Machine Vision and Applications (2008) 19:1–25
DOI 10.1007/s00138-007-0072-4

ORIGINAL PAPER

Real-time edge-enhanced dynamic correlation and predictive open-loop car-following control for robust tracking

Javed Ahmed · M. N. Jafri · Mubarak Shah · Muhammad Akbar

Received: 31 May 2006 / Accepted: 22 November 2006 / Published online: 5 May 2007

Abstract We present a robust framework for a real-time visual tracking system, based on a BPNN-controlled fast normalized correlation (BCFNC) algorithm and a predictive open-loop car-following control (POL-CFC) strategy. The search for the target is carried out in a *dynamically* generated resizable search-window. In order to achieve the robustness, we use some *edge-enhancement* operations before the correlation operation, and introduce an adaptive template-updating scheme. The proposed tracking algorithm is compared with various correlation-based techniques and (in some cases) with the mean-shift and the condensation trackers on real-world scenarios. A significant improvement in efficiency and robustness is reported. The POL-CFC algorithm *approximates* the current velocity of an open-loop pan-tilt unit, computes the predicted relative-velocity of the object using Kalman filter, and generates the precise control signals to move the camera accurately towards the maneuvering target regardless of its changing velocity. The proposed system works in real-time at the speed of 25–200 frames/second depending on the template size, and it can persistently track a distant or near object even in the presence of object fading, low-contrast imagery, noise, short-lived background clutter, object-scaling, changing object-velocity, varying illumination, object maneuvering, multiple objects, obscuration, and sudden occlusion.

Keywords Visual tracking · BPNN-controlled fast normalized correlation · Dynamic search-window · Robust template-updating · Predictive open-loop car-following control

J. Ahmed (✉)
Computer Vision Lab, University of Central Florida,
Orlando, FL 32816, USA
e-mail: javed@ieee.org

J. Ahmed
Department of Electrical (Telecom.) Engineering,
Military College of Signals, Adiala Road, Rawalpindi,
46000, Pakistan

M. N. Jafri
Electrical (Telecom.) Engineering Department,
National University of Sciences & Technology,
Rawalpindi, Pakistan
e-mail: mnjafri@mcs.edu.pk

M. Shah
School of Electrical Engineering & Computer Science,
University of Central Florida, Orlando, FL 32816, USA
e-mail: shah@cs.ucf.edu
URL: http://www.cs.ucf.edu/∼vision

M. Akbar
Engineering Division, National University
of Sciences & Technology, Rawalpindi, Pakistan
e-mail: makbar_mcs@nust.edu.pk

1 Introduction

In a typical visual tracking system, the video frames acquired from the camera are analyzed to find the (x, y) pixel-coordinates of the object of interest. The coordinates are then directly (or indirectly using a state estimator, e.g. Kalman filter) sent to a control algorithm which moves a pan-tilt unit (PTU) so that the camera (which is fitted upon the PTU) can move towards the object. As a result, the object is always projected at the center of the video frames.

Efficient tracking of an object in complex environments is a challenging task for the machine vision community. Some widely known applications of real-time visual tracking system are surveillance and monitoring [15], perceptual user interfaces [7], smart rooms [40,48], and video compression [18]. The computational complexity of the tracker is critical for most applications; only a small percentage of the system resources can be allocated for tracking while the rest is assigned to preprocessing stages or high-level tasks such as recognition, trajectory interpretation, and reasoning [39].

Several techniques have been proposed by the researchers for target tracking in the consecutive video frames. Most of these are either limited to tracking specific class of objects [2,3,14,17], or assume that the camera is stationary (and exploit back-ground subtraction) [41,49]. The trackers based on the particle filter or condensation [25,33,34,45] and active contours [26,49] do not assume constant background. They are reported to track the whole object instead of only the centroid or a portion of the object [49]. However, keeping in mind the present power of a high-end computer, they are computationally too expensive to be exploited for a practical real-time tracking application. The mean shift tracker [11,12] has gained a significant influence in the computer vision community in recent years, because it is fast, general-purpose and does not assume static background. Mean-shift is a nonparametric 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 [37]. There are variants, e.g. [35], to improve its localization by 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 highly depends on the discriminating power of the histograms that are considered as the objects' probability density function [37]. Another issue in the mean shift tracker is inherent in its use of histogram, which does not carry the spatial information of the pixels [1]. The integral histogram based tracker [36] 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 neighborhood of the previous target-position. On a P4 3.2 GHz machine, this tracker works with the speed of about 18 fps, and the mean shift tracker works with the speed of about 66 fps [36]. 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 [37], 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 half of the object information is lost (which can be crucial for a very small target) and 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 [37]) is only 7 frames/second on a high-end PC (P4, 3.2 GHz).

There are also the widely used classic trackers, such as the edge tracker, the centroid tracker, and the correlation tracker. A good introduction to these trackers can be found in [6], where it is reported that the correlation tracker has proved to be the most robust of the three, especially in a noisy and cluttered scene. However, the *standard* correlation tracker has also some inherent problems. Firstly, it is prone to the template-drift problem; secondly, its performance tremendously deteriorates in the presence of varying illumination conditions; thirdly, if the template is kept constant throughout the tracking session, the detection performance declines especially when the object changes its shape, size, and orientation. Therefore, it is not adequately robust without some preprocessing, adaptive template, and above all the modification in the basic correlation formulation [6,47]. As far as its implementation is concerned, the correlation operation in *spatial* domain can be computationally expensive, if the search-window-size or the template-size or both are too large. In order to speed up the computation, the *standard* correlation can be implemented in frequency domain using the convolution theorem of the discrete Fourier transform [6,23,47]. However, the modified correlation metrics, which are more robust than the standard correlation, have no direct counterparts in frequency domain. Moreover, it is not necessary that the correlation in frequency domain is always faster than its spatial domain implementation, as pointed out in Sect. 2.

This paper (i.e. its sections on target detection) focuses on enhancing the efficiency and robustness of the classic correlation tracker by addressing its problems mentioned above. The correlation metrics are reviewed in Sec. 2, in which we introduce BPNN-controlled fast normalized correlation (BCFNC) algorithm. The BPNN (back-propagation neural network) decides if the correlation should be performed in spatial domain or in frequency domain to achieve the desired efficiency. If the correlation is performed in frequency domain, it is efficiently normalized exploiting the concept of summed-area-table (SAT) or integral image [13,28,36], which allows us to compute the normalizing factor using only four addition operations. Later on, we also propose a novel and effective template *updating* scheme which changes the template gradually with time using the history of the template and the current best-match. This template updating scheme minimizes the template drift phenomenon, and copes with short-lived occlusion and background clutter. We also introduce a novel algorithm to dynamically determine the position and size of the search window using the prediction and the prediction-errors of a Kalman filter.

The PTU (pan-tilt unit) motion control algorithm is also an integral part of a successful target tracking system. If the control is not smooth and precise, the object in the video will oscillate to and fro around the center, and in the worst case it may get out of the frame. One approach is to use a classic PID controller; however, its design is rather complex and requires a mathematical model of the system (to be controlled). Besides, it necessitates a sensitive and rigorous tuning of its proportional, differential and integral gain parameters, especially when they are to be optimized for use with all the zoom levels of the camera. An alternative approach is to use a fuzzy controller that does not require the system model [14], 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 [24], but it is heavily dependent on the *quality* and the *variety* of the *examples* in the training dataset, which can accurately represent the complete behavior of the controller in *all* possible scenarios, including the varying zoom-levels of the camera. Furthermore, the traditional control algorithms, e.g. [32], are generally implemented based on the difference between the centre (reference) position and the current target position in the image. They 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*). We present a predictive open-loop car-following control (POL-CFC) algorithm. Its basic idea is borrowed from the car-following control (CFC) strategy.[1] The CFC assumes that the actual velocity of the PTU is *observable* through a velocity sensor. However, our POL-CFC does not make this assumption and simply *approximates* the current PTU velocity as the previous velocity command. Then, it computes the relative velocity of the target from the predicted position provided by the Kalman filter, and 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 offers 0% overshoot, 0 steady-state tracking error, and 1.7 s rise-time.

We have implemented the proposed tracking system using multiple threads. The image processing operations and Kalman filter are implemented in one thread and the PTU motion control algorithm is implemented in another thread. This approach exploits the parallel processing power available in the micro-processor in a standard PC. As a result, the processing speed of 25 to 200 frames per second (depending on the size of the template) has been achieved, when the frame size is 320×240 pixels. The proposed system has been field-tested for more than a year for indoor and outdoor experiments, where it could reliably track a distant or near object even in the presence of temporary object fading, significant background clutter, object-scaling, changing object-velocity, varying illumination conditions, significant object maneuvering, multiple objects, obscuration, short-lived occlusion, and low-contrast imagery.

[1] Basic Control Law for PTU to Follow a Moving Target, Application Note 01, Directed Perception Inc (1996)

The paper is organized as follows. Sect. 2 reviews various correlation-based target detection techniques, and explains the proposed methodology in detail. The assumed state-space model of the target dynamics for the Kalman predictor is discussed in Sect. 3. Section 4 presents a novel approach for creating the dynamic search window. The evaluative comparison is performed in Sect. 5. Section 6 describes the camera motion control in detail. The experimental results of the proposed target tracking system are presented in Sect. 7. The implementation of the overall system is summarized in Sect. 8. Finally, we present some future directions and conclude in Sect. 9.

2 Correlation-based template matching

In this section, we present a brief review of the correlation metrics for the template matching application, and discuss the proposed BPNN-controlled fast normalized correlation (BCFNC). The BCFNC is an integral part of the proposed target tracking algorithm, which is explained and compared with the other methods in Sect. 5.

2.1 Correlation metrics

Consider a template (a gray-level image of the object), t, and a search window (a gray-level image in which the object is supposed to exist), s. The position of the target can be detected by matching the template with every template-size section in the search window. There are various template matching schemes, but we focus on only the correlation-based methods. The standard 2-D correlation metric is given as [6,23]:

$$c(m,n) = \sum_{i=0}^{K-1}\sum_{j=0}^{L-1} s(m+i, n+j)t(i,j), \tag{1}$$

where $c(m,n)$ is the element of the correlation surface (i.e. matrix) at row m and column n, K is the number of rows in the template, L is the number of columns in the template, $m = 0, 1, 2, \ldots, M-K+1$, $n = 0, 1, 2, \ldots, N-L+1$, M is the number of rows in the search window such that $M \geq K$, and N is the number of columns in the search window such that $N \geq L$.

The standard correlation can be efficiently computed in frequency domain as:

$$c = real(ifft(S.*T^*)), \tag{2}$$

where S and T are the 2-D discrete Fourier transforms (DFT) of s and t, respectively. The superscript $(^*)$ over T indicates its conjugate, the symbol $(.^*)$ indicates the element-by-element multiplication, $ifft(.)$ is the inverse fast Fourier transform function, and the $real(.)$ function extracts the real part of a complex matrix. The real part is extracted because the imaginary part of the resulting complex matrix is almost zero, since s and t are real 2-D signals. Note that s and t must be appropriately zero padded before getting their transforms to obtain correct results, because of the periodic nature of the *discrete* Fourier transform [23]. The minimum size of the zero-padded images should be $P \times Q$, where $P = M + K - 1$ and $Q = N + L - 1$.

Once the correlation surface is built, the maximum value in the correlation matrix, $c_{\max}$, is found at (m^*, n^*) which indicates the position of the top-left corner of the *best-match* (i.e. the image section that matches best with the template) in the search window.

However, the standard correlation described above is highly sensitive to varying illumination conditions because it produces $c_{\max}$ at the brightest spot in the search image. Furthermore, the correlation value is dependent on the size of the template, and is not normalized to be in the range $[-1.0,\ 1.0]$. Thus, we can not have an absolute measure of confidence for further decision making (e.g. when to update the template, or when to vary the size of the search window, etc.). Therefore, some researchers, e.g. [47], use the normalized correlation (NC):

$$c(m,n) = \frac{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1} s(m+i,n+j)t(i,j)}{\sqrt{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1} s^2(m+i,n+j)}\sqrt{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1} t^2(i,j)}}. \tag{3}$$

The numerator of NC is the same as (1) which is normalized by two factors. These are basically the square-roots of the energies of the image section and the template itself, respectively. The NC has two salient features. Firstly, it is less sensitive to varying illumination conditions than (1). Secondly, its values are within the range $[-1.0,\ 1.0]$, therefore further decision making is possible for template-updating, etc. However, its counterpart in the frequency domain does not exist, so it is computationally more expensive than (1).

Additionally, there is a normalized correlation *coefficient* (NCC) [6,22,23,28], that is more robust to varying illumination conditions than (3), and its values are also normalized within the range $[-1.0,\ 1.0]$. It is defined as:

$$c(m,n) = \frac{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1} [s(m+i,n+j)-\mu_s][t(i,j)-\mu_t]}{\sqrt{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1} [s(m+i,n+j)-\mu_s]^2}\sqrt{\sum_{i=0}^{K-1}\sum_{j=0}^{L-1} [t(i,j)-\mu_t]^2}}, \tag{4}$$

where μ_s and μ_t are the mean intensity values of the image section and the template itself, respectively. However, this metric has two disadvantages. Firstly, it requires that the intensity values of s or t must not be constant, otherwise the correlation value will be infinity or indeterminate (but this problem is not so serious in real-world imagery because of the inherent sensor noise). Secondly, its direct implementation is computationally more expensive than (3) (but there is an efficient method [28] to calculate it using FFT and the concept of summed-area table [13]).

Some researchers have used symmetric phase-only matched filter (SPOMF) or *phase correlation* [10,27,38], defined as:

$$c = real\big(ifft\big((S./\|S\|).*\big(T^*./\|T\|\big)\big)\big), \tag{5}$$

where $\|.\|$ operator computes the magnitude of every complex number in its input matrix, and the ./ is the *element-by-element* division operator. In this approach, the transform coefficients are normalized to unit magnitude prior to computing correlation in the frequency domain. Thus, the correlation is based only on the phase information and is insensitive to changes in image intensity. It has an interesting property that it yields a sharp peak at the best-match position and attenuates all the other elements in the correlation surface to almost zero. Although this approach has proved to be successful, it has a drawback that all transform components are weighted equally, whereas one might expect that insignificant components should be given less weight [28]. It is shown in [42,43] and this paper that the SPOMF may produce false alarms, and a small peak (significantly less than 0.5) even at the correct position depending upon the scene content. Moreover, it does not provide us with a reliable confidence metric for further decision making. However, its false-alarm rate has been reduced to some extent in [47] by phase-correlating the *edge images* of the search window and the template, rather than their gray-level images.

2.2 BPNN-controlled fast normalized correlation (BCFNC)

In this section, we discuss the proposed BCFNC algorithm, which is used to perform the template matching in our target detection algorithm (to be discussed next).

2.2.1 *Efficient implementation of NC using FFT and SAT*

The template matching process with NC in spatial domain using (3) is a computation intensive process. Therefore, we

use a faster implementation using FFT and (summed-area table) (SAT) [13]. The same method has been exploited in [28], but for implementing the normalized correlation *coefficient* given by (4). The idea is that the numerator of (3) is computed in frequency domain using (2), and the second factor in the denominator is pre-calculated only once for each video frame (because the template remains same for an entire iteration). However, the first factor varies with (m, n), so it has to be calculated for every section of the search image under the template. It is basically the square-root of the *local energy* of the search window under the template. For efficient calculations, we use the concept of SAT [13], which was used in [28,36], in which it is equivalently referred to as "running sum image" or "integral image".
The SAT of the $M \times N$ search window is a matrix of the size $(M+1) \times (N+1)$. The elements in its 0th row and column are initialized to 0. All other elements are calculated recursively as:

$$I(i, j) = s(i-1, j-1) + I(i, j-1) + I(i-1, j) - I(i-1, j-1), \tag{6}$$

where I is the SAT, s is the search window, $i = 1, 2, \ldots, M$, and $j = 1, 2, \ldots, N$. Once the SAT is computed, we can easily calculate the sum of all the elements in any rectangular section in the search-window by algebraically adding only the four corner elements of the corresponding rectangular section in its SAT. For example, if we want to calculate the sum of elements contained in an $L \times K$ rectangular section (in the search window) with top-left element $s(i, j)$, top-right element $s(i, j+L-1)$, bottom-right element $s(i+K-1, j+L-1)$, and bottom-left element $s(i+K-1, j)$, the sum is computed from its SAT as:

$$sum = I(i+K, j+L) + I(i, j) - I(i+K, j) - I(i, j+K). \tag{7}$$

We need to calculate the *local energies* (instead of the *local sums*) of the search window. Thus, we first compute the SAT of the *square* of the search window, and later find the local sums of the *squared* search image using *its* SAT. As a result, we get a matrix of size $(M-K+1) \times (N-L+1)$ containing the local energies of the original search-window. Note that the size of this matrix is exactly the same as that of c. If we compute the square-root of the elements of this matrix, and multiply the resulting matrix with the pre-calculated "second factor of the denominator of (3)", we get the normalizing matrix. Finally, if we divide the numerator of (3) (already computed using FFT) by this normalizing matrix *element-by-element*, we obtain the *normalized* correlation surface, c.

2.2.2 Performance comparison

Let t_p be the time (in ms) required for the proposed efficient implementation of NC, and t_d be the time (in ms) required for direct implementation. Then, the *speed gain* of the proposed implementation is calculated as:

$$G_\mathrm{p} = \frac{t_\mathrm{d}}{t_\mathrm{p}}. \tag{8}$$

Furthermore, let S_t, S_s, and R_ts be defined as:

$$S_\mathrm{t} = \sqrt{KL}, \quad S_\mathrm{s} = \sqrt{MN}, \quad \text{and } R_\mathrm{ts} = \frac{S_\mathrm{t}}{S_\mathrm{s}}. \tag{9}$$

Assuming that the width and height of the zero-padded images are optimal integers greater than or equal to P and Q (Sect. 2.1) respectively, it has been observed that G_p is a nonlinear function of mainly S_t and R_ts, as shown in the surface plot in Fig. 1. The surface plot has been obtained by experimentally acquiring the speed gain for $S_\mathrm{s} = 40, 80, 120, \ldots, 600$, and $R_\mathrm{ts} = 0.025, 0.05, 0.075, \ldots, 1.0$ for every value of S_s. If P and Q, individually, are power of 2, or if they contain only small prime factors (e.g. 2, 3, or 5), then the FFT computation is very efficient, and the *speed gain* is drastically increased as illustrated by various peaks in the surface plot. For example, when S_t is 153 and S_s is 360, R_ts becomes 0.425, and the size of the zero-padded images is set to be 600×600 (Note that 600 contains small prime factors: 2, 2, 2, 3, 5, and 5), then t_d is 2629.6 ms, while t_p is only 66.2 ms; thus, the speed gain (G_p) is 39.72 (as illustrated by the highest peak in the middle of the surface plot and mentioned in Table 1). The flat valley (with the darkest color) indicates $G_\mathrm{p} \leq 1.0$, while all the other regions in the surface indicate $G_\mathrm{p} > 1$. It shows that the frequency domain computation can be sometimes slower than the spatial domain computation of

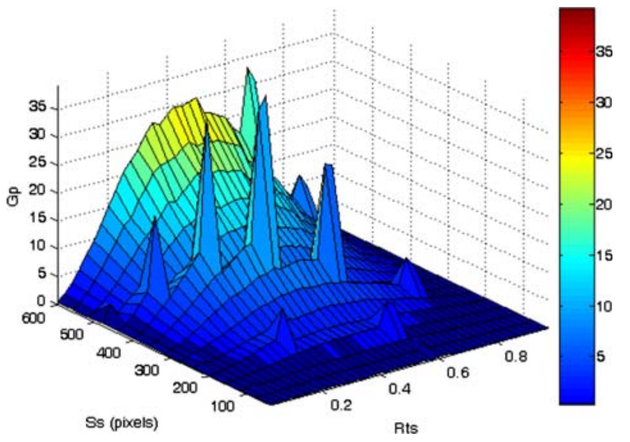


Fig. 1 Surface plot of G_p as a function of R_ts and S_s, where G_p is the speed-gain of frequency-domain NC implementation relative to its spatial-domain implementation, R_ts is the ratio of template-size to search-window-size, and S_s is the search-window-size

Table 1 BPNN Response Validated by the Experimental Results

$M \times N$	$K \times L$	R_{ts}	d	t_d (ms)	t_p (ms)	G_p
50×50	30×30	0.6000	↓	0.8	3.2	0.25
75×75	25×25	0.33	↓	4	8.81	0.45
320×240	10×10	0.036	↓	28.68	82.28	0.35
320×240	51×51	0.18	↑	260	80	3.25
320×240	75×75	0.27	↑	450	90	5
320×240	100×100	0.3610	↑	601.8	100.4	5.99
320×240	125×125	0.44	↑	681	100	6.81
320×240	300×200	0.884	↓	135.2	282.4	0.48
360×360	153×153	0.4250	↑	2629.6	66.2	39.72
512×512	60×30	0.0820	↑	776.34	317	2.45
512×512	205×205	0.4	↑	7631	390	19.56
640×480	400×300	0.6245	↑	10111	701	14.42

NC, therefore we should not use a single approach (direct or proposed) for all cases. In order to harness the best features, we propose a solution in the next section.

2.2.3 BPNN Controller

We propose a BPNN controller that can suggest which implementation of NC will perform faster for the images at hand, before actually computing the correlation surface. We used BPNN, because it can learn a non-linear multi-dimensional mapping/classification problem better as compared to the other conventional neural networks [2,3,20,24]. The BPNN has been trained on the experimental data that was used to generate the non-linear surface plot in Fig. 1. It is emphasized that we used only three parameters (S_s, R_{ts}, and G_p) to generate the surface plot in Fig. 1, and that the *content* of the images was not used for training the neural network. The architecture of the BPNN is designed as shown in Fig. 2. It contains an input layer, a hidden layer, and an output layer. The input layer has two nodes (i.e. $m_0 = 2$), because we will input in it

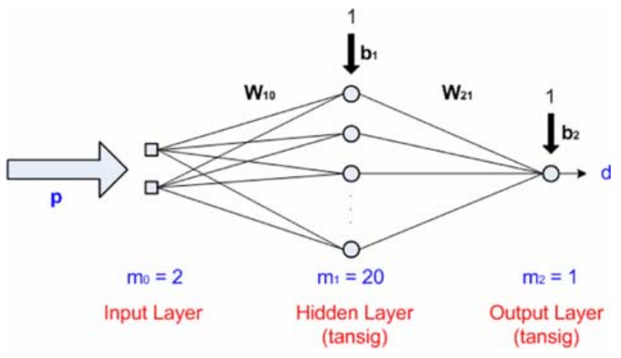


Fig. 2 The proposed architecture of the BPNN classifier, where *tansig* is the activation function used for the neurons in the hidden and the output layers (see [48])

the pattern **p** consisting of two elements, as described in (2). The output layer has one neuron (i.e. $m_2 = 1$), because we want the BPNN to output a single binary decision (either a positive or a negative value). The number of neurons in the hidden layer depends on the difficulty level of the mapping or classification problem at hand. After few experiments, we found out that only 20 hidden neurons (i.e. $m_1 = 20$) could solve our problem satisfactorily. The activation function of the neurons in the hidden layer and the output layer is chosen to be the tangent-sigmoid function, because it is non-linear (which is necessary to solve a non-linear classification problem) and it speeds up the learning process of the neural network (because it supports negative as well as positive values) [16,20,24]. The training of the BPNN was carried out using the efficient scaled-conjugate learning algorithm [30] with the goal of mean-squared-error (mse) set to 0.01. The neural network accepts a pattern (**p**) as its input, defined as:

$$\mathbf{p} = \left[\frac{S_s}{600} R_{ts} \right]^T, \tag{10}$$

where the S_s is normalized by 600, which was its maximum value in the experimental data. Thus, the maximum size of the search window that it can support is 600×600 pixels (which is more than the size we need, because we process the video frames of size 320×240 pixels). The output, d, of the BPNN classifier in its application phase can be easily determined as:

$$d = \text{tansig}\{\mathbf{W}_{21}.\text{tansig}(\mathbf{W}_{10}.\mathbf{p} + \mathbf{b}_1) + \mathbf{b}_2\}, \tag{11}$$

where $\mathbf{W}_{10}$, $\mathbf{b}_1$, $\mathbf{W}_{21}$, and $\mathbf{b}_2$ are the $m_1 \times m_0$, $m_1 \times 1$, $m_2 \times m_1$, and $m_2 \times 1$ matrices, respectively. Each row of $\mathbf{W}_{10}$ contains the synaptic weights of its corresponding neuron in the hidden layer. The elements of the row vector $\mathbf{W}_{21}$ are the synaptic weights of the output neuron. The column vector $\mathbf{b}_1$ contains the bias weights of the neurons in the hidden layer, and $\mathbf{b}_2$ is the bias weight of the output neuron. All these synaptic weights are adapted and optimized according to the training dataset during the learning phase of the BPNN. The tangent sigmoid (*tansig*) activation function is a fast approximation of *tanh* function.[2] It is defined as:

$$\text{tansig}(n) = \frac{2}{1 + e^{-2n}} - 1. \tag{12}$$

The output of the BPNN, d, will be either a positive or a negative value. If $d > 0$, the proposed implementation of NC will be faster than its *direct* implementation for the particular sizes of the images, and vice versa, if $d < 0$. The response of the BPNN has been tested with all the patterns from its

[2] MATLAB 7.0 On-line Help Documentation

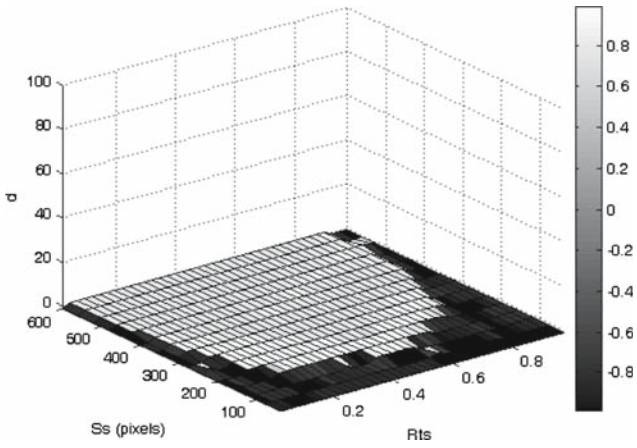


Fig. 3 Surface plot showing the decisions of the BPNN classifier when it was provided with various combinations of the search-window-size and the size-ratio as a two-element input pattern

training dataset, and the resulting surface plot is shown in Fig. 3.

If we compare the surface plots in Figs. 1 and 3, we can observe that all the darkest regions at the valley (where $G_p \leq 1$) in Fig. 1 corresponds to all the blackish regions (where $d < 0$) in Fig. 3. Similarly, all the higher regions (where $G_p > 1$) in Fig. 1 correspond to all the whitish regions (where $d > 0$) in Fig. 3. The BPNN is well *generalized*, that is, it can produce the right decisions, even for those input patterns, which were not included in its training dataset. The testing results and experimental validations are listed in Table 1. It may be noted that t_d is the time taken by the optimized *direct* implementation of NC in OpenCV.[3]. The downward arrow (↓) and the upward arrow (↑) indicate $d < 0$ and $d > 0$, respectively. The testing was carried out on a PC with 1.7 GHz processor and 512 MB RAM. The value of G_p, for every case listed in the table, validates the corresponding decision (d) of the BPNN. At first sight, it may seem that the spatial-domain implementation is applied rarely; however, it is the one which is applied frequently especially when the template size is small in case of distant object tracking, or when the template size is very large in case of near (and smoothly moving) object.

2.3 Proposed target detection algorithm

In this section, the proposed target detection algorithm is explained in detail. For the first frame, steps 1 to 4, given below, are applied on both the search-window and the template. However, for the second frame and onward, the steps 1 to 4 will be applied on the search-window only, because the template for these frames will be generated adaptively using the best-match section of the previous edge-enhanced search-window and the previous template (as discussed in Sect. 2.5).

Step 1: Apply $w \times w$ Gaussian smoothing filter with standard deviation, σ_w. Larger size of the filter results in more smoothing, but it trades off with the computational speed. Our experiments show that $w = 9$ works very fine in almost all scenarios. If the value of σ_w is too low, the pixels (in the search window) corresponding to the boundary coefficients of the Gaussian mask get too small weight, and vice versa. We use an efficient method[3] for automatically calculating the value of σ_w. It produces the effective weights of the Gaussian mask with a desirable property that the sum of all the coefficients in the resulting mask equals 1. The formula is given as:

$$\sigma_w = 0.3\left(\frac{w}{2} - 1\right) + 0.8. \tag{13}$$

The smoothing process attenuates the undesirable artifacts and/or sensor noise which occur especially when the ambient light around the sensor is low. Thus, it reduces the risk of the generation of the false edges due to the noisy pixels.

Step 2: Apply horizontal and vertical Sobel masks [23,44] on the smoothed image, and get the two images, E_h and E_v.

Step 3: Use E_h and E_v to compute the edge image as:

$$E(x, y) = \sqrt{E_h^2(x, y) + E_v^2(x, y)}, \tag{14}$$

where $x = 0, 1, 2, \ldots, U$, and $y = 0, 1, 2, \ldots, V$. For the template, $(U, V) = (L, K)$, and for the search-window, $(U, V) = (N, M)$. However, we have used its efficient approximation (given below) which produces almost identical edge image [23,44].

$$E(x, y) = |E_h(x, y)| + |E_v(x, y)|. \tag{15}$$

Step 4: The experiments have shown that the *dynamic range* of the edge image is often too narrow towards darker side as compared to the available range [0, 255], especially in low-contrast imagery. Conventionally, the edge image is converted into a binary image using a predefined threshold; however, this approach does not work well in a template matching application, because some weak, but valuable, features are lost. Furthermore, all the pixels with the gray level slightly greater than the threshold will become 1 in this approach. Thus, the rich content of gray level edge-features of the object is also lost. In order to make the object detection algorithm robust, we enhance the edges, using a normalization procedure given by:

$$E_n(x, y) = \left(\frac{255}{E_{\max} - E_{\min}}\right)\{E(x, y) - E_{\min}\}, \tag{16}$$

[3] OpenCV v4: Intel's Open Source Computer Vision Library http://www.sourceforge.net/projects/opencvlibrary

where $E_{\min}$ and $E_{\max}$ are the minimum and maximum values in the un-normalized edge image, E, respectively, and E_n is the normalized image. The normalization effectively stretches (scales) the histogram of the image linearly in the whole range [0, 255]. As a result, the contrast between the object and the background is also enhanced. Now, the edges of the object almost always have the values greater than 100 in any scenario. Nevertheless, it is safe to assume a threshold $\tau = 50$, for quenching the remaining false and very weak edges due to smoothed noise and artifacts. The thresholding operation is given by:

$$E_{nt}(x, y) = \begin{cases} E_n(x, y) & \text{if } E_n(x, y) > \tau \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where E_{nt} is the normalized and thresholded image. It may be noted that E_{nt} is not a binary image, but an edge-enhanced gray-level image adequately containing the important features of the object. Figure 4a shows a 320×240 real-world image containing a small and dim helicopter having very low contrast with its background. Fig. 4b is its edge image obtained by applying only the Sobel masks without Gaussian smoothing and edge-enhancement. It may be noted that the edges of the object are very weak and almost invisible, and that it also contains the undesirable edges of the noise and artifacts. Figure 4c illustrates that the smoothing, normalization, and thresholding operations enhance the edges of the object and takes care of the edges due to artifacts and noise.

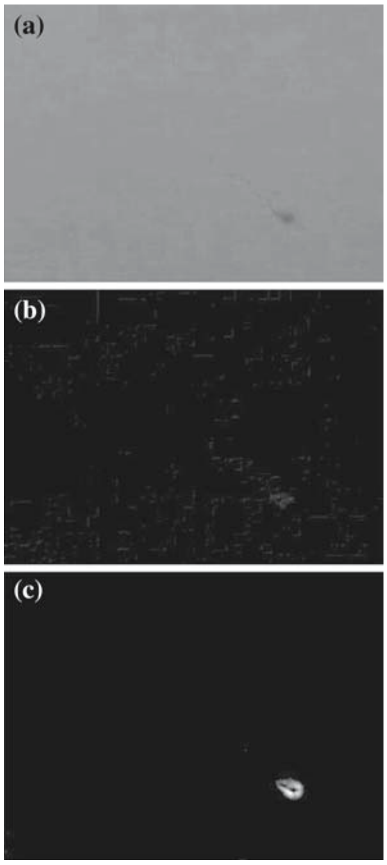


Fig. 4 Effect of edge-enhancement. **a** A 320×240 gray level image containing a very low-contrast object, **b** Edges of the image without using the proposed edge-enhancement operations, **c** Edges of the same image after the proposed edge-enhancement operations

Step 5: Match the edge-enhanced template with the edge-enhanced search-window using the proposed BCFNC algorithm (Sect. 2.2).

Step 6: Locate the peak value $c_{\max}$ in the normalized correlation matrix, and denote its position by (m^*, n^*), which is basically the top-left position of the best-match in the search-window.

Step 7: Assuming that K and L (i.e. height and width of the template, respectively) are odd integers to have an exact center point, locate the center of the best-match using:

$$(m_c, n_c) = \left(m^* + \frac{K-1}{2}, n^* + \frac{L-1}{2}\right). \tag{18}$$

Step 8: Assuming that x_t and y_t are the column (horizontal) and row (vertical) coordinates of the top-left corner of the search-window in the frame respectively, locate the center of the best-match with respect to the top-left frame origin at (0, 0), using:

$$(x, y) = (x_t + n_c, y_t + m_c). \tag{19}$$

This (x, y) position is considered as the target location in the current frame.

2.4 Template Updating

As time progresses, the shape, size, orientation, etc. of the object may change during its motion in the video; therefore a constant template can not work for a long and good tracking session. It must be adapted with time according to the change in the image of the object. In this section, we will first describe some conventional template updating schemes and then the proposed one. In all cases, let $b[k]$ be the best-match section in the current search window that produced the highest peak in the correlation surface, and let $t[k]$ and $t[k+1]$, respectively, be the current and the updated template. The $c_{\max}$ is the peak value in correlation surface, as previously defined. Finally, let τ_t be some threshold, such that $0 < \tau_t < 1$. Typically, we get satisfactory results by setting $\tau_t = 0.84$.

2.4.1 Simple template updating method

In this scheme, the template is updated as:

$$t[k+1] = \begin{cases} b[k] & \text{if } c_{\max} > \tau_t \\ t[k] & \text{otherwise} \end{cases} \tag{20}$$

This approach assumes that the best-match provided by the correlation is always the true target; which is not true because sometimes the nearby clutter can produce a higher correlation value than the actual object. Thus, the template is corrupted by the clutter, and the actual object walks-off the template very soon.

2.4.2 α -Tracker template updating method

In this approach, the template is updated as:

$$t[k+1] = \begin{cases} t[k] + \alpha(b[k] - t[k]) & \text{if} c_{\max} > \tau_t \\ t[k] & \text{otherwise} \end{cases} \tag{21}$$

A larger value of α, close to 1.0, will cause a greater change in the template than a smaller value. If $\alpha = 0$, the template will not be updated. In [9,21], a small constant value for α (e.g. 0.02) is used which reduces the effect of short-lived noise or neighboring object by smoothing the update of the template over time. However, if the tracked object is rapidly changing its shape, α should be large so as to avoid stagnation on the object's previous shape. In [47], the value of α is set to be equal to $c_{\max}$; however, this approach is not adequately robust because $c_{\max}$ is generally greater than 0.9 in a good tracking session which makes this method almost equivalent to the *simple template updating* scheme.

2.4.3 The proposed template updating method

We propose a robust template updating scheme which uses a low-pass IIR (Infinite Impulse Response) filter [4,31] with *adaptive* coefficients, β and $(1-\beta)$:

$$t[k+1] = \begin{cases} \beta b[k] + (1-\beta)t[k] & \text{if } c_{\max} > \tau_t \\ t[k] & \text{otherwise} \end{cases} \tag{22}$$

where $\beta = \lambda c_{\max}$. The value of λ is recommended to be in the range (0.0, 0.3] to get effective results. If the frame rate is adequately high (e.g. 25 fps), a reasonable value of λ is typically 0.15. In fact, the updated template is a weighted-sum of the current best-match and the current template (and the weights are adaptively changing). The current template itself is *not* the previous best-match, but weighted-sum of the previous best-match and the previous template. Thus, our approach uses the history of the template, and it does not quickly assign the best-match as the new template. The amount of change to be introduced in the updated template is determined by the quality of the correlated object. A stronger match will introduce a larger change in the template. The proposed template updating method copes with the short-lived clutter and sudden occlusion as described below.

Short-lived Clutter: When the camera is tracking an object, the neighboring background pixels of the clutter will be continuously changing randomly; therefore these pixels will not become the dominant part of the template due to the low value of β. On the contrary, the pixels belonging to the object do not change as rapidly, so their effect will become more and more dominant in the template with time. As a result, the template will contain only the object and not the clutter.

Sudden Occlusion: If the object is suddenly occluded by another object for only a few frames, the correlation value will suddenly drop below the threshold, and the template will not be updated at all. When the object re-appears after the sudden occlusion, there is a high probability that the shape of the object will not change significantly, so it will again produce the highest correlation value and the template will be updated according to (22).

3 Target model for Kalman predictor

To further enhance the robustness of the tracking system, we use the Kalman filter which estimates the position of the target in the next frame. The predicted position is exploited:

- To search the object of interest in the next frame around the predicted position only (see Sect. 4), so that the probability of picking-up a similar object moving in a different direction can be minimized,
- To create a dynamic search window of optimal size (see Sect. 4), and
- To make the motors of PTU ready (one step ahead of time) to move towards the predicted position (see Sect. 6).

The dynamic model of the target used normally is "constant velocity with random walk", but we use 2-D "constant-acceleration with random walk [8]" model with six states, because it provided better accuracy when we tested it for various target trajectories. We use the following target state equation and the observation equation, respectively:

$$\mathbf{X}_{n+1} = \mathbf{\Phi} \boldsymbol{X}_n + \mathbf{U}_n, \tag{23}$$

and

$$\mathbf{Y}_n = \mathbf{M}\mathbf{X}_n + \mathbf{V}_n, \tag{24}$$

where $\mathbf{X}_n$ is the proposed state vector containing six states (position, velocity, and acceleration in x and y direction), defined as:

$$\mathbf{X}_n = \begin{bmatrix} x_n & \dot{x}_n & \ddot{x}_n & y_n & \dot{y}_n & \ddot{y}_n \end{bmatrix}^{\mathrm{T}}. \tag{25}$$

The state transition matrix, Φ, used in our study, is:

$$\Phi = \begin{bmatrix} 1 & T & \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{26}$$

where T is the sampling time (which is simply the inverse of the frame rate). It may be noted that the x_n and y_n are expressed in terms of second order approximation of their Taylor expansions, respectively [8]. $\mathbf{U}_n$ is the system noise vector, given by:

$$\mathbf{U}_n = \begin{bmatrix} 0 & 0 & u_{xn} & 0 & 0 & u_{yn} \end{bmatrix}^{\mathrm{T}}, \tag{27}$$

where u_{xn} and u_{yn} are the uncorrelated zero-mean Gaussian noise elements with variances σ_{ux}^2 and σ_{uy}^2, respectively. They account for the small uncertainty in the acceleration of the object. $\mathbf{Y}_n$ is the measurement vector given by:

$$\mathbf{Y}_n = \begin{bmatrix} x_n & y_n \end{bmatrix}^{\mathrm{T}}, \tag{28}$$

where x_n and y_n are the target coordinates obtained from the proposed target detection algorithm at current time step n. $\mathbf{M}$ is the observation matrix given by:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{29}$$

and $\mathbf{V}_n$ is the observation noise vector given by:

$$\mathbf{V}_n = \begin{bmatrix} v_{xn} & v_{yn} \end{bmatrix}^{\mathrm{T}}, \tag{30}$$

where v_{xn} and v_{yn} are the uncorrelated zero-mean Gaussian noise elements with variances σ_{vx}^2 and σ_{vy}^2, respectively. If the variances are set to higher values, the Kalman predictor will give the measurement coordinates lower importance than the predicted coordinates while correcting its prediction after the arrival of the new measurement [46]. Further details of the Kalman filter can be obtained from [6,8,29,46].

4 Dynamic search window

The search window (in which the target will be searched for in the next iteration) is kept smaller than the whole frame to save CPU time and to get rid of the false alarms that may be produced due to the clutter present in the background. However, it should not be too small; otherwise there is a risk of losing the track of the target [6]. Conventionally, the size of the search window is set to be *constant* throughout the tracking session and its centre is located at the centre of the current best-match [19] or at the predicted position [6], but these approaches have some drawbacks: if the target is moving and maneuvering very fast, it may go out of the search window, and if the target is moving and maneuvering very slow, the redundant background pixels and clutter in the search-window will deteriorate the template matching process. To eliminate these problems to a significant extent, we propose to determine the location and an optimum size of the search window, dynamically, as described below.

We set the center of the search window at the position $(x_{\mathrm{p}}, y_{\mathrm{p}})$ predicted by Kalman filter in the current iteration for the next iteration. Now, let $(x_{\mathrm{t}}, y_{\mathrm{t}})$ and $(x_{\mathrm{b}}, y_{\mathrm{b}})$ be the coordinates of the top-left and the bottom-right corners of the search window for the next frame, and assume that L (template-width) and K (template-height) are odd integers. Then, we propose to set the coordinates of the two corners of the search window for the next iteration as:

$$\begin{aligned}(x_{\mathrm{t}}, y_{\mathrm{t}}) = \Bigg(& x[n+1|n] - \left(\frac{L-1}{2} + \kappa + a_{\mathrm{t}x}|\varepsilon_x|\right), \\ & y[n+1|n] - \left(\frac{K-1}{2} + \kappa + a_{ty}|\varepsilon_y|\right)\Bigg),\end{aligned} \tag{31}$$

$$\begin{aligned}(x_{\mathrm{b}}, y_{\mathrm{b}}) = \Bigg(& x[n+1|n] + \left(\frac{L-1}{2} + \kappa + a_{\mathrm{b}x}|\varepsilon_x|\right), \\ & y[n+1|n] + \left(\frac{K-1}{2} + \kappa + a_{\mathrm{b}y}|\varepsilon_y|\right)\Bigg),\end{aligned} \tag{32}$$

where κ is the minimum width of the border pixels around the best-match. In our case, the PTU is continuously moving the camera towards the target, so we use $\kappa = 13$ pixels only. However, if the camera were fixed, κ could have a little larger value. Furthermore, $x[n+1|n]$ and $y[n+1|n]$ are the target coordinates predicted by the Kalman filter in the current iteration for the next iteration. The ε_x and ε_y are the prediction errors defined as:

$$\varepsilon_x = x[n] - x[n|n-1], \quad \text{and } \varepsilon_y = y[n] - y[n|n-1], \tag{33}$$

where $x[n|n-1]$ and $y[n|n-1]$ are the target coordinates predicted by Kalman filter in the previous iteration for the current iteration, respectively. The $x[n]$ and $y[n]$ are the target coordinates calculated by our target detection algorithm in the current iteration. Moreover, $a_{\mathrm{t}x}$, $a_{\mathrm{t}y}$, $a_{\mathrm{b}x}$, and $a_{\mathrm{b}y}$ are the scaling factors, which compensate for the possible prediction errors in case of a sudden maneuvering of the object. If any of the scaling factors is positive, the minimum-size search-window will be expanded from the corresponding side proportional to the corresponding prediction error. We set their values as:

$$(a_{tx}, a_{bx}) = \begin{cases} (a_1, a_2) & \text{if } \varepsilon_x \geqslant 0 \\ (a_2, a_1) & \text{otherwise} \end{cases}, \tag{34}$$

$$\left(a_{ty}, a_{by}\right) = \begin{cases} (a_1, a_2) & \text{if } \varepsilon_y \geqslant 0 \\ (a_2, a_1) & \text{otherwise} \end{cases}, \tag{35}$$

where $a_1 = -0.25$ and $a_2 = 1.25$. For example, if the prediction error in x-axis is $\varepsilon_x = +8$, the target position (provided by our target detection algorithm) is to the right of the predicted position, thus the minimum-size search window will be contracted by 2 pixels from left (using $a_{tx} = -0.25$), and expanded towards right side by 10 pixels (using $a_{bx} = 1.25$).

5 Evaluative comparison

In this section, we compare the robustness of the proposed target detection method with the conventional correlation based methods. First, we compare them on a single low-contrast image with small and dim target. Then, we compare them on challenging real-world image-sequences. Some of these sequences were also used in [45] to test the robustness of the *mean-shift* and the *condensation* trackers.

5.1 Comparison on Single Image

Figure 5 shows the gray-level template, its edges, and its edge-enhanced version. The gray-level template was extracted from the gray-level image (i.e. search-window) shown in Fig. 4a. The top-left corner of the template in the search-window was at $(m, n) = (169, 224)$, where m and n are the row and column coordinates, respectively. In this section, we compare all the correlation-based target-detection methods reviewed in this paper, and evaluate which technique produces a correct and clean peak at this location in the correlation surface.

Figure 6a illustrates the correlation surface obtained when the gray-level template was matched with the gray-level search-window using simple correlation given by (1) or (2). This method failed to locate the correct position of the object, because it produced the highest peak of 8,871,601 value at (10, 11) instead of (169, 224) along with infinite number of other false peaks at all those spots which were brighter than the object in the search window. In fact, the correlation value is lower (i.e. 8,226,021) at (169, 224) position, where the object actually lies, since the object is darker than the background. Moreover, the correlation values are not normalized in the range [−1.0, 1.0].

Figure 6b shows the correlation surface obtained when the gray-level template was matched with the gray-level search-window, using SPOMF given by (5). The method failed to locate the correct location of the object because it produced the highest peak of only 0.14 value at (10, 11) instead of (169, 224) along with many other false peaks.

Figure 6c illustrates the correlation surface which is the result of matching the gray-level template with the gray-level search window using NC given by (3). The object is located correctly at (169, 224), and the highest peak has value of 1.0. If we observe the surface closely, we can find that the minimum correlation value in the whole surface is also too high, i.e. 0.9945. This behavior of NC with gray level images may result in the template-drift problem or detection of a wrong target instead of the true target in critical situations.

Figure 6d shows the correlation surface obtained when the gray-level template was matched with the gray-level search-window using NCC given by (4). This method detects the correct position of the target at (169, 224) with the peak correlation value of 1.0. There are other positive as well as negative peaks within the range [−1.0, 1.0], but their values are not near the value of the correct peak as they were in the NC approach with gray-level images, discussed above.

Finally, Fig. 6e illustrates a nice and clean correlation surface resulting from the proposed edge-enhanced correlation (see Sect. 2.3), in which the edge-enhanced template (Fig. 5, right) is matched with the edge-enhanced search window (Fig. 4c) It is evident that there is only one peak with the correlation value of 1.0, which is exactly at (169, 224). Figure 6f shows the exact location of the helicopter detected. The position of the top-left corner of the best-match is shown by the black cross-hair at $(m^*, n^*) = (169, 224)$, while the position of the center of the best-match is indicated by the white cross-hair at $(m_c, n_c) = (179, 235)$.

5.2 Comparison on real-world video

For the real-world image sequences, we compare our target tracking algorithm only with the normalized correlation coefficient (NCC) due to space constraint, because NCC proved to be the next best method after our algorithm in the previous section. However, some of these sequences were also used in [45] to test the robustness of various recent trackers, so their results are cited for comparison wherever applicable. For simplicity, let A_1 be the target detection algorithm based on NCC with α-tracker template updating method using $\alpha = c_{\max}$ as in [47], and A_2 be the proposed target detection algorithm and template updating scheme (without Kalman filter and dynamic search window). In both the

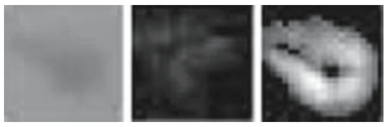


Fig. 5 The 23×21 template (*left*), its edges (*middle*) and its edge-enhanced version (*right*)

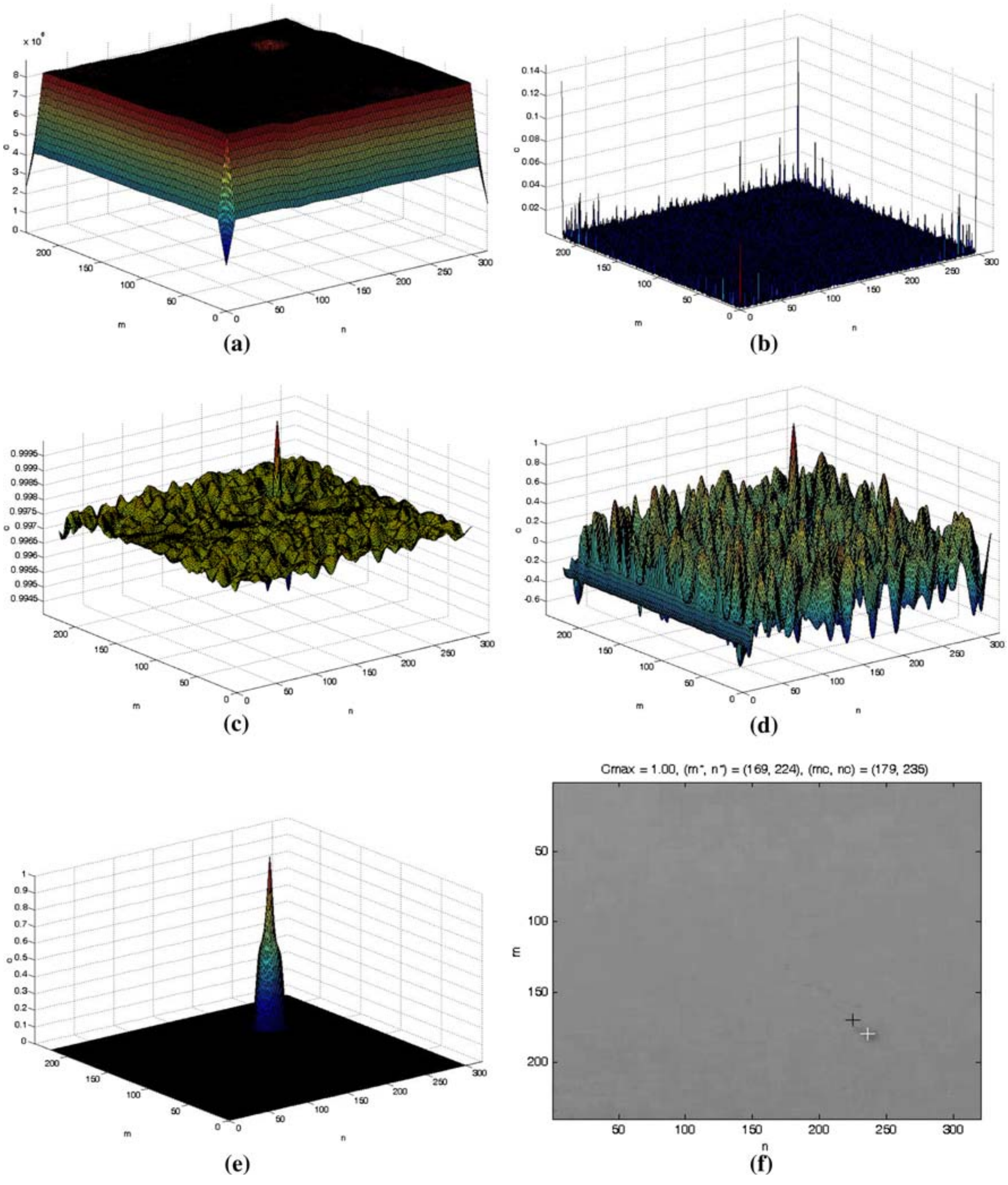


Fig. 6 Comparison of various correlation methods with the proposed edge-enhanced BCFNC algorithm. The search-window and the template were edge-enhanced images for our algorithm, while they were gray-level images for the other methods. **a** Standard correlation surface, **b** phase correlation surface, **c** normalized correlation surface, **d** normalized correlation coefficient surface, **e** the proposed edge-enhanced BCFNC surface, and **f** overlay of the + signs on the target coordinates found by the edge-enhanced BCFNC on the search-window, where the *black* and the *white* + signs represent the top-left coordinates (m^*, n^*) and the centre-coordinates (m_c, n_c) of the best-match, respectively

algorithms, the center of the search window (for the next iteration) is assumed to be the center of the current best-match and the size of the search window is kept three times larger than that of the template. In both cases, a 25×19 template is selected from the same position in the initial frame, and $\tau_t = 0.85$. First, we evaluate A_1 and A_2 exhaustively on two challenging image sequences S_1 and S_2. Then, we evaluate them on some other well-known sequences. The sequence S_1 contains 300 frames and was recorded by us with a handy-cam. During its recording, we continuously and

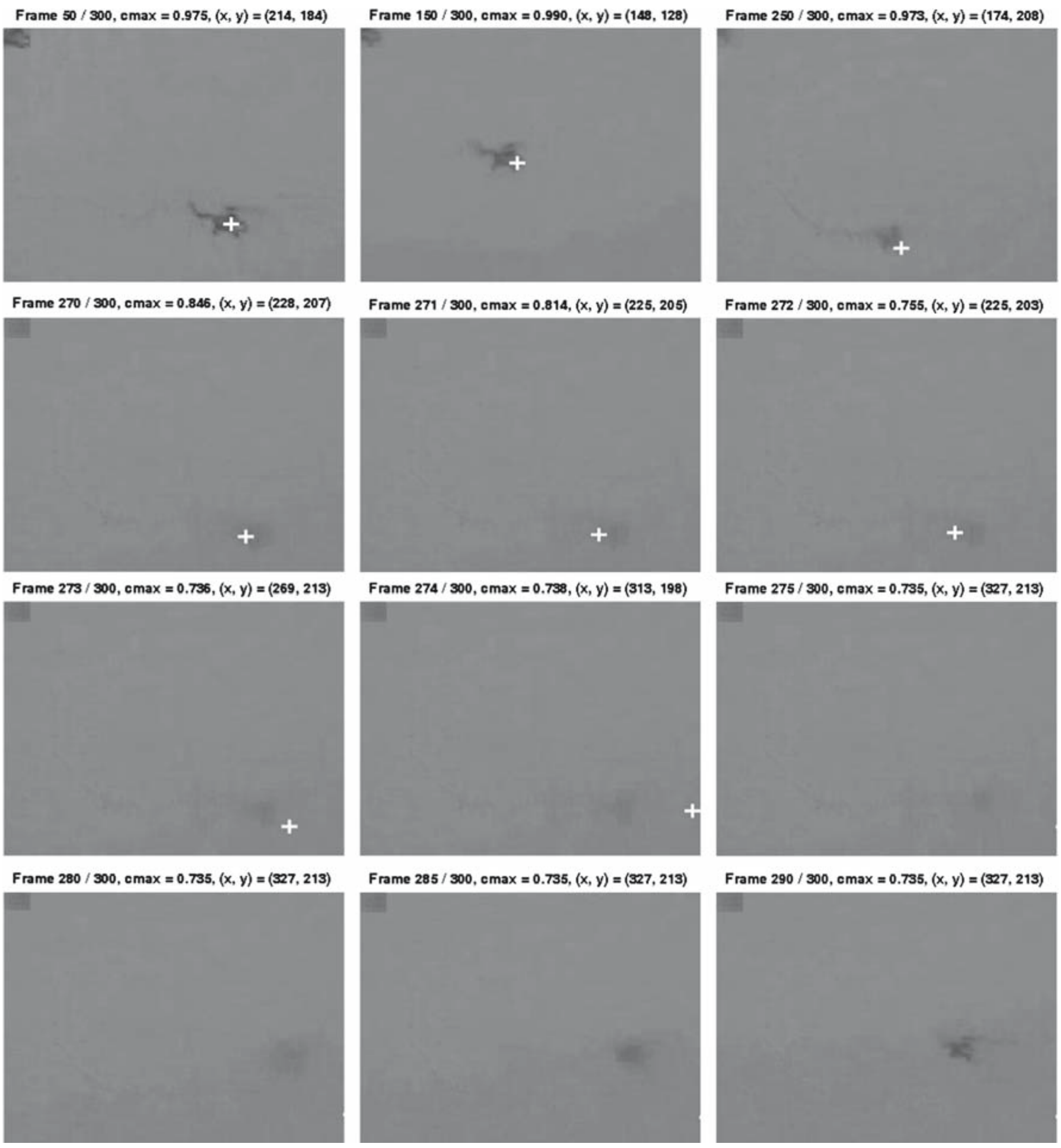

Frame 50 / 300, cmax = 0.975, (x, y) = (214, 184)
Frame 150 / 300, cmax = 0.990, (x, y) = (148, 128)
Frame 250 / 300, cmax = 0.973, (x, y) = (174, 208)
Frame 270 / 300, cmax = 0.846, (x, y) = (228, 207)
Frame 271 / 300, cmax = 0.814, (x, y) = (225, 205)
Frame 272 / 300, cmax = 0.755, (x, y) = (225, 203)
Frame 273 / 300, cmax = 0.736, (x, y) = (269, 213)
Frame 274 / 300, cmax = 0.738, (x, y) = (313, 198)
Frame 275 / 300, cmax = 0.735, (x, y) = (327, 213)
Frame 280 / 300, cmax = 0.735, (x, y) = (327, 213)
Frame 285 / 300, cmax = 0.735, (x, y) = (327, 213)
Frame 290 / 300, cmax = 0.735, (x, y) = (327, 213)

Fig. 7 Result of A_1 algorithm for S_1 image sequence, showing the template drift problem starting from Frame 271 and its failure starting from Frame 273

randomly moved the camera very fast and changed its zoom level frequently to introduce the object fading, fast and unpredictable object-motion, and blur effects (all at the same time). The S_2 sequence contains 390 frames, in which an aircraft is taking-off, during which its size is changing and the background is cluttered with trees and small buildings.[4] To evaluate the accuracy of the algorithms, the ground truth (i.e. the dataset of true target-coordinates) was generated manually for every frame in both the sequences.

[4] Available at http://www.fastpasses.com

Figure 7 is the result of A_1 for S_1 sequence. The template is overlaid at the upper-left corner in every frame. The frame index, correlation value, and the (x, y) coordinates of the target location are also shown at the top of every frame. We can see that the white target sign slowly keeps drifting away from the helicopter and the track is lost when the object is suddenly faded in Frame 272 (up to Frame 285). It can be observed that during the fading time the object is almost invisible. The resulting trajectory of the helicopter in S_1 is illustrated in Fig. 8, in which the template drifting and track-loss is observable after Frame 272. Figures 9 and 10 show the efficiency of the proposed target detection and

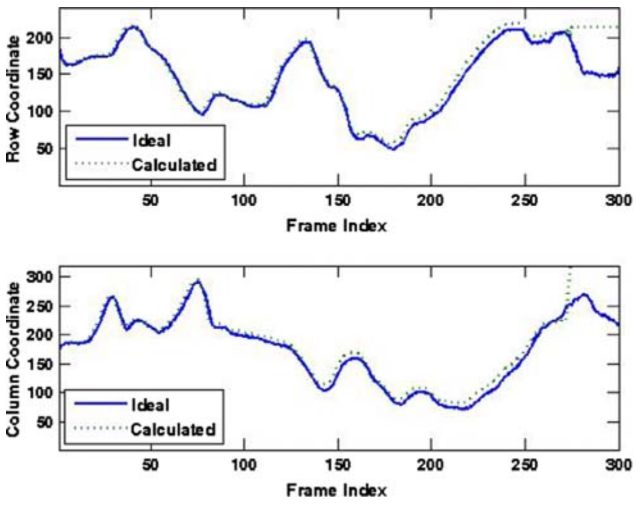


Fig. 8 Target trajectory for S_1 sequence by produced by A_1 algorithm showing the failure from Frame 273 through the last frame of the image sequence

template-updating schemes that keep a very good track of the target with no template-drift even during the severe fading of the object in the presence of fast and random object-motion in the low contrast imagery.

Figure 11 shows some frames from the image sequence S_2, when A_1 was tested on them. We can see that the target sign is exactly at the middle of the airplane in Frame 1, but it slowly drifts off backward in the subsequent frames, and the track is lost in Frame 93, because the white roof of the building comes suddenly in the upper neighborhood of the template which is not included in the current template as shown at the top left corner of Frame 93. The complete trajectory of the airplane calculated by A_1 is compared with the true trajectory in Fig. 12, which illustrates the failure of the algorithm starting from Frame 93. Figures 13 and 14 show the efficiency of the proposed target detection and template updating schemes that keep a good track of the airplane with negligible template drifting, even during the sudden appearance of the white roof of the building and the surrounding clutter. The reason is that the proposed algorithm gives importance to the enhanced edges of the object and background, rather than the intensity levels of their whole structures.

To evaluate the accuracy of both the algorithms for both of the image sequences, we carried out a post regression analysis [16] which provides (1) the correlation between the true and the calculated coordinates (R), and (2) the best-fit linear equation between them consisting of a slope (m) and intercept (C). Ideally, $(R, m, C) = (100\%, 1.0, 0)$. The results of the post-regression analysis are summarized in Table 2 which shows that our algorithm (A_2) outperforms A_1 algorithm in accuracy.

Now, suppose A_3 represents the algorithm which is the combination of our A_2, Kalman predictor and the proposed dynamic search window. The third row in TABLE 2 summarizes the results of the post-regression analysis, when algorithm A_3 was tested on S_1 and S_2 image sequences. The accuracy of A_3 is almost equivalent to that of A_2 for the sequence S_1. However, the real advantage of the Kalman predictor and the dynamic search window in A_3 can be observed in case of the sequence S_2 (which was full of clutter and contained significant object-scaling). The increased accuracy is also illustrated by the trajectories in Fig. 15.

We have tested our target tracking algorithm (A_3) also on other numerous real-world image sequences, but due to space constraint we present only some of them for further evaluation.

Figure 16 shows how A_3 persistently tracks a person in the presence of other persons in the test video "ShopAssistant2cor.mpg" from CAVIAR dataset[5] until the person goes out of the scene. In Frame 200, it can be seen that the target person is occluded by another person; even then the tracking is continued.

Figure 17 depicts the robust tracking of a tiny car moving along the road in a low-contrast, noisy and shaky video sequence recorded from an unmanned aerial vehicle (UAV). The whole scene (including the car) is rotating and translating simultaneously due to the motion of the UAV in 6 degree-of-freedom. Furthermore, in Frame 190, there is a glare effect (varying illumination). Never-the-less, the proposed algorithm tracks the car persistently and robustly.

Figure 18 illustrates some frames from the sequence $seq\_fast.avi$.[6] Here, a person moves his face right and left very fast (with slight rotation). It is illustrated in Fig. 6 in [45] that the *mean shift* and the *condensation* trackers could not track the fast moving face in this sequence. However, our tracker is able to track it without any difficulty as shown. The tracking algorithm proposed in [45] exploits a particle filter using an appearance model based on spatial-color Mixture of Gaussians (SMOG). Its results are comparative to those of our algorithm, but it is not a real-time tracker.

Figure 19 shows some frames from the sequence $seq\_mb.avi$,[6] in which the face of a girl is occluded with that of another person. In Fig. 7 in [45], it is reported that the *mean shift* and the *condensation* trackers could not robustly track the face of the girl during and after this occlusion. However, the tracker proposed in [45] could track it robustly. Our tracker has also successfully survived the occlusion with the comparative results, with the additional benefit of speed. The edge-enhanced template is shown at the top-right corner of

[5] CAVIAR (Context Aware Vision using Image-based Active Recognition) test video clips 2003–2004, available at http://www.homepages.inf.ed.ac.uk/rbf/CAVIAR/

[6] Available at http://vision.stanford.edu/~birch/headtracker/seq/

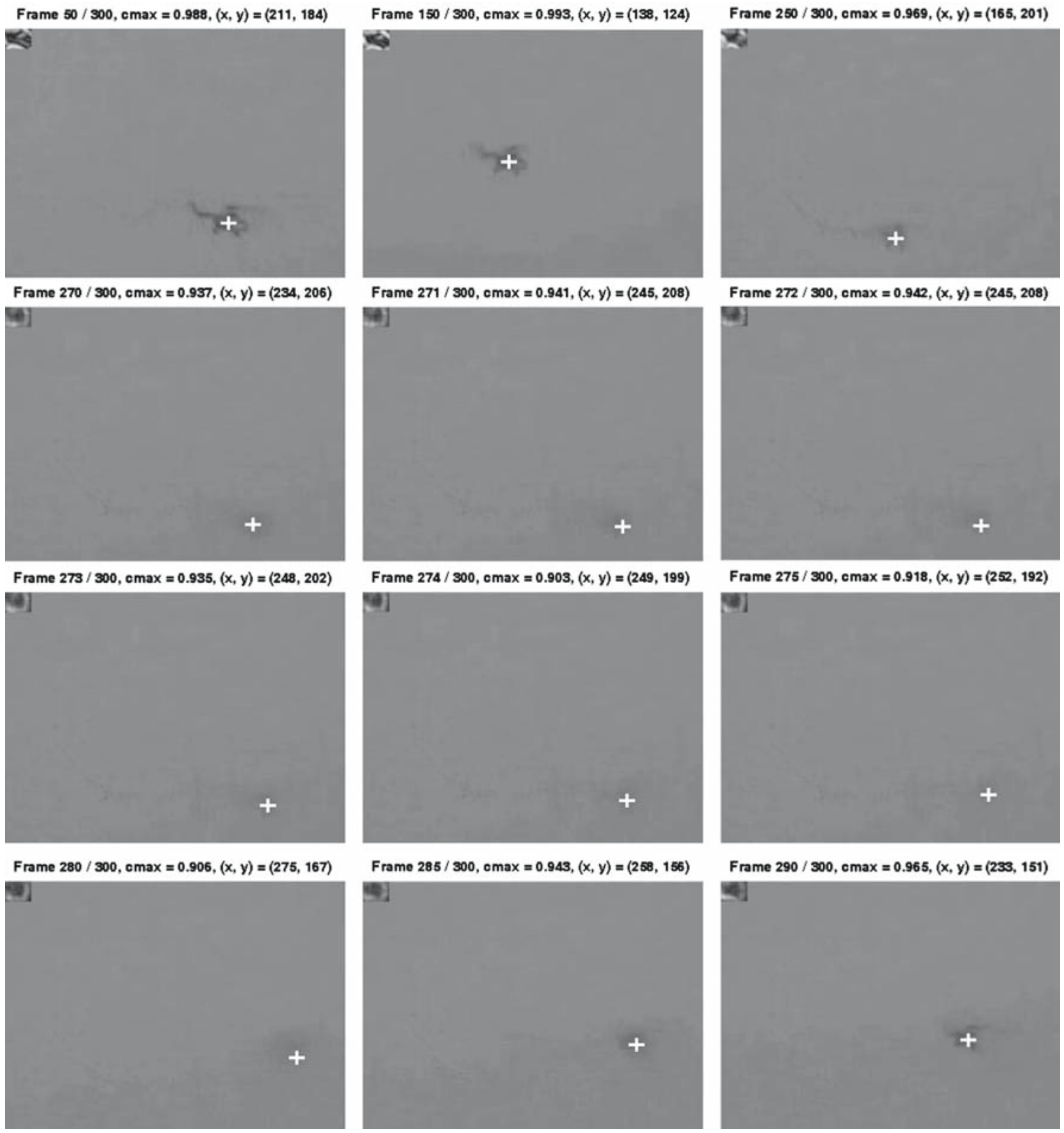

Frame 50 / 300, cmax = 0.988, (x, y) = (211, 184)
Frame 150 / 300, cmax = 0.993, (x, y) = (138, 124)
Frame 250 / 300, cmax = 0.969, (x, y) = (165, 201)
Frame 270 / 300, cmax = 0.937, (x, y) = (234, 206)
Frame 271 / 300, cmax = 0.941, (x, y) = (245, 208)
Frame 272 / 300, cmax = 0.942, (x, y) = (245, 208)
Frame 273 / 300, cmax = 0.935, (x, y) = (248, 202)
Frame 274 / 300, cmax = 0.903, (x, y) = (249, 199)
Frame 275 / 300, cmax = 0.918, (x, y) = (252, 192)
Frame 280 / 300, cmax = 0.906, (x, y) = (275, 167)
Frame 285 / 300, cmax = 0.943, (x, y) = (258, 156)
Frame 290 / 300, cmax = 0.965, (x, y) = (233, 151)

Fig. 9 Result of our A_2 algorithm for S_1 image sequence. The proposed target tracking method successfully tracks the helicopter in all the frames even during the severe object fading in very low-contrast video without any template-drift problem

each frame, and it can be observed how smoothly and robustly it is being updated, without introducing significant effects due to the occluding face.

6 Camera motion control

In this section, a simple, but very smooth and accurate motion controller is described. It generates the precise pan-tilt velocity commands for the PTU to move the camera smoothly and accurately towards the target. Its basic idea is adopted from the so-called "Car-following control (CFC) Law".[1]

6.1 Car-following control (CFC)

Suppose a car, F, is moving with a velocity V_F and is following another car, L, moving with a velocity V_L, as shown in Fig. 20. Let the positional error between the cars be denoted by e. Then, the basic CFC law is then defined as:[1]

$$V_F(new) = V_L + f(e), \tag{36}$$

where $f(e)$ is a function of the positional error. The control law states that "the speed of the *follow* car, F, should be set to the speed of the *lead* car, L, plus a speed adjustment based

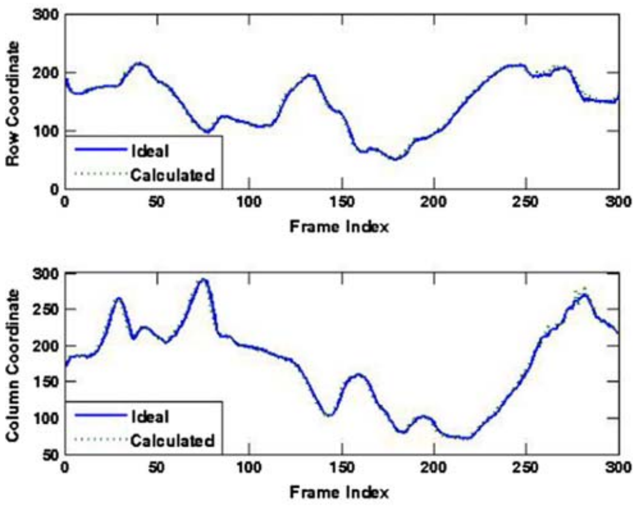


Fig. 10 Target trajectory (row and column coordinates) for S_1 sequence produced by our A_2 algorithm. Note that the computed trajectory is perfectly matching the ground truth trajectory for almost all the frames

upon the positional error, e, between them". This simple control strategy has some desirable characteristics:

- By attempting to match the *lead* velocity, the *follow* car can maintain *smooth* tracking while the *lead* car is in transit. This is not the case for "bang-bang" control laws (e.g. [32]) that drive only based upon the positional error (e).
- When the *lead* car stops, the $f(e)$ term still drives F to a proper steady state.
- Direction reversal of L can result in proper control of F.
- The function $f(e)$ can use closed forms to determine what correction term is required to minimize e, within a given time bound and given current velocities.

Proposed Implementation of CFC: Given a servo-motor PTU and a moving object in the video frames, the relative velocity of the object in terms of PTU units (i.e. degree/second) has two components: relative pan velocity (vrp) and relative tilt velocity (vrt). We compute them as:

$$v_{\text{rp}}[n] = C_{\text{spp}}\frac{\Delta x}{\Delta t} = C_{\text{spp}}\left(\frac{x[n] - x[n-1]}{T}\right), \tag{37}$$

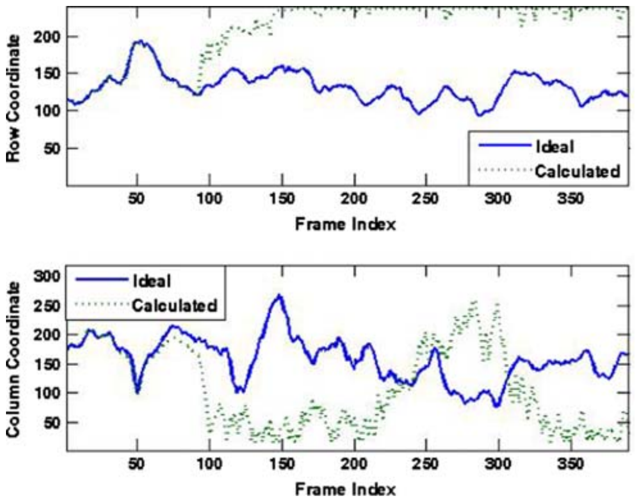


Fig. 12 Target trajectory (row and column coordinates) for S_2 sequence produced by A_1 algorithm, showing its track-failure starting from Frame 93

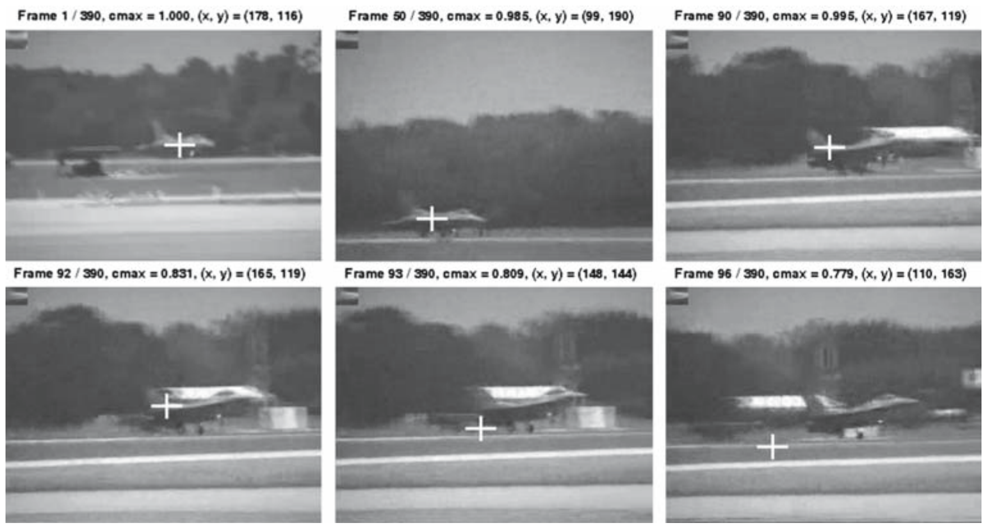

Frame 1 / 390, cmax = 1.000, (x, y) = (178, 116)
Frame 50 / 390, cmax = 0.985, (x, y) = (99, 190)
Frame 90 / 390, cmax = 0.995, (x, y) = (167, 119)
Frame 92 / 390, cmax = 0.831, (x, y) = (165, 119)
Frame 93 / 390, cmax = 0.809, (x, y) = (148, 144)
Frame 96 / 390, cmax = 0.779, (x, y) = (110, 163)

Fig. 11 Result of A_1 algorithm for S_2 image sequence. Note that the template-drift problem starts from Frame 90 and the track-failure starts from Frame 93 due to background clutter

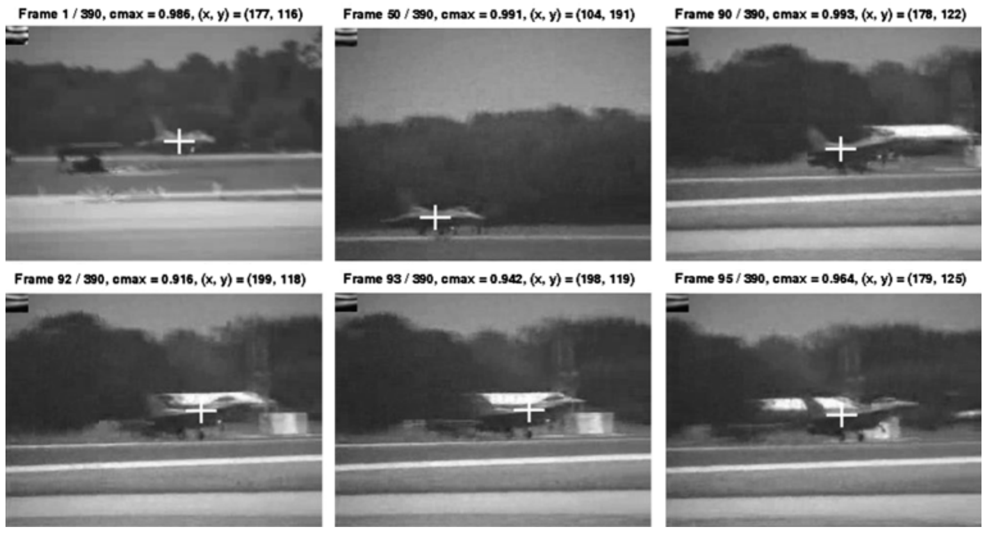

Frame 1 / 390, cmax = 0.986, (x, y) = (177, 116)
Frame 50 / 390, cmax = 0.991, (x, y) = (104, 191)
Frame 90 / 390, cmax = 0.993, (x, y) = (178, 122)
Frame 92 / 390, cmax = 0.916, (x, y) = (199, 118)
Frame 93 / 390, cmax = 0.942, (x, y) = (198, 119)
Frame 95 / 390, cmax = 0.964, (x, y) = (179, 125)

Fig. 13 Result of our A_2 algorithm for S_2 image sequence, showing how persistently it tracks the airplane in all the frames even in the presence of high background clutter

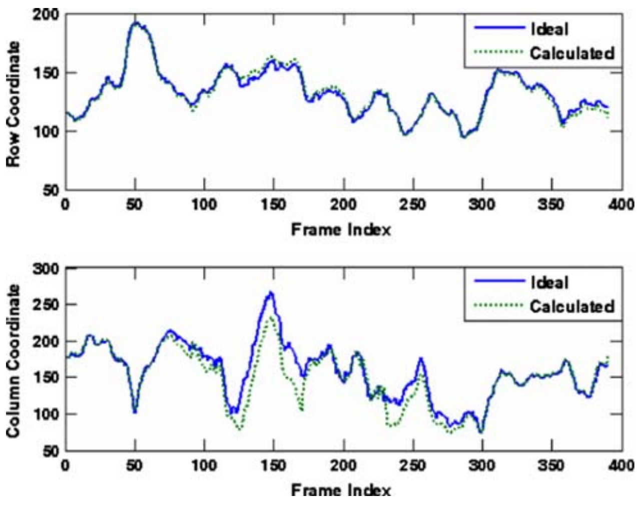


Fig. 14 Target trajectory computed for S_2 sequence by our A_2 algorithm. The computed target coordinates match the ground truth trajectory. However, there is a little deviation in column (i.e. horizontal axis) coordinates for some frames, due to the small-size template and the similar body structure of the airplane in its middle part, where the initial template was selected from

and

$$v_{\mathrm{rt}}[n] = C_{\mathrm{spp}} \frac{\Delta y}{\Delta t} = C_{\mathrm{spp}} \left(\frac{y[n] - y[n-1]}{T} \right), \tag{38}$$

where T is the sampling time which is the inverse of the frame rate, and C_{spp} is the conversion factor (obtained by camera calibration procedure) that converts the units of the velocities from image pixels/second into PTU degrees/second. It may be noted that $x[n]$ and $y[n]$ are the target coordinates coming from our target detection algorithm. The new pan and tilt velocities of the PTU (i.e. the *follow* velocities) are calculated as:

$$v_{\mathrm{p}}[n] = \left(v_{\mathrm{fp}}[n] - v_{\mathrm{rp}}[n] \right) + K e_x[n], \tag{39}$$

$$v_{\mathrm{t}}[n] = \left(v_{\mathrm{ft}}[n] + v_{\mathrm{rt}}[n] \right) - K e_y[n], \tag{40}$$

where e_x and e_y are the positional errors in x- and y-axis, respectively:

$$e_x[n] = r_x - x[n] \quad \text{and } e_y[n] = r_y - y[n], \tag{41}$$

Table 2 Post-Regression Analysis for Accuracy of A_1, A_2, and A_3

Algo	S_1						S_2					
	x			y			x			y		
	$R(\%)$	m	C	$R(\%)$	m	C	$R(\%)$	m	C	$R(\%)$	m	C
A1	94.8	1	−1.3	94.5	1	7.6	34.6	0.6	−0.8	−0.16	−0.4	258
A_2	99.8	1	−0.4	99.7	1	0.4	93.6	0.9	1.2	99.1	1	−2.1
A_3	99.8	1	−0.2	99.7	1	0.4	97.4	0.95	8.9	99.3	1	0.4

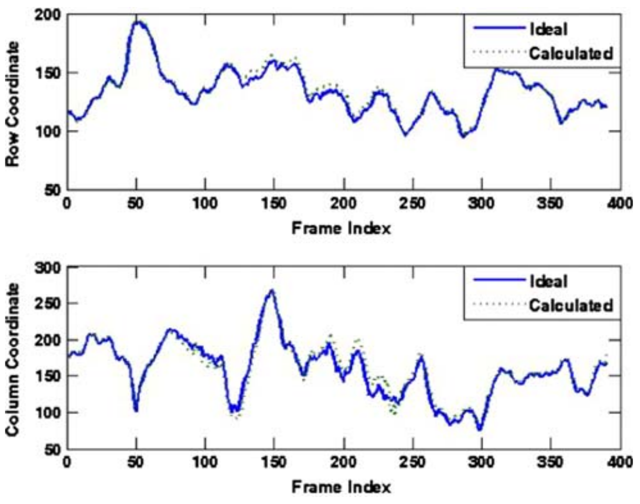


Fig. 15 Target trajectory computed for S_2 sequence by our A_3 algorithm. The computed target trajectory accurately follows the ground truth trajectory in almost all the frames

where (r_x, r_y) are the coordinates of the reference point (or *track-point*) which is normally the centre of the video frame, $(v_{\mathrm{fp}}, v_{\mathrm{ft}})$ are the current velocity components of the PTU fed back by speed sensors and K is the proportional gain with the unit of "degree.second/pixel". Using the car-following control equations (39) and (40), we can command the PTU to move the camera towards the object so that the object remains always at the track-point. The terms inside the parentheses in (39) and (40) represent the *absolute* lead velocities, while the remaining terms represent the $f(e)$ functions (or *approaching* velocities), as mentioned in (36). The plus and minus signs compensate the difference between the directions in the pixel coordinate system and the PTU coordinate system. We assume that the pan velocity of the PTU is positive if it is moving towards left, and its tilt velocity is positive when it is moving downwards.

6.2 Predictive Open-Loop CFC (POL-CFC)

The basic CFC [as implemented in the form of (39), (40) and (41)] assumes that we are getting the actual pan-tilt velocities (v_{fp} and v_{ft}) of the PTU through velocity sensors. But, in practice, we did not have servo-mechanism. We had a stepper-motor mechanism, which does not feedback its actual velocities. In control theory, such a system is referred to as "open-loop" system. The proposed POL-CFC algorithm does

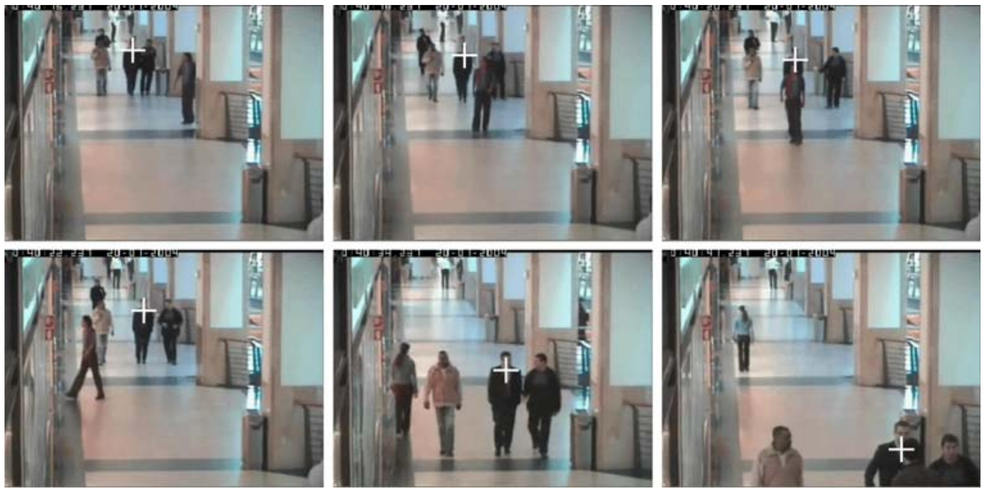


Fig. 16 Frames 100, 150, 200, 250, 550, and 725 (in raster scan order) of "ShopAssistant2cor.mpg" video clip from CAVIAR dataset, illustrating the robustness of our A_3 algorithm even in the presence of occlusion, multiple similar objects, varying illumination, clutter, and object scaling

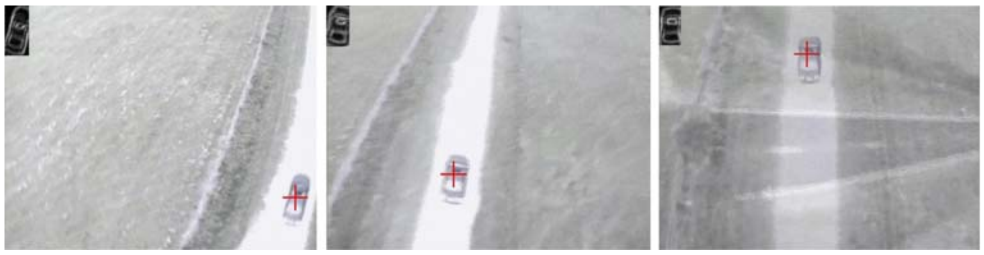


Fig. 17 Frames 1, 100, and 190 of a shaky video sequence recorded from an unmanned aerial vehicle (UAV) showing a small car being tracked perfectly by our algorithm in the presence of blur, glare, and UAV motion in 6 degree-of-freedom

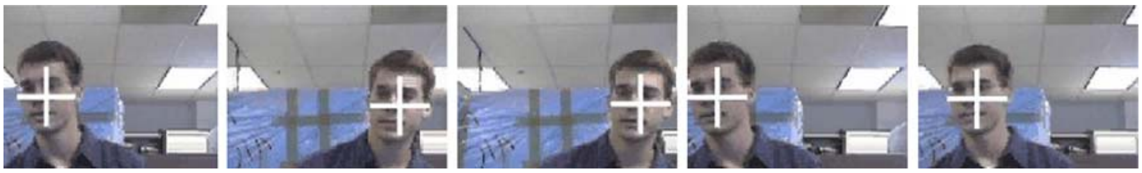


Fig. 18 Frames 1, 8, 12, 20, and 25 of the *seq_fast.avi* sequence, in which the proposed algorithm tracks the face even during its fast left and right motion. However, the mean-shift and condensation trackers could not track the fast-moving face (see Fig. 6 in [45])

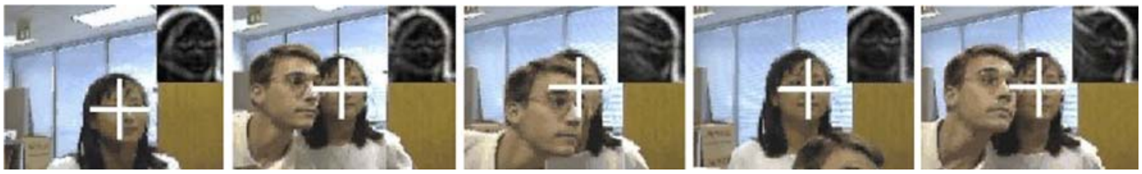


Fig. 19 Frames 1, 31, 40, 53, and 74 of the *seq_mb.avi* sequence. The proposed algorithm tracks the face of the girl even during occlusion. However, the mean-shift and condensation trackers could not robustly survive the occlusion in this sequence (see Fig. 7 in [45])

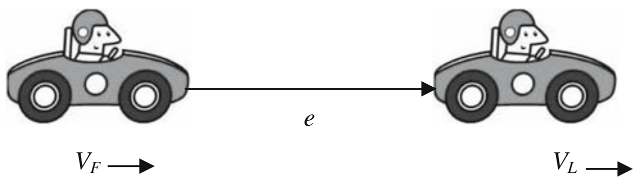


Fig. 20 Demonstration of "Car-Following Control Law"

not require the feed-back velocities. It replaces these velocities with the previous control-generated pan-tilt velocities, and uses η factor to control the amount added to the previous control-generated velocities. Furthermore, we use the predicted target position in the image rather than its current position, so that the PTU can be prepared one step ahead of time to accurately reach the angular position corresponding to the 3-D position of the target at the same time. We present the POL-CFC algorithm as follows. Let the notation $\hat{x}[n+1|n]$ be defined as a target state predicted by Kalman filter at iteration n for iteration $n+1$. Then, we propose to generate the pan-tilt velocities as:

$$v_{\mathrm{p}}[n] = v_{\mathrm{p}}[n-1] + \eta\big(K\hat{e_x}[n+1|n] - \hat{v_{\mathrm{rp}}}[n+1|n]\big), \tag{42}$$

$$v_{\mathrm{t}}[n] = v_{\mathrm{t}}[n-1] + \eta\big(\hat{v_{rt}}[n] - K\hat{e_y}[n+1|n]\big), \tag{43}$$

where η is a small positive constant (typically 0.1) which controls the amount of velocity added to the *previous* velocity command,

$$\hat{e_x}[n+1|n] = r_x - \hat{x}[n+1|n] \quad \text{and} \quad \hat{e_y}[n+1|n] = r_y - \hat{y}[n+1|n] \tag{44}$$

are the predicted tracking errors in both axes, and

$$v_{\mathrm{rp}}[n] = C_{\mathrm{spp}}\left(\frac{\hat{x}[n+1|n] - \hat{x}[n|n-1]}{T}\right), \tag{45}$$

$$v_{\mathrm{rt}}[n] = C_{\mathrm{spp}}\left(\frac{\hat{y}[n+1|n] - \hat{y}[n|n-1]}{T}\right) \tag{46}$$

are the predicted relative velocities of the target in both axes. It may be noted that the previous velocity commands ($v_{\mathrm{p}}[n-1]$ and $v_t[n-1]$) are only the approximations of the actual current velocities of the PTU because it is not necessary that the PTU will attain the previously commanded velocity within a small sampling time, T, due to some practical limitations (e.g. the motor response, the inertia of the PTU and camera, etc). Thus, we implemented the following heuristic rule for $v_p[n]$ and $v_{\mathrm{t}}[n]$ to address the problem and eliminate the initial overshoots:

$$v[n] = \begin{cases} \tau_v & \text{if } n < 12 \text{AND} v[n] > \tau_v \\ v[n] \end{cases} \tag{47}$$

where n is initialized when the object to be tracked is selected by the user.

The motion controller has been calibrated for 1× to 25× zoom levels of the video camera. The resolution of the pan-tilt stepper motors in our PTU is 0.01285°/step. The maximum steady-state error of the tracking system controlled by POL-CFC is 0 for zoom-levels 1× to 6×, ±1 pixels for 7× to 15×, ±2 pixels for 16× to 19×, and ±3 pixels for 20× to 25×. The steady state error is defined as the deviation of the target coordinates from the track-point at time after the transient period has finished. Furthermore, the POL-CFC algorithm offers 0% overshoot and 1.7 s rise-time. The percent overshoot and rise-time are the parameters to test the control system in its transient (initial) period. The percent overshoot is defined as:

$$\%\text{OS} = 100 \times \frac{(p - r)}{r}, \tag{48}$$

where p is the peak value and r is the reference or track-point, and the rise time is referred to as the time taken by the system to rise from 10 to 90% of the reference [5].

7 Experimental Results

In order to evaluate the performance of the proposed target tracking system (including the camera motion control), we selected a stationary object from the top-left section of the live video from the camera. Figure 21 shows the trajectory (position) of the object in the video frame, the control action (velocity) generated by the proposed controller and the positional error in both the axes, while the object was being centralized. Note that our PTU moves to the left if $v_\text{p} > 0$, and upward if $v_\text{t} < 0$. The curves illustrate that the controller generates a constant velocity initially at time $t = 0$ to start the motors from rest. When the controller senses that the positional error is not reduced adequately, it increases the speed quickly at $t = 0.4$ s. When the controller senses that the error is now reduced significantly and the current PTU velocity is greater than it should be for the current position of the object, it reduces the velocity gracefully until the object approaches the centre position (160, 120) in the video frame. We can observe that there is no overshoot, the rise time is 1.7 s, and the steady state error is zero. Thus, the stationary target selected from the position farthest from the center of the frame is centralized within 3 s.

We tested the performance of the proposed system also for various moving objects such as helicopters, airplanes, vehicles, walking and running persons, etc. in real-world scenarios. In Fig. 22, we present the results for a person with varying walking speed. We show the curves in horizontal axis only. We can observe that the object is initially centralized within t = 2 s and it remains centralized accurately regardless of the increasing velocity of the object. The small vibrations are due to the jerky motion of the walking person. Figure 23 shows how accurately and smoothly the modified car-following controller moves the camera towards a helicopter which is kept always at the center of the frame regardless of its changing velocity. The overlaid content at the top of every frame is "c_max, (x, y), zoom level, W (showing that the search is carried out in the small search *window*), (v_p, v_t)" where the pan and tilt velocities (v_p, v_t) are expressed in degrees/second. The edge-enhanced template is illustrated at the bottom-left of the frame. The helicopter is caught from the upper-left section of Frame 1, and is centralized within 1.76 s

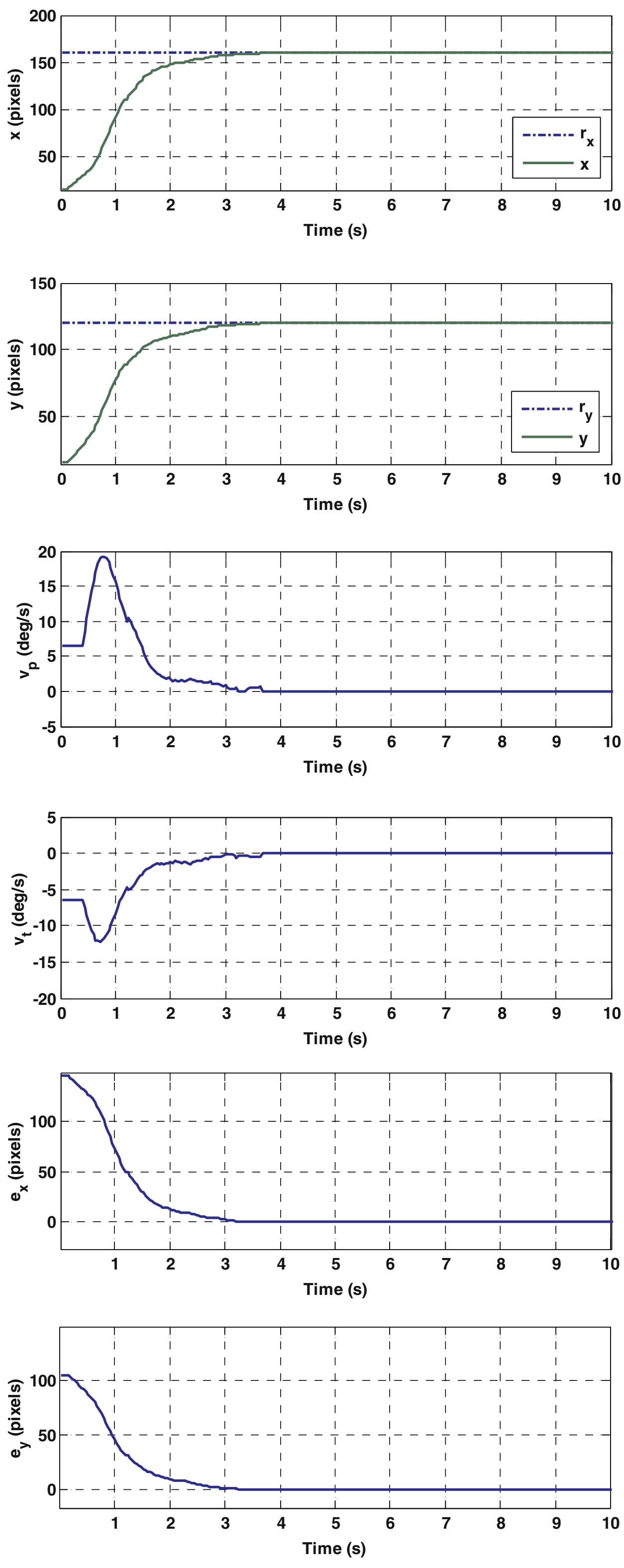


Fig. 21 Position, velocity and error curves in both axes, when a stationary object was being centralized in the video frames by the proposed tracking system

(Frame 44). The tracking is continued even when the helicopter is very faded in low-contrast video. Figure 24 shows

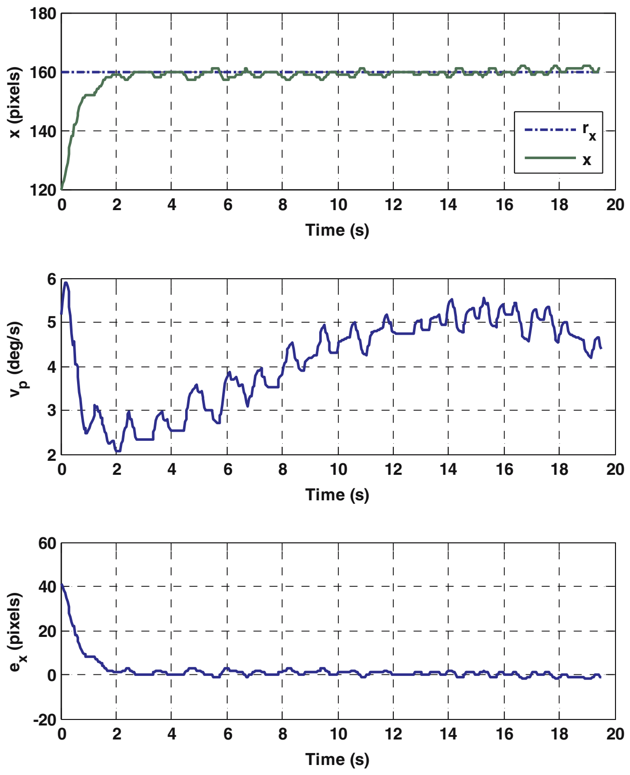


Fig. 22 Position, velocity and tracking error curves, when a moving object (a walking man) was being tracked

some frames from a tracking session in which a running boy is being tracked automatically by the moving camera, using the proposed target detection and tracking algorithm, in the presence of background clutter. It may be noted that the pan velocity varies abruptly according to the instantaneous velocity of the running boy. Figure 25 depicts some tracking frames in which a crow, which is a highly deformable object and flies with abrupt velocity changes, is being tracked automatically and robustly by our system. Figure 26 illustrates some frames from a tracking video, in which a station wagon is being tracked automatically by our moving camera in the presence of a lot of clutter, rotation, and scale changes, using the proposed target detection and tracking algorithm.

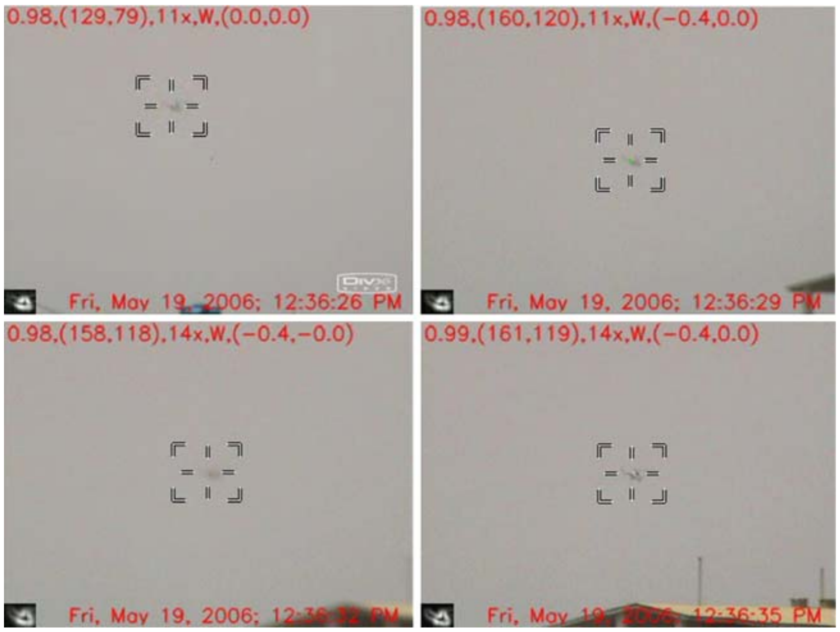
7.1 Implementation

The proposed object tracking application has been simulated in MATLAB, and then implemented for real-time field operation using C/C++ and LabVIEW. There are three threads in our LabVIEW program, all running in parallel, exploiting the parallel processing capabilities of the processor in a standard PC. The image processing operations and Kalman predictor run in the first thread, the POL-CFC algorithm for motion control along with the serial-communication with the PTU executes in the second thread, and the graphical-user-interface is implemented in the third thread. We have tested the application on a standard PC with 1.7 GHz processor and 512 MB RAM. The code can easily process 25 to 200 frames per second (fps), depending on the size of the template selected by the user while initialization, when the frame size is 320×240 pixels.

8 Conclusion and future directions

We introduced a robust and real-time framework to track an object of interest with a moving camera. The proposed

Fig. 23 Frames 1, 44, 88, and 128 (in raster scan order) of a real-time real-world tracking video, in which a very distant, dim and small helicopter is being tracked in low-contrast scenario automatically by a moving camera, using the proposed target detection and tracking algorithm. The edge-enhanced template is shown at the bottom-left of each frame. It may be noted that the helicopter is initially centralized in the frame within 44 frames (1.76 s)



0.98,(129,79),11x,W,(0.0,0.0)
Fri, May 19, 2006; 12:36:26 PM
0.98,(160,120),11x,W,(-0.4,0.0)
Fri, May 19, 2006; 12:36:29 PM
0.98,(158,118),14x,W,(-0.4,-0.0)
0.99,(161,119),14x,W,(-0.4,0.0)
Fri, May 19, 2006; 12:36:35 PM

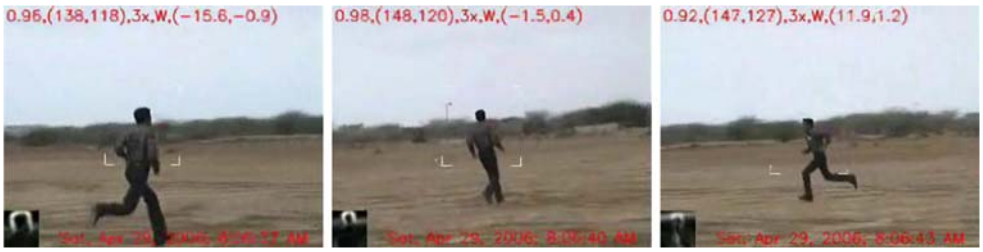

0.96,(138,118),3x,W,(−15.6,−0.9)
0.98,(148,120),3x,W,(−1.5,0.4)
0.92,(147,127),3x,W,(11.9,1.2)

Fig. 24 Frames 140, 210, and 280 of a tracking video, in which a running boy is being tracked automatically by a moving camera, using the proposed target detection and tracking algorithm

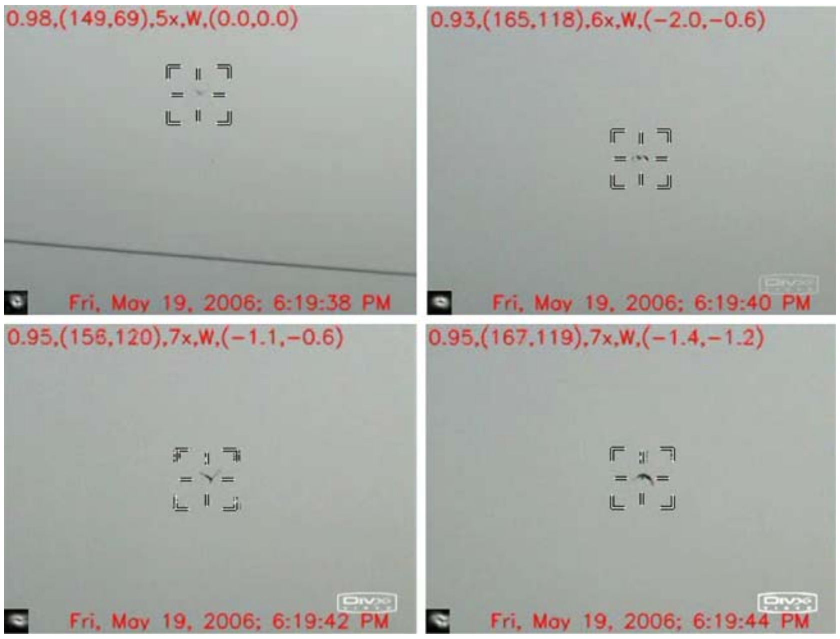

0.98,(149,69),5x,W,(0.0,0.0)
Fri, May 19, 2006; 6:19:38 PM
0.93,(165,118),6x,W,(−2.0,−0.6)
Fri, May 19, 2006; 6:19:40 PM
0.95,(156,120),7x,W,(−1.1,−0.6)
Fri, May 19, 2006; 6:19:42 PM
0.95,(167,119),7x,W,(−1.4,−1.2)
Fri, May 19, 2006; 6:19:44 PM

Fig. 25 Frames 1, 35, 64 and 91 (in raster scan order) of a tracking video, in which a crow (which is a highly deformable object and flies with abrupt velocity changes) is being tracked automatically by our moving camera tracking system. Note that the object in Frame 1 has very low-contrast with the back-ground

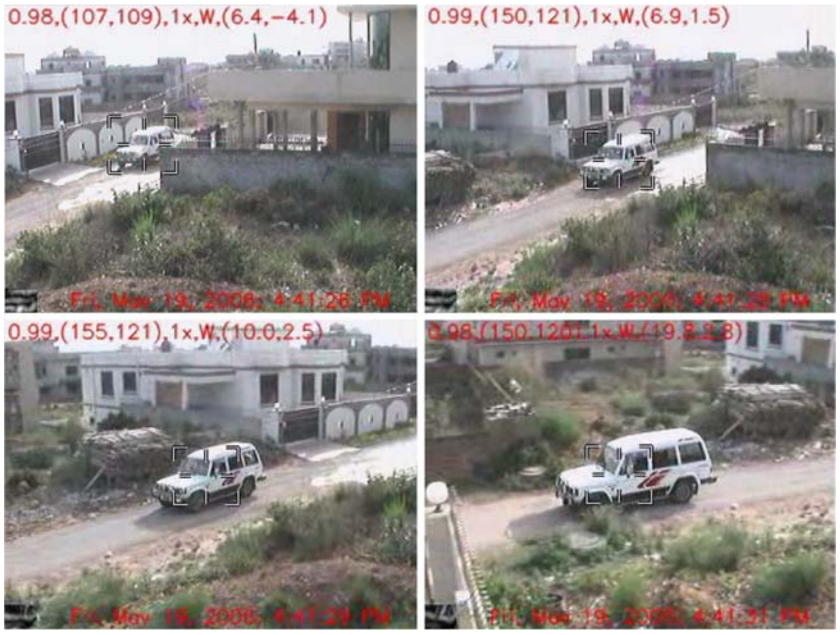

0.98,(107,109),1x,W,(6.4,−4.1)
0.99,(150,121),1x,W,(6.9,1.5)
0.99,(155,121),1x,W,(10.0,2.5)

Fig. 26 Frames 1, 25, 50 and 75 (in raster scan order) of a tracking video, in which a station-wagon is being tracked automatically by our moving camera tracking system in the presence of a lot of clutter, rotation, and scale changes

target tracking algorithm exploits Gaussian smoothing with an automatic standard deviation parameter, Sobel edge detector, normalization, thresholding and BCFNC. An effective and smooth method for updating the template is also introduced. The algorithm has been further enhanced using a Kalman predictor. A novel method is proposed to dynamically determine the location and size of the search-window depending upon the prediction and the prediction-error of the Kalman filter. The proposed algorithm has been compared with the other correlation-based target tracking techniques, and (for some sequences) the mean-shift and the condensation trackers. The results of the post-regression analysis proves that our target tracking algorithm is significantly more robust and accurate even in the presence of temporary object fading, significant background clutter, variations in the size of the object, variations in the illumination conditions, significant object maneuvering, multiple objects, obscuration, and partial occlusion of the object. However, a significant improvement would be to address the situation in a more systematic way when the object is *completely* occluded by other object(s) for a long time-duration. Moreover, if the template size is varied automatically according to the size of the object, the target tracking algorithm is likely to be even more robust.

Furthermore, a predictive open-loop car-following control (POL-CFC) for maneuvering the PTU has been presented. The controller calculates the predictive velocity of the object to be tracked using Kalman filter, and then smoothly adjusts the velocity of the PTU accordingly (without using any velocity feedback from the motors). As a result, the object remains always locked to the line-of-sight of the camera with good accuracy, regardless of the change in the velocity of the object. The POL-CFC algorithm offers 0% overshoot, 0 steady-state tracking error, and 1.7 s rise-time.

The overall algorithm has been implemented using multiple threads, and it runs with the speed of 25–200 fps depending on the size of the template, where each frame is of size 320×240 pixels.

References

1. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral Histogram. In: IEEE conference on computer vision and pattern recognition (2006)
2. 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, Melbourne (2005)
3. 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 in 4th world enformatika conference, Istanbul (2005)
4. Oppenheim, A.V., Schafer, R.W., Buck, J.R.: Discrete-Time Signal Processing, 2nd edn, Prentice Hall, Englewood cliffs (1999)
5. Kuo, B.C.: Automatic Control Systems, 7th edn. Wiley (1995)
6. Blackman, S., Popoli, R.: Design and Analysis of Modern Tracking Systems, Artech House, Boston, pp. 309–313 (1999)
7. Bradski, G.R.: Computer vision face tracking as a component of a perceptual user Interface. In: IEEE Workshop on Applic. Comp. Vis., Princeton, pp. 214–219 (1998)
8. Brookner, E.: Tracking and Kalman Filtering Made Easy. Wiley, NewYork (1998)
9. Brunson, R.L., Boesen, D.L., Crockett, G.A., Riker, J.F.: Precision trackpoint control via correlation track referenced to simulated imagery. Society of Photo-Optical Instrumentation Engineers, Bellingham (1992)
10. Chen, Q.-s., Defrise, M., Deconinck, F.: Symmetric phase-only matched filtering of Fourier–Mellin transforms for image registration and recognition. IEEE Trans. Pattern Anal. Mach Intell. **16** (1994)
11. Comaniciu, D., Visvanathan, R., Meer, P.: Kernel based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–575 (2003)
12. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: Proceedings, IEEE conference on computer vision and pattern recognition, Hilton Head, vol. 1, pp. 142–149 (2000)
13. Crow, F.: Summed-area tables for texture mapping. Comput. Graph. **18**(3), 207–212 (1984)
14. Cuevas, E.V., Zaldivar, D., Rojas, R.: Intelligent tracking. Technical Report (2003)
15. 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, Bombay, pp. 2–9 (1998)
16. Demuth, H., Beale, M.: Neural Network Toolbox for Use with MATLAB: User's Guide (v. 4), The Mathworks, Inc. (2001)
17. Doulamis, A., Doulamis, N., Ntalianis, K., Kollias, S.: An efficient fully unsupervised video object segmentation scheme using an adaptive neural-network classifier archtecture. IEEE Trans. Neural Netw. (2003)
18. 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)
19. Fagiani, C., Gips, J.: An evaluation of tracking methods for human-computer interaction, Senior Thesis, Computer Science Department, Boston College, Fulton Hall, Chestnut Hill, 02467, 2002
20. Fausett, L.: Fundamentals of Neural Networks: Architectures. Algorithms, and Applications, Prentice Hall, Englewood Cliffs (1994)
21. Fitts, J.M.: Precision correlation tracking via optimal weighting functions. In: 18th IEEE conference on decision and control including the symposium on adaptive processes (1979)
22. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digtal Image Processing Using MATLAB, Pearson Education Pte. Ltd., Singapore (2004)
23. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd edn, Prentice-Hall, Inc., Englewoodcliffs (2002)
24. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Pearson Education, Delhi (1999)
25. Isard, M., Blake, A.: CONDENSATION-conditional density propagation for visual tracking. Int. J. Comput. Vision **29**(1), 5–28 (1998)
26. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. Int. J. Comput. Vis. **1**(4), 321–331 (1988)
27. Kuglin, C., Hines, D.: The Phase Correlation Image Alignment Method. In: Proceedings of International conference cybernetics and society, pp. 163–165 (1975)
28. Lewis, J.P.: Fast Normalized Cross-Correlation. Industrial Light& Magic (1995)

29. Grewal, M.S., Andrews, A.P.: Kalman Filtering: Theory and Practice Using MATLAB, 2nd edn. J. Wiley, New York (2001)
30. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw, **6**, 525–533 (1993)
31. Hayes, M.H.: Digital Signal Processing. McGraw-Hill, New York (1999)
32. Mir-Nasiri, N.: Camera-based 3D tracking. In: Proceedings of IEEE Region 10: Tencon'05 Conference, Melbourne (2005)
33. Nummiaroa, K., Koller-Meierb, E., Gool, L.V.: An adaptive color-based particle filter Image Vision Comput. **21**, 99–110 (2003)
34. Perez, P., et al.: Color-based probabilistic tracking. European Conference on Computer Vision, pp. 661–675 (2002)
35. Porikli, F., Tuzel, O.: Multi kernel object tracking. In: Proceedings of IEEE International conference on multimedia and Expo, Amsterdam (2005)
36. Porikli, F.: Integral histogram: a fast way to extract histograms in cartesian spaces. In: IEEE conference on computer vision and pattern recognition (2005)
37. Porikli, F., Tuzel, O., Meer, P.: Covariance tracking using model update based on lie algebra. In: IEEE Conference on Computer Vision and Pattern Recognition (2006)
38. Ritter, G.X., Wilson, J.N.: Handbook of Computer Vision Algorithms in Image Algebra. CRC Press, Boca Raton (1996)
39. Rosales, R., Sclaro, S.: 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In: IEEE Conf. Comp. Vis. And Pat. Rec., vol. 2, pp. 117–123, Fort Collins (1999)
40. Intille, S.S., Davis, J.W., Bobick, A.F.: Real-time closed-world tracking, In: IEEE Conference on Comp. Vis. and Pat. Rec., Puerto Rico, pp. 697–703 (1997)
41. Stauffer, C., Grimson, W.: Learning patterns of activity using real time tracking. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 747–767 (2000)
42. Stone, H.S., Tao, B., McGuire, M.: Analysis of image registration noise due to rotationally dependent aliasing. NEC Research (2000)
43. Stone, H.S.: Fourier-based image registration techniques. NEC Research (2002)
44. Umbaugh, S.E.: Computer Imaging: Digital Image Analysis and Processing, CRC Press, Boca Raton (2005)
45. Wang, H., Suter, D., Schindler, K.: Effective appearance model and similarity measure for particle filtering and visual tracking. European Conference on Computer Vision (2006)
46. Welch, G., Bishop, G.: An Introduction to the Kalman Filter. TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill 27599–3175 (2004)
47. Wong, S.: Advanced correlation tracking of objects in cluttered imagery. In: Proceedings of SPIE, vol. 5810 (2005)
48. 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)
49. Yilmaz, A., Li, X., Shah, M.: Contour-based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Trans. Pattern Anal. Mach. Intelli. **26**(11), 1531–1536 (2004)

Author Biographies

Javed Ahmed received his bachelor's degree in Electronics Engineering from NED University of Engineering& Technology, Karachi (Pakistan), at the end of 1994. Then, he obtained MSc in Systems Engineering with distinction under the fellowship program at Pakistan Institute of Engineering & Applied Sciences, Islamabad (Pakistan), in 1997. He joined National Engineering and Scientific Commission in 1997 and worked in the fields of real-time embedded systems and signal processing. His current research areas are image processing, machine vision, control systems, signal processing, neural networks, fuzzy logic, and genetic algorithm. Presently, he is pursuing his PhD at National University of Sciences and Technology, Rawalpindi (Pakistan) under the NUST Endowment Fund Scholarship program, where he is focusing his work on real-time object detection and tracking. Besides, he is a research associate at Computer Vision Lab, University of Central Florida, Orlando (USA) since July 2006, and a member of IEEE and SPIE. He has presented two research papers in 4th World Enformatika Conference and IEEE Tencon Region 10 Conference held in Istanbul (Turkey) and Melbourne (Australia), respectively, in 2005.

Dr. M. Noman Jafri , who is Professor of Electrical Engineering at National University of Sciences and Technology, received his M.Sc. and PhD degrees both in Electrical Engineering with specialization in Control System Engineering from University of Ottawa, Canada in 1964 and 1967 respectively. Since then, he has been engaged in academics and various R&D and D&D activities, such as establishment of Postgraduate School of Electronics, establishment of Instrumentation, Computer and Control Complex and Performance and Reliability Analysis Division in an R&D organization. He has been responsible for design, development, testing and commissioning of full scope training simulator of a power plant for plant operators and safety analyzers. His main interests are in control, communications, information technology, image processing, artificial intelligence and robotics. He is member of IEEE and EIC (Canada) and has a number of publications in all the above fields.

Dr. Mubarak Shah received his M.S. and Ph.D. degrees both in Computer Engineering from Wayne State University, Detroit, Michigan, respectively in 1982 and 1986. Since 1986 he has been with the University of Central Florida, where he is currently Agere Chair Professor of Computer Science, and the director of Computer Vision lab. Dr. Shah is a *fellow of IEEE* and *IAPR*, was an IEEE Distinguished Visitor Speaker, and is often invited to present seminars, tutorials and invited talks all over the world. He received the Harris Corporation *Engineering Achievement Award* in 1999, the *IEEE Outstanding Engineering Educator Award* in 1997, *TOKTEN Awards* from United Nations Development Program in 1995, 1997, and 2000, *Teaching Incentive Award* in 1995 and 2003, *Research Incentive Award* in 2003, *Millionaires' Club Award* in 2005 and 2006, and *PEGASUS Professor Award* in 2006, an *honorable mention* for the ICCV 2005 *Where Am I?* Challenge Problem, and was nominated for the *best paper award* in ACM Multimedia Conference in 2005. He is an editor of international book series on "Video Computing"; editor in chief of *Machine Vision and Applications* journal, associate editor of ACM Computing Surveys, Area editor of Wiley Encyclopedia of Computer Science and Engineering, and an associate editor *Pattern Recognition* journal. He was an associate editor of the *IEEE Transactions on PAMI*, and a guest editor of the special issue of *International Journal of Computer Vision* on Video Computing.

Dr Muhammad Akbar received the B.S. degree in telecommunication engineering from University of Engineering and Technology, Lahore, Pakistan, in 1979. He received the M.S.E.E and Ph.D degrees from Michigan State University in 1984 and 1992. Since 1992, he has been a member of the faculty at College of Signals, NUST, Pakistan where he is currently a Professor. His research interests are in the areas of image processing, parallel processing, digital systems and wireless meshed networks. He is a member of Eta Kappa NU.