Draft Title: Activity Recognition

Sameera Ramasinghe, *Member, IEEE,* John Doe, *Fellow, OSA,* John Doe, *Fellow, OSA,* Ranga Rodrigo, *Member, IEEE,* and Ajith A. Pasqual, *Member, IEEE*

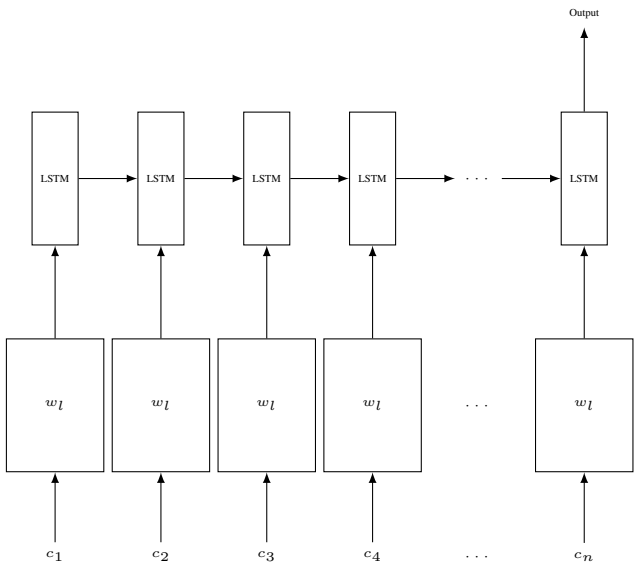


Fig. 1. Test Figure

***Abstract*—This is the abstract ...**

***Index Terms*—Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM).**

I. Introduction

Automatic action recognition, in videos, has caught an extensive research interest recently, in computer vision, mostly due to its wide range of real world applications, such as, sport applications, health care, video surveillance, robot vision, human-computer interaction etc. Furthermore, with the rapid growth of digital video data, in modern world, the need for automatic classification and indexing of video data, has become vital than ever.

Despite the increasing interest, in this field, state-of-the-art systems are still far from human performance, in contrast to image classification [1], [2]. This is partially due to the fact that, videos in nature, consist of complex intra-class variations. Some obvious causes behind this dilemma are, variability of the view point change, background clutter, high dimensionality of data, lack of larger datasets, and low resolution.

But, despite the fact, that most of these issues, up to some extent, also exists in automatic image classification, it has achieved immense success, in recent years, largely owing to the rise of deep learning techniques. Therefore, it is worth to investigate, what is holding back video classification, and find solutions. In this study, mainly, three such key areas are addressed.

A complex activity is typically composed of several sub activities. But, most of the existing accomplished approaches, try to classify a video, treating it as a single, high level activity [3], [4], [5], [6]. As the action becomes complex, the behavior, and the temporal evolution of its underlying sub events, becomes more complicated. For example, cooking may involve sub actions: cutting, turning on cooker, stirring etc. They may not always preserve this same low level order, for same action class. Rather, they may contain a higher order temporal relationship among them. For example, turn on cooker and cutting, may change the order in another video in the same class. Therefore, this temporal pattern of sub events, is not easy to capture through a simple time series analysis. These patterns can only be identified, by observing a large number of examples, using a system, which as an infinite dynamic response. Therefore, we argue that, it is critically important to model this higher order temporal progression, and capture temporal dynamics of these sub events, for better recognition of complex activities.

Recognition accuracy of a complex activity, largely depends on the recognition accuracy of underlying sub events. In contrast to image classification, the information contained in videos, are not in a single domain. Both, the motion patterns of the actors and objects, as well as the static information, such as, background, interactive/still objects, are important for determining an action. For example, the body movements of a group of people fighting, may closely relate with body movements of a sports event. In such a case, it is tough to distinguish between the two activities, solely by looking at the motion patterns. Inspecting the background setting, and objects are crucial in such a scenario. Therefore it is necessary to engineer powerful features, from both motion and static domains. In addition, these features must be complementary, and one feature domain should not influence other domain. In other words, they should be self explanatory, and mutually exclusive, to an utmost level. Also, these motion features should be able to capture activities of each actor independently, so later, the distributional properties of these actions, over shorter and longer time ranges, can be used to identify high level actions. In this proposed method, such a feature crafting mechanism, is provided.

Furthermore, how to combine these descriptors, in two different domains, remains a key challenge. The combined descriptor, should contain the essence of both domains, and should not have a negative interference on each other. Re-

S. Ramasinghe was with the Department of Electronic and Telecommunication Engineering, the University of Moratuwa, 10400, Sri Lanka, e-mail: (see http://www.michaelshell.org/contact.html).

J. Doe and J. Doe are with Anonymous University.

Manuscript received January XX, 201X; revised XXXX 26, 201X.

cently, there has been attempts to answer this aspect [6], [5]. One major requirement, is to have a high level intuitive interpretation, on how much contribution does each domain provide to the final descriptor, and have the authority to control that contribution, in which, we put our special focus. This makes it possible to deeply investigate the the contribution of each domain, for a better accuracy.

In this work, we focus specially on the problematic areas, mentioned above, and provide novel and efficient methods to overcome these issues. In order to examine the temporal evolution of sub activities later, we create video segments with constant overlap. Afterwards, static and motion descriptors are engineered to represent each segment. We propose **motion tubes**, a dense trajectory [3] based tracking mechanism, to identify and track candidate moving areas, separately. This enables independent modeling of the activities, in each moving area. In order to capture motion patterns proficiently, histogram oriented optic flows (HOOF) [7] , are used, inside motion tubes. Then, a bag of words method is applied on the generated features, to capture the distributional properties of micro actions, and create high level, discriminative motion features.

Inspired by the power of CNNs, in the task of object recognition, we create a 7 layer deep convolutional neural net, and pre-train in on the popular Imagenet dataset [8]. Afterwards, this trained CNN is used to create deep features, which are used for creating static descriptors.

Then, a computationally efficient, yet powerful mathematical model is proposed to fuse these two feature vectors. This model provides the ability, to control the contribution of each domain, to the final fused discriptor. Therefore, utilizing this intuition, we investigate the optimum contribution of each domain, for a better accuracy, and solidly justify that, these features are complementory, and contribute to the final accuracy.

In order to model the temporal progression of sub events, the fused vectors are then fed to a LSTM network, where, underlying temporal patterns of the sub events are discovered, and high level actions are classified. We feed the same vectors to a classifier, which does not capture temporal dynamics, and show that, modeling temporal progression of sub events, indeed contribute for a better accuracy.

We carry out our experiments on the two popular action datasets, UCF-11 [9] and Hollywood2 [10]. Our results exceeds existing best state-of-the-art results on these datasets, to the best of our knowledge.

The key contributions of this paper, can be summarized as follows:

- We propose an end to end system, which simultaneously extracts static and motion information, fuse, and models the temporal evolution of sub activities, and does action classification.
- We propose a novel, moving actor/object tracking mechanism, called *motion tubes*, which enables the system to track each actor/object independently, and model the motion patterns individually. This allows the system to model actions, occurring in a video, in micro level, and utilize these learned dynamics at a high level, afterwards.
- We propose a novel, simple and an efficient mathematical model for fusioning two vectors, in two different domains, based on Choleskey transformation. This method provides the ability to govern the contribution of each domain, for the fused vector, precisely in exact numbers, rather than empirically. Utilizing this advantage, we prove that, both static and motion information are complementary and vital for activity recognition, through experiments.
- We model the underlying temporal evolution of sub events, for complex action recognition, using a LSTM network. Through our experiments, we prove that, capturing these dynamics, indeed benefits the final accuracy.
- With the proposed technique, and novel techniques, we outperform the existing best results for the popular datasets UCF-11 [9] and Hollywood2 [10].

II. Related work

There has been not many approaches in activity recognition, which highlight the importance of, exclusively engineering static and motion features. Most of the work relies on generating spatio-temporal interest regions, such as action tubes [11], tubelets [12], dense trajectory based methods [13], [14], spatio-temporal extensions of spatial pyramid pooling [15], or spatioi-temporal low level features [16], [17], [18], [3], [19], [20]. Action tubes in [11] is quite similar to our motion tubes, but our motion candidate regions are chosen, based on more powerful dense trajectories, instead of raw optic flows. Also, we employ a tracking mechanism of each moving area, through motion tubes, isolating actions of each actor, throughout the video. Also, our static interest regions are independent from motion, unlike in [11], where we are able to extract background scenery information, for action recognition. One of the common attributes of these methods, is that, motion density is the dominant factor, for identifying candidate regions, for engineering both motion and static features. In contrast, in this study, motion and static features are treated as two independent domains, and this dominant factor is eliminated.

Recently, few attempts are made on exclusive crafting and late fusion of motion and static features. In [5], a video is first decomposed in to spatial and temporal components, based on RGB and optical flow frames. Two deep ConvNets are then applied on these two components separately, to extract spacial and temporal information. Each network, operates mutually exclusively, and performs action classification independently. Afterwards, softmax scores are coalesced by late fusion.

Work done in [21], is also similar. Instead of late fusion, they fuse the two domains, in a convolutional layer. Both these approaches, rely explicitly on automatic feature generation, in increasingly abstract layers. While this has provided promising results on static feature generation, we argue that motion patterns can be more more superiorly extracted by hand crafted features. Our logic behind this philosophy is, temporal dynamics extend to a longer range, unlike spatial variations. It is not possible to capture and discriminate motion patterns in to classes, by a system, which has a smaller temporal

support. There are models, which employ 3D convolution [22], [23], which extends the traditional CNNs, into temporal domain. [6] applies CNNs on optic flows, and [24] on low level hand-crafted inputs (spatio-temporal outer boundaries volumes), to extract motion features. But even by generating hierarchical features, on top of pixel level features, it is not easy to discriminate motion classes, as it does not extend to a long range. Also, tracking and modeling actions of each actor separately, in longer time durations, is not possible with these approaches. Our motion features, on the other hand, are capable of capturing motion patterns, in longer temporal durations. Furthermore, with the aid of *motion tubes*, our system tracks and models the activities of each moving area separately.

Regarding video evolution, [25] postulate a function, capable of ordering the frames of a video, temporally. They learn a ranking function per a video, using a ranking machine, and use the learned parameters, as a video descriptor. Several other methodologies, for example, HMM ([26], [27]), CRF-based methods ([28]), also have been employed in this aspect. These methods model the video evolution in frame level. In contrast, attempts for temporal ordering of atomic events, also has been carried out, such as [29], [30]. In [29], transition probabilities of a series of events, are encoded statistically with a HMM model. In [30], they identify low level actions, using dense trajectories, and assign concept ID probabilities for each action. Then a LDS is applied on these generated concept vectors, to exploit temporal dynamics of the low level actions. [31] uses simple dynamical systems [32], [33] to create a dictionary of low-level spatio-temporal attributes, and these attributes are later used as a histogram to represent high level actions. Our approach lies closely with these latter mentioned approaches, as we also generate descriptors, for sub events, and then extract temporal progression of these sub events. But, instead of simple statistical models, which has a finite dynamic response, we use a LSTM network [34] to capture these dynamics.

Such models, are starting to make appearance in the literature of action recognition, e.g. in [35] the LSTM network model the dynamics of the CNN activations, and in [36], the LSTM network learns the temporal dynamics of low level features, generated by a CNN.

III. METHOD1

This section outlines our approach. The overall methodology is illustrated in figure

First, a video is segmented in to smaller segments, with a constant frame overlap, and features are engineered for each of these sub segments. We create features, for describing both motion and static domains.

For extracting motion features, we create *motion tubes*, across frames, where we track each moving area along the frames, using action boxes. Action boxes are square shaped regions, which contain significant motion, in each frame. These candidate areas are choosen by, first creating dense trajectories, for each frame, and then clustering trajectory points, after a significant amount of pre processing. This process is explained in detail, in next section. These action boxes are linked across the frames to create motion tubes. Then, histogram oriented optic flows (HOOF) features are calculated within these motion tubes, and a *bag of features* method is applied on these features, to create a descriptor for each video segment.

For extracting static features, we train a deep convolutional neural net, on the popular dataset Imagenet, and then apply this convolutional neural net, on the frames of a video segment, to retrieve deep features from it. Then, these features are used to create a static descriptor for the video segment.

Afterwards, we combine these features, using a mathematical model.

A video can then be represented as a vector time series $C = [c_{t_0}, c_{t_1}, ...c_{t_{n-1}}]$. , where n is the number of segments. Then, a recurrent neural net is applied on these features, and dynamics of time evolution of the combined vector are exploited, by applying a recurrent neural net. Then, by classifying the dynamics of this time series, the actions are predicted.

IV. METHOD2

This section is organied as follows. First, the motion and static vector generation procedure is discussed. Then, our proposed method for fusioning of these two vectors, is explained. Afterwards, the process of exploiting temporal dynamics of sub events, is discussed.

A. Motion Features

1) Low level motion descriptor: First, it is needed to have a pixel level descriptor to identify moving points in the video. Dense trajectories is a powerful pixel level algorithm, which captures the motion, and track it across several frames. Therefore, in this work, dense trajectories is chosen as the low level descriptor, for capturing raw motion.

2) Clustering: Initially, dense trajectories are created for every frame in the video. Then, in order to isolate each sub area in a frame, which contains significant movement, DBScan clustering is applied on the calculated trajectory points. Our clustering algorithm is illustrated in the pseudo code Algorithm 1. Empirically, we use 8 and 10 as epsilon and minpoint parameters.

Even though there are many regions, which contain motion, in a video, some are not significant, nor descriptive. Those moving regions can be neglected without loss of performance. Therefore, after clustering each trajectory point in to cluster groups, non-significant cluster groups are neglected. This is done, in order to prevent the algorithm focusing on small random moving areas in a video, and to only create motion tubes along areas, which are significant and descriptive. Therefore, all the clusters, which are not at least 50% in area size, of the largest cluster of the frame, are discarded.

After identifying the initial candidate clusters, for creating action boxes, further processing is done to each cluster to ensure that they contain only the important moving areas, according to the Algorithm 2. For each point, the chebychev distance, from the centroid of the cluster, is calculated, as in

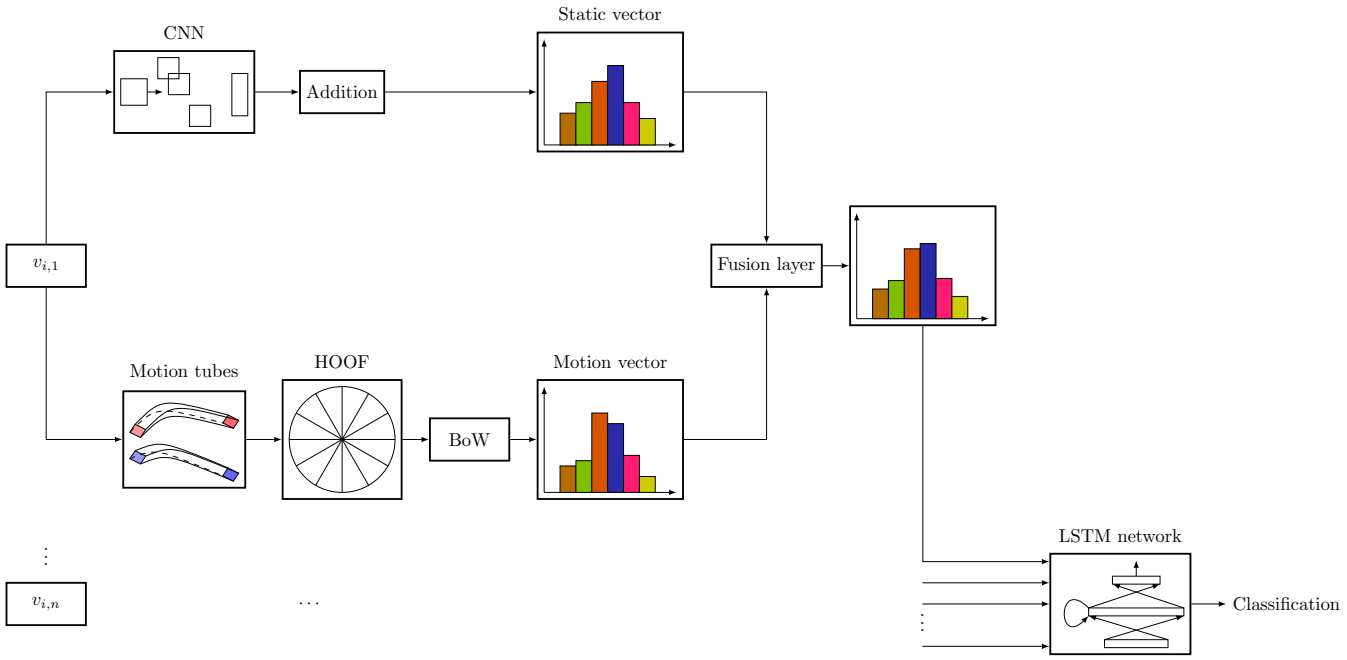


Fig. 2. Test Figure

equation 1, and, furthest 20% of the points, with respect to the total number of points, are discarded from the cluster. The reason behind the choice of chebychev distance, over euclidean distance, is that, a symmetric square shaped cluster groups are obtained, rather than a circle. Which makes it easier to track moving areas, and create motion tubes.

$$D_{chebychev} = max(|X_2 - X_1|, |Y_2 - Y_1|) \tag{1}$$

After identifying square shaped interest regions (action boxes), each of them is represented with a vector $b = (x, y, r, f)$, where, x, y are the coordinates of the top left corner of the box, r is the height/width of the box, and f is the frame number.

3) Motion Tubes: Since our work models the time evolution of sub activities, within a video, we divide each input video V_i into temporal segments $f(V_i) = [v_{i,1}, v_{i,2}, ...v_{i,n}]$, and create features for each individual segment separately. Therefore, after creating the action boxes for each video segment, the action boxes within a segment $v_{i,t}$ can be represented as per equation,

$$\begin{aligned} g(v_{i,t}) = \big\{&[b_{t,1,1}, b_{t,1,2}, ..., b_{t,1,q}], \\ &[b_{t,2,1}, b_{t,2,2}, ..., b_{t,2,p}], ..., [b_{t,n,1}, b_{t,n,2}, ..., b_{t,n,k}]\big\} \end{aligned} \tag{2}$$

Where $b_{t,j,k}$ is the k^{th} action box in j^{th} frame of the t^{th} video segment. Note that the number of action boxes differ from frame to frame.

Therfore, Before linking the action boxes to create motion tubes, further pre processing is needed to ensure same number of action boxes exists in every frame, within a video segment. Otherwise, different motion tubes could become joined halfway though the video, and the effort to capture dynamics of each moving object separately, is disturbed.

First, the rounded down mean number of action boxes, per a frame, is calculated. And we iterate through each frame from frame number 1, until we come to a frame, which has that number of action boxes.

Then from that frame, we propagate forward and back along the frames, either to eliminate, or add action boxes.The procedure is explained below.

If the action box count in a particular frame, is larger than the previous frame, the smallest excess number of action boxes are removed. In the case, where action box count is lower than the previous frame, linear regression for each value x, y and r of vector $b = (x, y, r, f)$, up to that frame, is used to create artificial action boxes, until the number of action boxes match the mean number of action boxes, as in equation 3.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ .. \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ ..... \\ 1 & X_n \end{bmatrix} * \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ .. \\ \epsilon_n \end{bmatrix} \tag{3}$$

Where, Y, X would be either x, y or r. Then least square method is used to minimize the error matrix $\sum \epsilon^2$, and find the values for β_1 and β_0. Afterwards, using these values, missing action boxes are predicted. Note that n increases as the observed action boxes increases, hence the prediction becomes more accurate. Note, after this processing, equation 2 is transformed in to equation 4, which illustrates that the number of action boxes per frame is equal for all the frames, within a video segment.

$$\begin{aligned} h(g(v_{i,t})) = \big\{&[b_{t,1,1}, b_{i,1,2}, ..., b_{t,1,k}], \\ &[b_{t,2,1}, b_{t,2,2}, ..., b_{t,2,k}], ..., [b_{t,n,1}, b_{t,n,2}, ..., b_{t,n,k}]\big\} \end{aligned} \tag{4}$$

The following procedure is used to link the action boxes in consecutive frames.

Algorithm 1

function DBSCAN($Dataset, epsilon, MinPoints$)
  $Cluster = 0$
  for *each* P in $Dataset$ **do**
    if P is *visited* **then**
      continue
    end if
    mark P as *visited*
    $NeighborPts = getAllPoints(P, epsilon)$
    if $size(NeighborPts) < MinPoints$ **then**
      mark P as $NOISE$
    else
      $Cluster$ = next cluster
      $addToCluster(P, NeighborPts, Cluster, epsilon, MinPoints)$
    end if
  end for
end function
function ADDTOCLUSTER($P, NeighborPts, Cluster, epsilon, MinPoints$)
  $addPtoclusterCluster$
  for *each point* np in $NeighborPts$ **do**
    if P is *not visited* **then**
      mark np as *visited*
      $NeighborPts' = getAllPoints(np, epsilon)$
      if $size(NeighborPts') >= MinPoints$ **then**
        $NeighborPts = NeighborPtsjoinedwithNeighborPts'$
      end if
    end if
    if np is *not* yet member of any cluster **then**
      add np to cluster $Cluster$
    end if
  end for
end function
function GETALLPOINTS($P, epsilon$)
  $return$ all points within P's $epsilon$-neighborhood (including P)
end function

Algorithm 2

function FORMATCLUSTER($inCluster, maxChebichevDistance$)
  $maxD_{chebichev}$ = $maxChebichevDistance$
  $totalPoints$ = $getTotalPoints(inCluster, maxD_{chebichev})$
  $currentPoints$ = $totalPoints$
  while $true$ **do**
    if $number_of_count_currentPoints < number_of_count_totalPoints * 80/100$ **then**
      return $allthepointsbellongtocurrentPoints$
    end if
    $maxD_{chebichev}$ = $maxD_{chebichev}$ - 1
    $currentPoints = getTotalPoints(inCluster, maxD_{chebichev})$
  end while
end function

Assume $b_{t,k,1}$,$b_{t,k,2}$,...,$b_{t,k,n}$ and $b_{t,k+1,1}$,$b_{t,k+1,2}$,...,$b_{t,k+1,n}$ are action boxes in two consecutive frames at time $k, k+1$. Then, following distance matrix is calculated.

$$D = \begin{bmatrix} D_{11} & D_{12} & D_{13} & \dots & D_{1k} \\ D_{21} & D_{22} & D_{23} & \dots & D_{2k} \\ \dots & \dots & \dots & \dots & \dots \\ D_{k1} & D_{k2} & D_{k3} & \dots & D_{kk} \end{bmatrix} \tag{5}$$

Where, $D_{i,j}$ is the euclidean distance, between the centroids of i^{th} action box in k^{th} frame and j^{th} action box in $(k+1)^{th}$ frame. Then, u^{th} action box at $k+1$, and 1^{st} action box at k is linked, where u is calculated using,

$$u = \underset{j \in J}{argmin_j}\{D_{1,j}\}, J = \{1, 2, ..., l\} \tag{6}$$

Then, the 1^{st} row and the u^{th} column is removed from the distance matrix, and the same process is applied repeatedly to link each of the action boxes at k with $k+1$. By this removal process, we avoid combining of motion tubes in halfway through the video segment, and keep them isolated from each other, which is vital to capture the dynamics separately, for each moving object.

Finally, we create matrix M_i which encodes all the information of motion tubes, in a particular video segment.

$$M = \begin{bmatrix} 1 & 1 & x_{1,1} & y_{1,1} & r_{1,1} \\ 1 & 2 & x_{1,2} & y_{1,2} & r_{1,2} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & n & x_{1,n} & y_{1,n} & r_{1,n} \\ 2 & 1 & x_{2,1} & y_{2,1} & r_{2,1} \\ 2 & 2 & x_{2,2} & y_{2,2} & r_{2,2} \\ \dots & \dots & \dots & \dots & \dots \\ 2 & n & x_{2,n} & y_{2,n} & r_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ z & 1 & x_{z,1} & y_{z,1} & r_{z,1} \\ z & 2 & x_{z,2} & y_{z,2} & r_{z,2} \\ \dots & \dots & \dots & \dots & \dots \\ z & n & x_{z,n} & y_{z,n} & r_{z,n} \end{bmatrix} \tag{7}$$

Where, columns represent, the frame number, action box number, x coordinate of the top left corner, y coordinate of the top left corner, and the width/height of the action box, respectively.

4) Histogram Oriented Optic Flows(HOOF): Since each action box, in a particular motion tube, may differ in size, we take $R = max(r_i)$, for $\forall i$ where r_i is the i^th action box of the motion tube. Then we redraw the action boxes around thier centroids, having width/length as R. After identifying the k number of motion tubes (k is a variable) for each video segment $v_{i,n}$, the optic flows along each motion tube, , are calculated, using Lucas *et. al*. After that, Histogram-Oriented-Optic-Flows, for each motion tube is created. We create HOOFs for every action box within a motion tube. Each optic flow vector within a spatio-temporal action box within a motion tube, is binned according to its primary angle from the horizontal axis and weighted according to its magnitude. Therefore, all optical flow vectors, $z = [x, y]^T$ with direction, $\theta = tan^{-1}(\frac{x}{y})$ in the range,

$$-\frac{\pi}{2}+\pi\frac{b-1}{B}\leq\theta<-\frac{\pi}{2}+\pi\frac{b}{B} \tag{8}$$

will contribute by a weight of $\sqrt{x^2+y^2}$ to the sum in bin b, $1\leq b\leq B$ out of a total of B bins. Finally, the histogram is normalized. We choose 100 as the bin number.

5) Bag of HOOFs: We use a bag of features method to create a motion descriptor for each video segment. First, we create a code book for HOOF vectors. 100000 vectors are randomly selected, from all the HOOF vectors, of all the video segments, in all video classes. Then these 100000 vectors are clustered using k-means clustering, and 1000 cluster heads are identified. We choose the number of cluster heads as 1000, because, the dimensions of final motion descriptors are needed to be same as static descriptors, which is explained later. Then for each video segment $v_{i,n}$, a histogram is calculated as following.

for each,

$$k\in\{1,2,.....l\} \tag{9}$$

we calculate,

$$p=\underset{j\in J}{argmin_j}(T_j-h_{n,k}), J=\{1,2,...,1000\} \tag{10}$$

Where, $h_{n,k}$ is the k^{th} HOOF vector of the n^{th} video segment, and T_j is the j^{th} cluster head. And then, histogram values are incremented as in,

$$H_n(p)=H_n(p)+1 \tag{11}$$

where $H_n(p)$ is the p^{th} value, $1<p<1000$, of histogram of n^{th} video segment $v_{i,n}$. After calculating the histogram vector H_n, for every video segment $v_{i,n}$, this $H=[H_1,H_2,.....H_n]$ is the vector time series, which encodes the time evolution of motion information, in the video.

B. Static Features

In order to create static descriptors, we create a CNN of 1000 output classes, and train it on image-net dataset. The architecture is shown in figure. After 30 epochs, the training is stopped and used on each individual frame of each video segment. Then, the output vectors of the CNN, are averaged along indexes, and a static descriptor s_i is obtained foe each video segment v_i. Following the procedure for every v_i, finally, we have a vector time series $S=[s_1,s_2,...s_n]$, representing the static time evolution of the whole video.

C. Fusioning of static and motion features

This work strongly relies on the philosophy, that both static and motion information are vital for action recognition. Therefore, the following mathematical models is used to fuse the static and motion vectors. We do our mathematical derivation based on the Cholesky transformation. An abstract version of Cholesky transformation is described next.

Let P and Q be two random variables of unknown correlation. These random variables can be transformed into two new random variables (R and S) with a known correlation of ρ, where the value of ρ can be chosen at will. The transformation can be performed as follows.

$$\begin{bmatrix}R\\S\end{bmatrix}=\begin{bmatrix}1&0\\\rho&\sqrt{1-\rho^2}\end{bmatrix}*\begin{bmatrix}P\\Q\end{bmatrix}$$

Therefore,

$$R=P \tag{12}$$

$$S=\rho P+\sqrt{1-\rho^2}Q \tag{13}$$

The Cholesky transformation guarantees that the correlation between the two random variables R and S is ρ.

Based on the above properties of the Cholesky transformation, we propose the following methodology to fuse the static and motion vectors.

Let X and Y be static and motion vectors, respectively. Cholesky transformation can be applied to the two vectors X and Y, with the correlation value ρ_1.

$$\begin{bmatrix}R\\S\end{bmatrix}=\begin{bmatrix}1&0\\\rho_1&\sqrt{1-\rho_1^2}\end{bmatrix}*\begin{bmatrix}X\\Y\end{bmatrix}$$

$$R=X \tag{14}$$

$$S=\rho_1 Y+\sqrt{1-\rho_1^2}Y \tag{15}$$

Similarly, the transformation can be applied to Y and X, with the correlation value ρ_2.

$$\begin{bmatrix}A\\B\end{bmatrix}=\begin{bmatrix}1&0\\\rho_2&\sqrt{1-\rho_2^2}\end{bmatrix}*\begin{bmatrix}Y\\X\end{bmatrix}$$

$$A=Y \tag{16}$$

$$B=\rho_2 X+\sqrt{1-\rho_2^2}X \tag{17}$$

Again, the Cholesky transformation guarantees the follwing two properties.

1) The correlation between X and S is ρ_1.
2) The correlation between Y and B is ρ_2.

Therefore, if the values of ρ_1 and ρ_2 are chosen in such a way that they obey the following rule,

$$\rho_2=\sqrt{1-\rho_1^2} \tag{18}$$

It can be guaranteed that $S=B$, $\forall X,Y,\rho_1,\rho_2$. Hence, the resultant vector Z can be obtained by,

$$Z=S=B \tag{19}$$

Where the correlation between Z and X is ρ_1 and the correlation between Z and Y is ρ_1. This derivation leads us to an important intuition. Here X and Y represent the static and the motion vectors whereas Z represents the resultant vector. By choosing the value of ρ_1, we can choose the degree, in which the static features and the motion features contribute, in deriving the resultant vector. In section 4, it is shown, how this property is used to explore, the optimal contribution of static and motion domain information, for recognizing actions.

D. Capturing temporal evolution

In our work, a video is represented as n fixed-length segments (n differ for videos of different lengths), with over-lapping frames. Each segment is represented with a fusioned vector c_t. Therefore, each video can be represented as a vector time series $C = [c_{t_0}, c_{t_1}, ...c_{t_{n-1}}]$.

Now, each vector time series could be analyzed using traditional time series modeling techniques, such as Auto Regressive Moving Average [1] etc. Where it is possible to obtain features or model parameters to describe the vector time series. But the main drawback of these methods is, they model current values of a series as a function of past values, and have finite dynamic response to time series input. Also they lack the ability to grasp the internal state representations of a complex time series. Recurrent neural nets, on the other hand, maintain hidden layers with directed feedback connections, and hence, have an infinite dynamic response. While training, it learns internal states of a sequence, and are usually performed better in modeling complex dynamic temporal patterns of long sequences.

But, it is not ideal to train standard RNNs to solve problems, which require, learning of long-term temporal dependencies. This is because of the vanishing-gradient problem, which occurs due to the exponential decay of gradient loss of the function, with respect to time.

In practice, LSTM(Long short term memory) networks, typically perform better in such cases. LSTM networks are a special type of RNN, which include a 'memory cell', and as the name suggests, it can maintain a certain state, in memory, for longer periods of time. Also, it has a set of control gates, which is a structured way of controlling removal or addition information to the cell state. This special architecture gives them the ability to capture longer-term dependencies. In the next section, the operation of a LSTM network is revisited.

The most important structure of a LSTM unit is its memory cell C_t, which preserves the state. Basic structure of a LSTM unit is shown in figure. The memory cell is self connected, and it has three gates(multiplicative units), i.e. input gate, forget gate and output gate, which are used to control, how much to store, remove or output long range contextual information of a temporal sequence.

The detailed activation process of the memory cell and three gates are illustrated as follows:

$$i^t = \sigma(W_{xi}x^t + W_{hi}h^{t-1} + W_{ci}c^{t-1} + b_i) \quad (20)$$

$$f^t = \sigma(W_{xf}x^t + W_{hf}h^{t-1} + W_{cf}c^{t-1} + b_f) \quad (21)$$

$$c^t = f^t c^{t-1} + i^t tanh(W_{xc}x^t + W_{hc}h^{t-1} + b_c) \quad (22)$$

$$o^t = \sigma(W_{xo}x^t + W_{ho}h^{t-1} + W_{co}c^{t-1} + b_o) \quad (23)$$

$$h^t = o^t tanh(c^t) \quad (24)$$

Where W is the connection weight between two units and $\sigma(.)$ is the sigmoid function.

Since the LSTM network is used only for capturing the temporal dynamic patterns between sub actions, one LSTM layer is enough. Our LSTM network is shown in figure. As it is seen, the network consists of an input layer, a 128 unit LSTM layer with 0.8 dropout, and a fully connected softmax output layer. As we have a sequence of activites per one classification, we use many to one approach, for feeding the fusioned vectors to the network, as shown in figure.

V. Experiments and Results

This section details our experimental methodology and the video datasets used. We evaluate our approach on the two popular datasets Youtube Action Dataset and the Olympic Sports dataset. On both datasets, we show that our work exceeds the current state-of-the-art results, by comparing with other popular techniques. Also, we vary the contribution of static and motion features, for the calculation of combined vector series, and explore what is optimum contribution from each domain. By this step, we also prove that both static and motion features are complementery, and provide vital information about the actions occuring in a video. Furthemore, we highlight the importance of considering the time evolution of sub activities, in order to identify a complex event, by comparing the results of RNN and other methodologies, which does not capture the temporal dynamics.

A. Datasets

Holywood 2: It consists of 12 classes of human actions distributed over 1500 video clips: *Answer phone, drive car, eat, fight person, get out car, hand shake, hug person, kiss, run, sit down, sit up,* and *stand up*. The dataset is composed of video clips from 69 movies and provides a challenging task, in automatic action detection.

UCF-11: It consists over 1000 sports and home videos from YouTube. This dataset contains 11 action classes: *basketball shooting, cycle, dive, golf swing, horse back ride, soccer juggle, swing, tennis swing, trampoline jump, volleyball spike*, and *walk with a dog*. Each of the action sets is subdivided into 25 groups sharing similar environment conditions. This also is a challenging dataset with camera jitter, cluttered backgrounds and variable illumination.

B. Contribution of Static and Motion domains

The derivation done in section 4, for fusioning the static and motion vectors, provides us a insightful intuition. That is, we can controll the contribution of motion and static domains, to the fusion vector, by varying the ρ value. The derivation of ρ values for different contribution ratios, is illustrated in V-B

Results, for these ρ values, for UCF-11 and Hollywood2 datasets, are shown in table and table, respectively. We use accuracy and mean average precision as performance matrics, for UCF-11 and Hollywood2, respectively.

Also, an overview distribution of the overall performance, over ρ values, for both datasets, is shown in table. We can see that, the performance change, for different contribution precentages of motion and static domain. Also, the optimum contribution may change depending on the nature of the data. For example, if the motion patterns are indistinguishable accross actions, static information plays a critical role, for determining the action, and vise versa. This highilghts our hypothesis, that being able to control this contibution explcitly, is vital for an action recognition system.

Contribution to Z	**ρ value**	**Fusion vector**
80% Motion, 20% Static $\rho_1 = 4\rho_2$	$\frac{1}{4}\rho_1 = \sqrt{1-\rho_1^2}$ $\rho_1 = \frac{4}{\sqrt{17}}$	$Z = \frac{4}{\sqrt{17}}M + \frac{1}{\sqrt{17}}S$
60% Motion, 40% Static $2\rho_1 = 3\rho_2$	$\frac{2}{3}\rho_1 = \sqrt{1-\rho_1^2}$ $\rho_1 = \frac{3}{\sqrt{13}}$	$Z = \frac{3}{\sqrt{13}}M + \frac{2}{\sqrt{13}}S$
40% Motion, 60% Static $3\rho_1 = 2\rho_2$	$\frac{3}{2}\rho_1 = \sqrt{1-\rho_1^2}$ $\rho_1 = \frac{2}{\sqrt{13}}$	$Z = \frac{2}{\sqrt{13}}M + \frac{3}{\sqrt{13}}S$
60% Motion, 40% Static $4\rho_1 = \rho_2$	$4\rho_1 = \sqrt{1-\rho_1^2}$ $\rho_1 = \frac{1}{\sqrt{17}}$	$Z = \frac{1}{\sqrt{17}}M + \frac{4}{\sqrt{17}}S$

TABLE I
DERIVATION OF CONTRIBUTION OF STATIC AND MOTION DOMAINS TO THE FUSION VECTOR

TABLE II
MY CAPTION

Class	rho	rho	rho	rho	rho
B shooting					
Biking					
Diving					
G swinging					
H riding					
S juggling					
Swinging					
T swinging					
T jumping					
V spiking					
W dog					
Overall Accuracy					

C. Comparison with the state-of-the-art

able 2 compares our results to state of the art. $\rho = \frac{1}{\sqrt{2}}$ is used to combine the static and motion vectors, since it gave the best results, for this dataset. On UCF-11, we significantly outperform the state of the art [] by %. A mean average precision of % is achieved by our system, which outperforms the state-of-the-art by %.

Per action class results, are also compared in table and table. In UCF-11, our method excells in 8 out of 11 classes, when compared with [], [], [] and [] We only fall behind in asd and asda classes. In Hollywood2, we calculate the average precision of each class, and compare with [], [], [] and []. We achieve best results in 8 out of 12 classes in this case.

TABLE III
MY CAPTION

Class	rho	rho	rho	rho	rho
AnswerPhone					
DriveCar					
Eat					
FightPerson					
GetOutCar					
HandShake					
HugPerson					
Kiss					
Run					
SitDown					
SitUp					
StandUp					
mAP					

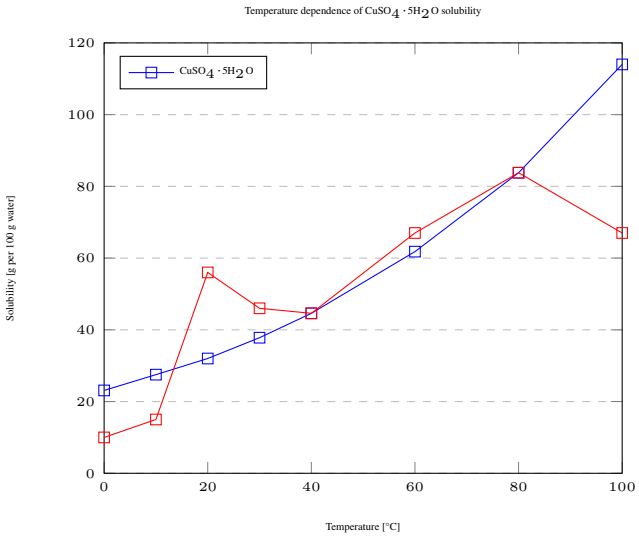
TABLE IV
MY CAPTION

UCF-11		Hollywood2	
Liu et al.	71.2%	Vig al.	59.4%
Ikizler-Cinbis et al.	75.21%	Jiang et al.	59.5%
Dense trajectories	84.2%	Mathe al.	61.0%
Sameera et al		Jain al	62.5%
		Dense	58.3%
		Improved	64.3%
Our method		Our method	

D. Effectiveness of Capturing Time Evolution

As discussed in earliar sections, complex actions are composed of sub activities, preserving a temporal pattern. In this work, we try to capture those underlying patterns, by a LSTM netwok. It is interesting to verify, whether this stratergy has an impact on the accuracy of the classification. Here, we directly feed the fused vectors to a random forest classifier, which does not capture sequencial dynamic patterns, and compare it with the results obtained by the LSTM network. The results are shown in table

As indicated by the results in chart, LSTM network significantly outperforms the random forest classifier, for both datasets. Therefore, it can be concluded that, exploiting temporal patterns of sub activities, benifits complex action classification.



VI. CONCLUSION

The conclusion goes here. This paper presents an end to end system, for action classification, which operates on both static and motion features. Our approach relies on deep features, for creating static vectors, and *motion tubes* for motion features. *motion tubes* are a novel concept, we introduce in this paper, which can be used to track individual actors/objects across frames, and model micro level actions. We present three novel methods: Cholesky transformation,aas,asd, for efficient combining of features, from different domains, which is a vital requirement in action classification. Additionally, Cholesky provide the power to control the contribution of each domain, in exact numbers. We utilize this ability, and run experiments

TABLE V
MY CAPTION

Class	Ours	KLT	Wang et al.	kizler-Cinbis	Sameera et. al.
B shooting					
Biking					
Diving					
G swinging					
H riding					
S juggling					
Swinging					
T swinging					
T jumping					
V spiking					
W dog					
Overall Accuracy					

TABLE VI
XSFSDFSDF

Class	Ours	KLT	Wang et al.	Ullah
AnswerPhone				
DriveCar				
Eat				
FightPerson				
GetOutCar				
HandShake				
HugPerson				
Kiss				
Run				
SitDown				
SitUp				
StandUp				
mAP				

to show that, optimum contribution of static and motion domains, may vary, depending on the scenario.

Through our experiments, we show that, our static and motion features are complementory, and contribute to the final result. We also compare our three fusion algorithms, and show that choleky, is superioir, although all three of them gives impressive results.

we also model the temporal progression of subevents, using an LSTM network. Experimental results indicate that this is indeed benificary, compared to a flat temporal structure.

Comparison of our work with mutiple state-of-the-art algorithms, on the two popular datasets, UCF-11 and Hollywood2, conclude the superiority of our proposed system.

In the future, it is interesting to improve the motion tubes, so that, it can maintain an identity over each actor object. While, even at the present state, it is mostly the case, there is no mathematical guarentee for that. Also explore more powerful methods to describe micro actions, inside motion tubes.

APPENDIX A
PROOF OF THE FIRST ZONKLAR EQUATION

Appendix one text goes here.

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENT

The authors would like to thank the National Research Council of Sri Lanka for support received under the Grant NRC/XX

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3169–3176.

[4] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.

[5] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[6] S. Ramasinghe and R. Rodrigo, "Action recognition by single stream convolutional neural networks: An approach using combined motion and static information," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov 2015, pp. 101–105.

[7] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1932–1939.

[8] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei, "Imagenet large scale visual recognition competition 2012 (ilsvrc2012)," 2012.

[9] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos in the wild," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1996–2003.
[10] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2929–2936.
[11] G. Gkioxari and J. Malik, "Finding action tubes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 759–768.
[12] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek, "Action localization with tubelets from motion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 740–747.
[13] J. van Gemert, M. Jain, E. Gati, and C. Snoek, "Apt: Action localization proposals from dense trajectories," in *BMVC*, vol. 2, 2015, p. 4.
[14] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4305–4314.
[15] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.
[16] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3. IEEE, 2004, pp. 32–36.
[17] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1. IEEE, 2005, pp. 166–173.
[18] E. Shechtman and M. Irani, "Space-time behavior based correlation," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 405–412.
[19] A. Klaser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *BMVC 2008-19th British Machine Vision Conference*. British Machine Vision Association, 2008, pp. 275–1.
[20] T.-H. Yu, T.-K. Kim, and R. Cipolla, "Real-time action recognition by spatiotemporal semantic and structural forests." in *BMVC*, vol. 2, no. 5, 2010, p. 6.
[21] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *arXiv preprint arXiv:1604.06573*, 2016.
[22] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.
[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 4489–4497.
[24] H.-J. Kim, J. S. Lee, and H.-S. Yang, "Human action recognition using a modified convolutional neural network," in *International Symposium on Neural Networks*. Springer, 2007, pp. 715–723.
[25] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5378–5387.
[26] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1310–1323, 2011.
[27] D. Wu and L. Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 724–731.
[28] Y. Song, L.-P. Morency, and R. Davis, "Action recognition by hierarchical sequence summarization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3562–3569.
[29] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele, "Script data for attribute-based recognition of composite activities," in *European Conference on Computer Vision*. Springer, 2012, pp. 144–157.
[30] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, and M. Shah, "Recognition of complex events: Exploiting temporal dynamics between underlying concepts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2235–2242.
[31] W. Li, Q. Yu, H. Sawhney, and N. Vasconcelos, "Recognizing activities via bag of words for attribute dynamics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2587–2594.
[32] E. A. Jackson, *Perspectives of nonlinear dynamics*. CUP Archive, 1992, vol. 1.
[33] T. Kailath, "A view of three decades of linear filtering theory," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 146–181, 1974.
[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[35] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.
[36] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

PLACE PHOTO HERE

Sameera Ramasinghe Biography text here.

PLACE PHOTO HERE

Michael Shell Biography text here.

PLACE PHOTO HERE

Michael Shell Biography text here.

PLACE PHOTO HERE

Ranga Rodrigo Biography text here.

Ajith A. Pasqual Biography text here.