A BIOLOGICALLY INSPIRED OPTICAL FLOW SYSTEM FOR MOTION DETECTION AND OBJECT IDENTIFICATION

A Thesis presented to the faculty of the Graduate School
University of Missouri-Columbia

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

By

Vishal Rijhwani

Dr. Guilherme DeSouza, Thesis Supervisor

December 2007

The undersigned, appointed by the dean of the Graduate School, have examined the [thesis] entitled

A BIOLOGICALLY INSPIRED OPTICAL FLOW SYSTEM FOR MOTION DETECTION AND OBJECT IDENTIFICATION

Presented by Vishal Rijhwani,

A candidate for the degree of Masters of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

_____

Dr. Guilherme DeSouza, Assistant Professor, Electrical and Computer Engineering

_____

Dr. Zhihai He, Assistant Professor, Electrical and Computer Engineering

_____

Dr. Chi-Ren Shyu, Associate Professor, Computer Science

ACKNOWLEDGEMENTS

Looking back on my master's research years, I realized that I have gone through a great program which strengthened my academic knowledge and gave me a broader scope of what the electrical engineering discipline really is. Needless to say, I have faced a number of situations that seem to hard to overcome. However, I was lucky enough to have great faculty members and friends who always are ready to help me out.

First of all, I deeply thank my Lord for that He always listens to me whenever I pray for myself.

Especially, I feel a great thankfulness for my adviser Dr. Guilherme DeSouza. His precious advice and support always encouraged me in the right direction of my research goal. It would not have been possible for me to finish my work without his kind helps and wise suggestions through all my master's years.

I would like to express my acknowledgement to Dr. Zhihai He & Dr. Chi-Ren Shyu for kindly agreeing to join my thesis committee.

I am thankful my lab mates: Yuanqiang Dong, Hui Peng, Kyumin Han, Youyou Wang and Thomas Konig.

Last, but not the least, I deeply thankful to my family members in India: My father Kishanchand Rijhwani, my mother Maya Rijhwani, my sister Shilpa Dodani for cheering me up to confront new challenges in my life.

TABLE OF CONTENTS

BIOLOGICALLY INSPIRED OPTICAL FLOW SYSTEM FOR MOTION DETECTION AND OBJECT IDENTIFICATION

Vishal Rijhwani

Dr. Guilherme DeSouza, Dissertation Supervisor

ABSTRACT

Optical flow is possibly the best known method for motion segmentation. However its application is restricted to offline processing as it requires extensive computational resources and time. This thesis explores an optical flow method derived from observation on vision system of diptereous insect. The proposed method , Biological Optical flow (BioOF) was implemented using series of first order filters, and, therefore is much faster than any existing machine coded optical flow algorithm beside being hardware implement able.

Like other optical flow methods, the output of proposed BioOF has two components: horizontal optical flow and vertical optical flow; both of them can be combined in order to get a better final result in terms of motion segmentation. Unfortunately, this combined output of the BioOF can be heavily coupled with noise. So, in order to remove the noise, intensive image processing had to be performed. The result was an algorithm that can provide a good contour of the segmented object in an image.

Finally the object contour is converted to a Fourier feature space leading to a representation that is rotational and translational invariant. Over this feature space various classification algorithms including SVM, feature subset forward selection, Scatter matrix, and a simple linear classifier using principal component analysis and Mahanabolis distance were investigated

1.0 INTRODUCTION

This Thesis proposes a novel system to determine the presence of vehicular and human activity in remote locations. The main constraint for the system operation at these remote locations is in the requirement for battery operation and therefore the system design must be highly power efficient.

The relevant components of this system are shown in Figure 1. In the figure it is depicted the imaging subsystem, which consists of Optics, Image Plane (visible and near IR sensor), Detection Stage processing, and Recognition Stage processing. The function of the Detection Stage is to determine whether human or vehicular activity is occurring in the scene. This processing can be carried out in the form of advanced motion detection, as long as it can also determine whether the object has the relevant size and motion characteristic of either a human or a vehicle.

The Image Plane and the Detection Stage are the low power segments of the system. However, in addition to low power design, low average power must also be achieved by power cycling the sensor and Detection Stage. This requirement indicates that the algorithms developed for detection must operate with a minimum number of frames of video (ideally 1-3 frames).

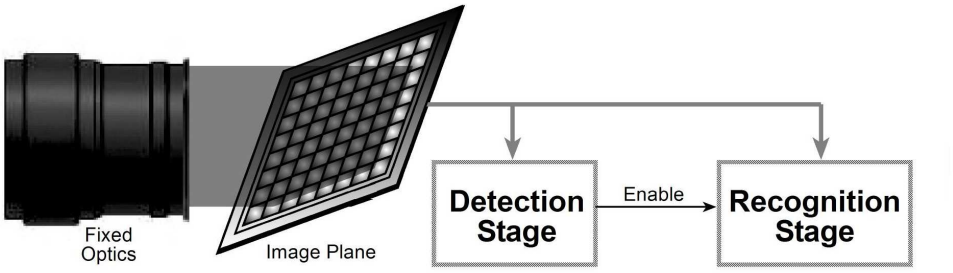


Figure 1 Imaging Subsystem Top Level View

Once an object has been determined to have the characteristics that could indicate human or vehicular activity the Recognition Stage is activated for final determination of the activity type and object(s) involved. Since the algorithmic set required for this recognition function is significantly more complicated and processing intensive, it is assumed that the Recognition Stage will incur in a much higher power drain.

The scope of this effort was mainly to evaluate the different approaches for the Detection Stage, but also to suggest future developments for both the Detection Stage and the Recognition Stage.

2.0 MOTION DETECTION

As we mentioned above, the Detection Stage is expected to perform a simple identification of the target object in order to decide whether to *awake* the Recognition Stage or not. Therefore, we divided the Detection Stage into two complementary modules: *Motion Detection* and *Object Identification*.

For the first module, we exploited a Biologically-inspired Optical Flow method (BioOF). As we will explain further, the main advantage of the BioOF method is that it **requires the storage of a quite small number of frames** (works well for just 1 frame) and it lends itself **easily to a hardware implementation** through the use of a small number of successive low and high pass filters.

For the Object Identification module, we explored various approaches to classification, including Support Vector Machines, Linear Classifiers using Mahalanobis Distance, Scattered Matrices, etc. In all cases, we employed a technique to encode the target objects using the Fourier Descriptors of their detected contour – in some cases the object contour was enhanced by a Snakes algorithm.

In the next few sections, we will present each of the methods investigated in this effort and the results obtained so far. At the end, we point out the shortcomings of these methods and suggest future developments.

2.1 Motion Detection using BioOF

Optical flow is the distribution of apparent velocities of brightness patterns in an image due to the projection of moving objects onto the image plane. Therefore, the optical flow arises from the relative motion between objects and the viewer (camera) in space.

Ideally, when different objects in a scene present different spatial velocities, their optical flow will also present different directions and/or intensities. This property of the optical flow can be exploited for, for example, object segmentation, motion detection, etc.

The first work on optical flow was by Horn & Schnuck [18]. Their formulation imposes two constraints for the detection of optical flow. The two constraints were called the Brightness Constancy and the Motion Smoothness Constraints. The first constraint assumes that the brightness of any given object or background pixel remains constant throughout the observed time interval, while the second assumes that neighboring pixels belong to the same object and therefore, the velocities of neighboring pixels are the same and that the velocity field of the brightness patterns varies smoothly everywhere. If these constraints are violated then the results obtained are expected to be poor.

In many practical cases however, brightness variation will occur due to, among other reasons: non uniformity in the illumination; inter and intra objects shadowing; reflectance changes due to motion of objects; inter reflections; etc. In order to take into account these changes in brightness, Gennert and Negahdaripour [10] proposed a model that incorporates the effects from any event that may change the image brightness. Similarly, in order to lift the smoothness constraint, a.k.a. the spatial coherence constraint, Black and Anandan [2] introduced a regularization term that forces the local flow vector to be close to the average of its neighbors.

However, in spite of these and many other efforts, many problems still arise depending on the specific situation and application, and, therefore, many other approaches using local or global estimators have been proposed in the literature [21]. The

major problem with most of these approaches is that they required enormous computation resources, which usually confines their application to the cases of offline processing.

Given our previous experience with some of the above robust algorithms and after having implemented a simplified version of such algorithms, it became clear early in this effort that such approaches would not be easily ported into hardware, as required. Therefore, we moved towards a completely different method which finds its inspiration in biologic organisms.

2.2 Biological Optical Flow (1D Model)

Over fifty years ago, Hassenstein and Reichardt [15] studied the optomotor response of insects and proposed a mathematical model (HR) that represents such responses to elementary motion detections. The HR model has been improved and used by various scientists since then. Recently, Higgins et al [17] have advanced these studies on dipterous insects by expanding the HR model and incorporating direction-selective circuits, also known as elementary motion detectors (EMD), to a newly proposed computational model.

In this effort, we propose an optical-flow-based segmentation method using Higgins' model for dipterous insects [17]. Like the eyes of real dipterous insects (Figure 2), our model forms a compound eye consisting of many individual facets called ommatidia. Each of the ommatidia consists of eight photoreceptors referred to as R1 through R8. The lens in each ommatidia focuses light onto the photoreceptor which detects light through a chemical photo transduction process and sends a signal to the axons of the visual ganglia.

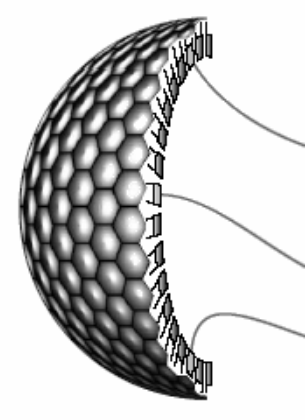


Figure 2 - Stylised representation of the ommatidae's compound eye [22].

Out of the eight photoreceptor, six of them R1 – R6 are believed to be the input to the elementary motion detectors (EMD) which compute motion by comparing changes in light intensity from adjacent visual units. The optical flow patterns are detected by neurons in the fly's lobula plate, called lobula plate tangential cell. These cells are, therefore, responsible for the sensitiveness to the direction, orientation and intensity of the motion

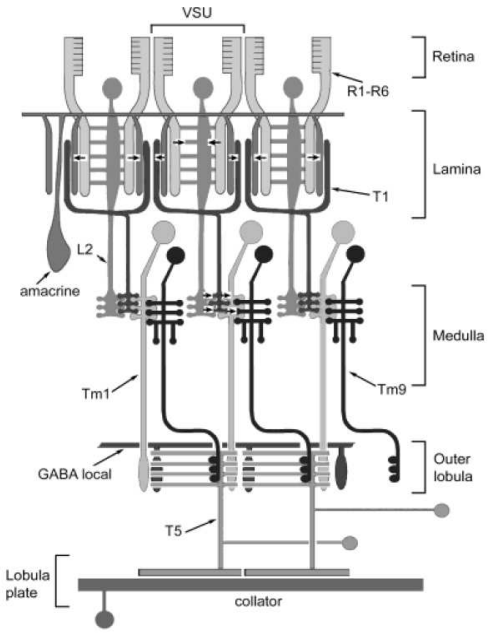


Figure 3 - Anatomical model for dipteran EMD circuit [1].

As Figure 3 indicates, the anatomical model for the EMD circuit of dipterous insects consists of the lamina amacrine cells (Am), lamina monopolar cells (L2), the Basket T cells T1, the transmedullary cells Tm1 and Tm9, the T5 bushy T-cell with an inhibitory interneuron.

For more information on this anatomical model, the reader is strongly encouraged to check the above mentioned works [1, 17, 22, 15]. However, in order to justify the computational model used in this work, we will explain the basic biological principles behind such model. As for example, that the amacrine cells (Am) receive photoreceptor inputs and synapse onto the T1 basket T-cell. Because the T1 cells show an inverted response to the input from the photoreceptors which also present a small DC component, the signal from the amacrine cell (Am) in the model is represented by an inverted and relaxed high pass filter (-RHPF). That is, the output of the amacrine cells is obtained by applying a high pass filter along with a small component of a low pass filter directly to the output from the photoreceptors (Figure 4).

Similarly, the lamina monopolar cell L2 also receives input from the photoreceptor, but its output does not contain any sustained response. In the computational model, this behaviour can be obtained using an inverted high pass filter (-HPF). Both the L2 and T1 cells are pre synaptic to Tm1, where the outputs of L2 and T1 are combined using a simple summation .

From the anatomical model, it can be inferred that the signal from two non-adjacent photoreceptors are combined through the T1 cell before it reaches the Tm1 cell. But the second pre-synapse at Tm1 gets its signal directly from the photoreceptor (L2 cell). That is, one of the pre synapses at Tm1 comes directly from the L2 cell, while the

other signal passes through the T1 cell before it is sensed by the Tm1 cell. This extra step creates a delay which was modeled by a low pass filter in series with the relaxed high-pass filter of the amacrine output (Am). Also, Tm1 shows no sensitivity to directional motion and histological studies show that Tm1 acts as an excitatory input to the T5 cell, with a second transmedullary cell Tm9 likely to be an inhibitory input.

Finally, it was observed that both Tm1 and Tm9 start at the same point (output of Tm1), but the Tm9 unit is delayed – which was again modeled by a low pass filter at the output of Tm1. Also, the shunting inhibition from Tm9 produces nonlinearity, which is required for direction selectivity. That is, the complete direction selectivity at the T5 level is achieved by inhibitory interneuron from the opposite side of the model. And, in order to get the output of T5 cell, both excitatory and inhibitory signals are half-wave rectified and then multiplied (dirty multiplication).

All filters in the computational model are first order filter. The time constant of all filters before Tm1 is 1/3 of the frame rate (for ex. 11 ms) and after Tm1 the constant is twice of the first (or 22 ms).

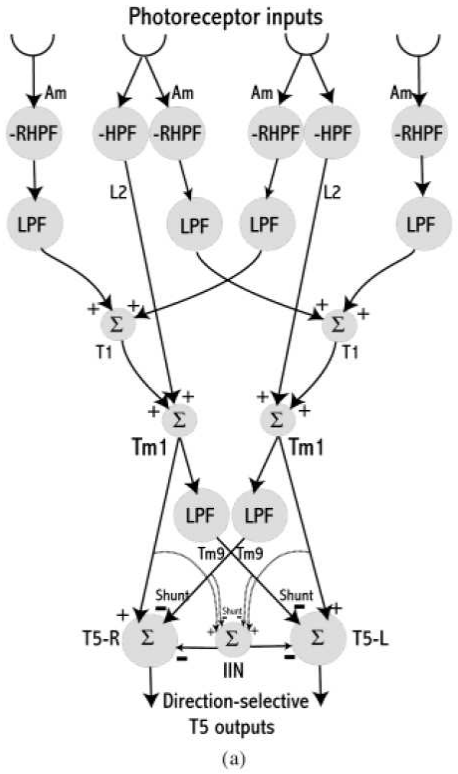


Figure 4 - Higgins' computation approach for the EMD model for dipterous insects [6].

In mathematical terms, the transfer function of the above high pass filters, in Laplace domain, is given by

$$H_{hpf}(s) = \frac{s * \tau}{(1 + s\tau)}$$

And the transfer function of low pass filter in Laplace domain is given by

$$H_{lpf}(s) = \frac{1}{(1 + s\tau)}$$

As mentioned earlier, the relaxed high pass filter allows a small sustained component (i.e. the output of LPF preceded by a HPF). Therefore, the transfer function of the relaxed high pass filters was obtained by

$$H_{rhpf}(s) = \frac{s\tau}{(1 + s\tau)} + \frac{k}{(1 + s\tau)} \quad \text{or}$$

$$H_{rhpf}(s) = \frac{k*(1+s\tau_2)}{(1+s\tau)}$$

Where $\tau_2 = \frac{\tau}{k}$ and $k = 0.1$

Again, the simple model for the shunting inhibition is mathematically expressed in terms of the so called *dirty* multiplication, where the output of Tm1 is the excitatory signal I_e; the output of Tm9 is the inhibitory signal I_s; and the total shunting inhibition is expressed as:

$$S(I_e, I_s) = pos(I_e) * \frac{(1 - pos(I_s))}{I_{s\max}}$$

where *pos* represent the half wave rectification.

Finally, the inhibitory interneuron gets the same input as the T5 cells combined and returns a weight inhibition. Mathematically this is given by

$$T5_L^{'} = T5_L - a*(T5_L + T5_R)$$

$$T5_R^{'} = T5_R - a*(T5_L + T5_R)$$

Where $T5_L$ and $T5_R$ represent the inputs to two adjacent T5 cells, while $T5_L^{'}$ and $T5_R^{'}$ are the same signals with inhibition.

For an a = 0.5, then

$$T5_L^{'} = 0.5*(T5_L - T5_R)$$

$$T5_R^{'} = 0.5*(T5_R - T5_L)$$

The most important characteristic of this biologically-inspired approach is its potential simplicity and effectiveness in terms of an implementation in hardware. In order to demonstrate this fact, we first simulated the 1D version of the model (Figure 4) using

different sampling rates. Also, we implemented in hardware the same 1D model using OpAmps as low and high pass filters, adders, etc.

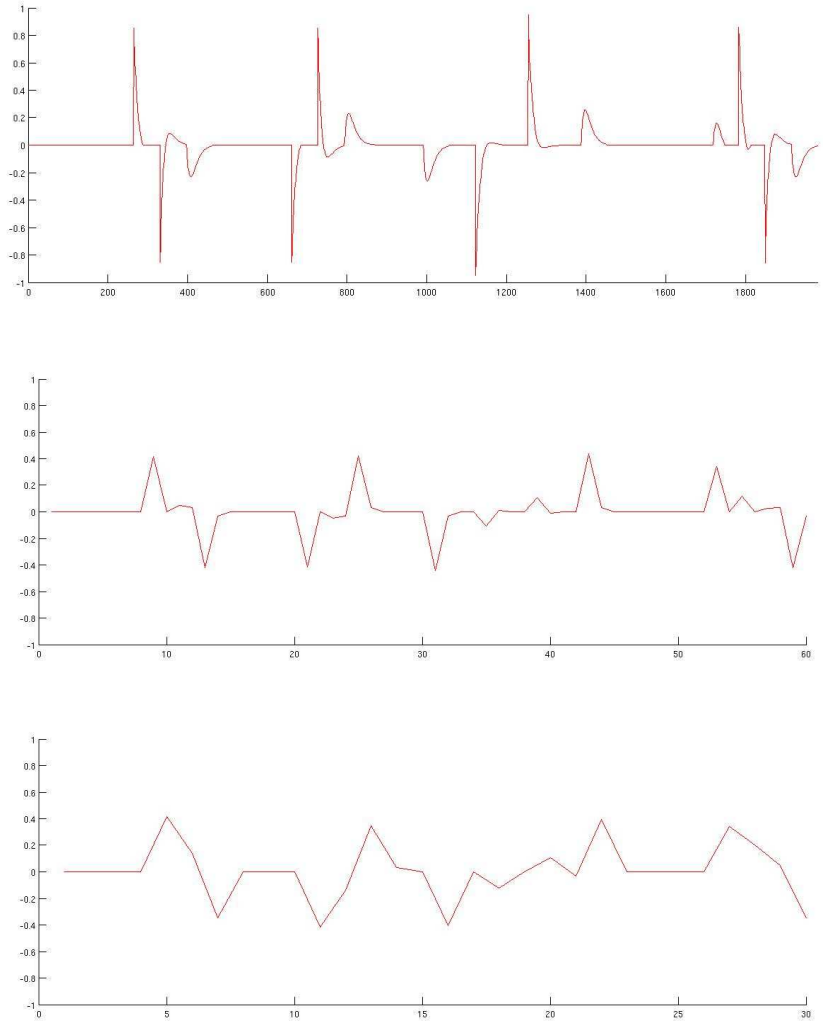


Figure 5 – Output (T5) of the 1D circuit: on top, at 30 samples per period; middle, at 3 samples per period; and bottom, 2 samples per period.

Figure 5 depicts the output (T5) of the circuit for three different sampling rates. First, at the "maximum" sampling rate of 30 samples per period, the output clearly indicates the direction of the motion by either two consecutive *positive-negative* or a *negative-positive* spikes. That is, on the top portion of Figure 5, we can see a *positive-negative* sequence of spikes followed by two *negative-positive* spikes followed again by a fourth, last, *positive-*

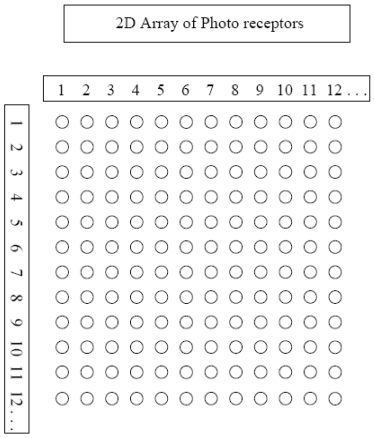
negative sequence of spikes – which indicates an object moving over the photoreceptors from left to right followed by two other objects moving from right to left and finally a fourth object also moving from left to right.

The same sequence of object motions captured at 3 samples per period (middle) and 2 sample per period (bottom) still produce a distinguishable sequence of spikes. That fact allowed us to implement a 2D version of the model that can capture horizontal and vertical motions using only one previous frame – a quite small memory storage requirement for the implementation of the various recursive filters used in this approach.

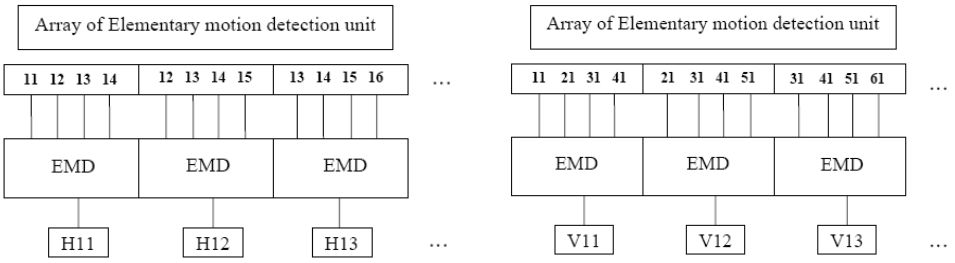
2.3 Biological Optical Flow (2D Model)

For this research, we expanded the model described in the previous section (Figure 4) into the 2D arrangement of Figure 6. In this 2D arrangement, two sets of 2D signals are computed for each image captured by the camera: a horizontal set and a vertical set. The sets contain one output signal (T5) for each pixel (*photoreceptor*) in the original image. That is, groups of four horizontally (or vertically) consecutive pixels are combined and used as the photoreceptors in the original EMD model to produce one output. The output (T5) for each group is then associated to the corresponding element in the 2D array that forms the horizontal (or vertical) optical flow component.

This idea is conveyed in Figure 6a and 6b (6c), where three of the various EMD blocks are shown and the corresponding horizontal (vertical) groups of 4 pixels in the input image – for example, pixels 11, 12, 13, and 14 (11, 21, 31, and 41) – produce the corresponding optical flow components in the output array, that is, H_{ij} (V_{ij}).



a)



b) c)

Figure 6 – 2D model as the arrangement of various simple EMD elements in Figure 4.

2.4 Preliminary Results

Figure 7a and 7b illustrate the typical result obtained from the Bio-Optical Flow method. The top portion of the figures contains the original grayscale image, while the middle and the bottom portion contain, respectively, the horizontal and the vertical components of the calculated optical flow.

a) b)

Figure 7 – Two examples of motion detection using Bio-Optical Flow.

It is important to stress the fact that the results above are the direct response from the BioOF method. There is absolutely no thresholding, filtering or any other post-processing being applied to these images. That is the case for the images above, as well as all other images related to the BioOF method in the Results section.

The only processing applied to the BioOF outputs is within the Object Identification Module. As it will be explained later, this processing was required in order to create a 1-pixel wide contour that could be used to create a feature space (Fourier Descriptors) for classification.

Another important point to make is that by using first order filters and only one sample per period, we guarantee that there is only one iteration per frame and that the algorithm only needs to store one previous frame. That is, when applying the various time

filters, whether we use a convolution technique or a recursive filter (which is indeed the case), there is only one iteration in time.

These choices raise a quite valid question regarding the possibility of implementing the time filters using look-up tables. While the use of such tables is obviously a great form to speed up the algorithm in software or hardware, we did not pursue this option yet simply because of the flexibility attained in doing otherwise.

3.0 OBJECT IDENTIFICATION

3.1 Contour Extraction From BioOF to Snake

As mentioned above, the BioOF algorithm does not contain any intrinsic filters or extrinsic post-processing to remove noise. The BioOF figures presented so far and presented in the results section consist of the simple and immediate combined outputs from the BioOF. Also, despite these figures may seem clean and free of noise, they do contain a reasonable amount of background noise. So, in order to perform object identification we need to first segment the object out from the background. Also, as it will be explained next, since all identification techniques approached relied on having a well defined contour of the object, a Snake algorithm had to be employed to create a 1-pixel wide object contour.

In summary, these are the steps of the Contour Extraction Algorithm:

- Foreground segmentation
- Foreground localization by pixel spatial distribution
- Median filtering
- Image Intensity Adjustment
- Image thresholding
- Connected Component
- Noise removal by image masking
- Image centralization and snake algorithm initialization
- Snake Active Contour

Foreground Segmentation

In the first step of the Contour Extraction, the algorithm determines where the maximum pixel concentration is located. That is, we apply a simple centroid localization using the pixel spatial distribution. This process is done along the rows and columns of the image and besides the centroid coordinate itself, it also returns a ROI (region of interest) – a bounding rectangular portion of the image where the object can be found.
This step is important because, by eliminating most of the low-intensity noisy pixels on the background, it changes drastically the pixel distribution. This new distribution is guaranteed to present the higher concentration of pixel as the foreground pixels – which will be required for the next steps of the contour extraction.

This step can also be used to determine if there is indeed an object in the scene. That is, if a ROI can not be found, then that means that there is no moving object present in the scene to be identified.

As we have just explained, the result from the previous step is a blank image with a rectangle in the middle. This rectangle contains the object pixels as well as some of the *"salt-and-pepper"* noise still present from the BioOF output. So, the next step of the process is a simple median filtering of size 3 by 3 [12].

It was observed that the Intensity of image is restricted in a very small region of the gray scale image, so it was decided to stretch the intensity level of image to entire gray scale of image & saturate low & high end of Intensity, doing so gave us a good separation between noise pixels & image data

After that, we used a simple image thresholding to remove some of the remaining noise pixels. However, in order to preserve weak edges of the object contour, this

thresholding was not very aggressive and further processing was required. So, a connected component step was used to remove small and isolated pixels in the image. For that, we used a 8-connect criterion.

To further remove the noise, the images obtained by the above steps were heavily thresholded to guarantee that none of the noise components can further be present. The images so created are then morphologically dilated to form a blob and the resulting blob was then used as a mask over the original image to eliminate all the pixel outside the mask.

After the steps above, the final result was a noise-free image with 1-pixel wide edges. However, in the process a few edges may get disconnected. That is, the contour is well defined, but it may also be *"broken up"*. That characteristic can create a large number of undesired high-frequency coefficients in the Fourier descriptors. So, in order to re-shape the object contour, we applied the Snake Algorithm. For that, the objects in the image had first to be centered and the snake algorithm was initialized with a radius of 5 pixels more than the distance of the farthest point in the image. In order to guarantee that the contour would be exactly over the image pixels, the image was converted into a square image by zero padding a few lines at the top of the original image.

- **Snake Algorithm**

Active Contours [20, 3, 28, 29] is a family of energy minimization methods which are used to create well-defined contours around objects. The way by which these active contours *slither* while minimizing their energy function resembles the slithering of a snake and hence these methods became known as *Snake Algorithms*.

A Snake algorithm exhibits dynamic behavior because of its energy minimization function. So, for better performance, the user has to initialize the snake close to the intended contour in order for the energy minimization to be able to carry the process the rest of the way. In our case, the output of the filtering process above (the ROI) also gave us a good initialization for the snake algorithm.

The snake model is basically a spline, where some forces push the snake towards salient image features like line, edges, etc, while other forces push the snake away. The process is controlled by the so called image forces and external constraint forces and it proceeds until it reaches equilibrium of the energy function – i.e. until it adapts to the contour of the object. The total energy functional of the snake is given by

$$E_{Snake} = \int_0^1 E_{Snake}(v(s))ds$$

Where v(s) represents the parametric position of the snake in pixel coordinates. This same energy function can be decomposed into:

$$E_{Snake} = \int E_{\text{int}}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))$$

The internal energy, E_{int}, gives rise to the forces that discourage the spline from bending and stretching. The image energy, E_{image}, represents the attractive image forces, while the constraint energy, E_{con} , can be used to model additional forces such as pressure forces. The constraint forces along with the image forces form the external forces. It is the external forces that pull the snake towards the desired image contour. In this research we did not use any constraint energy.

The internal energy is given by

$$E_{\text{int}} = (\alpha(s) \mid v_s(s) \mid^2 + \beta(s) \mid v_{ss}(s) \mid^2) / 2$$

The above energy is composed of a first order term $v_s(s)$ and a second order term $v_{ss}(s)$. The first order term is controlled by $\alpha(s)$ while the second order term is controlled by $\beta(s)$.

In the original snake algorithm [20], the image energy component was made of three different energy functionals. These energy functionals were: line, edge and termination; and they were combined in a weighted summation given by:

$$E_{image} = w_{line} E_{line} + w_{edge} E_{edge} + w_{term} E_{term}$$

The line functional is directly derived from the image intensity and is given by

$$E_{line} = I(x, y)$$

Depending on the sign of w_{line} the snake can be made attracted to light lines or dark lines. The edges in a image are basically abrupt discontinuities in grey levels, therefore such discontinuity can be easily detected by the image gradient:

$$E_{edge} = - \mid \nabla I(x, y) \mid^2$$

The E_{term} functional helps to find the termination of line segments and corners and it uses the curvature of the level lines:

$$E_{term} = \partial \theta / \partial n_{\perp}$$

where θ is the gradient angle and $n_{\perp} = (-\sin\theta, \cos\theta)$ is the unit vector perpendicular to the gradient direction.

By changing the weights of the energy functionals, snakes can be attracted from a fairly large distance towards the desired image features. However, if defined as above, snakes present a slow convergence around concave portions of an object. That is because,

in general, the external forces point towards the object boundary – as expected – but within a concave portion of the object, the forces will point horizontally and in opposite direction. Image, for example, that the object has a U-shape. Inside the "U", the curve gets pulled apart and the snake cannot conform to the concavity of the object. In order to address this problem, the Gradient Vector Flow functional were proposed [13, 14].

The GVF force is formalized by first defining an edgemap given by

$$f(x,y) = |\nabla[G_\sigma(x,y) * I(x,y)]|$$

where $G_\sigma(x,y)$ is a two-dimensional Gaussian function.

Since the BioOF already provides an edgemap, in our case, $f(x,y)$ was made the output from the above foreground-segmentation steps applied to the response from the BioOF.

Next, the GVF is defined as the vector $v(x,y) = [u(x,y), v(x,y)]$ that minimizes the energy functional

$$\varepsilon = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |v - \nabla f|^2 \, dxdy$$

From the above equation, when ∇f is small the energy is dominated by the first term in the integral: the square of the partial derivatives of the vector field – resulting in a smooth field. When ∇f is large the second term dominates the integration and ε is minimized by setting $v = \nabla f$.

Therefore, like the original functional, the GVF field points towards the object boundary when it is near to the image border, but in homogeneous region it varies smoothly. By doing so, the GVF forces have high covering range from either side of the boundary, but they can also force the contour to move into concave regions.

3.2 Feature Space

In object recognition – whether it is a simple identification as we propose here, or a more complete classification – it is required that we first define the feature space in which to perform the recognition [8]. In our case, we had yet two other options to choose in terms of the portion of the image that should be used: region based; or contour based [32]. In region-based techniques all the pixels within a shape are taken into account, while in contour-based, object representation is derived only from the boundary of the object. Given the output (Figure 7) of the Detection Stage, BioOF, it became quite clear that we should use a contour-based representation.

In terms of contour, the methods are further classified into: global shape descriptor; shape signatures; and spectral descriptors [32]. While simple to compute and rich in representation, global discriminator are effective only if the dissimilarity between the object classes is high.

On the other hand, most shape signature techniques are noise prone and therefore not robust. Additionally they require intensive computation during the similarity matching step, especially when hard-to-perform normalizations are included to provide rotational and/or scale invariance.

Finally, spectral descriptors, including Fourier Descriptor and Wavelet Descriptor [23, 31, 26, 30, 31], are obtained by applying spectral transformations on the shape signatures, such as: central distance; complex coordinates; curvature function; cumulative angles; etc. In most cases, the most prominent shape attributes of the object are represented by the first few low-frequency terms, while the high-frequency terms capture finer details of the shape, which usually is not a significant factor in the

classification process. That is exactly the reason why Fourier Descriptors overcome the problems of noise sensitivity of other shape representation methods. That is also the case of Wavelet Descriptor [26, 30, 12], however, this latter may require intensive computations in order to be made rotation invariant.

Since it was imperative in this effort that the object identification be made invariant to scale and rotational, we opted to use a Fourier Shape Descriptor. In fact, this choice should not only solve the problem of rotational and scale invariance, but it also converts the original 2D image space into a 1D space – the contour – thus making the classification task even simpler.

3.2.1 Fourier Descriptor

Assuming we have a noise-free, 1-pixel wide, continuous contour of the objects, we can now move to the feature representation of the object for classification.

The central distance or centroid distance function is expressed as the distance of the boundary points to their centroid and it's given by:

$$r(t) = \sqrt{(x(t) - xc)^2 + (y(t) - yc)^2}$$

By definition, this centroid distance representation is invariant to translation and rotation of the object, but it is not invariant to occlusions. Moreover, since all the points are expressed with respect to their distance to the centroid, small variations in the shape may result in large changes in the centroid function, causing a significant error in object identification. Initially, we employed the centroid difference function to compute the signatures for the images, but this approach was later dropped due to the problems above.

On the other hand, the curvature function is expressed as the second derivative of the object boundary, which is equivalent to the first derivative of the boundary tangent and can be approximated within a window of size *w* by:

$$k(t) = \theta(t) - \theta(t-1)$$

Where $\vartheta(t)$ is the arctangent of the local slope of the boundary and can be approximated by:

$$\theta(t) = \arctan(\frac{(y(t) - y(t-1))}{(x(t) - x(t-1))})$$

The curvature function so defined presents a discontinuity at size 2π In order to avoid such discontinuity, a cumulative angular function is introduced instead. This function is the net amount of angular bend between the starting position z(0) and the current position z(t) along the shape boundary. The cumulative curvature function works better also because it does not result into large error due to small changes in the shape (e.g. noise). Therefore, the new k(t) can be redefined as:

$$\varphi(t) = [\theta(t) - \theta(0)] \bmod (2\pi)$$

$$k(t) = \varphi(t) - \varphi(t-1)$$

Finally, the Discrete Fourier transform of the shape signature k(t) is given

$$U_n = \frac{1}{N}\sum_{t=0}^{N-1} k(t) \exp\left(-\frac{j2\pi nt}{N}\right)$$

For *n = 0, 1… N-1*

By the definitions above, the shape signature using curvature is translation invariant. Therefore, the representation of the shape using Fourier Descriptor is also translation invariant.

In order to achieve rotational invariance, we ignore the phase information by taking only the magnitude values of the Fourier descriptors. Also, since both the centroid and the curvature difference signatures are real valued, only the first *N/2* Fourier Descriptors needs to be indexed. Finally, the scale invariance is obtained by dividing the magnitude values of the Fourier descriptors by their D.C. component.

$$F = \frac{|FD_1|}{|FD_0|}, \frac{|FD_2|}{|FD_0|}, \ldots\ldots\ldots \frac{|FD_{N/2}|}{|FD_0|}$$

In the next sections, we report the experimental results that we obtained for the Fourier Descriptor using the curvature function signature.

Once we have defined the feature space to be used, we can move to the actual identification (classification) phase. For this phase, we explored various classification algorithms, such as using neural networks, linear classifiers, scattered matrix classifier, etc… In the next subsections, we will present all the classification algorithms explored in this effort.

3.3 Identification Using Support Vector Machines

Classification of objects using SVM [16, 14, 5] is usually a neural network based approach. The support vector machine can detect a linear (or nonlinear) optimal hyper plane that maximizes the separability of the classes in the data.

Suppose we want to separate two classes given by the feature vectors $\vec{x} = [x_n]$, where *n* is the dimension of the feature space, with a hyper plane in $\Re^n$.

The equation of the optimal linear hyper plane is given by

$$w^T[x_n] + b = 0 \qquad (1)$$

That is,

$$w^T \vec{x}^{(i)} + b > 0 \qquad \text{for } \forall \vec{x}^{(i)} \in \text{ Class } C_1 \qquad (2)$$

$$w^T \vec{x}^{(i)} + b < 0 \qquad \text{for } \forall \vec{x}^{(i)} \in \text{ Class } C_2$$

Figure 8 illustrates how this optimum hyper plane is constructed as a function of the support vectors $\boldsymbol{x}^{(s)}$ for the trivial case of a two-dimensional feature space.

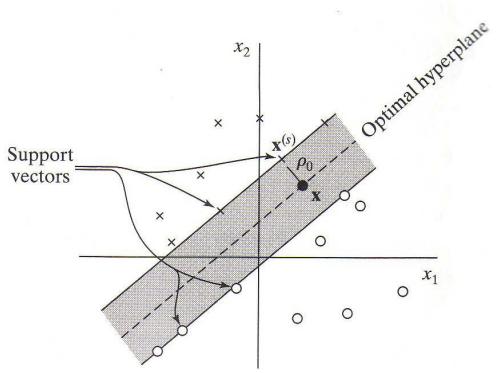


Figure 8 – Example of a SVM for a two-class, two-dimension classification problem.

Moreover, given the above equation of the hyper plane, the distance of $\vec{x}^{(i)}$ to this hyper plane is given algebraically by:

$$\vec{x}^{(i)} = x_p + \rho * (\frac{w_0}{\| w_0 \|}) \qquad (3)$$

Where x_p is the projection of $\vec{x}^{(i)}$ onto the optimal hyper plane. Once again, by definition, a positive value for ρ indicates that $\vec{x}^{(i)}$ is on the positive side of hyper plane and therefore, it belongs to class C_1, while a negative value for ρ indicates that $\vec{x}^{(i)}$ belongs to class C_2. Therefore, the classification function can be expressed as:

$$g(\vec{x}^{(i)}) = w_0^T \vec{x}^{(i)} + b = \rho * \| w_0 \| = \begin{cases} > 0 & if\ class\ C_1 \\ < 0 & if\ class\ C_2 \end{cases} \qquad (4)$$

Also, as shown in Figure 9, the distance from the origin to the optimal hyperplane is given by $b/\|w_0\|$ and if $b>0$, then the origin lies on the positive side of the

optimal hyperplane. Otherwise, if $b<0$, then the origin lies on the negative side of the hyperplane. Finally, if $b=0$, then the optimal hyperplane passes through the origin.

So, finding a SVM becomes simply the problem of finding the parameters w_0 and b and a set of supporting vectors $\boldsymbol{x}^{(s)}$ that minimize the error in equation (4) above or maximize the margin of separation (shaded area in Figure 8) between the two classes. In that case, the data points for which the constraint equation approximates zero are the support vectors, $\vec{x}^{(s)}$. These vectors lie very close to the optimal data points and are difficult to classify, so, instead, we define the support vectors as:

$$g(\vec{x}^{(s)}) = w_0^T \vec{x}^{(s)} + b = \begin{cases} +1 & for\ C_1 \\ -1 & for\ C_2 \end{cases}$$

As in (4), the optimal distance of a support vector $x^{(s)}$ to the hyper plane is given by

$$\rho = \frac{g(x^{(s)})}{\| w_0 \|}$$

$$\rho = \frac{1}{\| w_0 \|} \qquad \text{if} \quad d^{(s)} = +1$$

$$\rho = -\frac{1}{\| w_0 \|} \qquad \text{if} \quad d^{(s)} = -1$$

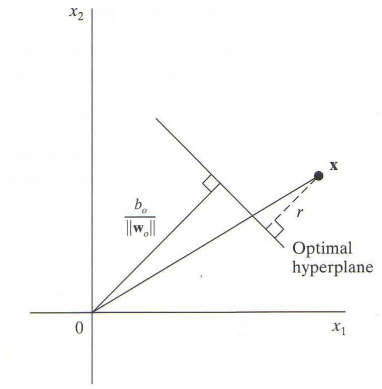


Figure 9 – Distance of a point $\vec{x}^{(i)}$ to the optimal hyperplane

Finally, the margin of separation between the two classes that constitute the training set is therefore:

$$2\rho = \frac{2}{\| w_0 \|}$$

And we conclude that maximizing this margin of separation between the classes is equivalent to minimizing the Euclidean norm of weight vector, w_0.

3.4 Identification Using Feature Subset Forward Selection

One common problem in any classification method is with the dimensionality of the feature space. Large feature spaces may imply in long and costly searches, while small feature spaces usually leads to loss in power of discrimination (class separability) [8].

For the objects used in this effort, the feature space define by the Fourier descriptors have 100 dimensions. However, not all of these 100 dimensional coefficients are useful for classification. That is, some of the coefficients capture high frequency changes in the shape of the object, which may not be very useful for its identification. This fact in addition to the stringent real-time constraints of this effort led us to select only the most effective coefficients for classification.

By brute force, the best subset of *m* coefficients out of the *n* initial dimensions can be found by evaluating the class seperability under all possible combinations of m coefficients, C_m^n. That number, of course, can become quite large even for small values of *m* and *n.* So, instead, a *forward selection* procedure was chosen.

The procedure starts by evaluating the class separability using a single feature as shown in following figure.

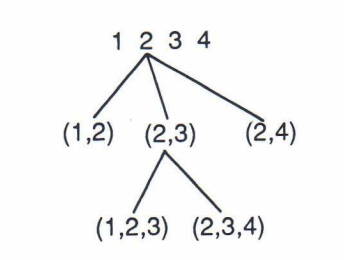


Figure 10 – Forward Selection

In this example, if it is required that 3 out of 4 features be selected, the best choice (best class separability) using a single feature is achieved using feature number 2. Next, the algorithm selects a second feature, which is combined with the other features already chosen, in this case, (1,2) (2,3) and (2,4) are selected. Again, if the pair (2,3) gives the least classification error, this pair is selected and the algorithm proceeds to select another feature from the remaining unselected features until the desired number of features is reached – in this example, 3 features are desired.

The criterion used to measure the class separability in the forward selection algorithm was the Mahalanobis distance. That is, for each class in the training set and for each subset of features, we chose as the next feature the one that minimized the overall error in classification based on the Mahalanobis distance.

3.5 Identification Using Scatter Matrices

In discriminant analysis, it is often common to use *within class*, *between class* and *mixture scatter matrices* as the criteria for class separability.

The within class scatter matrix shows how scattered samples are around class expected vector and it is expressed by:

$$S_w = \sum_{i=1}^{L} P_i \sum_i$$

Where L is the number of classes, P_i is the probability of class C_i – since usually all classes have equal distribution, the probability of each class is assumed to be 1/L – and $\sum_i$ is the covariance matrix of class C_i.

Similarly, the between class scatter matrix measures how scattered the expected vectors of each class are around the mixture mean. It is given by

$$S_b = \sum_{i=1}^{L} P_i (M_i - M_0)(M_i - M_0)^T$$

Where M_i is the mean of each class, and M_0 represents the expected vectors of the mixture distribution. M_0 is given by

$$M_0 = \sum_{i=1}^{L} P_i M_i$$

The mixture scatter matrix is simple the sum of the within class matrix and the between class matrix above. That is:

$$S_m = S_w + S_b$$

A common criteria for class separability using scatter matrices should maximize the between class scatter while (at the same time) minimizing the within class scatter. That property can be expressed by a single scalar as:

$$J = trace(S_w^{-1} S_b)$$

When there exist only two classes to choose, the matrix above, $(S_w^{-1} S_b)$, is rank one and the eigenvector associated to the only non-zero eigenvalue defines a hyperplane similar to the one found for the SVM. In this case, however, the eigenvector can be analytically expressed by:

$$\Phi = \frac{S_w^{-1}(M_2 - M_1)}{\left\| S_w^{-1}(M_2 - M_1) \right\|}$$

And the decision function is simply:

$$g(\vec{x}^{(i)}) = \frac{(M_2 - M_1)^T S_w^{-1}}{\left\| S_w^{-1}(M_2 - M_1) \right\|} \vec{x}^{(i)} = \begin{cases} > 0 & \textit{for class } C_1 \\ < 0 & \textit{for class } C_2 \end{cases}$$

For this effort, the distribution of humans and vehicles using the Fourier descriptor as the feature space proved to be highly overlapped.

3.6 Identification Using Principal Component Analysis (Kahunen- Loéve Transform)

The last identification method using in this effort was the PCA method [8]. In this approach, all feature vectors – in this case, the Fourier coefficients of the shapes – are transformed by being multiplied by the eigenvectors of the covariance matrix. That is, the covariance matrix of all samples is decomposed into its eigenvector and eigenvalue matrices.

$$\Sigma = \Phi^T \Lambda \Phi$$

Where Σ is the covariance matrix and Λ and Φ are respectively the eigenvalue and eigenvector matrices.

Then, the new feature vector is obtained by multiplying $\vec{x}^{(i)}$ by Φ. That is:

$$\vec{y}^{(i)} = \Phi^T \vec{x}^{(i)}$$

The new feature vector is truncated and only the *n* first (principal) components of the new vector – corresponding to the *n* largest eigenvalues of the covariance matrix – are used for classification, which can be carried out using any of the classification methods

discussed above or a simple Euclidean distance criterion. For this effort we applied the PCA followed by a feature subset forward selection and the Mahalanobis distance.

4.0 EXPERIMENTAL RESULTS MOTION DETECTION

4.1. Motion Detection Results Overview

This section describes the results for the experiments on object detection using the BioOF method. Table 1 shows the list of videos to which the methods were applied. All videos have a frame size of 720x480 and they were all converted to grayscale before being used. These sequences were selected from a slightly larger set, but they represent a comprehensive set in terms of the available situation, that is: camera distances (near/medium/far); types of target (vehicles/people); speed of motion; contrast (high/low); and presence of clouds.

It was observed that in general BioOF methods worked very well and detected at least some parts of the moving objects in most, if not all of the sequences. The moving clouds or their shadows did not cause misdetection, and the moving objects were not missed unless they were very far and have very low contrast. The night sequence videos were very challenging due to the existence of large noise and lack of structure, except for a couple of cases that are shown below.

This method has no parameters to be adjusted and the results reported here were obtained using only one sample per period for maximum performance and minimum memory storage requirements – as discussed in section 2.2

Table 1: List of video sequences.

#	File name	Type	view	#of frames	other information
1	HC_Truck	synthetic	far	428	white truck, daytime
2	HC_Truck_Clouds	synthetic	far	428	white truck, daytime, cloud shadows
3	LC_Truck	synthetic	far	428	dark truck, daytime
4	LC_Truck_Clouds	synthetic	far	428	dark truck, daytime, cloud shadows
5	LongShotCamo	synthetic	far	428	camouflage truck, daytime, cloud shadows
6	LongShotLC	synthetic	far	428	dark truck, daytime, cloud shadows
7	LongShotHC	synthetic	far	428	white truck, daytime, cloud shadows
8	MediumShotCamo	synthetic	medium	359	camouflage truck, daytime
9	MediumShotLC	synthetic	medium	359	dark truck, daytime
10	MediumShotHC	synthetic	medium	359	white truck, daytime
11	NearShotCamo	synthetic	near	251	camouflage truck, daytime
12	NearShotLC	synthetic	near	251	dark truck, daytime
13	NearShotHC	synthetic	near	251	white truck, daytime
14	Person_Med	synthetic	medium	428	slow moving person, 7x13 size,daytime
15	Person_Far_Clouds	synthetic	far	428	slow moving person, 4x7 size, daytime , cloud shadows
16	Long_Person_cloud	synthetic	far	428	same as previous, different texture
17	GroupDark_FarClouds	synthetic	far	428	slow moving group, daytime, cloud shadows
18	Long_GroupD_cloud	synthetic	far	428	same as previous, different texture
19	umc_test_sequence	real	medium	247	walking person, night shot, moving shadows
20	daytime_slice_00_10_edit	real	medium	7397	truck, person
21	daytime_slice_10_20_edit	real	near	994	truck, person, moving plants
22	daytime_slice_20_30_edit	real	far	15480	truck, person, slow motion, very far

23	daytime_slice_30_40_edit	real	far	4357	truck, person, slow motion, very far
24	daytime_slice_40_50_edit	real	far	5017	truck, person, slow motion, very far
25	daytime_slice_50_60_edit	real	far	8677	truck, person, very slow motion, very far
26	nighttime_slice_00_10_edit	real	near	7044	night time, very noisy
27	nighttime_slice_50_60_edit	real	near	2279	night time, very noisy

4.2 Implementation details

The BioOF was implemented in Matlab in a highly efficient manner: with few loops and exploring parallelism as much as possible. Zero logic was assigned as -3, while logic one was assigned -2. All the filter i.e high pass, low pass and relaxed high pass filters are first order recursive filters with the upper filter of the elementary motion detector having time constant of 10 milliseconds and the lower filter of the elementary motion detector of 20 milliseconds The sampling rate was 1 sampling period of 66 milliseconds. To make the filters more time efficient the filters were applied on the entire 2D signal at once.

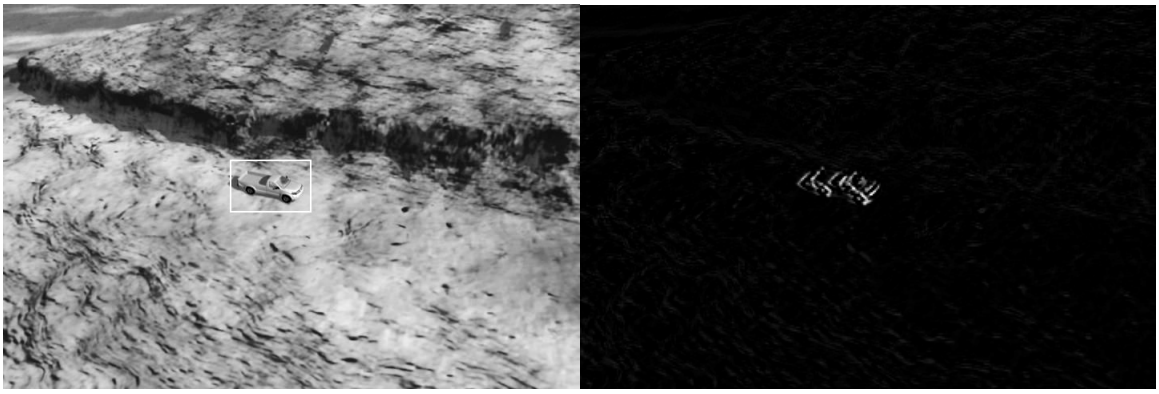
Due to the grouping of four input pixels for each pixel in the output, the horizontal and vertical optical flow did not match spatially. So, in order to combine them, the horizontal component had to be shifted by 3 pixels in the downward direction before it could be superimposed onto the vertical component. The median filter, connected component and image adjustment were used from the Matlab function for Image Processing toolbox. FFT function used in Fourier curvature signature is a built in Matlab function. The code for SVM was used with little modification from [14]. The snake program was used from [28, 29 , 30] also with little modification

4.3 Qualitative Results for Selected Synthetic Videos

Among all the synthetic sequences, the most challenging ones were the very far shots of moving person due to their very small size in pixels and the speed of the target. Some of these sequences contained cloud movements and the shadows of the clouds changed the illumination in the scene. However, our method was able to detect the

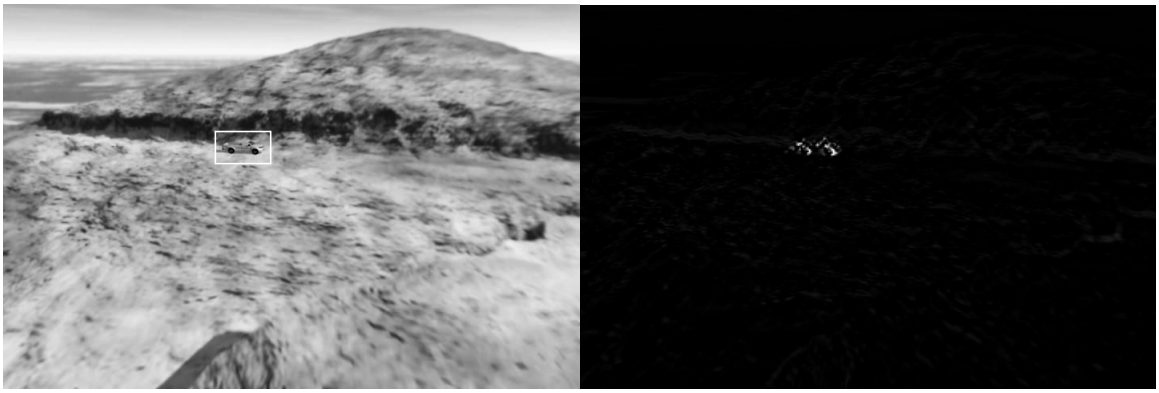
moving objects and was not affected by the moving clouds or their shadows. The smallest object in these sequences was a person in a far view scene with size 4x7 pixels.

The following figures illustrate the results from both methods using the synthetic video sequences by combining the vertical and horizontal outputs in one single frame.



a) b)

Figure 11: NearShotHC frame # 150: a) Original Image, b) BioOF

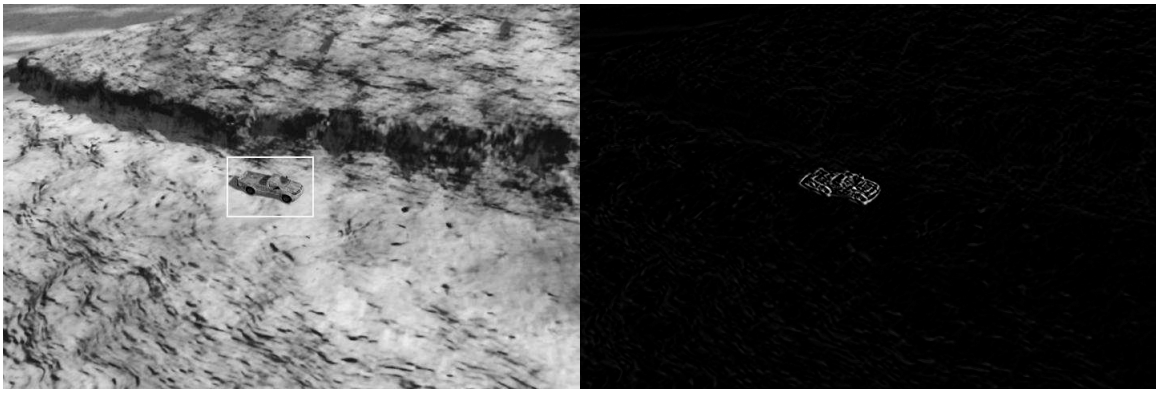


a) b)

Figure 12: MediumShotHC frame # 150: a) Original Image, b) BioOF

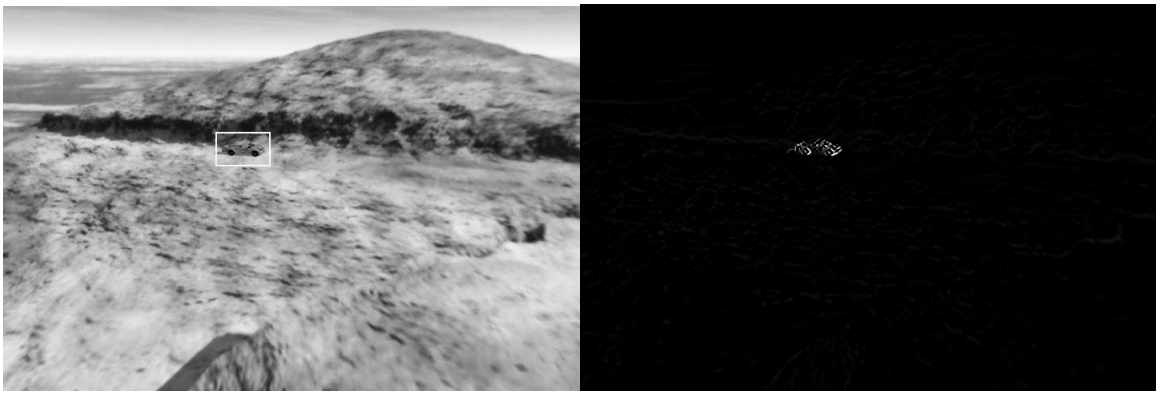
a) b)

Figure 13: LongShotHC frame # 150: a) Original Image, b) BioOF.



a) b)

Figure 14: NearShotCamo frame # 150: a) Original Image, b) BioOF

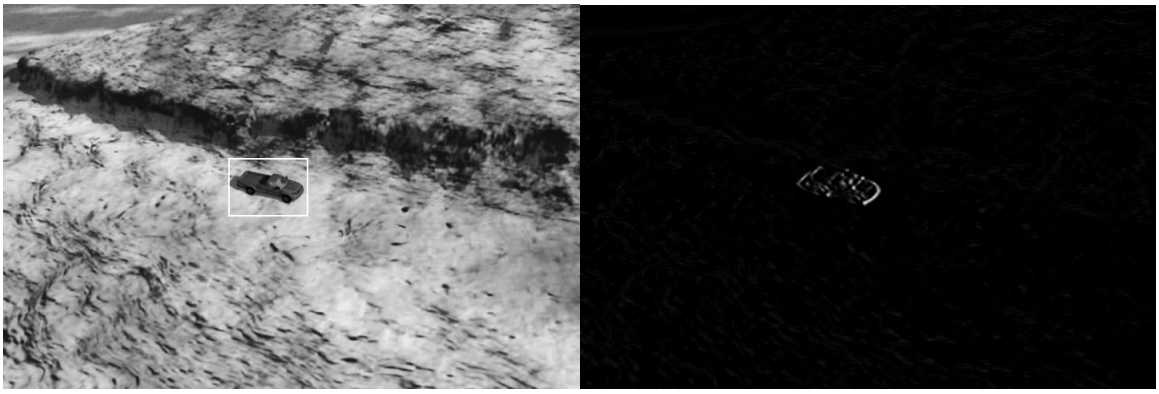


a) b)

Figure 15: MediumShotCamo frame # 150: a) Original Image, b) BioOF,

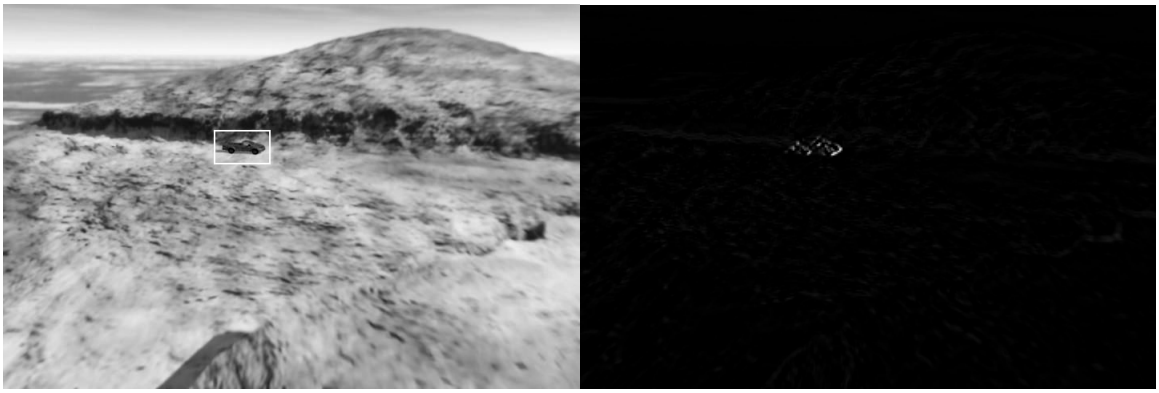
a) b)

Figure 16: LongShotCamo frame # 150: a) Original Image, b) BioOF



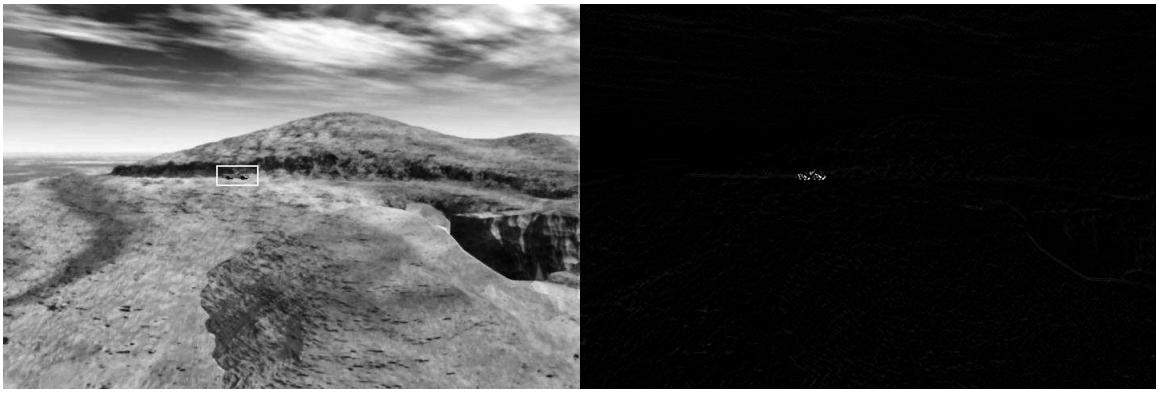
a) b)

Figure 17: NearShotLC frame # 150: a) Original Image, b) BioOF



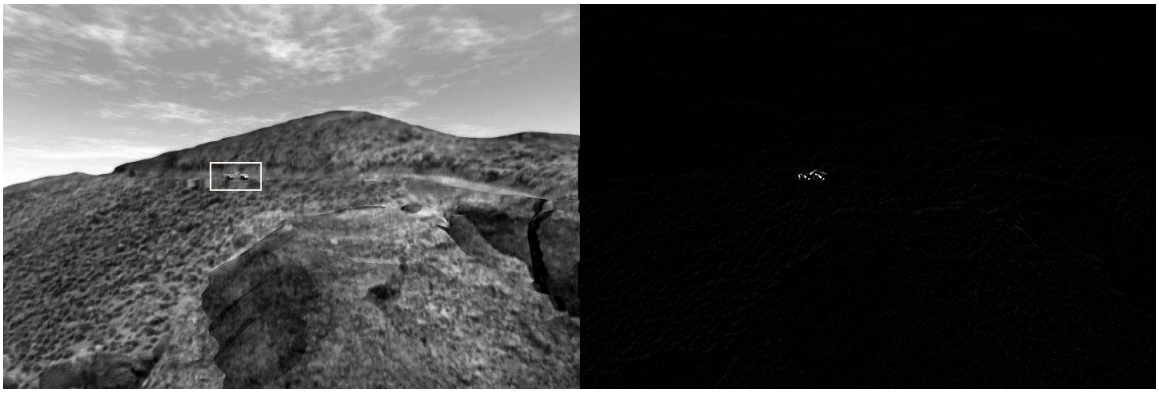
a) b)

Figure 18: MediumShotLC frame # 150: a) Original Image, b) BioOF.



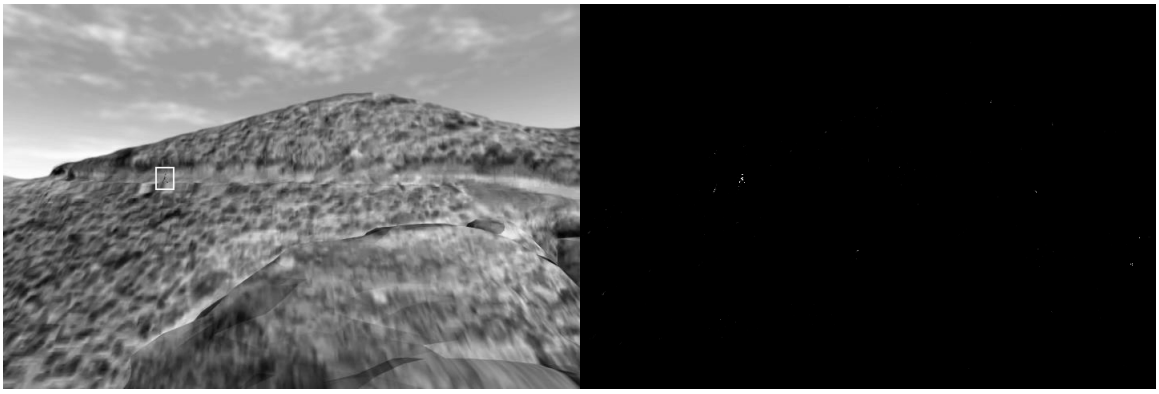
a) b)

Figure 19: LongShotLC frame # 150: a) Original Image, b) BioOF.



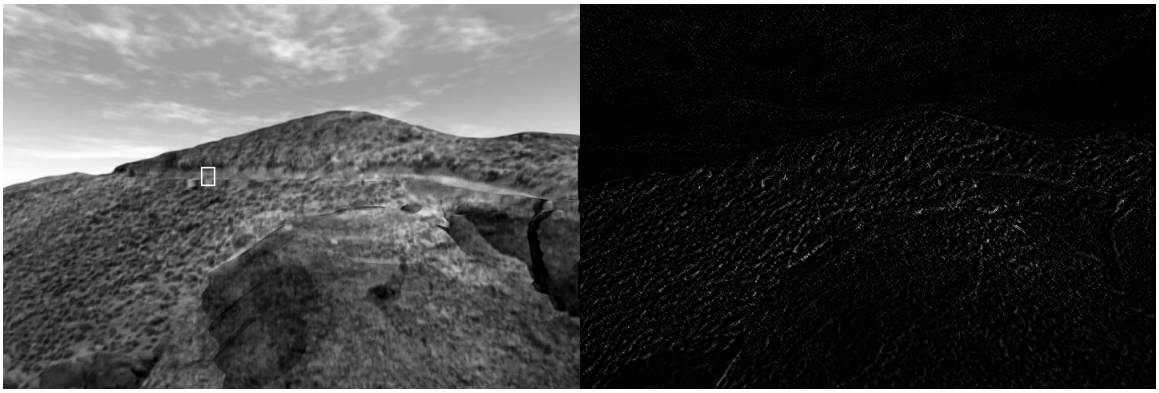
a) b)

Figure 20: LC_Truck_Clouds frame # 150: a) Original Image, b) BioOF.



a) b)

Figure 21: Person_Med frame # 150: a) Original Image, b) BioOF



a) b)

Figure 22: Person_Far_Cloud frame #200: a) Original Image, b) BioOF

a) b)

Figure 23: Long_Prsn_Cloud frame #200: a) Original, b) BioOF

a) b)

Figure 24: GroupDark_Far_Cloud frame #200: a) Original, b) BioOF

a) b)

Figure 25: Long_GroupD_Cloud frame #200: a) Original, b) BioOF

4.4 Qualitative Results for Selected Real Videos

The BioOF method detected moving objects very well in real videos except in the cases where the objects were very far, hence small in pixel size, moving very slowly and presented very low contrast against the background. In these cases, the BioOF results are usually: very faint; noisy; or missing altogether for a large number of frames.

One remedy that can improve the responses is to process the sequences at 15 fps. By doing so, the objects would appear as if in accelerated motion, and the detection would become much more pronounced.

a) b)

Figure 26: daytime_00_10_edit frame #100: a) Original, b) BioOF

a) b)

Figure 27: daytime_00_10_edit frame #560: a) Original, b) BioOF

a) b)

Figure 28: daytime_10_20_edit frame #100: a) Original, b) BioOF

a) b)

Figure 29: daytime_10_20_edit frame #470: a) Original, b) BioOF

a) b)

Figure 30: daytime_20_30_edit frame #12800: a) Original, b) BioOF

a) b)

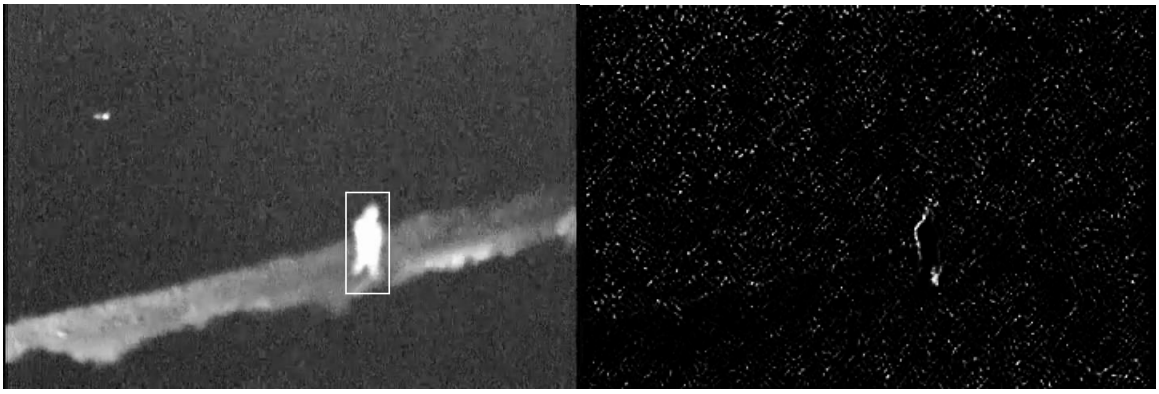
Figure 31: daytime_30_40_edit frame #3470: a) Original, b) BioOF

a) b)

Figure 32: daytime_50_60_edit frame #1000: a) Original, b) BioOF

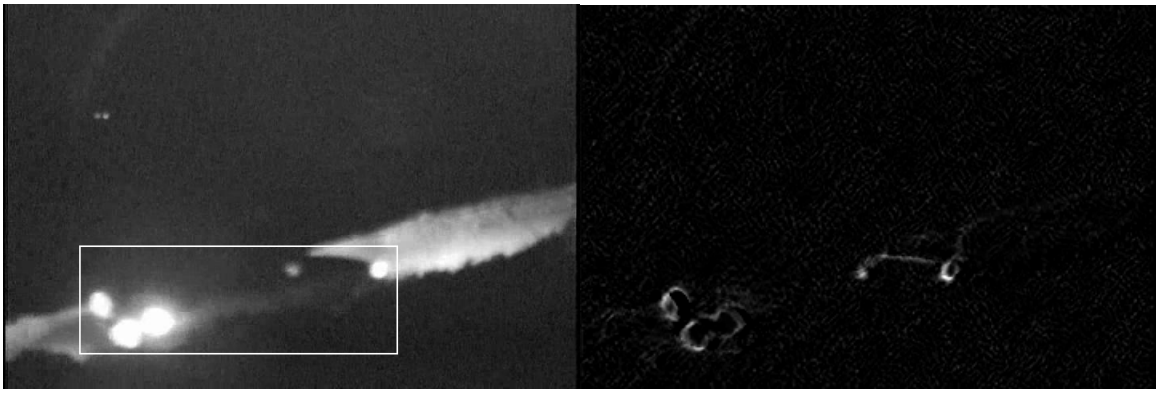
a) b)

Figure 33: daytime_50_60_edit frame #5960: a) Original, b) BioOF



a) b)

Figure 34: nighttime_00_10_edit frame #2772: a) Original, b) BioOF



a) b)

Figure 35: nighttime_50_60_edit frame #2035: a) Original, b) BioOF

a) b)

Figure 36: umc_test_seq frame #15: a) Original, b) BioOF

4.5 Quantitative Results for the Motion Detection Methods

In order to measure the performance of the motion detection, we manually created a *partial ground truth data* in the form of a bounding box around the target objects. The response bounding boxes of the motion detection algorithms were obtained from the binary mask. Next, the performance of each method was evaluated in terms of *Precision* and *Recall*. These metrics are defined as following:

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

where TP, FP and FN are the *true positive*, *false positive*, and *false negative*, respectively. In our case, TP is the overlap of the ground truth bounding box and the response bounding box derived from motion detection method; TP+FN is the area of the ground truth bounding box, TP+FP is the area of the response bounding box. That is:

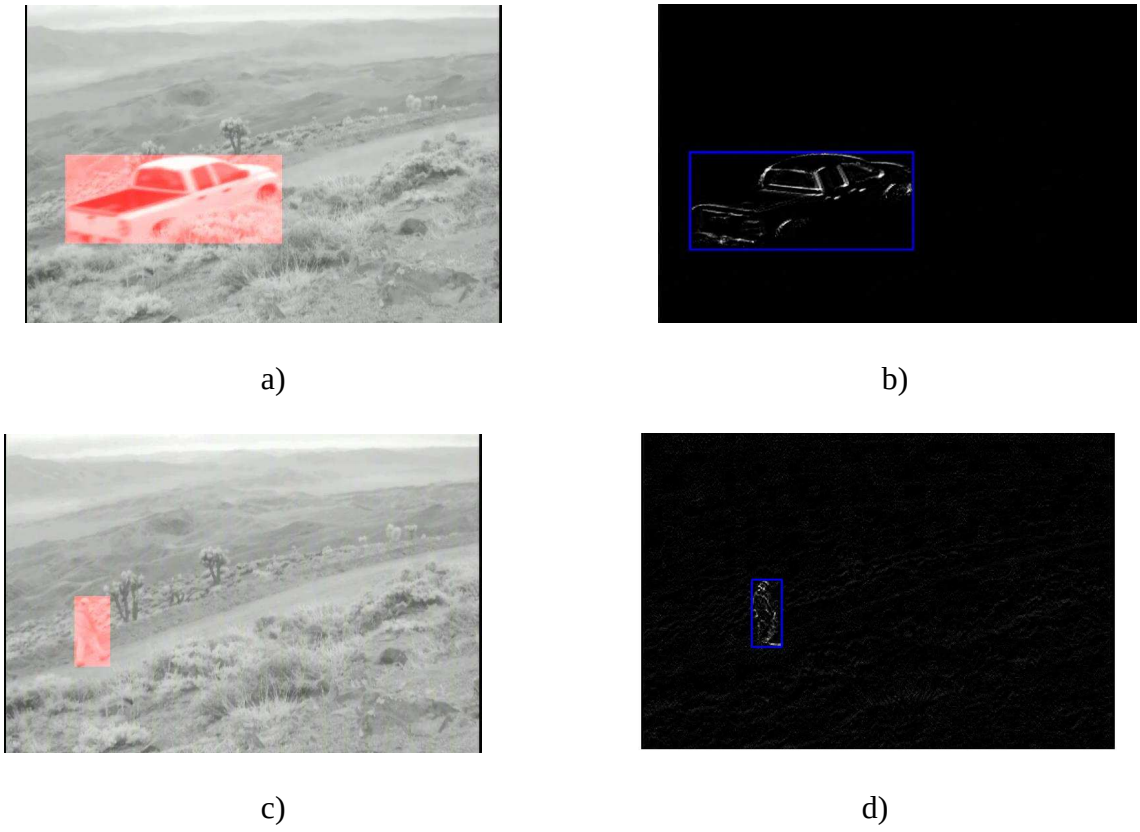
$$Recall = \frac{overlap}{area_{ground\_truth}} \qquad Precision = \frac{overlap}{area_{motion-delection}}$$

As can be inferred, Recall is a measure of the ability of the method to retrieve the ground truth. It is also known as the detection rate. Precision is a measure of the relevant portion of the detection with respect to all information obtained; in our case the overlapping area vis-à-vis the total area detected. False alarm rate (FAR) can be calculated as (*1-Precision*).

Neither of these metrics alone can provide a good picture of the performance of the method. For example , if one method consistently overestimates the bounding box in that case the Recall metric will be very high. However, for that same case, the Precision will suffer, since the FP will tend to be large

On the other hand, if a method detects just a small part of the target – which may not be a bad characteristic of the method for our application – its Recall index will be low, while the Precision will be very high – reflecting exactly the good characteristic of the method for this application.

We applied the above measurement to 370 images in the *daytime_slice_00_10_edit* sequence – 240 frames of a person walking from left to right, and 130 frames of a truck moving from left to right).

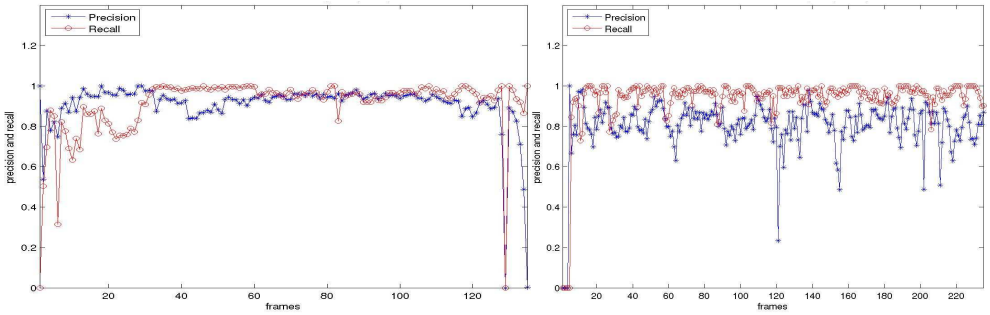


a) b)

c) d)

Figure 36: Samples from *daytime_00_10_edit* sequence. a) Original image with highlighted ground truth bounding box around the truck; b) BioOF with the response bounding box for the truck; c) Original image with highlighted ground truth bounding box around the person and d) BioOF with response bounding box for the person

Figure 36 shows the ground truth bounding boxes and the response bounding box of BioOF for an arbitrary image in the test data. Figure 37 presents the plots for the *Precision* and *Recall* metrics of the BioOF method for various frames of the truck and the person subsequences.



a) Truck b) Person

Figure 37: Precision and Recall for the BioOF

As seen in the figure above, Precesion of BioOF of the person near 150 sample drop to a low value this happens because around these samples person comes to almost stand still position & then he starts walking, motion of person becomes quite slow and that's why optical flow becomes weak & noise more predominant as a result detected bounding box of a person is far off to the ground truth bounding box. As expected, the BioOF method presented very high *Precision*, while maintaining an also quite high *Recall*. The low values of *Recall* at the beginning and end of the sequences correspond to the frames when the truck and the person were leaving the scene.

Table 4 compares the mean values of *Recall* and *Precision* obtained by each method.

Method	**Truck**		**Person**	
	Recall	Precision	Recall	Precision
BioOF	93.5%	89.9%	93.4%	80.2%

Table 4 – Mean values of *Recall* and *Precision*.

5.0 EXPERIMENTAL RESULTS OBJECT IDENTIFICATION

The object contour derived from the BioOF method was used for identification as described in section 2.3. That is, the contours from BioOF were processed via a sequence of noise filtering and the snake algorithm to extract the 1-pixel wide contours of the objects. After that, the snake contours were converted into Fourier Descriptors as also explained in section 2 and the different classification algorithms were applied.

A total of 40 images of the truck and 40 images of the human were used for training the various classification approaches and a separate set with 10 images of each object was used for testing. Appendix A presents all the images used in both Training and Testing sets.

The following table summarizes the percentage in correct classification using the various methods.

Method	Success Rate
SVM using Exponential Radial Basis Function	**65.00%**
SVM using Linear Function	**60.00%**
SVM using Polynomial function	**65.00%**
SVM using Annova	**65.00%**
Forward Feature Section and Mahanabolis Distance	**60.00%**
Principal Component Analysis and Mahanabolis Distance	**50.00%**
Scatter Matrices	**60.00%**

Table 5 – Percentage of combined (human and truck) corrected identification

As it can be observed from the table above, the success rate in all methods was somewhat low. The reason for such percentages is in the quality of the contours used in all methods. That is, close looks in the images from Appendix A (see Fig. 39) can demonstrate that the contours extracted – after the snake algorithm – do not preserve all the object features. Some of the most important object features, especially those that could provide a good discrimination – such the truck fender – are lost during the contour extraction.

Since all methods presented above rely on the Fourier Descriptors of the contours detected by the snake algorithm, if the contours are poorly extracted, the Fourier will fail to capture the same features over the training set, and consequently, the classification will suffer accordingly.

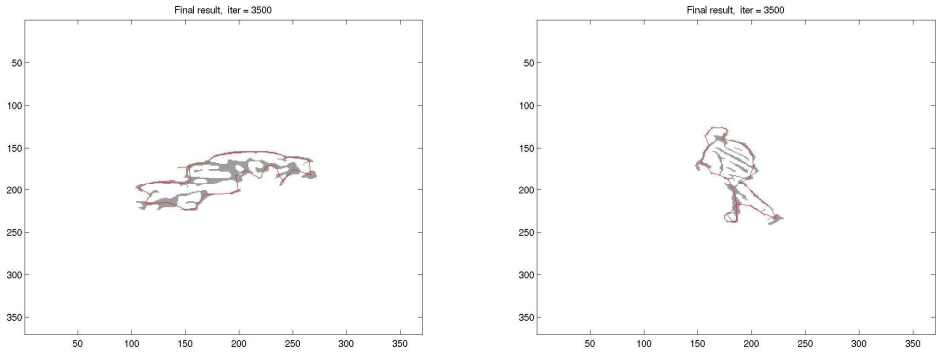


Figure 39 – Enlarged image from Apendix A. The images show the contours used in the Object Identification Module.

In future develops a better contour extraction algorithm is required, whether we use spatial features, temporal features or both in the identification module. One way in which we could improve not only the result from the snake algorithm, but also gain

performance – for instance, towards an integrated hardware implementation – would be by integrating the calculation of the GVF with the BioOF.

As we explained in section 2.3.1, the GVF filed is the argument *v* that minimizes the expression below.

$$\varepsilon = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |v - \nabla f|^2 \, dxdy$$

The first part of the integration is formed by the square of the first order derivatives of the pixel coordinates in the *x* and *y* direction. Such derivatives are present in the calculation of any Optical Flow method. Hence, by integrating the snake algorithm with the BioOF, those derivatives could be calculated over the original image, rather than over the edge map image – with broken edges.

6.0 FUTURE WORK

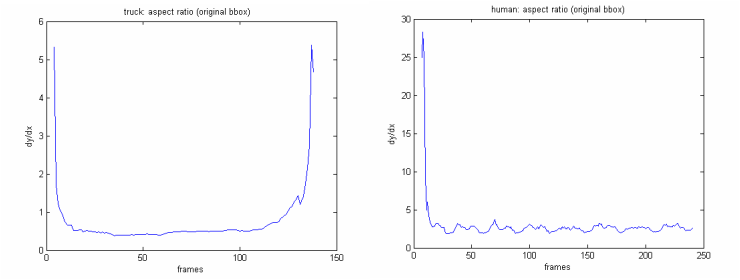
6.1 Discriminating Humans and Vehicles

Object classification based on motion properties receives a lot of attention due to the fact that temporal data can improve the performance of recognition and reduce the reliance on accurate spatial primitives [11, 19, 25, 6, 4, 23]. Motion based recognition can offer a more robust approach by incorporating the temporal variations in shape as a characteristic of certain types of objects. Especially in the case of discrimination between humans and vehicles, one key observation is that a vehicle exhibits rigid-body motion while a human exhibits articulated body motion. This can be used as a discriminating characteristic. The articulated motion that non-rigid natural objects (humans, animals, etc.) exhibit is in general periodic due to the usage of their limbs for locomotion. Hence detecting periodicity can serve as a discriminating feature between vehicles and humans. Another approach is to study the characteristics of local spatio-temporal neighborhoods in detected moving regions. Rigid bodies exhibit the same motion characteristics (such as velocities) in both spatial and temporal localities, whereas non-rigid bodies exhibit dissimilar motion characteristics. In the case of humans this is due to the distinct motion of limbs.

6.2 Periodicity of Human Motion

While features based on special features alone may not be reliable for classification, studying their temporal changes may provide a more robust feature for the same task. One such feature is the temporal change of aspect ratio. The periodic motion of a human manifests itself as a periodic change in the aspect ratio since movements of arms and legs change this feature from one frame to another. A simple approach is to find

the bounding box of the object and calculate its aspect ratio for each frame. Since the vehicle does not exhibit periodic motion, the temporal change of its aspect ratio will not contain any periodicity even though it may vary due to perspective distortion, rotation or poor detection.



a) b)

Figure 40: Change of aspect ratio for truck a) and human b).

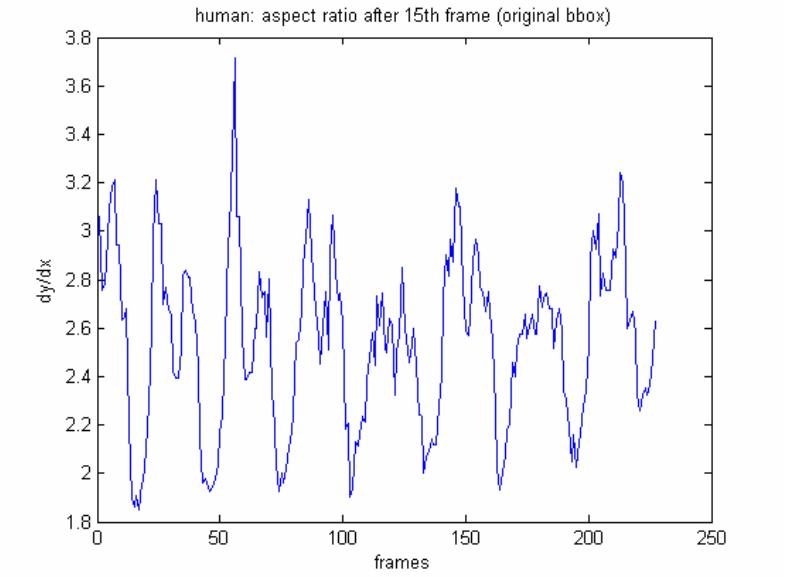


Figure 41: Change of aspect ratio for human after 15th frame.

Figure 40 depicts the changes in the aspect ratio for the ground truth bounding boxes described in section 3.1 for both human and truck. Since the first frames in the

sequence are not reliable due to only partial detection of the target, Figure 41 is a blown up view of Figure 40b).

As Figure 41 shows, the periodic character of the aspect ratio of humans is maintained throughout the sequence, whereas the aspect ratio of the vehicle does not exhibit any periodicity.

7.0 CONCLUSION

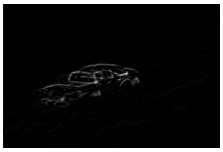
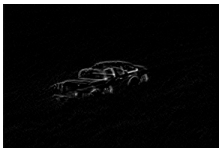
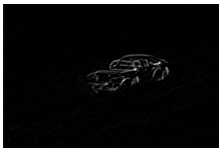
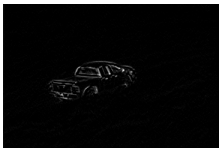
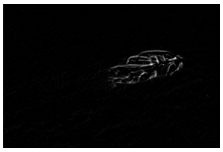
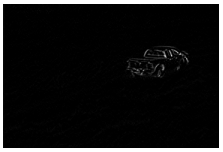
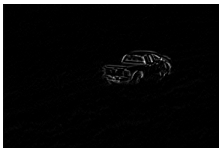
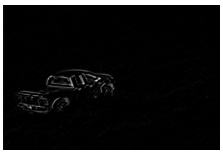
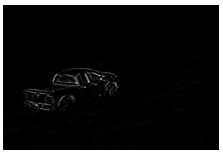
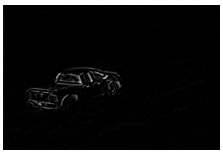
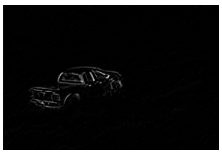
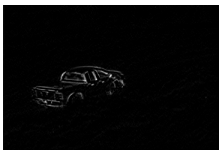
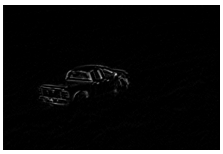
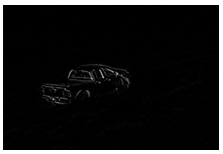
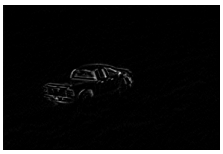
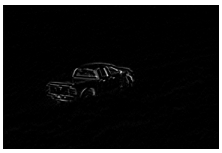
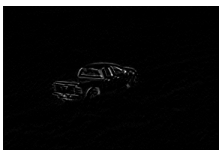
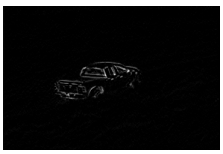
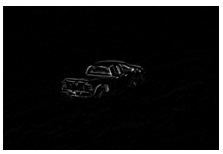
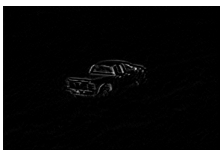
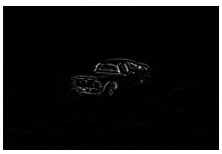
A Biologically inspired optical flow method for motion segmentation was presented. Such method is much faster than any existing machine vision optical flow algorithm and it is hardware convertible. The results are comparable to other existing optical flow algorithms.

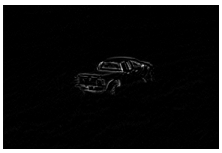
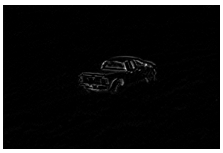
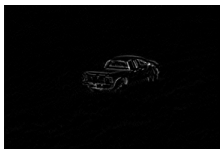
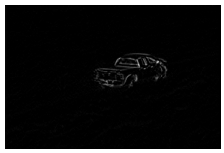
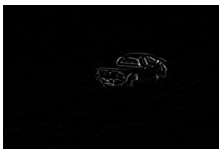
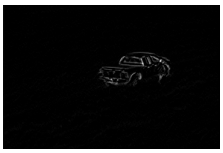
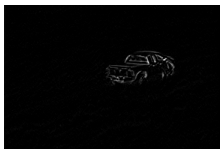
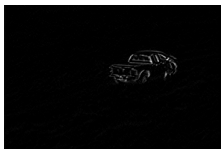
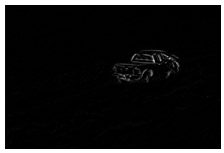
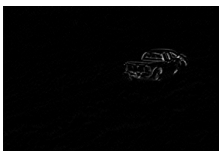
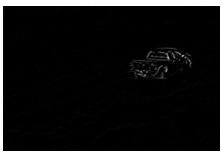
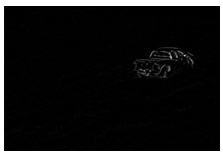
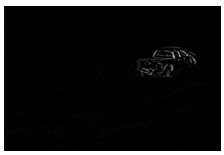
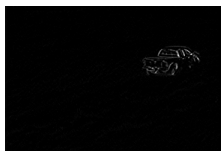
Various classification algorithms were investigated using a Fourier feature space. The classification algorithms were used to identify an object as human or vehicle. Despite the relatively good performance of the system, some limitations were found. Overcoming these limitations represents a challenge for future research in terms of, for example, obtaining better contours from the BioOF method by improving the output of the snake algorithm. Also, in terms of classification, exploiting temporal features seems very promising and one possible direction is to calculate the correlations for a set of frames and represent the dynamics of the underlying motion in a multi-dimensional space and to search for separable features in that space.

APPENDIX

Appendix A: TRAINING AND TESTING SETS FOR CLASSIFICATION

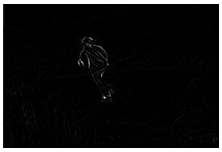
Truck Training Set Output from BioOF

 1	 2	 3	 4	 5
 6	 7	 8	 9	 10
 11	 12	 13	 14	 15
 16	 17	 18	 19	 20
 21	 22	 23	 24	 25

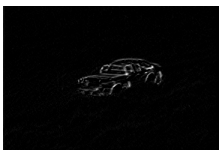
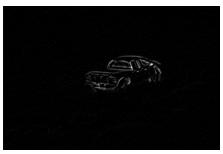
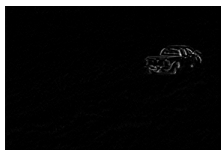
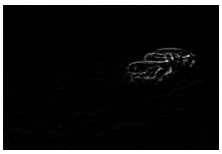
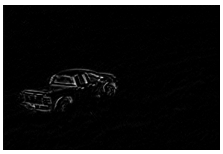
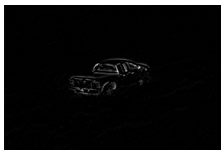
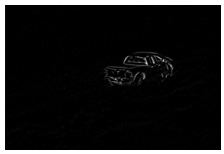
 26	 27	 28	 29	 30
 31	 32	 33	 34	 35
 36	 37	 38	 39	 40

Human Training Set – Output from BioOF

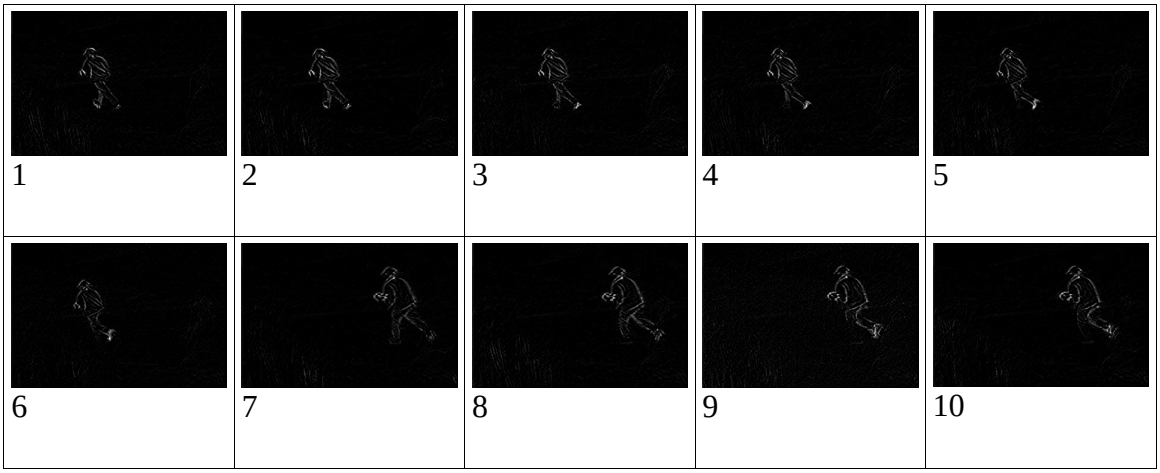
 1	 2	 3	 4	 5
 6	 7	 8	 9	 10
 11	 12	 13	 14	 15
 16	 17	 18	 19	 20

 21	 22	 23	 24	 25
 26	 27	 28	 29	 30
 31	 32	 33	 34	 35
 36	 37	 38	 39	 40

Truck Testing Set – Output from BioOF

 1	 2	 3	 4	 5
 6	 7	 8	 9	 10

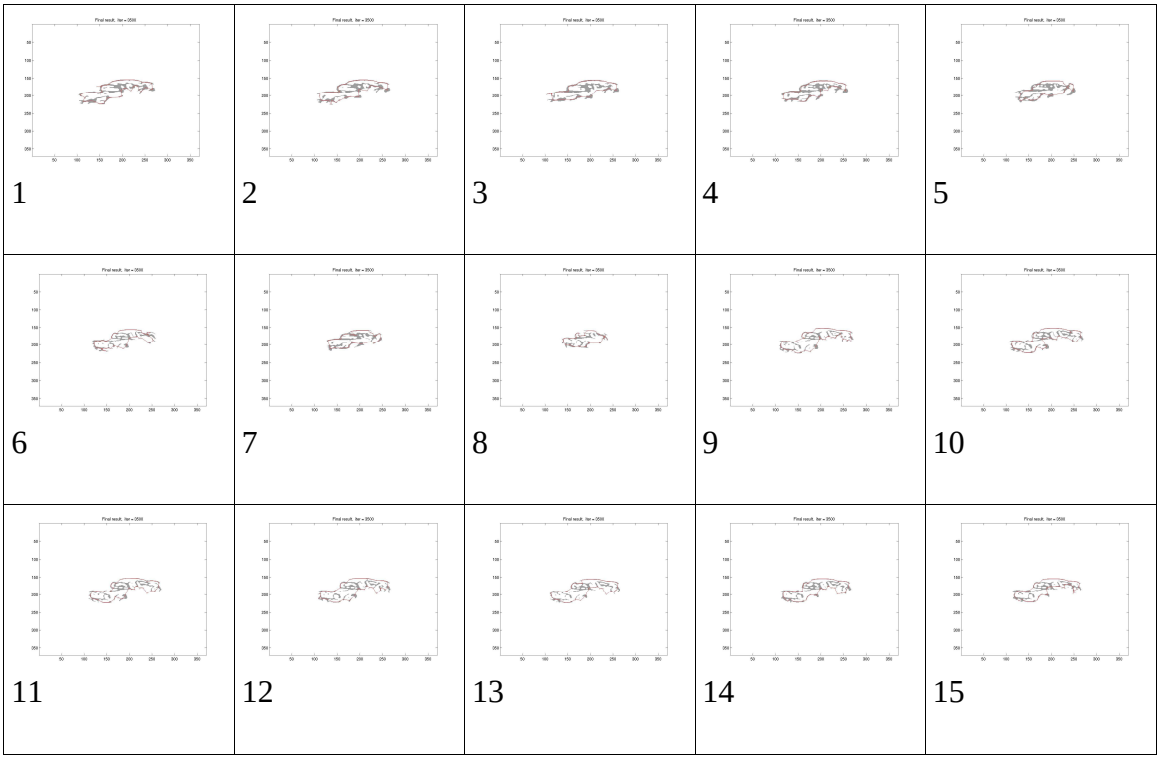
Human Testing Set – Output from BioOF



1 2 3 4 5

6 7 8 9 10

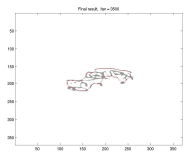
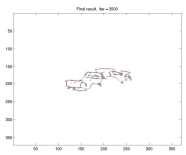
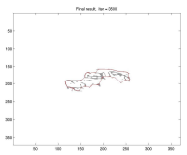
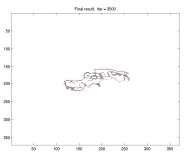
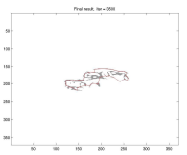
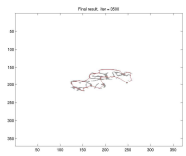
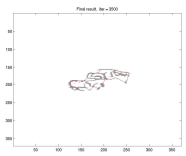
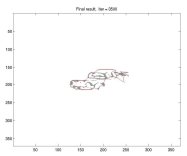
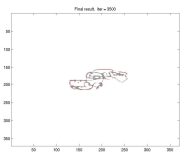
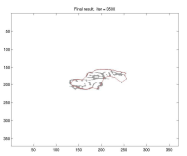
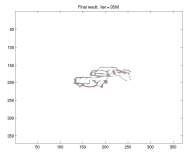
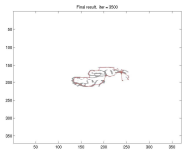
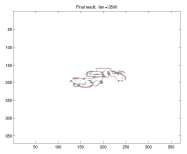
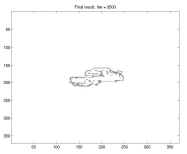
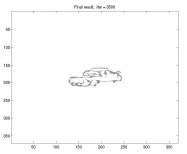
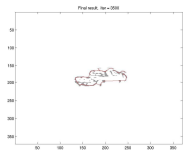
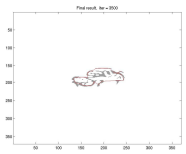
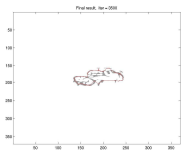
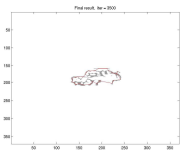
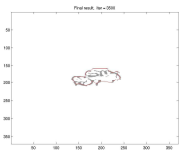
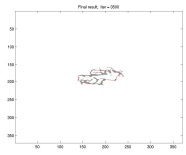
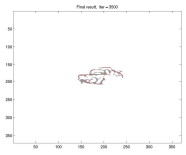
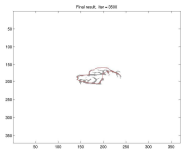
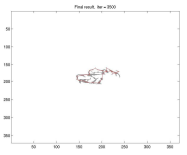
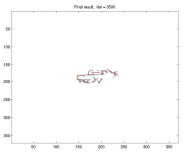
Truck Training Set – Output from the Snake Algorithm



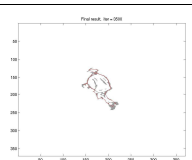
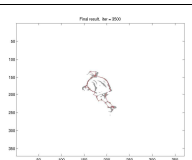
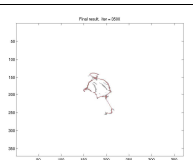
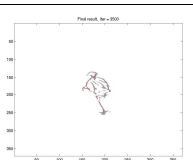
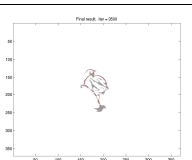
1 2 3 4 5

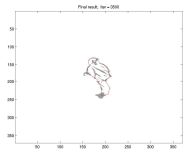
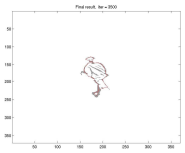
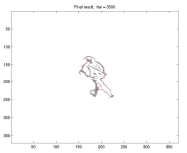
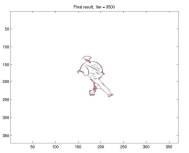
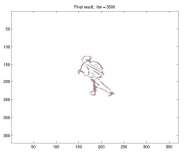
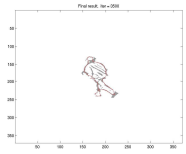
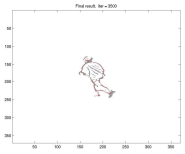
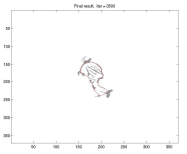
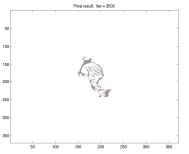
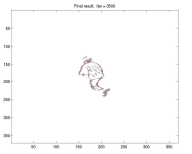
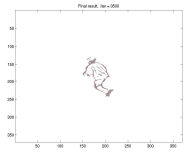
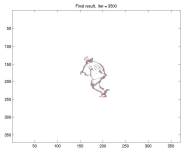
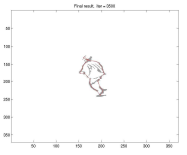
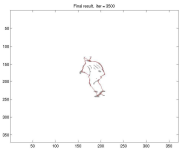
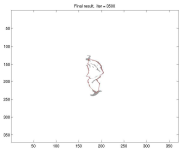
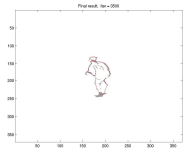
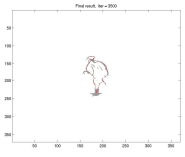
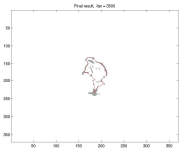
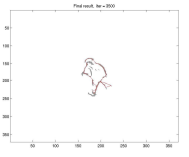
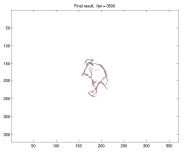
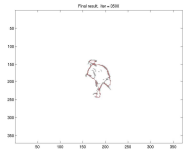
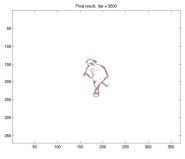
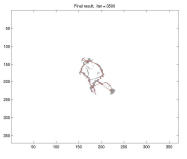
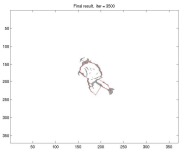
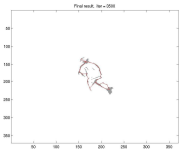
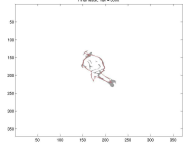
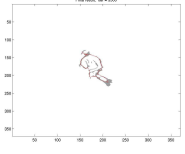
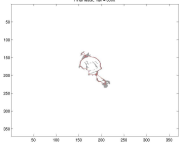
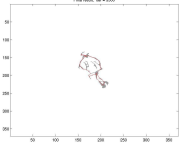
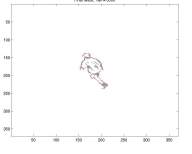
6 7 8 9 10

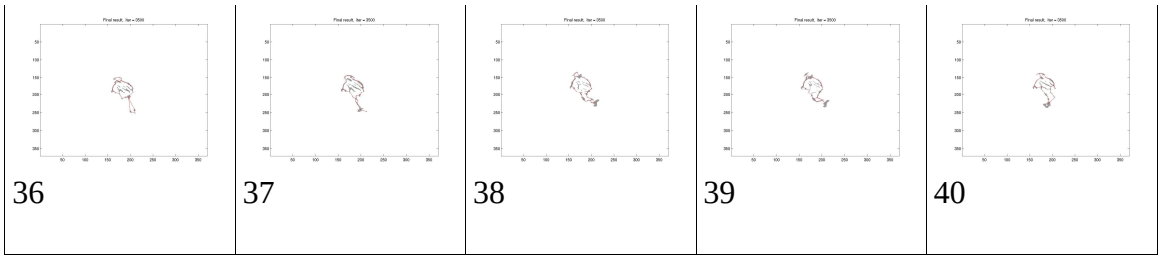
11 12 13 14 15

 16	 17	 18	 19	 20
 21	 22	 23	 24	 25
 26	 27	 28	 29	 30
 31	 32	 33	 34	 35
 36	 37	 38	 39	 40

Human Training Set – Output from the Snake Algorithm

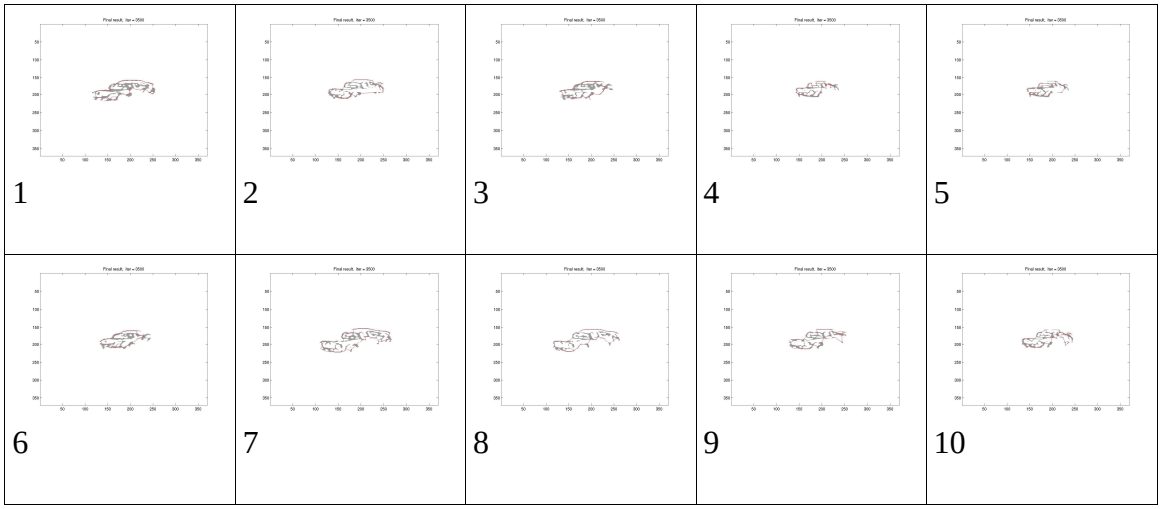
 ***1***	 ***2***	 ***3***	 ***4***	 ***5***

 6	 7	 8	 9	 10
 11	 12	 13	 14	 15
 16	 17	 18	 19	 20
 21	 22	 23	 24	 25
 26	 27	 28	 29	 30
 ***31***	 ***32***	 ***33***	 ***34***	 ***35***



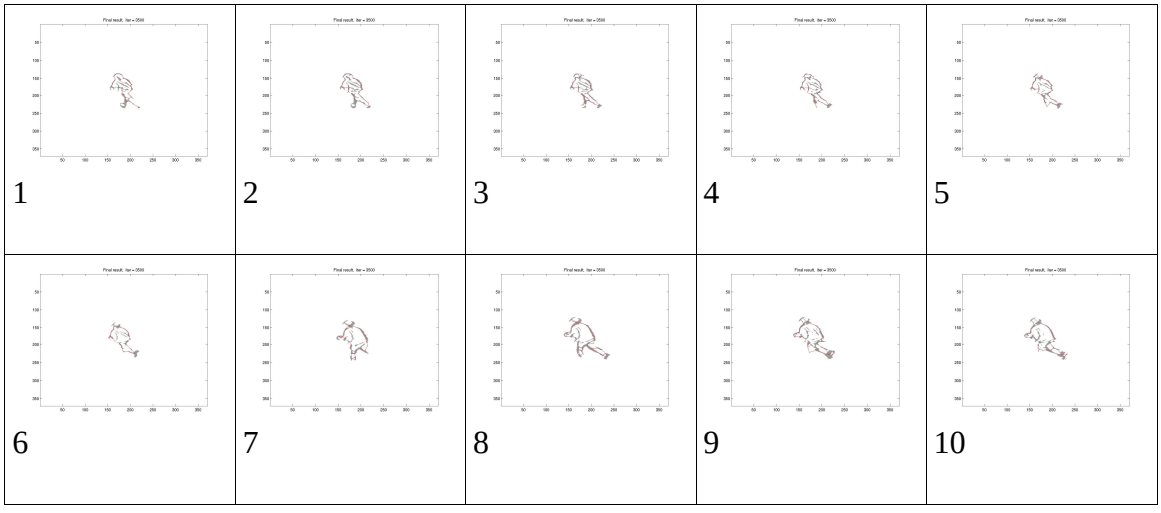
36	37	38	39	40

Truck Testing Set – Output from the Snake Algorithm



1	2	3	4	5
6	7	8	9	10

Human Testing Set – Output from the Snake Algorithm



1	2	3	4	5
6	7	8	9	10

Appendix B: MATLAB CODE

The following code creates sequence of image of a block in specified direction at specified speed, Initaily testing of BioOF was done on these images & BioOF code was optimized

```
function data = create_images(image_size, block_pos, block_speed,
n_frames)

block_size = [block_pos(3)-block_pos(1)+1, block_pos(4)-
block_pos(2)+1];
%block = rand(block_size(1),block_size(2))*255;
block = ones(block_size(1),block_size(2))*255;
block_p0 = [block_pos(1) block_pos(2)];

data = zeros(image_size(1), image_size(2), n_frames);

for t = 0:n_frames-1
   indx = t*block_speed + block_p0;
   ib = indx(1);
   ie = indx(1)+block_size(1)-1;
   jb = indx(2);
   je = indx(2)+block_size(2)-1;
   data(ib:ie,jb:je,t+1) = block;
end

clear;

n_frames = 30;
bg_size = [240, 320];
fg_size = [15, 15];
img_size = [120, 160];

%background = floor(rand(bg_size(1),bg_size(2))*255);
%background = floor(ones(bg_size(1),bg_size(2))*100);
background = floor(zeros(bg_size(1),bg_size(2)));

foreground1 = floor(rand(fg_size(1),fg_size(2))*255);
foreground2 = floor(rand(fg_size(1),fg_size(2))*255);
%foreground1 = floor(ones(fg_size(1),fg_size(2))*200);
%foreground2 = floor(ones(fg_size(1),fg_size(2))*50);

bg_speed = [1, -1];
fg1_speed = [1, 2];
fg2_speed = [1, -3];

bg_p0 = (bg_size - img_size)/2;
fg1_p0 = [5, 20];
fg2_p0 = [40, 130];
```

```
colormap([[0:1/256:1]' [0:1/256:1]' [0:1/256:1]']);

for t = 0:n_frames-1
   indx = t*bg_speed + bg_p0;
   img = background(indx(1):indx(1)+img_size(1)-
1,indx(2):indx(2)+img_size(2)-1);

   indx = t*(fg1_speed+bg_speed) + fg1_p0;
   img(indx(1):indx(1)+fg_size(1)-1,indx(2):indx(2)+fg_size(2)-1) =
foreground1;

   indx = t*(fg2_speed+bg_speed) + fg2_p0;
   img(indx(1):indx(1)+fg_size(1)-1,indx(2):indx(2)+fg_size(2)-1) =
foreground2;

   image(img);
   M(t+1) = getframe;
   write_pgm(['Blocks/test',num2str(t),'.pgm'], img);
end

%movie(M);
%movie2avi(M, 'demo.avi');
```

The following code creates elementary motion detector for one input data sequence, Input sequence was synthetic data created to test & modify basic working of all filters & basic elementary motion detector

```
function [T5D, T5U] = emd1D(in,tau1,tau2,sr)

[n_in, n_samples] = size(in);

Am = rlp(rrp(in,tau1,sr,0,0),tau1,sr,0,0);
L2 = rhp(in,tau1,sr,0,0);

Am(n_in+1,:) = Am(n_in,:);  % create padding at the bottom
Am(n_in+2,:) = Am(n_in,:);
Am(n_in+3,:) = Am(n_in,:);
L2(n_in+1,:) = L2(n_in,:);
L2(n_in+2,:) = L2(n_in,:);

T1_D = Am(1:n_in,:) + Am(3:n_in+2,:);
T1_U = Am(2:n_in+1,:) + Am(4:n_in+3,:);

Tm1_D = T1_D + L2(2:n_in+1,:);
Tm1_U = T1_U + L2(3:n_in+2,:);

Tm9_D = rlp(Tm1_D,tau2,sr,0,0);
Tm9_U = rlp(Tm1_U,tau2,sr,0,0);

Ie_D = Tm1_D .* (Tm1_D>0); % "Ie_D = pos(Tm1_D)"
Ie_U = Tm1_U .* (Tm1_U>0); % "Ie_U = pos(Tm1_U)"
```

```
Is_D = Tm9_U .* (Tm9_U>0); % "Ii_D = pos(Tm9_U)"
Is_U = Tm9_D .* (Tm9_D>0); % "Ii_U = pos(Tm9_D)"

T5_D = Ie_D .* (1 - (Is_D ./ (max(Is_D,[],2)*ones(1,n_samples))));
T5_U = Ie_U .* (1 - (Is_U ./ (max(Is_U,[],2)*ones(1,n_samples))));

T5D = 0.5*(T5_D-T5_U);
T5U = 0.5*(T5_U-T5_D);
```

The following code creates elementary motion detector for two input data sequence, Input sequence was synthetic data created to test & modify basic working of all filters & basic elementary motion detector

```
function [y1p y2p] = emd1_2inputs(in1,in2,tau1,tau2,sr)

ht1 = trp(in1,tau1,sr);
ht2 = thp(in1,tau1,sr);
%ht3 = ht1;
ht4 = trp(in2,tau1,sr);
ht5 = thp(in2,tau1,sr);
%ht6 = ht4;

ht7 = tlp(ht1,tau1,sr);
%ht8 = ht7;
ht9 = tlp(ht4,tau1,sr);
%ht10 = ht9;

ht11 = ht7 + ht9;
%ht12 = ht11;

ht13 = ht11 + ht2;
ht14 = ht11 + ht5;

ht15 = tlp(ht14,tau2,sr);
ht16 = tlp(ht13,tau2,sr);

%**********In the following step dirty multiplication is performed
hte1 = ht13 .* (ht13>0); %pos(ht13);
hte2 = ht14 .* (ht14>0); %pos(ht14);
hti1 = ht15 .* (ht15>0); %pos(ht15);
hti2 = ht16 .* (ht16>0); %pos(ht16);
himax1 = max(hti1);
himax2 = max(hti2);
% if himax1 == 0
%     himax1 = 1;
% end;
y1 = hte1 .* (1 - hti1/himax1);
% if himax2 == 0
%     himax2 = 1;
% end;
```

```
y2 = hte2 .* (1 - hti2/himax2);
%******************here ends dirty multiplication******
y1p = 0.5*(y1-y2);
y2p = 0.5*(y2-y1);
%y = y1p-y2p;
```

The following code creates elementary motion detector for four input data sequence, Input sequence was synthetic data created to test & modify basic working of all filters & basic elementary motion detector

```
function [y1p y2p] = emd1_4inputs(in1,in2,in3,in4,tau1,tau2,sr)

ht1 = rrp(in1,tau1,sr,0,0);
ht2 = rhp(in2,tau1,sr,0,0);
ht3 = rrp(in2,tau1,sr,0,0);
ht4 = rrp(in3,tau1,sr,0,0);
ht5 = rhp(in3,tau1,sr,0,0);
ht6 = rrp(in4,tau1,sr,0,0);

ht7 = rlp(ht1,tau1,sr,0,0);
ht8 = rlp(ht3,tau1,sr,0,0);
ht9 = rlp(ht4,tau1,sr,0,0);
ht10= rlp(ht6,tau1,sr,0,0);

ht11 = ht7 + ht9;
ht12 = ht8 + ht10;

ht13 = ht11 + ht2;
ht14 = ht12 + ht5;

ht15 = rlp(ht14,tau2,sr,0,0);
ht16 = rlp(ht13,tau2,sr,0,0);

%**********In the following step dirty multiplication is performed
hte1 = ht13 .* (ht13>0);
hte2 = ht14 .* (ht14>0);
hti1 = ht15 .* (ht15>0);
hti2 = ht16 .* (ht16>0);

himax1 = max(hti1);
if himax1 == 0
    himax1 = 1;
end;
y1 = hte1 .* (1 - hti1/himax1);

himax2 = max(hti2);
if himax2 == 0
    himax2 = 1;
end;
y2 = hte2 .* (1 - hti2/himax2);
```

```
%******************here ends dirty multiplication******
y1p = 0.5*(y1-y2);
y2p = 0.5*(y2-y1);
%y = y1p-y2p;
```

The following code creates elementary motion detector for one input data sequence for recursive filters,

```
function [T5D, T5U] = emd2D(in,tau1,tau2,sr,y,in0)

%%%%%%%    Process along the Vertical direction

if (exist('y','var') == 0)
   y = zeros(size(in,1),size(in,2),1);
   in0 = zeros(size(in,1),size(in,2),1);
end
Am = rlp(rrp(in,tau1,sr,y,in0),tau1,sr,y,in0);
L2 = rhp(in,tau1,sr,y,in0);

[n_rw, n_cl, n_samples] = size(Am);

Am(n_rw+1,:,:) = Am(n_rw,:,:);  % create padding at the bottom
Am(n_rw+2,:,:) = Am(n_rw,:,:);
Am(n_rw+3,:,:) = Am(n_rw,:,:);
L2(n_rw+1,:,:) = L2(n_rw,:,:);
L2(n_rw+2,:,:) = L2(n_rw,:,:);

T1_D = Am(1:n_rw,:,:) + Am(3:n_rw+2,:,:);
T1_U = Am(2:n_rw+1,:,:) + Am(4:n_rw+3,:,:);

Tm1_D = T1_D + L2(2:n_rw+1,:,:);
Tm1_U = T1_U + L2(3:n_rw+2,:,:);

Tm9_D = rlp(Tm1_D,tau2,sr,y,in0);
Tm9_U = rlp(Tm1_U,tau2,sr,y,in0);

Ie_D = Tm1_D .* (Tm1_D>0);
Ie_U = Tm1_U .* (Tm1_U>0);
Is_D = Tm9_U .* (Tm9_U>0);
Is_U = Tm9_D .* (Tm9_D>0);

max_Is_D = reshape(max(Is_D,[],3),n_rw*n_cl,1)*ones(1,n_samples);
max_Is_U = reshape(max(Is_U,[],3),n_rw*n_cl,1)*ones(1,n_samples);
max_Is_D = reshape(max_Is_D,n_rw,n_cl,n_samples);
max_Is_U = reshape(max_Is_U,n_rw,n_cl,n_samples);

T5_D = Ie_D .* (1 - (Is_D ./ max_Is_D));
T5_U = Ie_U .* (1 - (Is_U ./ max_Is_U));

T5D = 0.5*(T5_D-T5_U); T5D(isnan(T5D)) = 0;
T5U = 0.5*(T5_U-T5_D); T5U(isnan(T5U)) = 0;

function [T5D, T5U, T5R, T5L] = emdC2D(in,tau1,tau2,sr,y,in0)
```

```
%%%%%%%    Process along the Vertical direction

if (exist('y','var') == 0)
   y = zeros(size(in,1),size(in,2),1);
   in0 = zeros(size(in,1),size(in,2),1);
end
Am = rlp(rrp(in,tau1,sr,y,in0),tau1,sr,y,in0);
L2 = rhp(in,tau1,sr,y,in0);

[n_rw, n_cl, n_samples] = size(Am);

Am(n_rw+1,:,:) = Am(n_rw,:,:);  % create padding at the bottom
Am(n_rw+2,:,:) = Am(n_rw,:,:);
Am(n_rw+3,:,:) = Am(n_rw,:,:);
L2(n_rw+1,:,:) = L2(n_rw,:,:);
L2(n_rw+2,:,:) = L2(n_rw,:,:);

T1_D = Am(1:n_rw,:,:) + Am(3:n_rw+2,:,:);
T1_U = Am(2:n_rw+1,:,:) + Am(4:n_rw+3,:,:);

Tm1_D = T1_D + L2(2:n_rw+1,:,:);
Tm1_U = T1_U + L2(3:n_rw+2,:,:);

clear T1_D T1_U; pack;

Tm9_D = rlp(Tm1_D,tau2,sr,y,in0);
Tm9_U = rlp(Tm1_U,tau2,sr,y,in0);

Ie_D = Tm1_D .* (Tm1_D>0);
Ie_U = Tm1_U .* (Tm1_U>0);
Is_D = Tm9_U .* (Tm9_U>0);
Is_U = Tm9_D .* (Tm9_D>0);

clear Tm1_D Tm1_U Tm9_D Tm9_U; pack;

max_Is_D = reshape(max(Is_D,[],3),n_rw*n_cl,1)*ones(1,n_samples);
max_Is_U = reshape(max(Is_U,[],3),n_rw*n_cl,1)*ones(1,n_samples);
max_Is_D = reshape(max_Is_D,n_rw,n_cl,n_samples);
max_Is_U = reshape(max_Is_U,n_rw,n_cl,n_samples);

max_Is_D(max_Is_D==0) = 1;
max_Is_U(max_Is_U==0) = 1;

T5_D = Ie_D .* (1 - (Is_D ./ max_Is_D));
T5_U = Ie_U .* (1 - (Is_U ./ max_Is_U));

clear Ie_D Ie_U Is_D Is_U max_Is_D max_Is_U; pack;

T5D = 0.5*(T5_D-T5_U);
T5U = 0.5*(T5_U-T5_D);

clear T5_D T5_U; pack;
```

```
%%%%%%%    Process along the Horizontal direction

Am(:,n_cl+1,:) = Am(:,n_cl,:);  % create zero padding at the right
Am(:,n_cl+2,:) = Am(:,n_cl,:);
Am(:,n_cl+3,:) = Am(:,n_cl,:);
L2(:,n_cl+1,:) = L2(:,n_cl,:);
L2(:,n_cl+2,:) = L2(:,n_cl,:);

T1_R = Am(1:n_rw,1:n_cl,:) + Am(1:n_rw,3:n_cl+2,:);
T1_L = Am(1:n_rw,2:n_cl+1,:) + Am(1:n_rw,4:n_cl+3,:);

Tm1_R = T1_R + L2(1:n_rw,2:n_cl+1,:);
Tm1_L = T1_L + L2(1:n_rw,3:n_cl+2,:);

clear Am L2 T1_R T1_L; pack;

Tm9_R = rlp(Tm1_R,tau2,sr,y,in0);
Tm9_L = rlp(Tm1_L,tau2,sr,y,in0);

Ie_R = Tm1_R .* (Tm1_R>0);
Ie_L = Tm1_L .* (Tm1_L>0);
Is_R = Tm9_L .* (Tm9_L>0);
Is_L = Tm9_R .* (Tm9_R>0);

clear Tm1_R Tm1_L Tm9_R Tm9_L; pack;

max_Is_R = reshape(max(Is_R,[],3),n_rw*n_cl,1)*ones(1,n_samples);
max_Is_L = reshape(max(Is_L,[],3),n_rw*n_cl,1)*ones(1,n_samples);
max_Is_R = reshape(max_Is_R,n_rw,n_cl,n_samples);
max_Is_L = reshape(max_Is_L,n_rw,n_cl,n_samples);

max_Is_R(max_Is_R==0) = 1;
max_Is_L(max_Is_L==0) = 1;

T5_R = Ie_R .* (1 - (Is_R ./ max_Is_R));
T5_L = Ie_L .* (1 - (Is_L ./ max_Is_L));

clear Ie_R Ie_L Is_R Is_L max_Is_R max_Is_L; pack;

T5R = 0.5*(T5_R-T5_L);
T5L = 0.5*(T5_L-T5_R);

clear T5_R T5_L; pack;
```

The following program assigns various values to logic 0 & logic 1 & test which one is better

```
function [out zero one] = logic(ina, v)
```

```
if (max(max(max(ina))) ~= 0)
   out = ina/max(max(max(ina)));
else
   out = ina;
end

if (v == 0)  %   0/1 logic
   zero = 0;
   one  = 1;
   return;
end

if (v == 1)  %  -1/1 logic
   out = out*2 - 1;
   zero = -1;
   one  =  1;
   return;
end

if (v == 2)  % -3/-2 logic
   out = out - 3;
   zero = -3;
   one  = -2;
   return;
end

if (v == 3)  % -2/-3 logic
   out = -2 - out;
   zero = -2;
   one  = -3;
   return;
end

if (v == 4)  %  -2/2 logic
   out = out*4 - 2;
   zero = -2;
   one  =  2;
   return;
end

if (v == 5)  %   -4/-2 logic
   out = out*2 - 4;
   zero = -4;
   one  = -2;
   return;
end
```

The function given below converts a group of images into avi file

```
function make_movie(root_filename, set_begin, number_images, file_type,
display_flag)
%
%   Copyright (C) 2006 ViGIR - Vision Guided and Intelligent Robotics
(http://vigir.missouri.edu)
```

```
%   Written by Guilherme N. DeSouza & Vishal Rijhwani
%
%   This program is free software; you can redistribute it and/or
modify
%   it under the terms of the GNU General Public License as published
by
%   the Free Software Foundation, meaning:
%                       keep this copyright notice,
%                       do not try to make money out of it,
%                       it's distributed WITHOUT ANY WARRANTY,
%                       yada yada yada...
%
%
%   Routine to convert a sequence of images into a .avi movie
%
%     Usage:  make_movie(root_filename, number_images, file_type,
display_flag)
%
%      Where:
%             number_images is the number of images saved as
"root_filename"%i."file_type"
%             and output movie file will be root_filename.avi
%
%

if (exist('display_flag') == 0)
   display_flag = 0;
end;

if (display_flag == 0)
      disp('Image Display Disabled')
else
      disp('Displaying image')
end;

filename = sprintf([root_filename,'%06d%s'], set_begin, file_type);
out_mtrx = imread(filename);
frame = im2frame(out_mtrx);
M(number_images) = frame;

for count_images=set_begin:(set_begin+number_images-1);
   filename = sprintf([root_filename,'%06d%s'], count_images,
file_type);
   disp(sprintf('Processing file: %s', filename));
   out_mtrx = imread(filename);
   if (display_flag ~= 0)
      figure(1);
      image(out_mtrx);
   end;
   frame = im2frame(out_mtrx);
   M(count_images-set_begin+1) = frame;
end;
%movie(M);
movie2avi(M,sprintf('%s.avi',root_filename));
```

The following program reads an image sequence and covert it it to image sequence of desired dimension & crop it also as specified Initially BioOF use to take long time, to reduce the time required for implementing BioOF the images were cropped & reduced in dimension to make the processing fast.

```
function data = read_video(filename, bg, n_frames, extension)

if (exist('extension') == 0)
   extension = '.pgm';
end

if (extension == '.pgm')
%newly added section
    lz = 5;
    name_in = sprintf(['%s%0' num2str(lz) 'd%s'],filename,bg,'.pgm');
    name_cv = '/tmp/temp.pgm';
    system(sprintf('convert %s %s',name_in,name_cv));
    data(:,:,n_frames) = double(imread(name_cv));
%*************************
    for ind = bg:bg+n_frames-1
        name_in = sprintf(['%s%0' num2str(lz)
'd%s'],filename,ind,'.pgm');
        disp(sprintf('Processing file: %s',name_in));
        name_cv = '/tmp/temp.pgm';
        system(sprintf('convert %s %s',name_in,name_cv));
        data(:,:,ind-bg+1) = double(imread(name_cv));
    end
elseif (extension == '.jpg')
    lz = 6;
    name_in = sprintf(['%s%0' num2str(lz) 'd%s'], filename, bg,
'.jpg');
    name_cv = '/tmp/temp.pgm';
    system(sprintf('convert %s %s', name_in, name_cv));
    data(:,:,n_frames) = double(imread(name_cv));
    for ind = bg:bg+n_frames-1
        name_in = sprintf(['%s%0' num2str(lz) 'd%s'], filename, ind,
'.jpg');
        disp(sprintf('Processing file: %s', name_in));
        name_cv = '/tmp/temp.pgm';
        system(sprintf('convert %s %s', name_in, name_cv));
        data(:,:,ind-bg+1) = double(imread(name_cv));
    end
else
    lz = 5;
    name_in = sprintf(['%s%0' num2str(lz) 'd%s'], filename, bg,
'.ppm');
    name_cv = '/tmp/temp.pgm';
    system(sprintf('convert -resize 320x240 %s %s', name_in, name_cv));
    data(:,:,n_frames) = double(imread(name_cv));
    for ind = bg:bg+n_frames-1
        name_in = sprintf(['%s%0' num2str(lz) 'd%s'], filename, ind,
'.ppm');
```

```
        disp(sprintf('Processing file: %s', name_in));
        name_cv = '/tmp/temp.pgm';
        system(sprintf('convert -resize 320x240 %s %s', name_in,
name_cv));
        data(:,:,ind-bg+1) = double(imread(name_cv));
    end
end
```

The following code implements recursive relaxed high pass filter

```
function [hc] = rhp(A,tau,sr,y,A0)
%
%  recursive high-pass filter of signal A with time constant tau and
sampling rate sr
%
%  A can be 1xN   = one signal with N time samples
%           MxN   = M signals with N time samples
%           RxCxN = a 2D signal (image) with N time samples
%
%  y is the output of the filter at t-1
%
%  A0 is the signal at t-1
%
[rw,cl,dp] = size(A);
d = exp(-sr/tau);
a0 = (1+d)/2;
a1 = -(1+d)/2;
b1 = d;

if (rw==1)
   hc = zeros(1,cl);
   hc(1,1) = a0*A(1,1) + a1*A0 + b1*y;
   for i = 2:cl
      hc(1,i) = a0*A(1,i) + a1*A(1,i-1) + b1*hc(1,i-1);
   end
   hc = -hc;
   return;
end

if (dp == 1)
   hc = zeros(rw,cl);
   hc(:,1) = a0*A(:,1) + a1*A0 + b1*y;
   for i = 2:cl
      hc(:,i) = a0*A(:,i) + a1*A(:,i-1) + b1*hc(:,i-1);
   end
   hc = -hc;
   return;
end

if (dp ~= 1)
   hc = zeros(rw,cl,dp);
   hc(:,:,1) = a0*A(:,:,1) + a1*A0 + b1*y;
   for i = 2:dp
      hc(:,:,i) = a0*A(:,:,i) + a1*A(:,:,i-1) + b1*hc(:,:,i-1);
   end
```

```
   hc = -hc;
   return;
end
```

The following code implements recursive low pass filter

```
[rw,cl,dp] = size(A);
d = exp(-sr/tau);
a0 = 1-d;
b1 = d;

if (rw==1)
   hc = zeros(1,cl);
   hc(1,1) = a0*A(1,1) + b1*y;
   for i = 2:cl
      hc(1,i) = a0*A(1,i) + b1*hc(1,i-1);
   end
   return;
end

if (dp == 1)
   hc = zeros(rw,cl);
   hc(:,1) = a0*A(:,1) + b1*y;
   for i = 2:cl
      hc(:,i) = a0*A(:,i) + b1*hc(:,i-1);
   end
   return;
end

if (dp ~= 1)
   hc = zeros(rw,cl,dp);
   hc(:,:,1) = a0*A(:,:,1) + b1*y;
   for i = 2:dp
      hc(:,:,i) = a0*A(:,:,i) + b1*hc(:,:,i-1);
   end
   return;
end
```

The following codes creates relaxed high pass filter

```
function [rc] = rrp(A,tau,sr,y,A0)
%
%  recursive relaxed high-pass filter of signal A with time constant
%         tau and sampling rate sr
%
%  A can be 1xN   = one signal with N time samples
%           MxN   = M signals with N time samples
%           RxCxN = a 2D signal (image) with N time samples
%
%  y is the output of the filter at t-1
%
%  A0 is the signal at t-1
%
% rc = -0.9*rhp(A,tau,sr,y,A1) + 0.1*rlp(A,tau,sr,y,A1);
```

```
% rc = -hc;

[rw,cl,dp] = size(A);
d = exp(-sr/tau);
al0 = 1-d;
bl1 = d;

ah0 = (1+d)/2;
ah1 = -(1+d)/2;
bh1 = d;

if (rw==1)
   hc = zeros(1,cl);
   lc(1,1) = al0*A(1,1) + bl1*y;
   hc(1,1) = ah0*A(1,1) + ah1*A0 + bh1*y;
   for i = 2:cl
      lc(1,i) = al0*A(1,i) + bl1*lc(1,i-1);
      hc(1,i) = ah0*A(1,i) + ah1*A(1,i-1) + bh1*hc(1,i-1);
   end
   rc = -0.9*hc - 0.1*lc;
   return;
end

if (dp == 1)
   hc = zeros(rw,cl);
   lc(:,1) = al0*A(:,1) + bl1*y;
   hc(:,1) = ah0*A(:,1) + ah1*A0 + bh1*y;
   for i = 2:cl
      lc(:,i) = al0*A(:,i) + bl1*lc(:,i-1);
      hc(:,i) = ah0*A(:,i) + ah1*A(:,i-1) + bh1*hc(:,i-1);
   end
   rc = -0.9*hc - 0.1*lc;
   return;
end

if (dp ~= 1)
   hc = zeros(rw,cl,dp);
   lc(:,:,1) = al0*A(:,:,1) + bl1*y;
   hc(:,:,1) = ah0*A(:,:,1) + ah1*A0 + bh1*y;
   for i = 2:dp
      lc(:,:,i) = al0*A(:,:,i) + bl1*lc(:,:,i-1);
      hc(:,:,i) = ah0*A(:,:,i) + ah1*A(:,:,i-1) + bh1*hc(:,:,i-1);
   end
   rc = -0.9*hc - 0.1*lc;
   return;
end
```

This is command line prompt to process various sequences of image in a desired manner

```
% clear; clc;close all;pack;
%
test_video('/home/pupil/vkr2m5/Software/BioOF/Videos/Blocks/Frames','te
st','.pgm',0,31);
% clear; pack;
```

```
%
test_video('~/Software/BioOF/Videos/Office/Frames','mvid_save','.ppm',1
,120);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/Office/Results_BioOF/movie',3,120-
2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/Office/Results_BioOF/movie_a',3,120
-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/umc_test_sequence/Frames','seq_','.
jpg',1,247);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/umc_test_sequence/Results_BioOF/mov
ie',3,247-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/umc_test_sequence/Results_BioOF/mov
ie_a',3,247-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/daytime_slice_00_10_edit/Frames','d
aytime_slice_00_10_edit_','.jpg',456,10);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_00_10_edit/Results_Bi
oOF/movie',3,1336-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_00_10_edit/Results_Bi
oOF/movie_a',3,1336-2,'.ppm',0);

% clear all;close all;clc; pack;  %  Actually:  there are 7397 frames
in this sequence
%
test_video('~/Software/BioOF/Videos/daytime_slice_00_10_edit/Frames','0
0','.jpg',85,70);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_00_10_edit/Results_Bi
oOF/movie',3,1336-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_00_10_edit/Results_Bi
oOF/movie_a',3,1336-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/daytime_slice_10_20_edit/Frames','d
aytime_slice_10_20_edit_','.jpg',795,60);
```

```
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_10_20_edit/Results_Bi
oOF/movie',3,187-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_10_20_edit/Results_Bi
oOF/movie_a',3,187-2,'.ppm',0);

clear all;clc;close all; pack;
test_video('~/Software/BioOF/Videos/daytime_slice_10_20_edit/Frames','0
0','.jpg',795,60);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_10_20_edit/Results_Bi
oOF/movie',3,187-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_10_20_edit/Results_Bi
oOF/movie_a',3,187-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/daytime_slice_20_30_edit/Frames','d
aytime_slice_20_30_edit_','.jpg',1,2788);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_20_30_edit/Results_Bi
oOF/movie',3,2788-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_20_30_edit/Results_Bi
oOF/movie_a',3,2788-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/daytime_slice_30_40_edit/Frames','d
aytime_slice_30_40_edit_','.jpg',1,791);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_30_40_edit/Results_Bi
oOF/movie',3,791-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_30_40_edit/Results_Bi
oOF/movie_a',3,791-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/daytime_slice_40_50_edit/Frames','d
aytime_slice_40_50_edit_','.jpg',1,909);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_40_50_edit/Results_Bi
oOF/movie',3,909-2,'.ppm',0);
% clear; pack;
```

```
%
make_movie('~/Software/BioOF/Videos/daytime_slice_40_50_edit/Results_Bi
oOF/movie_a',3,909-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/daytime_slice_50_60_edit/Frames','d
aytime_slice_50_60_edit_','.jpg',1,1566);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_50_60_edit/Results_Bi
oOF/movie',3,1566-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/daytime_slice_50_60_edit/Results_Bi
oOF/movie_a',3,1566-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/HC_Truck/Frames','hc_truck_','.jpg'
,1,428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/HC_Truck/Results_BioOF/movie',3,428
-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/HC_Truck/Results_BioOF/movie_a',3,4
28-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/HC_Truck_Clouds/Frames','hc_truck_c
louds_','.jpg',1,428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/HC_Truck_Clouds/Results_BioOF/movie
',3,428-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/HC_Truck_Clouds/Results_BioOF/movie
_a',3,428-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/LC_Truck/Frames','lc_truck_','.jpg'
,1,428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LC_Truck/Results_BioOF/movie',3,428
-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LC_Truck/Results_BioOF/movie_a',3,4
28-2,'.ppm',0);
```

```
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LC_Truck_Clouds/Frames','lc_truck_c
louds_','.jpg',1,428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LC_Truck_Clouds/Results_BioOF/movie
',3,428-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LC_Truck_Clouds/Results_BioOF/movie
_a',3,428-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/NewDesert1/Frames','NewDesert1_','.
jpg',1,429);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NewDesert1/Results_BioOF/movie',3,4
29-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NewDesert1/Results_BioOF/movie_a',3
,429-2,'.ppm',0);

% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongCloudNoTruck/Frames','00','.jpg
',1,428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongCloudNoTruck/Results_BioOF/movi
e',3,428-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongCloudNoTruck/Results_BioOF/movi
e_a',3,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongShotCamo/Frames','00','.jpg',1,
428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotCamo/Results_BioOF/movie',3
,428-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotCamo/Results_BioOF/movie_a'
,3,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongShotHC/Frames','00','.jpg',1,42
8);
```

```
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotHC/Results_BioOF/movie',3,4
28-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotHC/Results_BioOF/movie_a',3
,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongShotLC/Frames','00','.jpg',1,42
8);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotLC/Results_BioOF/movie',3,4
28-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotLC/Results_BioOF/movie_a',3
,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/MediumShotCamo/Frames','00','.jpg',
1,359);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotCamo/Results_BioOF/movie'
,3,359-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotCamo/Results_BioOF/movie_
a',3,359-2,'.ppm',0);
%
% clear; pack;% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongCloudNoTruck/Frames','00','.jpg
',1,428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongCloudNoTruck/Results_BioOF/movi
e',3,428-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongCloudNoTruck/Results_BioOF/movi
e_a',3,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongShotCamo/Frames','00','.jpg',1,
428);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotCamo/Results_BioOF/movie',3
,428-2,'.ppm',0);
% clear; pack;
```

```
%
make_movie('~/Software/BioOF/Videos/LongShotCamo/Results_BioOF/movie_a'
,3,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongShotHC/Frames','00','.jpg',1,42
8);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotHC/Results_BioOF/movie',3,4
28-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotHC/Results_BioOF/movie_a',3
,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/LongShotLC/Frames','00','.jpg',1,42
8);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotLC/Results_BioOF/movie',3,4
28-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/LongShotLC/Results_BioOF/movie_a',3
,428-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/MediumShotCamo/Frames','00','.jpg',
1,359);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotCamo/Results_BioOF/movie'
,3,359-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotCamo/Results_BioOF/movie_
a',3,359-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/MediumShotHC/Frames','00','.jpg',1,
359);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotHC/Results_BioOF/movie',3
,359-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotHC/Results_BioOF/movie_a'
,3,359-2,'.ppm',0);
%
% clear; pack;
```

```
%
test_video('~/Software/BioOF/Videos/MediumShotLC/Frames','00','.jpg',1,
359);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotLC/Results_BioOF/movie',3
,359-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotLC/Results_BioOF/movie_a'
,3,359-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/NearShotCamo/Frames','00','.jpg',1,
251);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotCamo/Results_BioOF/movie',3
,251-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotCamo/Results_BioOF/movie_a'
,3,251-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/NearShotHC/Frames','00','.jpg',1,25
1);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotHC/Results_BioOF/movie',3,2
51-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotHC/Results_BioOF/movie_a',3
,251-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/NearShotLC/Frames','00','.jpg',1,25
1);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotLC/Results_BioOF/movie',3,2
51-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotLC/Results_BioOF/movie_a',3
,251-2,'.ppm',0);

%
test_video('~/Software/BioOF/Videos/MediumShotHC/Frames','00','.jpg',1,
359);
% clear; pack;
```

```
%
make_movie('~/Software/BioOF/Videos/MediumShotHC/Results_BioOF/movie',3
,359-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotHC/Results_BioOF/movie_a'
,3,359-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/MediumShotLC/Frames','00','.jpg',1,
359);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotLC/Results_BioOF/movie',3
,359-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/MediumShotLC/Results_BioOF/movie_a'
,3,359-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/NearShotCamo/Frames','00','.jpg',1,
251);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotCamo/Results_BioOF/movie',3
,251-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotCamo/Results_BioOF/movie_a'
,3,251-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/NearShotHC/Frames','00','.jpg',1,25
1);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotHC/Results_BioOF/movie',3,2
51-2,'.ppm',0);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotHC/Results_BioOF/movie_a',3
,251-2,'.ppm',0);
%
% clear; pack;
%
test_video('~/Software/BioOF/Videos/NearShotLC/Frames','00','.jpg',1,25
1);
% clear; pack;
%
make_movie('~/Software/BioOF/Videos/NearShotLC/Results_BioOF/movie',3,2
51-2,'.ppm',0);
% clear; pack;
```

```
%
make_movie('~/Software/BioOF/Videos/NearShotLC/Results_BioOF/movie_a',3
,251-2,'.ppm',0);
```

The following implements high pass filter, low pass filter & relaxed high pass filter non recursively using convolution. Convolution was used to simulate the effect of past inputs on filter

```
function [hb,hc] = thp(A,tau,sr)
%
%  high-pass filter of signal A with time constant tau and sampling
rate sr
%
%  A can be 1xN   = one signal with N time samples
%           MxN   = M signals with N time samples
%           RxCxN = a 2D signal (image) with N time samples
%
tf = 2*tau;
[rw,cl,dp] = size(A);
t = 0:sr:(tf-sr);
h = exp(-t/tau);
su = sum(h);
h = h/su;

if (rw == 1)       % one signal
   ha = conv(A,h);
   hb = ha(1:cl) - A;
   return;
end

if (dp == 1)      % multiple one dimensional signals
   ha = conv2(1,h,A);
   hb = ha(:,1:cl) - A;
   return;
end

if (dp ~= 1)      % 2D signal
   hb = zeros(rw,cl,dp);
   for i = 1:cl
     ha = conv2(1,h,reshape(A(:,i,:),rw,dp));
     hb(:,i,:) = reshape(ha(:,1:dp),rw,1,dp) - A(:,i,:);
   end
   return;
end


function [hb, hc] = tlp(A,tau,sr)
%
%  low-pass filter of signal A with time constant tau and sampling rate
sr
%
%  A can be 1xN   = one signal with N time samples
```

```
%           MxN   = M signals with N time samples
%           RxCxN = a 2D signal (image) with N time samples
%
tf = 2*tau;
[rw,cl,dp] = size(A);
t = 0:sr:(tf-sr);
h = exp(-t/tau);
su = sum(h);
h = h/su;

if (rw == 1)       % one signal
   ha = conv(A,h);
   hb = ha(1:cl);
   return;
end

if (dp == 1)      % multiple one dimensional signals
   ha = conv2(1,h,A);
   hb = ha(:,1:cl);
   return;
end

if (dp ~= 1)      % 2D signal
   hb = zeros(rw,cl,dp);
   for i = 1:cl
     ha = conv2(1,h,reshape(A(:,i,:),rw,dp));
     hb(:,i,:) = reshape(ha(:,1:dp),rw,1,dp);
   end
   return;
end

function [hb,hc] = trp(A,tau,sr)
%
%  relaxed high-pass filter of signal A with time constant tau
%         and sampling rate sr
%
%  A can be 1xN   = one signal with N time samples
%           MxN   = M signals with N time samples
%           RxCxN = a 2D signal (image) with N time samples
%
tf = 4*tau;
[rw,cl,dp] = size(A);
t = 0:sr:(tf-sr);
h = exp(-t/tau);
su = sum(h);
h = h/su;

if (rw == 1)       % one signal
   ha = conv(A,h);
   hb = 0.9*ha(1:cl) - A;
   return;
end

if (dp == 1)      % multiple one dimensional signals
   ha = conv2(1,h,A);
   hb = 0.9*ha(:,1:cl) - A;
```

```
   return;
end

if (dp ~= 1)      % 2D signal
   hb = zeros(rw,cl,dp);
   for i = 1:cl
     ha = conv2(1,h,reshape(A(:,i,:),rw,dp));
     hb(:,i,:) = 0.9*reshape(ha(:,1:dp),rw,1,dp) - A(:,i,:);
   end
   return;
end
```

The following function performs half wave rectification on the signal

```
% The following function takes an array at input
% The positive element of array are retained &
% negative element are replaced by 0
function h = pos(A)
n = size(A,2);
h = zeros(1,n);
for i = 1:n
    if A(i) < 0
       h(i) = 0;
    else
        h(i) = A(i);
    end
end
```

The following program is a training program for forward subset selection based on mahanabolis distance criteria

```
%the program given below calculates the error based on Mahalanobis
distance
clc;clear all;close all;
n1 = 2;
ug = 82;
loba(1) = 41;
loba(2) = 41;
name = '/home/pupil/vkr2m5/Desktop/trainingseq/movie';
sn = 80;
sp = 1;
fdr = noisemod(name,sp,sn);
%this section picks up fourier descriptor multiple of 5 in sequence
hv = 1:51;
hv1 = [1 5 10 15 20 25 30 35 40 45];
for i = 1:length(hv1)
    Ivcc = find(hv == hv1(i));
    hv(Ivcc) = [];
end
hva = 62:112;
hv1 = [65 70 75 80 85 90 95 100 105 110];
for i = 1:length(hv1)
   Ivcc = find(hva == hv1(i));
```

```
   hva(Ivcc) = [];
end
xs = fdr([hv hva],:);
xs1 = xs(:,2:101);
%*********************************************************************
****
xv = size(xs1,1);
overerr = zeros(1,ug);
for g = 1:ug
    X1 = zeros(1,100)';
    X1(:,1) = 1:100;
    if g == 1
       X2 = X1;
    else
       for j = 1:g-1
           Iv = find(X1(:,1)==X3(1,j));
           X1(Iv,:) = [];
       end
       X2 = zeros(length(X1),g);
       for j = 1:length(X1)
           X2(j,1:g-1) = X3(1,:);
       end
       X2(:,g) = X1;
    end
    e1 = zeros(1,size(X1,1));
    for h = 1:size(X1,1)
        a = 1;
        c = zeros([xv,n1]);
        Xavg = zeros([n1,g]);
        sigma = zeros([g,g,n1]);
        inv_sigma = zeros([g,g,n1]);
        Xavg_m = zeros([xv,g,n1]);
        for i = 1:n1
           if i ~= 1
              a = a+loba(i-1);
           end
           b = (a + loba(i)-1);
           Xavg(i,:) = mean(xs1(a:b,X2(h,:)));
           sigma(:,:,i) = cov(xs1(a:b,X2(h,:)));
           inv_sigma(:,:,i) = inv(sigma(:,:,i));
           Xavg_m(:,:,i) = repmat(Xavg(i,:),xv,1);
           c(:,i) = diag((xs1(:,X2(h,:)) - Xavg_m(:,:,i)) *
inv_sigma(:,:,i) * (xs1(:,X2(h,:)) - Xavg_m(:,:,i))');
        end
        [minv,Iv] = min(c,[],2);
        pclassi = Iv;
        a = 1;
        e = zeros(1,n1);
        for i = 1:n1
            if i ~= 1
               a = a+loba(i-1);
            end
            b = (a + loba(i)-1);
            Y = pclassi(a:b);
            I = find(Y ~= i);
            e(i) = length(I);
        end
```

```
        e1(h) = sum(e)/xv;
    end
    [aa,I1] = min(e1);
     X3 = X2(I1,:);
     overerr(g) = aa;
end
[mierr,optnco] = min(overerr);
for i = 1:length(overerr)
    g(i) = i;
end
stem(g,overerr);
grid;
%this portion calculates the sigma ... sigma n & average1....averagen
for best number of
%coeifficients
clc;
clear Xavg;
clear sigma;
clear inv_sigma;
clear inv_sigma1;
clear Xavg_m;
a = 1;
c = zeros([xv,n1]);
Xavg = zeros([n1+1,optnco]);
sigma = zeros([optnco,optnco,n1]);
inv_sigma = zeros([optnco,optnco,n1]);
Xavg_m = zeros([xv,optnco,n1]);
Xavg(1,:) = X3(1:optnco);
 for i = 1:n1
     if i ~= 1
        a = a+loba(i-1);
     end
     b = (a + loba(i)-1);
     Xavg(i+1,:) = mean(xs1(a:b,Xavg(1,:)));
     sigma(:,:,i) = cov(xs1(a:b,Xavg(1,:)));
% %  inv_sigma(:,:,i) = eye(g,g);
     inv_sigma(:,:,i) = inv(sigma(:,:,i));
%      Xavg_m(:,:,i) = repmat(Xavg(i+1,:),xv,1);
  end
nob = zeros(1,length(Xavg(1,:)));
nob(1) = n1;
save traindat nob -ASCII -APPEND;
save traindat Xavg -ASCII -APPEND;
for i = 1:n1
    inv_sigma1 = inv_sigma(:,:,i);
    save traindat inv_sigma1 -ASCII -APPEND;
    clear inv_sigma1;
end;
```

The following program is a testing program of forward subset selection based mahanabolis distance

```
%this is testing of mahanabolis distance criteria applied in testdat7
```

```
clc;clear all;close all
dat = load('/home/pupil/vkr2m5/Desktop/C++practice1/traindat');
nob = dat(1,1);
for i = 1:(nob+1)
Xavg(i,:) = dat(1+i,:);
end
dp = nob+2;
l = length(Xavg(1,:));
for i = 1:nob
    for j = 1:l
        dp1 = (i-1)*l + j;
        inv_sigma(j,:,i) = dat(dp+dp1,:);
    end
end
name = '/home/pupil/vkr2m5/Desktop/trainingseq/movie';
sn = 20;
sp = 81;
fdr = noisemod(name,sp,sn);
hv1 = [1 5 10 15 20 25 30 35 40 45];
hv2 = [65 70 75 80 85 90 95 100 105 110];
%**************************************************************************
****
xh1 = fdr(hv1,2:101);
xa1 = size(xh1,1);
xc1 = fdr(hv2,2:101);
v1 = 1:2*xa1;
% even positions are given to xh1 & odd positions are given to xc1
v = find((mod(v1,2)) == 0);
for i = 1:length(v)
w = find(v1 == v(i));
v1(w) = [];
end
hucar1(v,:) = xh1;
hucar1(v1,:) = xc1;
xv1 = size(hucar1,1);
Xhavg_m = repmat(Xavg(2,:),xv1,1);
Xcavg_m = repmat(Xavg(3,:),xv1,1);
b1 = diag((hucar1(:,Xavg(1,:)) - Xhavg_m) * inv_sigma(:,:,1) *
(hucar1(:,Xavg(1,:)) - Xhavg_m)');
c1 = diag((hucar1(:,Xavg(1,:)) - Xcavg_m) * inv_sigma(:,:,2) *
(hucar1(:,Xavg(1,:)) - Xcavg_m)');
pclassi1 = b1>c1;
Y1 = pclassi1(v);
Iv1 = find(Y1 == 1);
ev1 = length(Iv1);
Y1 = pclassi1(v1);
Iv2 = find(Y1 == 0);
ev2 = length(Iv2);
ev = (ev1 + ev2)/length(pclassi1);
```

The following program is a training program for classification based on scatter matrix

```
clc;clear all;close all;
n1 = 2;
xd = 100;
```

```
%nos = 80;
loba(1) = 41;
loba(2) = 41;
name = '/home/pupil/vkr2m5/Desktop/trainingseq/movie';
sn = 80;
sp = 1;
fdr = noisemod(name,sp,sn);
%this section picks up fourier descriptor multiple of 5 in sequence
hv = 1:51;
hv1 = [1 5 10 15 20 25 30 35 40 45];
for i = 1:length(hv1)
    Ivcc = find(hv == hv1(i));
    hv(Ivcc) = [];
end
hva = 62:112;
hv1 = [65 70 75 80 85 90 95 100 105 110];
for i = 1:length(hv1)
   Ivcc = find(hva == hv1(i));
   hva(Ivcc) = [];
end
xs = fdr([hv hva],:);
xs1 = xs(:,2:101);
%**********************************************************************
****
xv = size(xs1,1);
xv1 = size(xs1,2);
a = 1;
Xavg = zeros([n1,xv1]);
sigma = zeros([xv1,xv1,n1]);
for i = 1:n1
    if i ~= 1
       a = a+loba(i-1);
    end
       b = (a + loba(i)-1);
       Xavg(i,:) = mean(xs1(a:b,:));
       sigma(:,:,i) = cov(xs1(a:b,:));
end
p = 1/n1;
sw = 0;
for i = 1:n1
    sw = sw + p*sigma(:,:,i);
end
sb = 0;
for i = 2:n1
   for j = 1:i-1
       sb = sb+(p*p)*(Xavg(i,:)-Xavg(j,:))'*(Xavg(i,:)-Xavg(j,:));
   end
end
A = (inv(sw))*sb;
[V,D] = eig(A);
phi = V(:,1:(n1-1));
phi(1:xv1,n1) = 0;
for i = 1:n1
   M1(:,i) = (phi(:,1:(n1-1)))'*Xavg(i,:)';
end
nob = zeros(1,n1);
nob(1) = n1;
```

```
nob(2) = xv1;
save traindatv2 nob -ASCII -APPEND;
save traindatv2 phi -ASCII -APPEND;
save traindatv2 M1 -ASCII -APPEND;
```

The following program is a testing classification program based on scatter matrix

```
%testing part based on scater n
clc;clear all;close all
dat = load('/home/pupil/vkr2m5/Desktop/C++practice1/traindatv2');
nob1 = dat(1,1);
nob2 = dat(1,2);
phi = dat(2:nob2+1,1:(nob1-1));
Me = dat(nob2+2:nob2+nob1,1:nob1);
name = '/home/pupil/vkr2m5/Desktop/trainingseq/movie';
sn = 20;
sp = 81;
fdr = noisemod(name,sp,sn);
%**********************************************************************
****
nos =1;
%**********************************************************************
****
hv1 = [1 5 10 15 20 25 30 35 40 45];
hv2 = [65 70 75 80 85 90 95 100 105 110];
hv = [hv1 hv2];
for sp = 1:20
hu1 = fdr(hv(:,sp),:);
xv1 = size(hu1,1);%rows size
hu = hu1(:,2:nob2+1);
y = phi'*hu';
v = size(Me,2);
y2 = repmat(y,1,v);
y1 = (Me-y2).^2;
y3 = sum(y1,1);
y4 = sqrt(y3);
[C,I] = min(y4);
class(sp,1) = I;
end
```

The following program is training classification program based on PCA based on mahanabolis distance

```
 %the program given below calculates the error based on pca analysis
clc;clear all;close all;
n1 = 2;
ug = 82;
loba(1) = 41;
loba(2) = 41;
name = '/home/pupil/vkr2m5/Desktop/trainingseq/movie';
sn = 80;
sp = 1;
fdr = noisemod(name,sp,sn);
hv = 1:51;
```

```
hv1 = [1 5 10 15 20 25 30 35 40 45];
for i = 1:length(hv1)
    Ivcc = find(hv == hv1(i));
    hv(Ivcc) = [];
end
hva = 62:112;
hv1 = [65 70 75 80 85 90 95 100 105 110];
for i = 1:length(hv1)
   Ivcc = find(hva == hv1(i));
   hva(Ivcc) = [];
end
xs = fdr([hv hva],:);
xs1 = xs(:,2:101);
%**************************************************************************
****
xv = size(xs1,1);
xv1 = size(xs1,2);
%**from here starts the section to calculate pca
components***************
S =  cov(xs1);
[V,D] = eig(S);
y = (V'*xs1(1:xv,:)')';
%**************************************************************************
****
overerr = zeros(1,ug);
for g = 1:ug
    X1 = zeros(1,100)';
    X1(:,1) = 1:100;
    if g == 1
       X2 = X1;
    else
       for j = 1:g-1
           Iv = find(X1(:,1)==X3(1,j));
           X1(Iv,:) = [];
       end
       X2 = zeros(length(X1),g);
       for j = 1:length(X1)
           X2(j,1:g-1) = X3(1,:);
       end
       X2(:,g) = X1;
    end
    e1 = zeros(1,size(X1,1));
    for h = 1:size(X1,1)
        a = 1;
        c = zeros([xv,n1]);
        Xavg = zeros([n1,g]);
        sigma = zeros([g,g,n1]);
        inv_sigma = zeros([g,g,n1]);
        Xavg_m = zeros([xv,g,n1]);
        for i = 1:n1
           if i ~= 1
              a = a+loba(i-1);
           end
           b = (a + loba(i)-1);
           Xavg(i,:) = mean(y(a:b,X2(h,:)));
           sigma(:,:,i) = cov(y(a:b,X2(h,:)));
           inv_sigma(:,:,i) = inv(sigma(:,:,i));
```

```
            Xavg_m(:,:,i) = repmat(Xavg(i,:),xv,1);
            c(:,i) = diag((y(:,X2(h,:)) - Xavg_m(:,:,i)) *
inv_sigma(:,:,i) * (y(:,X2(h,:)) - Xavg_m(:,:,i))');
        end
        [minv,Iv] = min(c,[],2);
        pclassi = Iv;
        a = 1;
        e = zeros(1,n1);
        for i = 1:n1
            if i ~= 1
               a = a+loba(i-1);
            end
            b = (a + loba(i)-1);
            Y = pclassi(a:b);
            I = find(Y ~= i);
            e(i) = length(I);
        end
        e1(h) = sum(e)/xv;
    end
    [aa,I1] = min(e1);
     X3 = X2(I1,:);
     overerr(g) = aa;
end
[mierr,optnco] = min(overerr);
for i = 1:length(overerr)
    g(i) = i;
end
stem(g,overerr);
grid;
%this portion calculates the sigma ... sigma n & average1....averagen
for best number of
%coeifficients
clc;
clear Xavg;
clear sigma;
clear inv_sigma;
clear inv_sigma1;
clear Xavg_m;
a = 1;
c = zeros([xv,n1]);
Xavg = zeros([n1+1,optnco]);
sigma = zeros([optnco,optnco,n1]);
inv_sigma = zeros([optnco,optnco,n1]);
Xavg_m = zeros([xv,optnco,n1]);
Xavg(1,:) = X3(1:optnco);
 for i = 1:n1
     if i ~= 1
        a = a+loba(i-1);
     end
     b = (a + loba(i)-1);
     Xavg(i+1,:) = mean(y(a:b,Xavg(1,:)));
     sigma(:,:,i) = cov(y(a:b,Xavg(1,:)));
     inv_sigma(:,:,i) = inv(sigma(:,:,i));
  end
nob = zeros(1,length(Xavg(1,:)));
nob(1) = n1;
save traindatv nob -ASCII -APPEND;
```

```
save traindatv Xavg -ASCII -APPEND;
for i = 1:n1
    inv_sigma1 = inv_sigma(:,:,i);
    save traindatv inv_sigma1 -ASCII -APPEND;
    clear inv_sigma1;
end;
save traindatv1 V -ASCII -APPEND;
```

The following program is testing classification program based on PCA based mahanabolis distance criteria

```
%the program given below calculates the error based on pca analysis
clc;clear all;close all
nos = 1;
dat = load('/home/pupil/vkr2m5/Desktop/C++practice1/traindatv');
V = load('/home/pupil/vkr2m5/Desktop/C++practice1/traindatv1');
nob = dat(1,1);
for i = 1:(nob+1)
    Xavg(i,:) = dat(1+i,:);
end
dp = nob+2;
l = length(Xavg(1,:));
%there might be some error in reading sigma
for i = 1:nob
    for j = 1:l
        dp1 = (i-1)*l + j;
        inv_sigma(j,:,i) = dat(dp+dp1,:);
    end
end
%uncomment this part in
thesis*****************************************
name = '/home/pupil/vkr2m5/Desktop/trainingseq/movie';
sn = 20;
sp = 81;
fdr = noisemod(name,sp,sn);
%*************************************************************************
****
hv1 = [1 5 10 15 20 25 30 35 40 45];
hv2 = [65 70 75 80 85 90 95 100 105 110];
hv = [hv1 hv2];
for sp = 1:20
hu1 = fdr(hv(sp),:);
te = hu1(:,1);
xv1 = size(hu1,1);%rows size
xv2 = size(hu1,2);%columns size
hu = hu1(:,2:xv2);
%************************this part deals with the pca
analysis************
y = (V'*hu(1:xv1,:)')';
b = zeros([xv1,nob]);
for i = 1:nob
Xavg_m(:,:,i) = repmat(Xavg((i+1),:),xv1,1);
```

```
b(:,i) = diag((y(:,Xavg(1,:)) - Xavg_m(:,:,i)) * inv_sigma(:,:,i) *
(y(:,Xavg(1,:)) - Xavg_m(:,:,i))');
%**************************uncomment this part to get
average***********
% b(:,i) = diag((hu(:,Xavg(1,:)) - Xavg_m(:,:,i)) * inv_sigma(:,:,xvv)
* (hu(:,Xavg(1,:)) - Xavg_m(:,:,i))');
%*********************************************************************
****
end
[minv,Iv] = min(b,[],2);
pclassi1(sp,1) = Iv;
end
```

Following program was used to get border contour coordinates of model images

```
clc;clear all;close all;
    sp = 1;
    sn = 69;
    vb = zeros(1,5);
for ixx = 1:6
    name = '/home/pupil/vkr2m5/Desktop/object4';
    fdr = zeros([sp+sn-1,60]);
    for ix = sp:(sp+sn-1)
        clear xb;
        clear yb;
        ft = 0;
        ft1 = 0;
        name1 =
strcat(name,'/',num2str(ixx),'/mod',num2str(ix),'.jpg');
        C = imread(name1);
        x = size(C,1);
        for i = 1:x
            Iv = find(C(i,:) > 200);
            C(i,Iv) = 255;
            Iv = find(C(i,:) < 200);
            C(i,Iv) = 0;
        end
        X = bwlabel(C,8);
        [Ir,Ic] = find(X==1);
        [Ir1,Ic1] = find(X>1);
        for i = 1:length(Ir1)
        X(Ir1(i),Ic1(i)) = 0;
        end
        [mav,Imv] = min(Ir);
        a = 1;
        b = 1;
        xb(a) = Ir(Imv);
        yb(a) = Ic(Imv);
        xb(a+1) = -40;
        yb(a+1) = -40;
        b = length(xb);
        while ((abs(xb(b)-xb(1))+abs(yb(b)-yb(1))~=0))
                aa = [xb(a),yb(a)];
                if (X(xb(a),yb(a)+1) ==0) & (X(xb(a)+1,yb(a)+1)==1) &
(isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a)+1,yb(a)+1])==0)
```

```
                    xb(a+1) = xb(a)+1;
                    yb(a+1) = yb(a)+1;
                    a = a+1;
                elseif (X(xb(a)+1,yb(a)+1)==0) & (X(xb(a)+1,yb(a))==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a)+1,yb(a)])==0)
                    xb(a+1) = xb(a)+1;
                    yb(a+1) = yb(a);
                    a = a+1;
                elseif (X(xb(a)+1,yb(a))==0) & (X(xb(a)+1,yb(a)-1)==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a)+1,yb(a)-1])==0)
                    xb(a+1) = xb(a)+1;
                    yb(a+1) = yb(a)-1;
                    a = a+1;
                elseif (X(xb(a)+1,yb(a)-1)==0) & (X(xb(a),yb(a)-1)==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a),yb(a)-1])==0)
                    xb(a+1) = xb(a);
                    yb(a+1) = yb(a)-1;
                    a = a+1;
                elseif (X(xb(a),yb(a)-1)==0) & (X(xb(a)-1,yb(a)-1)==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a)-1,yb(a)-1])==0)
                    xb(a+1) = xb(a)-1;
                    yb(a+1) = yb(a)-1;
                    a = a+1;
                elseif (X(xb(a)-1,yb(a)-1)==0) & (X(xb(a)-1,yb(a))==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a)-1,yb(a)])==0)
                    xb(a+1) = xb(a)-1;
                    yb(a+1) = yb(a);
                    a = a+1;
                elseif (X(xb(a)-1,yb(a))==0) & (X(xb(a)-1,yb(a)+1)==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a)-1,yb(a)+1])==0)
                    xb(a+1) = xb(a)-1;
                    yb(a+1) = yb(a)+1;
                    a = a+1;
                elseif (X(xb(a)-1,yb(a)+1)==0) & (X(xb(a),yb(a)+1)==1)
& (isthere(xb(2:length(xb)),yb(2:length(xb)),[xb(a),yb(a)+1])==0)
                    xb(a+1) = xb(a);
                    yb(a+1) = yb(a)+1;
                    a = a+1;
                end
                b = length(xb);
        end
```

The following program creates bounding box across original Image BioOF results & ground truth bounding box across original image. It also plot graph for quantitative analysis of human & truck.

```
clear all;close all;clc;
% name = '/home/pupil/vkr2m5/Desktop/trainim/movie';
name =
'/home/pupil/vkr2m5/Software/BioOF/Videos/daytime_slice_00_10_edit/Resu
lts_BioOF/movie';
sn = 235;
```

```
sp = 421;
for ix = sp:(sp+sn-1)
    ix1 = sprintf('%06d%s',ix);
    name2 = strcat(name,ix1,'.ppm');
    I1 = imread(name2);
    I1 = rgb2gray(I1);
    if ix < 421
     I3 = distn1(I1);%for humans use distn & for cars use distn1
     f2 = preproc1(I3);
    else
       I3 = distn2(I1);
       f2 = preproc(I3);
    end
    [r,c] = find(f2 == 1);
    if length(r) > 0
        r1 = max(r);
        r2 = min(r);
        c1 = max(c);
        c2 = min(c);
    else
        r1 = 0;
        r2 = 0;
        c1 = 0;
        c2 = 0;
    end
    x = [r2,r2,r1,r1,r2];
    y = [c2,c1,c1,c2,c2];
    name1 =
'/home/pupil/vkr2m5/Software/BioOF/Videos/daytime_slice_00_10_edit/Fram
es/';
    ix2 = sprintf('%08d%s',ix);
    name3 = strcat(name1,ix2,'.jpg');
    Ig = imread(name3);
    figure(1)
    imshow(Ig);
    hold on
    plot(y,x,'-b','linewidth',2);
    hold off
    cartbb =
load('/home/pupil/vkr2m5/Desktop/groundtruth/carbboxorig.txt');
    humtbb =
load('/home/pupil/vkr2m5/Desktop/groundtruth/humanbboxorig.txt');
    if ix > 420
     I = find(humtbb(:,1) == ix);
     r2a = humtbb(I,3);
     r1a = humtbb(I,3) + humtbb(I,5);
     c2a = humtbb(I,2);
     c1a = humtbb(I,2) + humtbb(I,4);
    else
     I = find(cartbb(:,1) == ix);
     r2a = cartbb(I,3);
     r1a = cartbb(I,3) + cartbb(I,5);
     c2a = cartbb(I,2);
     c1a = cartbb(I,2) + cartbb(I,4);
    end
     x1 = [r2a,r2a,r1a,r1a,r2a];
     y1 = [c2a,c1a,c1a,c2a,c2a];
```

```
figure(1);
hold on;
plot(y1,x1,'-r',y,x,'-b','linewidth',2);
hold off;
h = legend('Ground truth Bounding Box','Response Bounding Box',2);
set(h,'Interpreter','none');
title('bounding boxes (Truck)')
namev1 = strcat('Truckbb',ix1);
print('-djpeg',namev1);
figure(2)
imshow(I1);
hold on
plot(y,x,'b','linewidth',2);
title('Biooptical flow response Bounding Box (Truck)');
namev2 = strcat('Truckrbb',ix1);
hold off
print('-djpeg',namev2);
figure(3)
imshow(Ig);
hold on
plot(y1,x1,'r','linewidth',2);
hold off
title('Ground truth bounding box (Truck)');
namev3 = strcat('Truckgtbb',ix1);
print('-djpeg',namev3);
%    this section calculates true positive
xv = size(I3,1);
yv = size(I3,2);
if length(r) > 0
   tp = -1*ones([xv,yv]);
   dp = zeros([xv,yv]);
   tp(r2a:r1a,c2a:c1a) = 0;
   if length(r) > 0
       dp(r2:r1,c2:c1) = 1;
   end
   rp = tp+dp;
   [vor,voc] = find(rp == 1);
   trp = logical(zeros([xv,yv]));
   for i = 1:length(vor)
       trp(vor(i),voc(i)) = 1;
   end
else
   trp = logical(zeros([xv,yv]));
end
tp = logical(zeros([xv,yv]));
dp = logical(zeros([xv,yv]));
tp(r2a:r1a,c2a:c1a) = 1;
if length(r) > 0
   dp(r2:r1,c2:c1) = 1;
end
fp = logical(dp-trp);
fn = logical(tp-trp);
tn = trp+fp+fn;
tn = logical(tn);
clear vor;
clear voc;
[vor,voc] = find(tn > 0);
```

```
      Atn = length(vor);
      clear vor;
      clear voc;
      [vor,voc] = find(fn == 1);
      Afn = length(vor);
      clear vor;
      clear voc;
      [vor,voc] = find(fp == 1);
      Afp = length(vor);
      clear vor;
      clear voc;
      [vor,voc] = find(trp == 1);
      Atrp = length(vor);
      rec(ix-sp+1) = Atrp/(Atrp+Afn);
      if length(r) > 0
         prec(ix-sp+1) = Atrp/(Atrp+Afp);
      else
          prec(ix-sp+1) = 1;
      end
end
xaxv = 1:length(prec);
figure(5)
plot(xaxv,prec,'-b*',xaxv,rec,'-ro');
h = legend('Precision','Recall',2);
set(h,'Interpreter','none');
axis([1 length(prec) 0 1.4]);
xlabel('frames');
ylabel('precision and recall');
title('precesion & recall for bounding box (Person)');
namev = 'Qualitative analysis Person10';
print('-djpeg',namev);
```

The following program bring the image object to the center of image & calculates radius

of snake contour

```
function [H,rad] = cenrad(I)
cv1 = size(I,1);
cv2 = size(I,2);
if cv1 > cv2
    c = cv1;
else
    c = cv2;
end
[r1,c1] = find(I < 255);
if(size(r1,1) >= 20)
   nr = round(mean(r1));
   nc = round(mean(c1));
%    r2 = r1-nr+round(cv1/2);
%    c2 = c1-nc+round(cv2/2);
   r2 = r1-nr+round(c/2);
   c2 = c1-nc+round(c/2);
   H = uint8(255*ones([c,c]));
   for i =1:length(r2)
       if (r2(i) > 0) & (c2(i) > 0)
           H(r2(i),c2(i)) = I(r1(i),c1(i));
```

```
       end
    end
end
% %rad = max(max(r2),max(c2)) - round(c/2);
rad = sqrt((max(r2)-round(c/2)).^2 + (max(c2)-round(c/2)).^2);
if (rad+10 < cv1) & (rad+10 < cv2)
     rad = rad+2;
end
```

The following program isolates the image area from the whole image based on distribution

```
function [H,H1] = distn2(I)
v1 = size(I,1);
 v2 = size(I,2);
 I(:,v2-10:v2) = 0;
 I(:,1:5) = 0;
 %********* The parameter given below matters on how image
is**************
 imth = 20;
 zn = 15;

%*************************************************************************
***
 H = uint8(zeros([v1,v2]));
% the following section deals with distribution along column
 coll = sum(I,1);
 mc = mean(coll);
 coll = coll-(mc);
 Ic = find(coll < 0);
 coll(Ic) = 0;
Is = find(coll > 0);
colm(1) = Is(1);
count = 1;
for i = 2:length(Is)
    a = Is(i) - Is(i-1);
    if a > zn
       count = count+1;
       colm(count) = Is(i-1);
       count = count+1;
       colm(count) = Is(i);
    end
end
colm(count+1) = Is(length(Is));
% this newly added section it considers the distribution which contains
peak
visa = 0;
while visa < imth
    [mac,Iac] = max(coll);
    for a = 2:2:(length(colm))
        if (Iac < colm(a)) & (Iac > colm(a-1))
            Icv = colm(a-1);
            Icv1 = colm(a);
```

```
        end
        if (Iac == colm(a-1))
            Icv = colm(a-1);
            Icv1 = colm(a);
        end
        if (Iac == colm(a))
            Icv = colm(a-1);
            Icv1 = colm(a);
        end
    end
   visa = Icv1-Icv;
    if visa < imth
        coll(Icv:Icv1)= 0;
        ivv = find(colm == Icv);
        colm(ivv) = [];
        ivv = find(colm == Icv1);
        colm(ivv) = [];
    end
    if isempty(colm) == 1
       break;
    end
end
I1 = uint8(zeros([v1,v2]));
I1(:,(Icv-5):(Icv1+5)) = I(:,(Icv-5):(Icv1+5));
clear I;
I = I1;
clear I1;
%The following section deals with distribution along row
 roww = sum(I,2);
 mr = mean(roww);
 roww = roww-(mr);
 Ir = find(roww < 0);
 roww(Ir) = 0;
Irs = find(roww > 0);
rowm(1) = Irs(1);
count = 1;
for i = 2:length(Irs)
    a = Irs(i) - Irs(i-1);
    if a > zn
       count = count+1;
       rowm(count) = Irs(i-1);
       count = count+1;
       rowm(count) = Irs(i);
    end
end
rowm(count+1) = Irs(length(Irs));
%this is newly added section
visa = 0;
while visa < imth
    [mar,Iar] = max(roww);
    for a = 2:2:length(rowm)
        if (Iar < rowm(a)) & (Iar > rowm(a-1))
            Irv = rowm(a-1);
            Irv1 = rowm(a);
        end
        if (Iar == rowm(a))
            Irv = rowm(a-1);
```

```
            Irv1 = rowm(a);
        end
        if (Iar == rowm(a-1))
           Irv = rowm(a-1);
           Irv1 = rowm(a);
        end
    end
    visa = Irv1-Irv;
    if visa < imth
        roww(Irv:Irv1)= 0;
        ivv = find(rowm == Irv);
        rowm(ivv) = [];
        ivv = find(rowm == Irv1);
        rowm(ivv) = [];
    end
    if isempty(roww) == 1
       break;
    end
end
H(Irv:Irv1,Icv:Icv1) = I(Irv:Irv1,Icv:Icv1);
H1 = I(Irv:Irv1,Icv:Icv1);
```

The following program computes Fourier descriptor of the contour

```
function h = frdes(a)
x = a(1,:);
y = a(2,:);
xc = mean(x);
yc = mean(y);
x = x-xc;
y = y-yc;
theta = zeros(1,length(x));
w = 3;
for ivi = 1:length(x)
    ivi1 = ivi-w;
    if ivi1 <= 0
        ivi1 = length(x) + ivi1;
    end
    if (x(ivi) - x(ivi1)) ~= 0
        theta(ivi) = atan((y(ivi) - y(ivi1))/(x(ivi)-x(ivi1)));
    else
        theta(ivi) = 22/14;
    end
end
phi = mod((theta-theta(1)),(44/7));
kcurve = zeros(1,length(phi)-1);
for ivi = 2:length(phi)
    kcurve(ivi) = phi(ivi) - phi(ivi-1);
end
fd = fftn(kcurve);
Magfd = abs(fd);
xiv = length(Magfd);
fdx = round(xiv/2);
if Magfd(1) > 0.001
   h = Magfd/Magfd(1);
```

```
   h = h(2:fdx);
else
   h = Magfd(2:fdx);
end
```

The following graph shifts vertical optical flow by 3 pixels & combine horizontal & vertical optical flow

```
clc;clear all;close all;
name =
'/home/pupil/vkr2m5/Software/BioOF/Videos/daytime_slice_10_20_edit/Resu
lts_BioOF/movie_a';
% name =
'/home/pupil/vkr2m5/Software/BioOF/Videos/Blocks/Frames/movie_a';
for ix = 798:855;
ix = 854;
ix1 = sprintf('%06d%s',ix);
name1 = strcat(name,ix1,'.ppm');
I = imread(name1);
I = rgb2gray(I);
I1 = uint8(zeros([490,730]));
I1(11:490,11:730) = I(481:960,:);
I2 = I(961:1440,:);
I3 = I1(8:487,11:730) + I2(:,:);
figure(2)
imshow(imcomplement(I3));
name2 = '/home/pupil/vkr2m5/Desktop/newtrainingseq/movie';
ix = ix-797+62;
ix1 = sprintf('%06d%s',ix);
name3 = strcat(name2,ix1,'.ppm');
imwrite(I3,name3,'ppm');
end
```

The following program see if the coordinates are already present

```
function [h] = isthere(X,a)
h = 0;
vv = find(X(1,:) == a(1));
for i = 1:length(vv)
    if(X(2,vv(i)) == a(2))
        h =1;
        break;
    end
end
```

The following program calculates fourier descriptor from the image after preprocing it

```
function fdr = noisemod(name,sp,sn)
clear all;close all;clc;
name = '/home/pupil/vkr2m5/Desktop/newtrainingseq/movie';
```

```
sn = 1;
sp = 12;
ab = 1;
for ix = sp:(sp+sn-1)
    ix1 = sprintf('%06d%s',ix);
    name2 = strcat(name,ix1,'.ppm');
    I1 =imread(name2);
    I1 = rgb2gray(I1);
    [I3,I4] = distn2(I1);
    I4 = imadjust(I4);
    av = mean(mean(I4>0));
    av = 1-av;
    f2 = preproc5(I3,av);
%     f4 = preproc4(I3,av);
    name1 = '/home/pupil/vkr2m5/Desktop/snaketest/movie';
    name3 = strcat(name1,ix1,'.ppm');
    imwrite(f2,name3,'ppm');
    figure(1)
    imshow(f2);
%     [rvv1,cvv1] = find(f2 == 1);
%     if rvv1 >= 1;
       bi = round(snakemod(f2));
       namev = strcat('figurec',ix1);
       print('-djpeg',namev);
       ly = size(bi,2);
       bi(:,ly+1) = bi(:,1);
       b = round(connectpoly((bi(1,:)),(bi(2,:))));
       b = b';
       b = repbust(b);
       b(2,:) = max(b(2,:)) - b(2,:);
       fdc = frdes(b);
       if length(fdc) > 99
          fdr(ab,1) = ix;
          fdr(ab,2:101) = fdc(1,1:100);
          ab = ab+1;
       else
            disp('Noise is large data information is tough to
extract');
       end
%      else
%          disp('this is just a noisy clip');
%     end
end
% save svmfourd2 fdr -ASCII -APPEND;
```

The following program preprocess image to remove noise out of it

```
function H = preproc5(I,ad)
f = spfilt(I,'median',3,3);
thr = 15;
f = con(f,thr,0);
f = imadjust(f);
aa = max(max(f));
aa = ad*aa;
f1 = f;
```

```
f1(f<= aa) = 0;
se = strel('square',2);
H = imopen(f1,se);
H = spfilt(H,'median',4,4);
H = imerode(H,strel('square',1));
thr = 100;
H = con(H,thr,0);
H1 = H;
H1(H<150) = 0;
[junk threshold] = edge(H1,'log');
fudgeFactor = 1.5;
Bw = edge(H1,'log',threshold * fudgeFactor);
se90 = strel('line',30,90);
se0 = strel('line',30,0);
Bwdil = imdilate(Bw,[se90,se0]);
BWdfill = imfill(Bwdil,'holes');
BWnd = imclearborder(BWdfill,8);
seD = strel('diamond',3);
BWfinal = imerode(BWnd,seD);
BWfinal = imerode(BWfinal,seD);
H(BWfinal == 0) = 0;
```

The following program removes any repetitive coordinates if present

```
function h = repbust(v)
if size(v,1) > size(v,2)
    v = v';
end
if size(v,2) == 1
    h = v;
else
    a = 2;
    h(:,1) = v(:,1);
    for i = 1:(size(v,2)-1)
        if isthere(h,v(:,i+1)) == 0
           h(:,a) = v(:,i+1);
           a = a+1;
        end
    end
end
```

REFERENCES

[1] Avidrez, Z. R., *"Computational modeling of Neurons involved in fly motion detection"*, Master's thesis University of Arizona 2005.

[2] Black, M. J., and P. Anandan, *"A framework for the robust estimation of optical flow"*, In Proceeding of International Conference on Computer Vision, pp. 231-236, 1993

[3] Blake and Isard, "*Active Contours*", Springer Verlag, London Limited, ISBN 3-540-76217-5, 1998.

[4] Bobick, A., and J. Davis. "*The Recognition of Human Movements Using Temporal Templates*", IEEE Trans. on Pattern Anal. Machine Intelligence Vol 23, No. 3, March 2001.

[5] Burges, C. J. C., *"A tutorial on support vector machines for pattern recognition*", Data Mining and Knowledge Discovery, vol. 2, pp. 121 – 167, 1998

[6] Cedras, C., and M. Shah, "*Motion-based Recognition: A Survey*", Image Vision Computing 13 (2)1995) pages 129-155.

[7] Cutler, R., and L. Davis. "*Robust Real-time Periodic Motion Detection, Analysis, and Applications"*, IEEE Trans. on Pattern Anal. Mach.Intell. Vol 22, No. 8, 2000, 781-796.

[8] Fukunaga, "*Introduction to statistical pattern recognition*", 2nd Edition, 1990 ISBN 0-12-269851-7

[9] GVF Snake program website http://iacl.ece.jhu.edu/projects/gvf

[10] Gennert, M. A., and S. Negahdaripour, *"Relaxing the Brightness constancy Assumption in computing optical flow"*, MIT A.I.Lab Memo No. 975 June 1987

[11] Goldenberg, R., R., Kimmel, E. Rivlin and M. Rudzsky. "*Dynamism of a 'Dog on a Leash' or Behavior Classification by Eigen-decomposition of Periodic Motions*", in Proc. of 2005 European Conference on Computer Vision, ECCV 2002, pages 461-475, Copenhagen, May 2002.

[12] Gonzalez, R. C., & R. E. Woods, "*Digital Image processing*", 2nd Edition, Pearson Education, ISBN 81-7808-629-8, 2002.

[13] Gonzalez, R. C., R. E. Woods, and S. L. Eddins, "*Digital Image Processing Using Matlab*" Pearson Education, ISBN 7-5053-9876-8, 2004.

[14] Gunn, S. R., *"Support Vector machines for classification & regression*" Technical report University of Southampton

[15] Hassenstein, B., and W. Reichardt, "*Systemtheorische analyse der Zeit-, Reihenfolgen-und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers Chlorophanus. Zeitschrift für Naturforschung*" 11b, 513–524, 1956.

[16] Haykin, S., "*Neural Networks: A comprehensive foundation"* Chapter 6, 2nd Edition, 1999 Prentice Hall, Inc. ISBN 0-13-273350-1.

[17] Higgins, C., J. Douglass, and N. Strausfeld, "*The computational basis of an identified neuronal circuit for elementary motion detection in dipterous insects*". Visual Neuroscience (2004), 21, pp. 567- 586

[18] Horn, B. K. P., and B. G. Schunck, "*Determining optical flow"*. Atrtificial Intelligence Vol 17 1981

[19] Javed, O., and M. Shah. "*Tracking and Object Classification for Automated Surveillance"*, in Proc. of 2005 European Conference on Computer Vision, ECCV 2002, pages 343-357 Copenhagen, May 2002.

[20] Kass, M., A. Witkin and D. Terzopoulos, "*Snakes: Active Contour Models*", International Journal of Computer Vision, 321 – 331 (1988)

[21] Kim, Y., A. Martinez and A. Kak, *"A local Approach for robust optical flow estimation under varying illumination*", in the Proceedings of the British Machine Vision Conference 2004.

[22] Laughlin, S. B.,"*Visual motion: Dendritic integration makes sense of the world*", Current Biology 9:R15-R17, Elsevier Science Ltd ISSN 0960-9822, 1999.

[23] Lu, Z., W. Xie, J. Pei and J. Huang. "*Dynamic Texture Recognition by Spatio-Temporal Multiresolution Histograms"*, In Proc. of Workshop of Applications of Computer Vision, WACV 2005, pages 241-246, Breckenridge, CO, Jan. 2005

[24] Ogale, N. A., "*A survey of techniques for human detection from video"*, Department of Computer Science, University of Maryland, College Park, MD 20742

[25] Otterloo., P. J., "*A contour Oriented Approach to Shape Analysis*", Prentice Hall International(UK) Ltd. C1991

[26] Tieng, Q. M. and W. W. Boles, "*Recognition of 2D object contours Using the Wavelet Transform Zero crossing Representation"*, IEEE Transactions on Pattern Analysis and Machine Learning, vol. 19(8) Aug. 1997.

[27] Tooth, D. and T. Aach "*Detection and Recognition of Moving Objects using Statistical Motion Detection and Fourier Descriptors*", In Proceeding of the 12th International Conference on Image Analysis and Processing 2003.

[28] Xu, C. and J. L. Prince, "*Gradient Vector flow: A new external force for snakes*" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 66-71.

[29] Xu, C. and J. L. Prince, "*Snakes, Shapes, and gradient Vector Flow"* IEEE Transactions on Image Processing, Vol. 7, No. 3, March 1998

[30] Yang, H. S., S. U. Lee and K. M. Lee. "*Recognition of 2D object Contours Using Starting-Point – Independent Wavelet Coefficients Matching*". Journal of Visual

[31] Zhang, D., "*Image Retrieval Based on Shape*" PhD Thesis March 2002, Information Technology Department, Monash University Australia

[32] Zhang, D. and G. Lu "*A comparative study on shape retrieval using fourier descriptors with different shape signatures*", In Proceedings. of the Fifth Asian Conference on Computer Vision, pp.646-651, Melbourne, Australia, January 22-25, 2002.