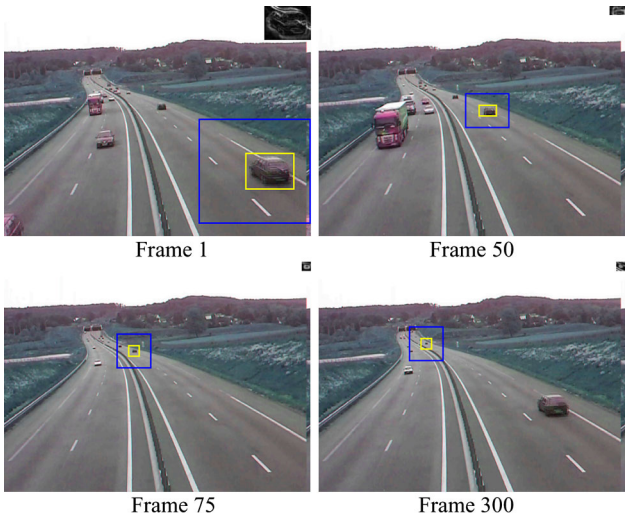


Fig. 16 A car is being tracked successfully by PCT, even when the scale (size) of the car is being reduced. The *yellow rectangle* represents BMR, and the *blue rectangle* shows the dynamic search window

Fig. 17 A car being tracked by PCT in the presence of background clutter and variation in the scale as well as the shape of the car

5.3 Performance of active camera tracking system

In this section, we present some experimental results of our ACTS to show its robust and accurate performance.

Fig. 18 The PCT tracks the face of the girl even during occlusion

Table 3 Comparison of the proposed algorithm with mean shift and condensation trackers

	seq_fast.avi video [53] (90 frames)	*seq_mb.avi* video [53] (105 frames)
Mean-shift tracker	Successful until frame 8	Successful until frame 53
Condensation tracker	Successful until frame 20	Successful until frame 74
Proposed tracker	Successful till the last frame	Successful till the last frame

Fig. 19 The proposed tracker is tracking a person in the presence of multiple similar objects, varying scene illumination, clutter, object scaling, and partial occlusion

Figure 6 shows some frames of a tracking session in which a helicopter is being tracked by our system. The BMR is represented by a white rectangle, and the frame center (i.e., optical axis of the camera) is represented by a white dot. The updated edge-enhanced template is overlaid at the bottom-right of every frame. The overlaid text at the top of the frames consists of the correlation peak value, the center coordinates of the BMR in the frame, zoom level of the camera, and finally the pan-tilt velocities of the camera mounted on a PTU in degrees/second. The pan velocity is positive, if the camera is rotating towards left. The tilt velocity is positive, if the camera is rotating downwards. We can see that the helicopter is automatically centralized very efficiently and smoothly in the video by increasing the pan velocity within the first 40 frames (i.e., 1.2 s), which is less than even the rise-time of our pan-tilt control system as

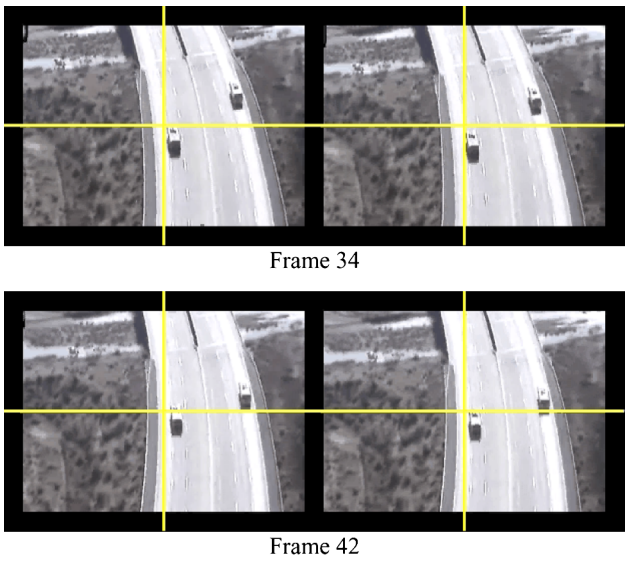


Fig. 20 Original (*left side*) versus stabilized (*right side*) frames of a video recorded from a vibratory flying helicopter

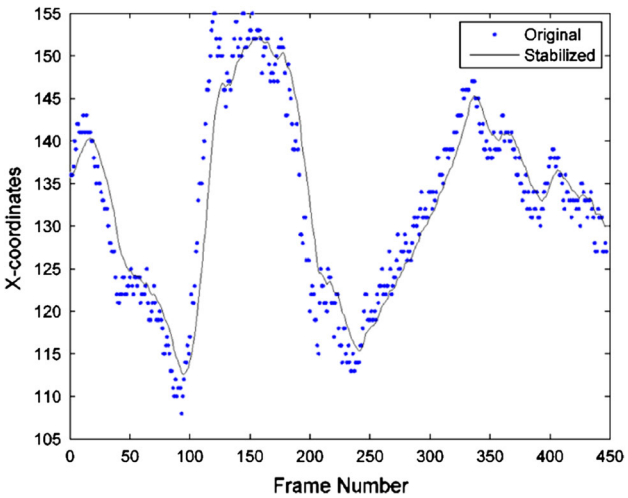


Fig. 21 Original versus stabilized *x*-coordinates of the left truck shown in Fig. 18

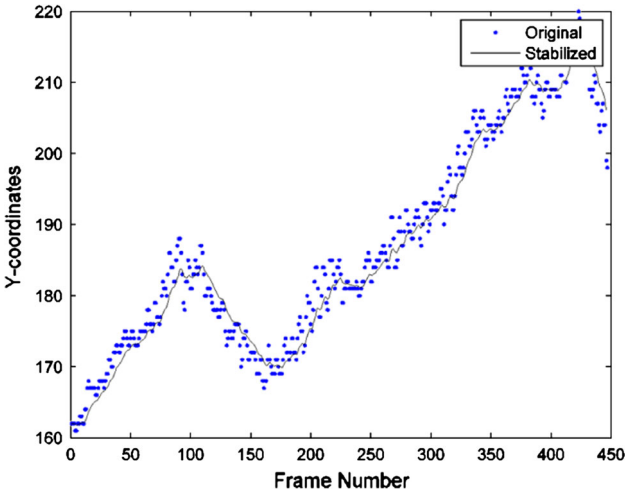


Fig. 22 Original versus stabilized *y*-coordinates of the left truck shown in Fig. 18

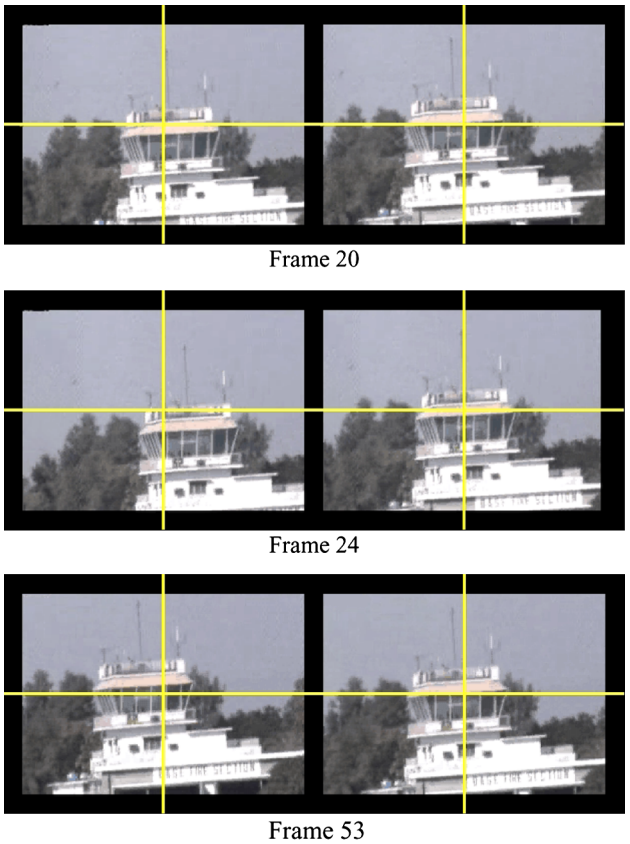


Fig. 23 Original (*left side*) versus stabilized (*right side*) frames of a video recorded from a vibratory hovering helicopter

mentioned in Sect. 3. After the initial automatic target centralization, the helicopter remains at the center of the frames throughout the tracking session, and it can be verified by the target coordinates shown at the overlaid text keeping in view that the frame size is 320 × 240. It may be noted that the helicopter is being tracked persistently and precisely with the proposed active camera tracking system even when: (1) the user had initialized the template inaccurately due to the motion of the helicopter in the video, and (2) the size, the appearance, and the velocity of the helicopter is continuously varying. The BMR-adjustment algorithm solves the incorrect initialization problem by resizing/relocating the BMR so that it tightly encloses the target very efficiently within the first 20 frames. Later on, the BMR is further dynamically resizing/relocating itself according to the current size of the helicopter by both the scale-handling method (Sect. 2.3) and the BMR-adjustment algorithm (Sect. 2.6).

Figure 24 illustrates how efficiently the proposed system tracks the face of a person, who is walking in a room with all the lights turned off. The only light, that was available in the room, was coming from the blinds shown in the frames. This natural light created a severe illumination variation in the video, since the camera was operating in its auto mode. Specifically, when the camera was looking in the direction of the bright window, the other things (persons, wall, etc.) became very dark (see frames 271 to 512), and when there was no bright window in the video frames, the whole scene became a little clearer. It may also be noted that there is noise and no detail in the whole video due to low light conditions. The target person and the occluding person are both walking in the *same* direction making the scenario even more complex. It can be further observed in frame 495, that the occlusion of the tracked person by the other person happens partly in the bright region and partly in the dark region of the video frame. Moreover, the track of the target person after the occlusion is resumed in *very much dark*, as shown in frame 512. Since the persons were very near to the camera, even a small movement of the persons was reflecting a large movement in the video frames. Thus, it was a challenging experiment for the pan-tilt control algorithm as well. All the problems (i.e., severe illumination variation, noise, low detail, full occlusion, and fast motion) are handled very efficiently and robustly by the proposed ACTS in real-time, and the face of the person of interest is always at (or near) the center of the video frames.

5.4 Performance of stabilized active camera tracking system

In this section, we demonstrate the results of our complete stabilized ACTS.

Figure 25 shows some of the frames of a long active camera tracking and stabilization session of a very distant airplane at 25× (highest) zoom level of the camera. We

Fig. 24 Tracking the face of a person during severe illumination variation, noise, low detail, and occlusion. All the lights in the room were turned off in this experiment to create a challenging scenario. The *dark yellow rectangle* in frame 495 indicates that the tracker is currently working in its occlusion handling mode

recorded the un-stabilized as well as the stabilized video frames in real-time for demonstration purpose. Left column depicts the resulting tracking video frames *without* stabilization, while right column shows the resulting tracking video frames *with* stabilization. Periodic vibratory force at the rate of 1 Hz was used to induce vibration on the PTU and thus on the real-time video. In the un-stabilized video fames, visual tracking and control algorithms always try to keep the target at the center of the image plane, but due to vibration the airplane is oscillating about the frame center. Video frames at the right column shows that the stabilization module of our system has successfully diminished the vibration in the target being tracked.

In Fig. 26, a pedestrian is being tracked in a cluttered environment at 11× zoom level. The vibratory source in this case is a Toyota 2400 cc Hilux engine working at 500 rpm. The synchronized tracking video frames without and with stabilization are again shown in the left and the right columns, respectively. Images in the left highlight the to-and-fro motion of the scene, because the ACTS is mounted on a vibratory vehicle. Images in the right column show that the tracked person and the varying background scene are stable and the vibration is significantly attenuated.

6 Conclusions

We have proposed a robust stabilized ACTS, consisting of a visual tracking module, a pan-tilt control module, and a video stabilization module. The visual tracking module can handle template drift, noise, object fading (obscuration), clutter, intermittent occlusion, varying illumination in the scene, high computational complexity, and varying shape, scale, and velocity of the manoeuvring target during its

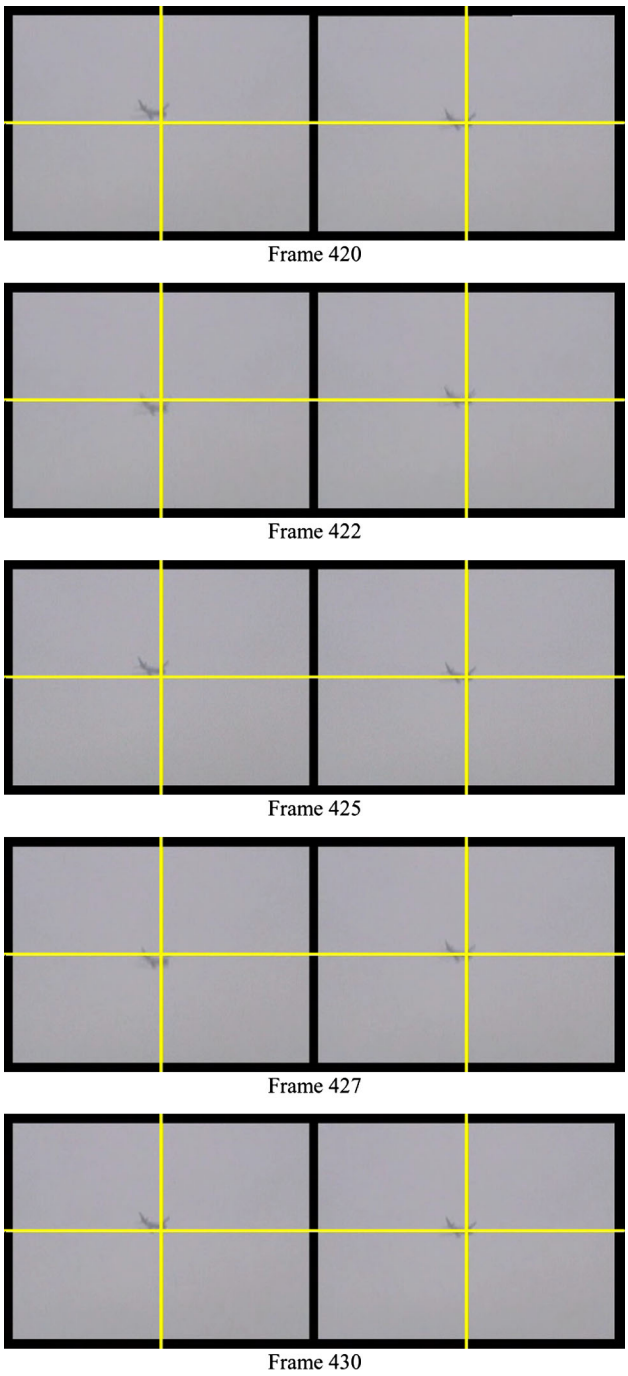


Fig. 25 Results of un-stabilized (*left column*) versus stabilized active camera tracking (*right column*) of a distant airplane

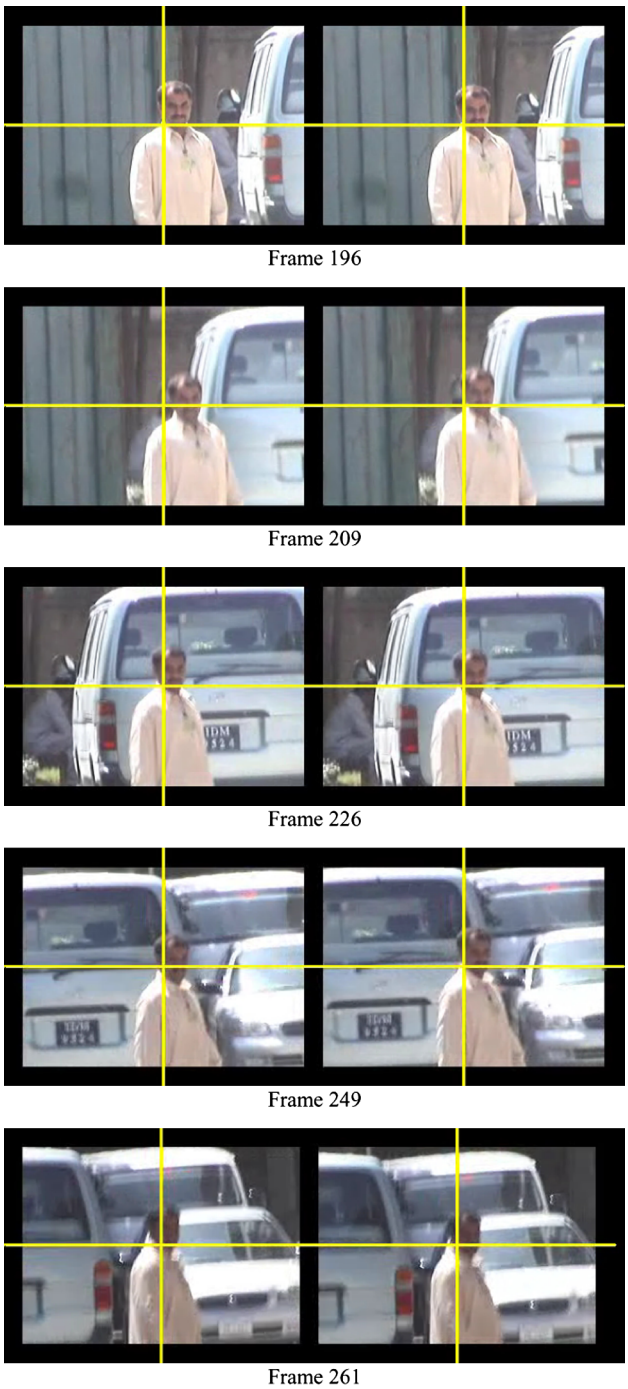


Fig. 26 Results of un-stabilized (*left column*) versus stabilized active camera tracking (*right column*) of a pedestrian

motion. The proposed pan-tilt control module is a POL-CFC, which moves the camera efficiently and smoothly so that the object being tracked is always at the center of the video frame. The control algorithm offers 0 % overshoot, negligible steady-state error, and 1.47 s rise-time. The video stabilization module handles the annoying vibratory motion in the image frame during tracking while the system is mounted on a vibratory platform (e.g., vehicle, helicopter, etc.). The complete proposed system has been successfully used for more than 2 years in indoor as well as out door scenarios, and it works in real-time at the full frame rate of 30 fps.

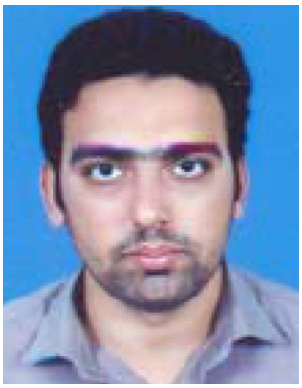
References

1. Kumar, P., Dick, A., Sheng T.S.: Real Time Target Tracking with Pan Tilt Zoom Camera. In: Proceedings of IEEE Conference on Digital Image Computing: Techniques and Applications (DICTA), December 2009
2. Dinh, T., Yu, Q., Medioni, G.: Real Time Tracking using an Active Pan-Tilt-Zoom Network Camera. In: IEEE International Conference on Intelligent Robots and Systems, October 2009
3. Cui, Y., Samarasekera, S., Huang, Q., Greienhagen M: Indoor Monitoring Via the Collaboration Between a Peripheral Sensor and a Foveal Sensor. In: IEEE Workshop on Visual Surveillance, pp. 2–9 (1998)
4. Bradski, G.R.: Computer Vision Face Tracking as a Component of a Perceptual User Interface. In: Proceedings of IEEE Workshop on Applications of Computer Vision, Princeton, pp. 214–219 (1998)
5. Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: PFinder: real-time tracking of the human body. IEEE Trans. Pattern Anal. Mach. Intell. **19**, 780–785 (1997)
6. Intille, S.S., Davis, J.W., Bobick, A.F.: Real-Time Closed-World Tracking. IEEE Conference on Computer Vision and Pattern Recognition, pp. 697–703 (1997)
7. Ahmed, J., Shah, M., Miller, A., Harper, D., Jafri, M.N.:A Vision Based System for a UGV to Handle a Road Intersection. In: Proceedings of AAAI-07, 22nd Conference on Artificial Intelligence, 22–26 July 2007
8. Eleftheriadis, A., Jacquin, A.: Automatic face location detection and tracking for model-assisted coding of video teleconference sequences at low bit rates. Signal Process. Image Commun. **7**(3), 231–248 (1995)
9. Rosales, R., Sclaro, S.: 3D Trajectory Recovery for Tracking Multiple Objects and Trajectory Guided Recognition of Actions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, vol. 2, pp. 117–123 (1999)
10. Doulamis, A., Doulamis, N., Ntalianis, K., Kollias, S.: An Efficient Fully Unsupervised Video Object Segmentation Scheme Using an Adaptive Neural-Network Classifier Archtecture. In: Proceedings of IEEE Transactions on Neural Networks, May 2003
11. Cuevas, E.V., Zaldivar, D., Rojas, R.: Intelligent Tracking. Technical Report B-03-15, Freie Universitat Berlin, Germany, 10 November 2003
12. Ahmed, J., Jafri, M.N., Ahmad, J., Khan, M.I.: Design and Implementation of a Neural Network for Real-Time Object Tracking. In: Proceedings of Machine Vision and Pattern Recognition Conference in 4th World Enformatika Conference, 2005
13. Ahmed, J., Jafri, M.N., Ahmad, J.: Target Tracking in an Image Sequence Using Wavelet Features and a Neural Network. In: Proceedings of IEEE Region 10: Tencon'05 Conference, 2005
14. Stauffer, C., Grimson, W.: Learning patterns of activity using real time tracking. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 747–767 (2000)
15. Wang, H., Suter, D., Schindler, K.: Effective Appearance Model and Similarity Measure for Particle Filtering and Visual Tracking. In: Proceedings of European Conference on Computer Vision, 2006
16. Perez, P., et al.: Color-Based Probabilistic Tracking. In: Proceedings of European Conference on Computer Vision, pp. 661–675 (2002)
17. Yilmaz, A., Li, X., Shah, M.: Contour-based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1531 (2004)
18. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. Int. J. Comput. Vis. **1**, 321–331 (1988)
19. Bleser, G., Becker, M., Stricker, D.: Real-time vision-based tracking and reconstruction. J. Real-Time Image Proc. **2**, 165–171 (2007). doi:10.1007/s11554-007-0034-0
20. Peddigari, V., Kehtarnavaz, N.: Real-time predictive zoom tracking for digital still cameras. J. Real-Time Image Process. **2**, 45–54 (2007). doi:10.1007/s11554-007-0036-y
21. Comaniciu, D., Visvanathan, R., Meer, P.: Kernel based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–575 (2003)
22. Comaniciu, D., Ramesh, V., Meer, P.: Real-time Tracking of Non-Rigid Objects Using Mean Shift. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head, SC, vol. 1, pp. 142–149 (2000)
23. Porikli, F., Tuzel, O., Meer, P.: Covariance Tracking using Model Update Based on Lie Algebra. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2006
24. Porikli F., Tuzel, O.:Multi-kernel Object Tracking. In: Proceedings of IEEE International Conference on Multimedia and Expo, 2005
25. Adam, A., Rivlin, E., Shimshoni, I.: Robust Fragments-based Tracking Using the Integral Histogram. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2006
26. Porikli, F.: Integral Histogram: A Fast Way to Extract Histograms in Cartesian Spaces. IEEE Conference on Computer Vision and Pattern Recognition, 2005
27. Nixon, M., Aguado, A.: Feature Extraction and Image Processing. Newnes, Oxford (2002)
28. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd Edn. Prentice-Hall Inc., USA (2002)
29. Ritter, G.X., Wilson, J.N.: Handbook of Computer Vision Algorithms in Image Algebra. CRC, Boca Raton (1996)
30. Kuglin, C., Hines, D.: The Phase Correlation Image Alignment Method. In: Proceedings on International Conference on Cybernetics and Society, pp. 163–165 (1975)
31. Wong, S.: Advanced Correlation Tracking of Objects in Cluttered Imagery. In: Proceedings of SPIE, vol. 5810 (2005)
32. Lewis, J.P.: Fast Normalized Cross-Correlation. Industrial Light and Magic, 1995
33. Ahmed, J., Jafri, M.N., Shah, M., Akbar, M.: Real-time edge-enhanced dynamic correlation and predictive open-loop car following control for robust tracking. Mach. Vis. Appl. J. **19**(1), 1–25 (2008)
34. Kuo, B.C.: Automatic Control Systems, 7th Edn. Wiley, USA (1995)
35. Ross, T.J.: Fuzzy Logic with Engineering Applications, 2nd edn. Wiley, England (2004)
36. Nguyen, H.T., Prasad, N.R., Walker, C.L., Walker, E.A.: A First Course in Fuzzy and Neural Control. Chapman & Hall/CRC, Boca Raton (2003)
37. Mir-Nasiri, N.: Camera-based 3D Tracking. In: Proceedings of IEEE Region 10: Tencon'05 Conference, Melbourne, Australia, 2005
38. Basic Control Law for PTU to Follow a Moving Target. Application Note 01, Directed Perception Inc., 1996
39. E. Vermeulen, "Real-Time Video Stabilization for Moving Platforms," 21st Aristotle UAV Systems Conference, April 2007
40. Tico, M., Vehvilainen, M.: Robust Method of Video Stabilization, In: EUSIPCO-07: European Signal and Image Processing Conference (2007)
41. Hu, R., Shi, R., Shen, I.f., Chen, W.: Video Stabilization using Scale Invariant Features. IV'07: In: Proceedings of IEEE 11th International Conference on Information Visualization (2007)
42. Inc., C., "Canon FAQ: What is vari-angle prism?" available at: http://www.canon.com/bctv/faq/vari.html (2011). Accessed 18 June, 2011

43. Battiato, S., Gallo, G., Puglisi, G.: Fuzzy-Based Motion Estimation for Video Stabilization Using SIFT Interest Points. In: Proceedings of SPIE, vol. 7250, 72500T; doi: 10.1117/12.805660 (2009)
44. Chang, H.-C., Lai, S.-H., Lu, K.-R.: A Robust and Efficient Video Stabilization Algorithm. ICME-'04: In: Proceedings of IEEE International Conference on Multimedia and Expo, June 2004
45. "Optical Flow", available at: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html (2011). Accessed 18 June, 2011
46. Buehler, C. Bosse, M., McMillian, L.: Non-metric Image Based Rendering for Video Stabilization. In: Proceedings of CVPR-01, IEEE Conference on Computer Vision and Pattern Recognition (2001)
47. Auberger, S., Miro, C.: Digital Video Stabilization Architecture for Low Cost Devices. In: Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis p. 474 (2005)
48. Jang, S.-W., Pomplun, M., Kim, G.-Y., Choi, H.-I.: Adaptive Robust Estimation of Affine Parameters from Block Motion Vectors. In: Image and Vision Computing, pp. 1250–1263, August 2005
49. Vella, F., Castorina, A., Mancuso, M., Messina, G.: Digital image stabilization by adaptive block motion vectors filtering. IEEE Trans. Consum. Electron. **48**, 796–801 (2002)
50. Blackman, S., Popoli, R.: Design and Analysis of Modern Tracking Systems., pp. 309–313. Artech House, Boston (1999)
51. Fagiani C., Gips, J.: An Evaluation of Tracking Methods for Human-Computer Interaction, Senior Thesis, Computer Science Department, Boston College, Fulton Hall, Chestnut Hill, MA 02467 (2002)
52. Brookner, E.: Tracking and Kalman Filtering Made Easy. Wiley, USA (1998)
53. BMP Image Sequences for Elliptical Head Tracking: http://vision.stanford.edu/~birch/headtracker/seq/ (2011). Accessed 18 June, 2011
54. Nummiaroa, K., Koller-Meierb, E., Gool, L.V.: An adaptive color-based particle filter. Image Vis. Comput. **21**, 99–110 (2003)
55. Isard, M., Blake, A.: CONDENSATION-conditional density propagation for visual tracking. Int. J. Comput. Vis. **29**(1), 5–28 (1998)
56. Bradski, G., Kaehler, A., Pisarevsky, V.: Learning-based computer vision with open source computer vision library. Intel. Technol. J. **9**(2) (2005)
57. Crow, F.: Summed-area tables for texture mapping. Comput. Graph. **18**(3), 207–212 (1984)
58. Ahmed, J., Jafri, M.N.: Best-Match Rectangle Adjustment Algorithm for Persistent and Precise Correlation Tracking. In: Proceedings of IEEE International Conference on Machine Vision, 28–29 December 2007
59. Available at: http://www.fastpasses.com (2005). Accessed 19 July, 2005
60. Lathoud, G., Odobez, J., Gatica-Perez, D.: AV16.3: an Audio-Visual Corpus for Speaker Localization and Tracking. LNCS 3361, pp. 182–195, Springer-Verlag, Berlin (2005)
61. PETS 2001 dataset, publicly available at the website: http://www.research.ibm.com/peoplevision/performanceevaluation.html (2006). Accessed 11 November, 2006
62. CAVIAR (Context Aware Vision using Image-based Active Recognition) test video clips 2003-2004, available at http://homepages.inf.ed.ac.uk/rbf/CAVIAR (2003). Accessed 23 December, 2003

Author Biographies

Javed Ahmed received his bachelor's degree in Electronics Engineering from NED University of Engineering and Technology, Karachi (Pakistan), in 1994. Then, he did MSc in Systems Engineering with distinction under the fellowship program at Pakistan Institute of Engineering and Applied Sciences, Islamabad (Pakistan), in 1997. He joined National Engineering and Scientific Commission in 1997 and worked in the areas of embedded systems and signal processing. He got his Ph.D. in Electrical (Telecom) Engineering from National University of Sciences and Technology, Islamabad, Pakistan, in May 2008. He also conducted an 8-month funded joint research with Computer Vision Lab at the University of Central Florida from July 2006 to February 2007. His current research areas are image processing, computer vision, signal processing, and soft computing (i.e., ANN, FL, and GA). He has published 10 international conference papers (two of them in AAAI-07 and CVPR-08) and two international journal papers (one of them in MVA). He is a member of IEEE, included in Who's Who in America 2011 (65th edition), and selected to be included also in Who's Who in Asia 2012 (2nd edition).



Ahmad Ali did his bachelor's degree in Computer Sciences (Hons.) from University of Engineering and Technology (UET), Lahore, Pakistan in 2002. He won National Fellowship Competition Award 2003–2005 and completed his MSc degree in Systems Engineering from Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan in 2005. He joined Centers of Excellence in Science and Applied Technologies (CESAT), Islamabad in 2005. He got IT and Telecom Endowment Fund Scholarship Award for Ph.D. in 2010. Currently he is doing Ph.D. from PIEAS. His areas of interest are image processing, computer vision, object tracking, artificial intelligence, and speech processing.

Asifullah Khan did MSc Physics in 1996 from University of Peshawar, Peshawar, Pakistan. He won National Fellowship Competition Award 1996–1998 and completed his M.Sc. Nuclear Engineering degree from PIEAS, Islamabad, Pakistan in 1998. He did MS in Computer System Engineering in 2003 from GIK Institute, Topi, K-Pakhtunkhwa, Pakistan. He completed his Ph.D. in Computer Systems Engineering in 2006 from GIK Institute, Topi, K-Pakhtunkhwa, Pakistan. He did Post-Doc Computer Science in 2009 from GIST, Gwangju, South Korea.