Implementation of a Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications

Yashpal Singh[A], Pritee Gupta[B], Vikram S Yadav[C]

Abstract

Detection of moving objects in video streams is the first relevant step of information extraction in many computer vision applications. Aside from the intrinsic usefulness of being able to segment video streams into moving and background components, detecting moving objects provides a focus of attention for recognition, classification, and activity analysis, making these later steps more efficient. We propose an approach based on self organization through artificial neural networks, widely applied in human image processing systems and more generally in cognitive science. The proposed approach can handle scenes containing moving backgrounds, gradual illumination variations and camouflage, has no bootstrapping limitations and achieves robust detection for different types of videos taken with stationary cameras.

Index Terms

Background subtraction, Motion detection, Neural Networks, Competitive Learning, self organization, Visual surveillance.

I. INTRODUCTION

Surveillance is the monitoring of behavior. Systems surveillance is the process of monitoring the behavior of people, objects or processes within systems for conformity to expected or desired norms in trusted systems for security or social control [1].

The word surveillance is commonly used to describe observation from a distance by means of electronic equipment or other technological means. However, surveillance also includes simple, relatively no- or low-technology methods such as direct observation, observation with binoculars, postal interception, or similar methods.

At a basic level, computers are a surveillance target because large amounts of personal information are stored on them. Anyone who can access or remove a computer can retrieve information. If someone is able to install software on a computer system, they can turn the computer into a surveillance device.

Figure: 1

Computers can be tapped by a number of methods, ranging from the installation of physical bugs or surveillance software to the remote interception of the radio transmissions generated by the normal operation of computers. Spyware, a term coined by self-proclaimed computer security expert

.CCTV is a collection of video cameras used for video surveillance. CCTV is generally used in areas where there is an increased need for security, such as banks, airports and town centers. A basic CCTV system comprises of the Camera, lens and power supply. Recording device, VCR or a digital video recorder and monitor. Closed-circuit television (CCTV) is the use of video cameras to transmit a signal to a specific place, on a limited set of monitors. One of the major market drivers for lawful authorized surveillance and infrastructure protection was the 9/11 terrorist attack. In light of this tragic event,

[A] Yashpal Singh is working as Reader with the department of Computer Science and Engineering, Bundelkhand Institute of Engineering and Technology, Jhansi, 284128, INDIA

[B] Pritee Gupta is working as Assistant Professor with department of Computer Science and Engineering, Skyline Institute of Engineering and Technology, Greater Noida,201306, INDIA

[C] Vikram S Yadav is working as Reader with the Applied Science and Humanities Department Bundelkhand Institute of Engineering and Technology, Jhansi, 284128, INDIA

Manuscript received March 5, 2010
Manuscript revised March 20, 2010

homeland security accelerated and endorsed privacy measures standardization/regulations. The main tasks in visual surveillance systems include motion detection, object classification, tracking, activity understanding and semantic description. Our focus here is on the detection phase of a general visual surveillance system using static cameras. The detection of moving objects in video streams is the first relevant step of information extraction in many computer vision applications. Aside from the intrinsic usefulness of being able to segment video streams into foreground and background components, detecting moving objects provides a focus of attention for recognition, classification, and activity analysis, making these later steps more efficient, since only moving pixels need be considered [2].The usual approach to moving object detection is through background subtraction, that consists in maintaining an up-to date model of the background and detecting moving objects as those that deviate from such a model. Compared to other approaches, such as optical flow [3], this approach is computationally affordable for real-time applications.

The background image is not fixed but must adapt to: Illumination changes

- **gradual**
- **sudden (such as clouds)**

Motion changes

- **camera oscillations**

- **high-frequencies background objects (such as tree branches, sea waves, and similar) Changes in the background geometry**

II. LITERATURE SURVEY

A. What is background subtraction

Identifying moving objects from a video sequence is a fundamental and critical task in many computer-vision applications. A common approach is to perform background subtraction, which identifies moving objects from the portion of a video frame that differs significantly from a background model. There are many challenges in developing a good background subtraction algorithm. First, it must be robust against changes in illumination. Second, it should avoid detecting non-stationary background objects such as moving leaves, rain, snow, and shadows cast by moving objects [4]. For example, by spatial and temporal smoothing, we can remove snow from a video as shown in Figure 2.1. Small moving objects, such as moving leaves on a tree, can be removed by morphological processing of the frames after the identification of the moving objects, as shown in Figure 2.2.

Figure 2.1 Video frames on left showing a traffic scene while snowing. Note the streaks in the image due to the snow flakes. The same video frame after spatial and temporal smoothing is on the right, without the snow streaks.

Figure 2.2 The video frame on the left highlights, in pink, the objects detected as moving. Note the movement of the leaves on the trees in the foreground. Morphological processing cleans up the video frame as shown on the right.

Some of the well-known issues in background maintenance, which will be specifically addressed in the sequel, include:

- ***Light changes***: the background model should adapt to gradual illumination changes;
- ***moving background***: the background model should include changing background that is not of interest for visual surveillance, such as waving trees;
- ***cast shadows***: the background model should include the shadow cast by moving objects that apparently behaves itself moving, in order to have a more accurate detection Of the moving objects shape;
- ***bootstrapping***: the background model should be properly set up even in the absence of a complete and static (free of moving objects) training set at the beginning of the sequence;
- ***Camouflage***: moving objects should be detected even if their chromatic features are similar to those of the background model.

Our implementation to moving object detection is based on the background model automatically generated by a self –organizing method [5].

B. RELATED WORK

Due to its pervasiveness in various contexts, background subtraction has been afforded by several researchers, and plenty of literature has been published. There is no unique classification of proposed methods. There are some advantages and tradeoffs of most existing methods, include the following ([6]-[8]).

- Parametric versus nonparametric: Parametric models are tightly coupled with underlying assumptions, not always perfectly corresponding to the real data, and the choice of parameters can be cumbersome, thus reducing automation.

On the other hand, nonparametric models are more flexible but heavily data dependent[9].

• Unimodal versus multimodal: Basic background models assume that the intensity values of a pixel can be modeled by a single unimodal distribution . Such models usually have low complexity, but cannot handle moving backgrounds, while this is possible with multimodal models at the price of higher complexity.

• Recursive versus nonrecursive: Nonrecursive techniques store a buffer of a certain number of previous sequence frames and estimate the background model based on the temporal variation of each pixel within the buffer, while recursive techniques and recursively update a single background model based on each input frame. In the first case, the background well adapts to eventual variations, but memory requirements can be significant; in the latter, space complexity is lower, but input frames from distant past can have an effect on the current background, and, therefore, any error in the background model is carried out for a long time period.

• Pixel-based versus region-based: Pixel-based methods assume that the time series of observations is independent at each pixel, while region- based methods take advantage of inter pixel relations, segmenting the images into regions or refining the low-level classification obtained at the pixel level. This step obviously increases the overall complexity. Our approach is based on the background model automatically generated by a self-organizing method and can be broadly classified as nonparametric, multimodal, recursive, and pixel based.

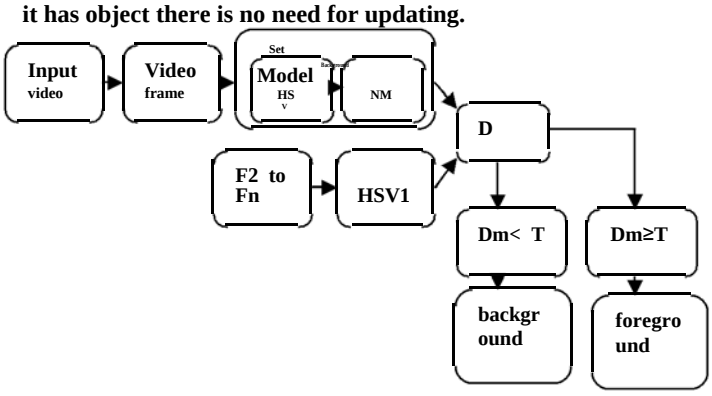
C. Winner takes all neural networks

Artificial Neural Networks (ANN) with unsupervised learning has been widely used in clustering tasks, dimensionality reduction, data mining, information extraction, density approximation, data compression, etc. A basic principle of unsupervised learning is the competition mechanism, in which the output units compete for activation. In some competitive learning algorithms only one output neuron is activated at any given time. This is realized by means of the so-called Winner-Takes-All (WTA) algorithm The principal problem in the implementation of the neural networks is the learning algorithm. There are basically two kinds of learning algorithm: supervised and unsupervised learning algorithm n in advance. The proposed model does not demand a teacher. Unsupervised neural network models used for clustering .In unsupervised models there is not an external signal (teacher) showing the desired value of an output unit or indicating whether the values of this output are appropriate or not (reinforcement signal). Thus, an unsupervised model should be able to discover, through, self-organization, patterns, characteristics, regularities, correlations or categories of the input data and encode them into the output . In winner-take-all (WTA) algorithm consider a two-layer neural network in which the input layer simply broadcasts the input stimuli and the output layer is a competitive one. The interlayer connections are modifiable and the competitive layer transforms an initially random categorization in a simple neuron responding to a group of samples.

- **During training, the output unit that provides the highest activation to a given input pattern is declared the winner and is moved closer to the input pattern, whereas the rest of the neurons are left unchanged.**
- **This strategy is also called winner-take-all since only the winning neuron is updated**
- **Output units may have lateral inhibitory connections so that a winner neuron can inhibit others by an amount proportional to its activation level**
- **Only the incoming weights of the winner node are modified.**
- **Winner = output node whose incoming weights are the shortest Euclidean distance from the input vector .**

III.PROPOSED WORK SPECIFICATION

In our project we have aimed to build such a surveillance system, which can detect motion even if the moving background, gradual illumination variations and camouflage into the background, this achieves robust detection for different types of videos taken with stationary cameras. To fulfill our aim we have used strong computing software called Matlab. We used Matlab to develop our work because Matlab provides image Acquisition and Image Processing Toolboxes which facilitate us in creating a good GUI and an excellent code. Input will be a video file in AVI format. First we will separate the frames from the video. Then we will convert it from RGB plane to HSV plane. Then we will form the neuronal map for the first frame (HSV) which will be our background. Then the remaining frames will be converted to HSV and it will be compared to the background by using the values given in the paper. After comparison we will decide whether the remaining frames consist of object or not. If it doesn't have object i.e. background we will update my background by using the values in paper and if it has object there is no need for updating.



Methodology:

- **Step 1:Convert the Input video into number of frames.**
- **Step 2:Set one frame as a background Image.**
- **Step 3: Calculate HSV for the frame and form Neuronal map 'NM' using weight vectors.**
- **Step 4: Calculate HSV1 for next frame.**
- **Step 5: Calculate Euclidean Distance 'D' between HSV1 and Neuronal map.**
- **Step 6: If Minimum distance is the best matchand less than T, Set the corresponding Pixel as background.**
- **Step 7: Update weight vectors.**
- **Step 8: Else set as object**
- **Step 9: Repeat Step 4 to Step 8 up to last frame**

IV. Study Analysis And Implementation

A. MODELING THE BACKGROUND BY SELF-ORGANIZATION

The main problem of the background subtraction approach to moving object detection is its extreme sensitivity to dynamic scene changes due to lighting and extraneous events. While the background model eventually adapts to these "holes," they generate false alarms for a short period of time. Therefore, it is highly desirable to construct an approach to motion detection based on a background model that automatically adapts to changes in a self-organizing manner and without *a priori* knowledge. We propose to adopt a biologically inspired problem-solving method based on visual attention mechanisms. Our approach defines a method for the generation of an active attention focus to monitor dynamic scenes for surveillance purposes. The idea is to build the background model by learning in a self-organizing manner many background variations, i.e., background motion cycles, seen as trajectories of pixels in time. Based on the learnt background model through a map of motion and stationary patterns, our implementation can detect motion and selectively update the background model. Each node computes a function of the weighted linear combination of incoming inputs, where weights resemble the neural network learning. Doing so, each node could be represented by a weight vector, obtained collecting the weights related to incoming links. In the following, the set of weight vectors will be called a *model*. An incoming pattern is mapped to the node whose model is "most similar" to the pattern, and weight vectors in a neighborhood of such node are updated. Therefore, the network behaves as a competitive neural network that implements a winner take-all function with an associated mechanism that modifies the local synaptic plasticity of the neurons, allowing learning to be restricted spatially to the local neighborhood of the most active neurons. For each color pixel, we consider a neuronal map consisting of n*n weight vectors. Each incoming sample is mapped to the weight vector that is closest according to a suitable distance measure, and the weight vectors in its neighborhood are updated. The whole set of weight vectors acts as a background model that is used for background subtraction in order to identify moving pixels.

Fig. 4.1 (Left) Simple image and the (right) neuronal map structure

B. Initial Background Model

In order to represent each weight vector, we choose the HSV color space, relying on the hue, saturation and value properties of each color. Such color space allows us to specify colors in a way that is close to human experience of colors. Let (h, s, v) be the HSV components of the generic pixel (x, y) of the first sequence frame I_0, and let C= (c1, c2…c_n^2) be the model for pixel (x, y). Each weight vector c_i, i=1,2… …..,n^2 is a 3-D vector initialized as c_i = (h, s, v).

The complete set of weight vectors for all pixels of an image I with N rows and M columns is represented as a neuronal map A with n x N rows and n x M columns, where the weight vectors for the generic pixel (x, y) of I are at neuronal map positions (i,j),i=nxx,……,nx(x+1)-1 and j=nxy,….,nx(y+1)-1. An example of such neuronal map structure for a simple image I with N=2 rows and M=3 columns obtained choosing n=3 is given in Fig. As depicted, the upper left pixel a of I (Fig. 4.1, left) has weight vectors (a_1,…….,a9) stored into the 3 x 3 elements of the upper left part of neuronal map (Fig. 4.1, right).

C. Subtraction and Update of the Background Model

After initialization, temporally subsequent samples are fed to the network. Each incoming pixel P_t of the t_{th} sequence frame I_t is compared to the current pixel model C to determine if there exists a weight vector that best matches it. If a best matching weight vector c_m is found, it means that P_t belongs to the background and it is used as the pixel encoding approximation, and the best matching weight vector, together with its neighborhood, is reinforced. In the latter case P_t is detected as belonging to a moving object (foreground). The above described background subtraction and update procedure for each pixel can be sketched as in the following algorithm.

Algorithm SOBS (Self-Organizing Background Subtraction)

Input: pixel value P_t in frame I_t,, t=0,1,2,……., LastFrame.

Output: background/foreground binary mask value B(P_t)

1. Initialize model C for pixel P_0 and store it into A
2. for t=1, Last Frame
3. Find best match c_m in C in *to current sample* P_t.
4. if (c_m found) then
5. B(P_t)=0 //*background*
6. *update A in the neighborhood of* c_m
7. else
8. B(P_t)=1 //*foreground*

In practice, we distinguish the whole process into two phases: a *calibration* phase and an *online* phase. The *calibration* phase involves the neural network initial learning, and consists in steps 1–6 executed on the first K+1 sequence frames (i.e., for t=1,K in step 2), with K< LastFrame .The *online* phase involves neural network adaptation and background subtraction, and consists in steps 2–8 executed on the last sequence frames (i.e., for t=K+1,LastFrame in step 2).The number K of sequence frames for the calibration phase basically depends on how many *static* (free of moving foreground objets) initial frames are available for each sequence.

Steps 3 and 6 of SOBS algorithm will next be described in detail.

1) Finding the Best Match in To Current Sample :

To determine which weight vector gives the best match, several metrics for detecting changes in color imagery could be adopted. Experiments lead us to employ the Euclidean distance of vectors in the HSV color hexcone , that gives the distance between two pixels P_i =(hi ,si, vi) and Pj=(hj, sj, vj) as

$$d(Pi,Pj)=\| (vi\ si\cos(hi), vi\ si\sin(hi), vi)-(vj\ sj\cos(hj), vj\ sj\sin(hj), vj) \|_2$$

Indeed, the representation of HSV values as vectors in the HSV color hexcone used in such distance measure allows to avoid problems connected with the periodicity of hue (that represents an angle) and with the instability of hue for small values of saturation (hue is undefined for null saturation) Weight vector c_m , for some m , gives the best match for the incoming pixel P_t if its distance from is minimum in the model C of P_t and is no greater than a fixed threshold

$$d(c_m, P_t) = \min_{i=1,..,n} d(c_i, P_t) \le \epsilon_2$$

The threshold € allows to distinguish between foreground and background pixels, and is chosen as

$$\epsilon = \begin{cases} \epsilon_1 & \text{if } 0 \le t \le K \\ \epsilon_2 & \text{if } t > K \end{cases} \quad (2)$$

with ϵ_1 and ϵ_2 small constants. Specifically , ϵ_1 should be greater than ϵ_2 , since higher values for ϵ_1 allow, within the first K sequence frames, to obtain a (possibly rough) background model including several observed pixel intensity variations, while lower values for ϵ_2 allow to obtain a more accurate background model in the online phase.

2) Updating in the Neighborhood of c_m *:* **If a best match c_m is found for current sample , P_t the weight vectors in the nxn neighborhood of c_m are updated according to selective weighted running average. In details, given the incoming pixel P_t (x ,y) at spatial position (x, y) and time t , if there exists a best match c_m in the model C of P_t , and c_m is present in the background model at position (x ,y), then weight vectors in the neighborhood of are updated according to**

$$A_t(i,j)=(1-\alpha_{i,j}(t))A_{t-1}(i,j)+\alpha_{i,j}(t)P_t(x,y) \quad (3)$$

Therefore, such updating allows to take into account spatial relationships among incoming pixel with its surrounding. In (3), $\alpha_{i,j}(t)=\alpha(t)w_{i,j}$, where $w_{i,j}$ are Gaussian weights in the neighborhood, that well correspond to the lateral inhibition activity of neurons. Moreover, $\alpha(t)$ represents the learning factor, chosen as

$$\alpha(t) = \begin{cases} \alpha_1 - t\,\dfrac{\alpha_1-\alpha_2}{K} & \text{if } 0 \le t \le K \\ \alpha_2, & \text{if } t > K \end{cases}$$

where $\alpha_1$and α_2 are predefined constants such that $\alpha_2 \le \alpha_1$.

D.Code With Function Explanation

The various codes used in this work are shown below. % --- Executes on button press in browse.

```
function browse_Callback(hObject, eventdata, handles)

% --- Executes on button press
in play. input=handles.input;
axes(handles.axes1);
movie(input);


% --- Executes on button press in
separation. filename=handles.filename;

%str1='frame';
str2='.bmp';
file=aviinfo(filename);  % to get inforamtaion

% --- Executes just before test is made
visible. a=ones(256,256);

axes(handles.axes1);
imshow(a);


% --- Executes on button press in browse.

[filename pathname]=uigetfile('*.avi','Select any
video'); if isequal(filename,0) | isequal(pathname,0)
   warndlg('Video is
not selected'); else
   input=aviread(filename);
```

```
   handles.input=input;

   handles.filename=filename;
   % Update handles structure
guidata(hObject, handles);
end
abt video file
frm_cnt=file.NumFrames          % No.of frames in the video
file
handles.frm_cnt=frm_cnt;
   h = waitbar(0,'Please wait...');
for i=1:frm_cnt
   frm(i)=aviread(filename,i);          % read the Video file
   frm_name=frame2im(frm(i));          % Convert Frame to
image file
   filename1=strcat(strcat(num2str(i)),str2);
   imwrite(frm_name,filename1);          % Write image file
    waitbar(i/frm_cnt,h)
end
 close(h)
 guidata(hObject, handles);
helpdlg('Frame seperation is Completed');

          waitbar(i/1000,h)
       end

% --- Executes on button press in map.
  input=imread('1.bmp');
 input=imresize(input,[256 256]);
[Ne_h, Ne_s, Ne_v]=neuro_map(input);
function          [neuro_map_h                    neuro_map_s
neuro_map_v]=neuro_map(input);%%% HSV
   hs=rgb2hsv(input);
   hue_v=hs(:,:,1);
   sat_v=hs(:,:,2);
   val_v=hs(:,:,3);
   [r c]=size(hue_v);
   %%%%% H-neuronal map with 3X3
   kk=1;
   jj=1;
   for rr=1:r
      for tt=1:c
         h=hue_v(rr,tt);
         s=sat_v(rr,tt);
         v=val_v(rr,tt);

         h1=h*ones(3);
         s1=s*ones(3);
         v1=v*ones(3);

         map_h(kk:kk+2,jj:jj+2) = h1;
         map_s(kk:kk+2,jj:jj+2) = s1;
         map_v(kk:kk+2,jj:jj+2) = v1;

        jj=jj+3;
      end

      jj=1;
      kk=kk+3;
   end


save Ne_h Ne_h;
save Ne_s Ne_s;
save Ne_v Ne_v;
msgbox('Neuronal map formation completed');

   % --- Executes on button press in detection.

K  = 15;
e1 = 0.1;
e2 = 0.006;
C2 = 0.05;
C1 = 1;
alpha1=1;
alpha2=0.05;
load Ne_h;
load Ne_s;
load Ne_v;
a='.bmp';
for i=2:20
   input=imread(strcat(num2str(i),a));
   input=imresize(input,[256 256]);

   hs=rgb2hsv(input);
   [r c p]=size(hs);
   Nh=Ne_h(2:769,2:769);
   Ns=Ne_s(2:769,2:769);
   Nv=Ne_v(2:769,2:769);
   row1=1;
   col1=1;
%    tic;
   for row=1:r
      for col=1:c
         H=hs(row,col,1);
         S=hs(row,col,2);
         V=hs(row,col,3);

         He=Nh(row1:row1+2,col1:col1+2);
         Se=Ns(row1:row1+2,col1:col1+2);
         Ve=Nv(row1:row1+2,col1:col1+2);
         col1=col1+3;
         if col1>=768
            row1=row1+3;
            col1=1;
         end
```

```
        d
=distance_calc1(H*ones(9,1),S*ones(9,1),V*ones(9,1),He
(:),Se(:),Ve(:));
%%%distance calculation
function d=distance_calc1(hi,si,vi,hj,sj,vj)
d(1:9) = ( ( vi(1:9) .* si(1:9) .* cos(hi(1:9)) ) -
( vj(1:9) .* sj(1:9) .* cos(hj(1:9)) ) ) .^2 + ...
      ( ( vi(1:9) .* si(1:9) .* sin(hi(1:9)) ) -
( vj(1:9) .* sj(1:9) .* sin(hj(1:9)) ) ) .^2 + ...
      ( ( vi(1:9) - vj(1:9) ) .^2)
 return;
        if i<=K
           e=e1;
           alpha=alpha1-(i.^((alpha1-
        alpha2)./K)); else
           e=e2;
           alpha=alpha2;
        end
   %%%% Minimum distance
        [dcm ind11]  = min(d);
        [i1 i2] = ind2sub([3 3],ind11);

        if dcm <= e
            %%%%%%%  set  corresponding  pixel  as
background and update  weight vectors
           cm = dcm;
           B(row,col) = 0;
%             tic;

[Ne_h(i1:i1+2,i2:i2+2),Ne_s(i1:i1+2,i2:i2+2),Ne_v(i1:i1+
2,i2:i2+2)] = updation...
(Ne_h(i1:i1+2,i2:i2+2),Ne_s(i1:i1+2,i2:i2+2),Ne_v(i1:i1+
2,i2:i2+2),hs(row,col,:),alpha);
function  [h  s  v]=
updation(neuro_map_h,neuro_map_s,neuro_map_v,P,alph
a);
h=neuro_map_h;
s=neuro_map_s;
v=neuro_map_v;

    return;
      else
           B(row,col) =1;
        % fore(row,col)=input(row,col);
        end
end
  end

%     tt1 = toc;
% fprintf('\nThe time for processing of one
frame %d\n',tt1);
 B=wiener2(B);
%    figure,imshow(B,[]);


cd output
imwrite(B,strcat(num2str(i),a));
cd ..
endhelpdlg('Detection completed');


% --- Executes on button press in Out.
frm_cnt=20;
number_of_frames=frm_cnt;
filetype='.bmp';
display_time_of_frame=1;
 cd 'output';
mov = avifile('new1.avi');
count=0;
for i=2:number_of_frames
name1=strcat(num2str(i),filetype);
   a=imread(name1);
   while count<display_time_of_frame
      count=count+1;
      imshow(a);
      F=getframe(gca);
   mov=addframe(mov,F); end
   count=0;
end
% close all
mov=close(mov);cd ..
```

V. Results And Discussion

Experimental results for moving object detection using the given approach have been produced for input image.

The number of weight vectors used to model each pixel has been fixed to 9 for all the reported experiments.

• Values for the distance thresholds should be chosen such that High values for allow to limit selectivity in the update of the background model during the calibration phase, enabling the inclusion into the initial background model of several observed pixel intensity variations. Lower values for should be chosen to obtain a more accurate background model in the online phase

• Learning factor in has been fixed to 1 for all the reported experiments, while, that depends on scene variability, has been experimentally chosen.

By applying the detection mask, we can observe that the object is almost perfectly detected, despite his camouflage and moving background. This is due to the fact that our model learns background motion trajectories captured by different weight vectors and, therefore, different color clusters, and to the adoption of distance, which allows a quite fair discrimination among colors.

VI. CONCLUSION

We have presented a Implementation of new self-organizing method for modeling background by learning motion patterns and so allowing foreground/background separation for scenes from stationary cameras, strongly required in video surveillance systems. Unlike existing methods that use individual flow vectors as inputs, my method learns background motion trajectories in a self organizing manner; this makes the neural network structure much simpler. Since we have a good result for the implementation of SOBs algo, the further study on shadow detection should be carry on in the future. Detection of images is popular problem until yet but now we have to find kernel that enhances high frequency parts. If we get the required result then we go for that is general case of. Future work will address techniques to get better result in moving object detection in video surveillance.

REFERENCE

[1] **J. M. Ferryman, Ed., in *Proc. 9th IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, 2006.**

[2] **R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson, "A system for video surveillance and monitoring," Tech.Rep. CMU-RI-TR-00-12, The Robotics Inst., Carnegie Mellon Univ.,**

[3] **J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 42–77, 1994.**

[4] **K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Proc. 7th IEEE Conf. Computer Vision*, 1999, vol. 1, pp. 255–261.**

[5] **L. Maddalena and A. Petrosino, "A self-organizing approach to detection of moving patterns for real-time applications," in *Proc. 2nd Int. Symp. Brain, Vision, and Artificial Intelligence*, 2007, pp. 181–190, Lecture Notes Comput. Sci. 4729.**

[6] **S.-C. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proc. EI-VCIP*, 2004, pp. 881–892.**

[7] **M. Piccardi, "Background subtraction techniques: a review," in *Proc. IEEE Int. Conf. Systems, Man, Cybernetics*, 2004, pp. 3099–3104.**

[8] **C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Realtime tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, May 1997.**

[9] **A. Elgammal, D. Hanvood, and L. S. Davis, "Nonparametric model for background subtraction," in *Proc. ECCV*, 2000, pp. 751–767.**