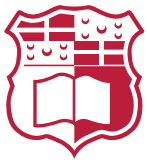
Understanding Activity in Private and Public Setups using 3D Video Content

Alexander Gutev

Supervisor: Prof. Carl James Debono

September 2024

Submitted in partial fulfilment of the requirements
for the degree of Ph.D. in Communications and Computer Engineering.



L-Università ta' Malta
Faculty of Information &
Communication Technology

Abstract

Human action recognition (HAR), which deals with classifying human action in video, is a core component of behaviour monitoring which has found applications in surveillance, security, sports, traffic management, medical monitoring and assisted living systems.

The state of the art in human action recognition depends on machine learning techniques, which require large datasets for training. This results in huge amounts of time spent in training as well as a large power consumption, which reduces the feasibility of adopting such methods in real-world applications. Most of the data present in video consists of background clutter that is irrelevant to human action classification. Thus, the training time can be significantly reduced by limiting the data that is processed to only those regions that capture the action taking place. A viable strategy for identifying these regions with low effort is motion saliency detection given that human action necessitates motion.

The main contributions of this work include a novel solution for identifying regions that capture human actions and a new HAR method that uses this solution to achieve a classification accuracy that is comparable to the state of the art however with a significant reduction in the time spent in training and inference. In this thesis various solutions to motion saliency detection were explored. A new motion saliency solution was developed as existing solutions were found to be too computationally intensive for any reduction in training time to be realised by their adoption in a HAR pipeline. The use of this motion saliency solution in HAR methods was explored. It was found that the highest classification is achieved with the Model-based Multimodal Network (MMNet) [1] method, which is a multimodal HAR method that fuses the classification results of the skeleton and colour modalities. A new HAR method, MMNet with motion saliency (MMNet-MS), was developed that is based on the MMNet method. Whilst MMNet relies on the OpenPose [2] tool that estimates skeleton joint coordinates, to identify the regions that are relevant to action classification, the proposed MMNet-MS identifies these regions using motion saliency detection to replace the computationally expensive OpenPose skeleton estimation step. Experimental results showed that the proposed MMNet-MS method achieves a comparable classification accuracy, on average 75.91%, to MMNet, which has an accuracy of 76.67% on average. In private settings such as the TST fall detection dataset [3], the accuracy of the proposed MMNet-MS, 55.69%, surpasses that of MMNet, 29.55%. A significant reduction in training and inference time is achieved where MMNet takes on average 28.63 hours to train and 27 milliseconds to classify an action in a video while MMNet-MS takes on average 14.92 hours to train and 9 milliseconds to classify an action. This reduction in

training time leads to a reduced power consumption in most cases particularly on the NTU-60 dataset where MMNet consumed on average 3.43 kWh during training while MMNet-MS consumed an average of 1.80 kWh during training.

Publications

Some elements of this work have appeared in or will appear in the following publications:

- “Motion saliency detection using depth information for human action recognition applications,” in 2023 International Symposium ELMAR, IEEE, Sep. 2023. [4]
- “Kalman filter aided depth-based motion saliency detection in human activity recognition applications,” in 2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON), IEEE, Jun. 2024. [5]
- “Motion saliency based human action recognition in RGBD videos,” in 2024 IEEE 20th International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE, Oct. 2024. [6]

Contents

List of Figures

List of Tables

1 Introduction

Human action recognition (HAR) can be defined as the task of classifying human actions in video. It is considered as an emerging field that has many applications which include surveillance, security, traffic management, medical monitoring and assisted living systems [7–10]. The goal of all HAR systems is to accurately and effectively analyse human behaviour in video [7]. Once this goal is realised, HAR technology, which forms the core of behaviour monitoring systems, has the potential to reshape many industries and our society as a whole. For example, an effective behaviour monitoring system can be used to automatically detect instances of uncivilised behaviour, examples of which include littering, stepping on a lawn and climbing on tourist attractions, as well as more serious offences such as dangerous driving [7]. This has the potential to not only reduce the burden of human monitoring for such behaviour but also reduces the burden to urban security and environmental maintenance caused by insufficient enforcement [7]. In other applications, behaviour monitoring systems have the potential for a profound impact on the lives of the elderly. While elderly homes provide care services for the elderly, they not only place restrictions on their lifestyle but are also a burden on the state, particularly in the case of public elderly homes, which are subsidised by the government [8]. Assisted living technology can allow the elderly to live, independently, in their own homes until a later age, thus not requiring the services of elderly care homes [8]. Whilst applications related to surveillance and security are focused on public settings where there are few restrictions on the video capturing equipment, assisted living applications are focused on private settings where the use of colour video introduces privacy concerns [8, 11]. As a result, colour video is generally not available in such applications and HAR systems are limited to depth and skeleton information only [3]. The application of HAR is not only limited to surveillance and assisted living applications but is attracting increasing interest in sports where HAR technology is used to monitor athletes for the purpose of analysing their skill and performance [12].

Despite the tremendous amount of research over the past years, HAR systems still face significant challenges to successful action recognition. One of the most significant challenges to the practical adoption of HAR systems is the long time required for training and inference as well as the expensive hardware requirements [13]. In order to maintain high classification accuracy while ensuring prompt recognition, HAR models are deployed on high performance cloud services [14]. Besides the increased cost of the computing resources, this setup requires a continuous high-quality and low-latency network connection which is not always available [14]. In some applications, for example autonomous vehicles and automotive

safety systems, local processing is required, where far fewer computing resources are available, for the low-latency requirements to be met [14]. As a result lower accuracy models are generally deployed in such applications [14]. If the time spent in inference can be reduced with a minimal cost to classification accuracy, there is a potential for deploying higher accuracy models in applications where local-processing is required and computing hardware is limited [14]. Furthermore, a reduction in the computational time has the potential for a reduction in the power consumption of high-performance cloud services [14].

The long training and inference times are caused by the large amount of data present in video that has to be processed by the classifier [15]. Most of this data consists of background clutter that is actually irrelevant to action classification [15]. Moreover, the presence of this data can also have a negative effect on the classification accuracy [1].

This thesis details research into the state of the art in human action recognition, along with the challenges faced by modern approaches to the problem. It is identified that the large training sets, required to effectively train a HAR classifier, lead to long training times as well as large power consumptions, which hinders the practical adoption of HAR systems. It is further identified that the training time and power consumption can be potentially reduced by limiting the data that is processed to only those regions of the video that capture the action taking place. Various strategies for reducing the size of the data that is processed by the HAR classifier are explored in order to reduce the time spent during training and inference while preserving the same classification accuracy. This is done with the goal of increasing the feasibility of adopting HAR systems in real world scenarios. Since the presence of action necessitate that motion is taking place, motion saliency detection is explored as the core strategy for reducing the data that is processed by the HAR classifier to only those regions capturing the action that is taking place.

The aim of this work is to develop a new HAR method, by building on existing methods, that achieves a comparable classification accuracy to existing methods but requires less time to train, is able to classify actions in video accurately in less time and consumes less power. The primary objective is to develop a HAR solution that relies on less data by identifying the regions in which the action is taking place. The second objective is to develop a solution for identifying these regions that can be incorporated in a HAR pipeline. The strategy is to reduce the size of the data that is processed by the HAR classifier by eliminating those regions of the video that are irrelevant to action classification. Thus, the negative impact on classification accuracy is expected to be minimal if any is observed at all while the time taken during training and inference is expected to be reduced significantly. The third objective is to optimise the HAR pipeline, particularly in the steps where the action regions are identified, such that the

training time, inference time and power consumption is reduced. The fourth objective is to determine the optimal parameters of the developed HAR solution that the classification accuracy is comparable to existing approaches with the final objective being to evaluate the performance of the solution, in terms of speed, computational time and power consumption, while comparing it to state of the art HAR approaches.

Existing solutions to motion saliency detection were explored and were found to be unsuitable for the task since they are either computationally intensive, which negates the reduction in processing time, or require training a classifier in addition to the main HAR classifier. Thus, a new motion saliency solution was developed for use in a HAR pipeline with the purpose of reducing the size of the data that is processed by the classifier. Two papers [4, 5] detailing the novel motion saliency solution were published.

Various recent HAR solutions were explored with extensive experimentation done to determine their performance, the benefit of motion saliency in reducing computational time and the impact on classification accuracy. It was found that the best performance was achieved using multimodal methods, in particular the Model-based Multimodal Network (MMNet) [1], which performs action classification by fusing the results of classification using skeleton joint and colour data. A new HAR system was developed by modifying the MMNet pipeline to include the motion saliency solution, developed during this work. This was done with the objective of replacing the use of the OpenPose [2] tool for estimating skeleton joint coordinates, which is a costly step, with motion saliency detection. Experimental results show that the developed pipeline achieves an action classification accuracy that is comparable to, and in some cases surpasses, MMNet but requires significantly less time for training and inference. Moreover, in most of the experiments the proposed pipeline consumed considerably less power during training. A paper [6] detailing the HAR method and the experimental results was published.

This thesis is organised as follows. Chapter 2 contains a detailed overview of the field of human action recognition, beginning with a brief overview of early work leading to the techniques behind the state of the art. Chapter 3 contains an in-depth review of recent approaches to HAR. Chapter 4 covers the research done into existing motion saliency detection solutions as well as a detailed explanation of the new motion saliency solution, that was developed during this work, and the experimentation done to evaluate its performance. Chapter 5 explores the use of the developed motion saliency solution in HAR methods to reduce the time required for training and the power consumption. This chapter concludes by presenting a new HAR method (MMNet-MS) which is based on MMNet but with a pipeline that is modified to include motion saliency detection. Chapter 6 details the experimentation done to evaluate the performance of the proposed HAR method in terms of classification

accuracy, training time, inference time and power consumption and compares its performance to that of MMNet and other methods. Chapter 7 concludes this thesis with a summary of the research done and the achievements of this work along with a discussion of potential avenues for future research.

2 Background

This section begins with a background on Convolutional Neural Networks (CNNs) and with a justification for why they are the preferred approach in computer vision tasks as well as the challenges faced when applying CNNs to image processing tasks such as object recognition. This is followed by a background on the application of CNNs to human action recognition (HAR) with an in-depth explanation of two-dimensional CNNs (2D-CNNs) and three-dimensional CNNs (3D-CNNs) with the advantages and limitations of both approaches discussed in detail.

Advances in deep-learning models have lead to CNNs, which were first described in [16], being established as the state of the art in most image processing tasks due to their tremendous success in object recognition and image classification [17–20]. CNNs are a variation of the standard feed-forward neural networks that are specifically adapted for processing 2D image data. Before explaining why CNNs are able to achieve a superior performance in image processing tasks, it is necessary to delve deeper into the weakness of standard feed-forward neural networks.

Before the introduction of CNNs, neural networks consisting of multiple fully connected layers were used to apply machine learning to image processing tasks, where each individual perceptron is fed the value of a single image pixel [16]. An overview of this architecture is shown in Figure 2.1. The network architectures of the time, which consisted of one input layer, one hidden layer and one input layer, lacked the discriminative power required to learn features from the large volume of data associated with image processing tasks [16]. However, increasing the number of layers increases the number of parameters to the point where a network with sufficient discriminative power is prohibitively expensive to train due to requiring an enormous set of training data and long training times [16]. Furthermore, avoiding overfitting becomes increasingly difficult with a larger number of parameters [16].

The core component of CNNs, which distinguishes them from their predecessors, is the convolution layer. It was observed that images generally consist of local features which are present in multiple regions of the same image [16]. Moreover, a feature detector which successfully detects a specific set of features, in a certain part of the image can also be used to detect the same feature set in another part of the image [16]. Thus, it is not necessary to dedicate a perceptron to every pixel in the input image but rather a feature detector, with its own set of weights, can be limited to a single block of pixels and applied to multiple regions of the image using the same weights [16]. This feature detector is implemented by a convolution layer which performs convolution of the input image with a 2D filter kernel. The kernel weights are learned using the same backpropagation algorithm that is used to train feed-forward

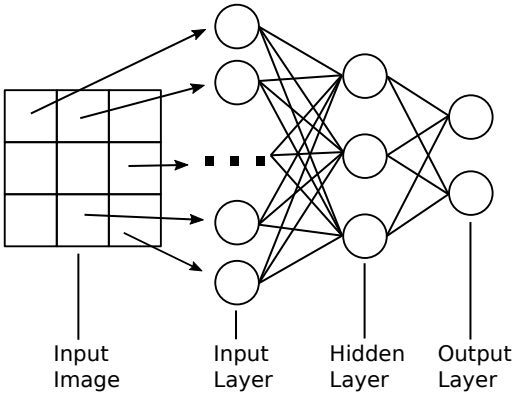


Figure 2.1 Standard feed-forward neural network architecture for image processing

neural networks. Each convolution layer serves to detect and extract a specific set of features using the 2D filter kernel. The initial layers detect local features while the upper layers combine multiple features, detected by the preceding layers, to form a global representation of the image [16]. Each convolution layer reduces the size of the data, effectively acting as a *squashing* operation [16] until the size of the data is reduced to the size of the output feature vector. An overview of the basic structure of a CNN is shown in Figure 2.2.

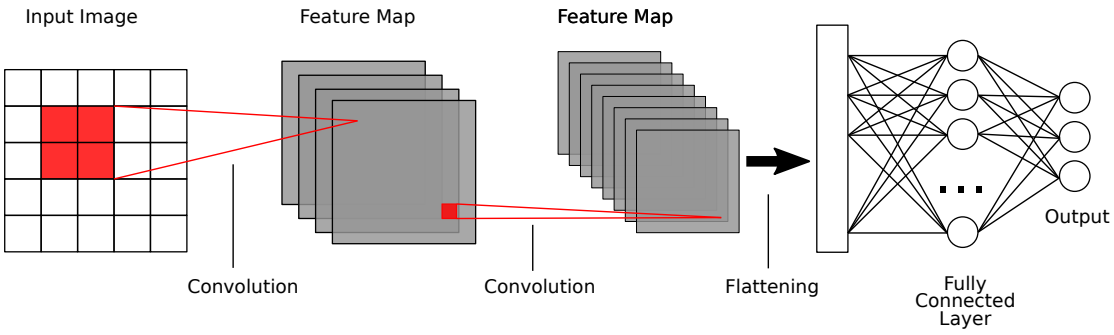


Figure 2.2 Basic structure of a Convolutional Neural Network

The major advantage of CNNs over standard feed-forward neural networks is that the number of parameters, and hence the required training set size, is reduced drastically making CNNs easier to train and more practical for real-world applications [16, 17]. Another significant advantage is that CNNs are shift-invariant, which means that the same result is produced even if the input is shifted [16]. This is attributed to the fact that the convolution kernel, in a given layer, uses the same weights regardless

of the location at which it is applied in the input image [16].

Even with these ground-breaking advances in neural network architectures, obtaining a sufficiently large training set is still a significant challenge [17]. For simple recognition tasks, such as digit and hand-writing recognition, the network can be trained using a dataset containing tens of thousands of annotated samples [17, 21–23]. However, these datasets are too small for object recognition in real-world settings, where the classifier has to distinguish between thousands of categories of objects each with considerable variability [17, 24]. To effectively train a classifier in such scenarios, larger datasets consisting of hundreds of thousands of samples, such as LabelMe [25], to millions of samples, such as ImageNet [26], are required [17]. Training with datasets of this size not only requires a model with a large training capacity but also a model which has prior knowledge to compensate for missing data [17]. These constraints are satisfied by CNNs [16, 22, 27–30], since their training capacity can be controlled by varying the depth of the network, i.e. the number of layers, and the size of the convolution kernels while the inherent kernel weight-sharing strategy means CNNs make correct assumptions about the nature of images, namely the spatial locality of pixel dependencies [17].

The success of CNNs is not limited to object recognition but with recent advancements, CNNs have achieved tremendous success in human action recognition and are considered the state of the art [18–20]. This is largely due to their inherent ability in extracting and combining low-level features as well as the recent availability of large-scale HAR datasets [18, 31, 32].

The challenges faced in training a CNN for object recognition, namely the large training set requirements, are also faced when training a CNN for human action classification but to a greater level due to the increased complexity of the task, the larger volume of data that has to be processed and the need to model temporal information as well as spatial information [18]. Recently a number of large-scale training datasets have emerged such as Kinetics-400 [31], Sports1M [33] and Moments-In-Time [34], which has led to rapid progress in video understanding and has facilitated research into the most optimal spatiotemporal models for video action recognition [18].

The two-stream CNN model is the most widely used deep architecture in video action recognition [18, 35]. This model consists of two separate CNN streams with one stream dedicated to process the colour (RGB) frame information, which contains spatial features, and another stream dedicated to process the temporal information in the form of optical flow [18, 32, 36]. The final action classification is delivered by late fusion of the two-streams in the upper layers [33]. The structure of a 2D-CNN for action classification is shown in Figure 2.3. The distinction between two-stream CNN based approaches to HAR lies in the strategies for identifying the subset of the colour

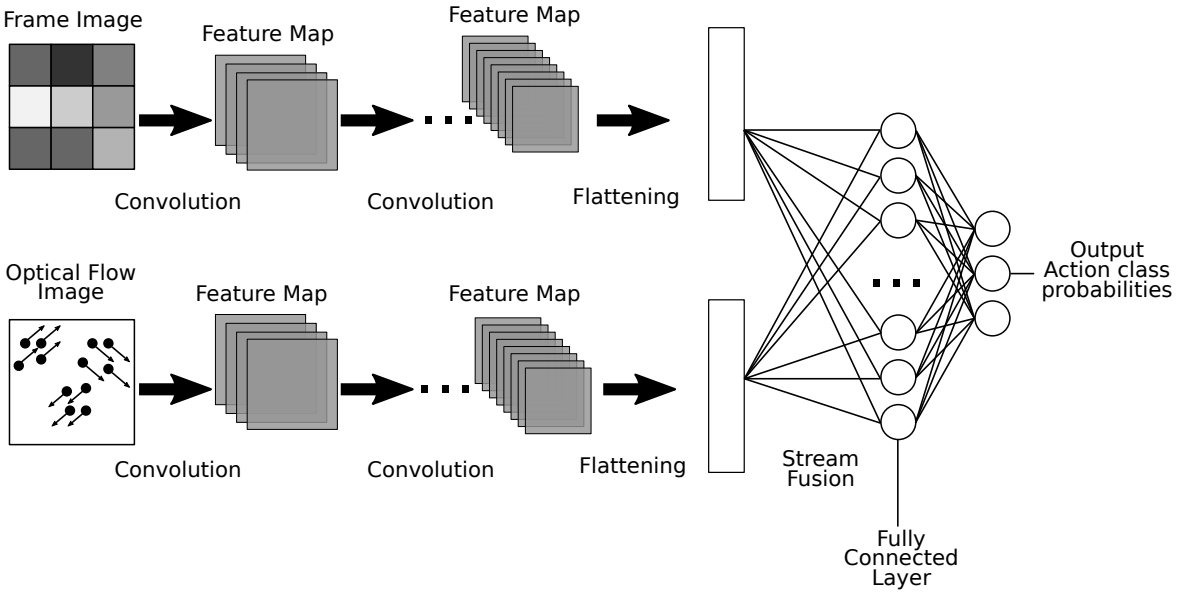


Figure 2.3 General structure of two-stream CNNs for HAR

and optical-flow that is fed into the CNN streams for processing. Ideally only those regions which are relevant to action classification are processed while the remaining data, which mostly consists of clutter, is removed. Pose based approaches such as [37] identify regions in the RGB and optical flow images which surround the individual body parts while separate two-stream CNN models learn features from each body part. This strategy has been shown to achieve a superior performance compared to strategies based on statistical representations of local features, such as [38–40], in distinguishing between fine-grained actions such as *washing hands* and *washing object* [37]. Approaches based on motion saliency, such as [41], apply techniques for identifying salient regions in video based on the presence of motion. The identified regions are used to reduce the colour and flow data to only those regions which contain motion. Since human action necessitates motion, those regions which do not contain motion are thus unlikely to contain the action and are thus removed from further processing by the CNN streams [41]. Long-term short-term memory (LSTM) memory models are widely used [42–45] to learn temporal information from both the colour and optical flow information. With these models the input sequence is represented as an ordered sequence of both colour frame images and optical flow images, while long-range temporal recursion is used to learn temporal features. It has been shown that these models are superior in modelling temporal dependencies between frames compared to models which learn features only from the visual domain [42, 43].

There are variations of the 2D-CNN approach to HAR which do not use a separate optical flow stream for learning temporal features but rather use temporal

aggregation of the frame images to encode the temporal information directly in the colour information [18]. One such approach, on which the Temporal Segment Network (TSN) [46, 47], Temporal Shift Module (TSM) [48] and other methods [36, 49–54] are based, is to extract short snippets of data from different frames over a long video sequence into a single stacked RGB image on which the classifier is trained. A core aspect of these methods, which distinguishes them from the two-stream based approaches, is the introduction of a sampling strategy to select the relevant information for action classification and build a long-range temporal structure [46]. In general a sparse sampling strategy is used, as opposed to a dense sampling strategy that involves sampling every frame, since consecutive frames are known to be highly redundant [46]. This strategy not only reduces the size of the data that has to be processed but also drastically reduces the computational cost of the system [46].

Whilst 2D-CNNs have achieved success in video-based human action recognition, they are recently being outperformed by three-dimensional (3D) CNN architectures [55, 56]. A 3D-CNN extends the traditional 2D-CNN to 3D space with the use of a 3D convolutional kernel that is applied on a 3D input [55]. These architectures are effective for human action recognition since the 3D kernel is capable of learning features directly from raw video input where the first two dimensions are the spatial dimensions with the third dimension being time [55]. The input is generally a time slice of a video in the form of a 3D matrix containing the 2D colour frame images stacked along a third time dimension. The ability of CNNs in performing automatic feature extraction allows 3D-CNNs to automatically extract both spatial and temporal (spatiotemporal) features directly with the 3D kernel while using only a single CNN stream [55, 56]. This is in contrast to the 2D-CNN based approaches which either require two separate streams for learning spatial and temporal features, or a sampling strategy to select specific frames from which to learn temporal relations. An overview of the general structure of 3D-CNNs is shown in Figure 2.4.

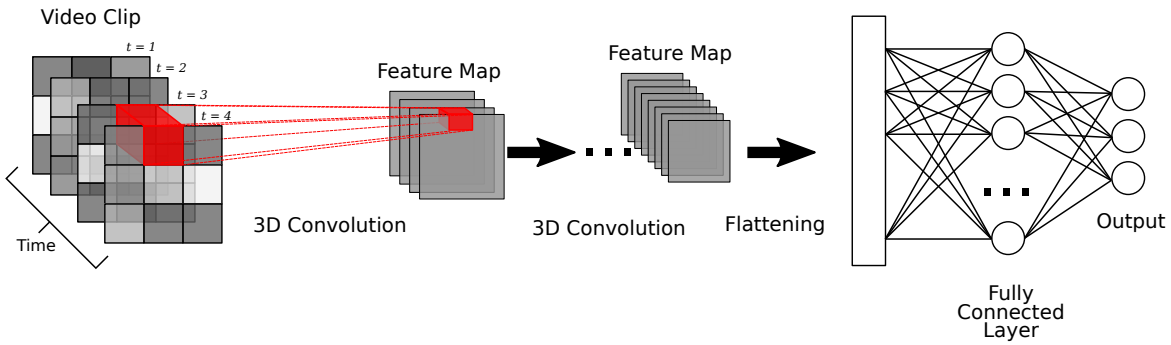


Figure 2.4 General structure of a 3D-CNN

A variety of 3D-CNN architectures exist, each with their own advantages and

disadvantages. Convolutional 3D (C3D) [57] is one of the most widely used 3D-CNN architectures [55], which uses a $3 \times 3 \times 3$ kernel in all layers. This architecture is the first 3D-CNN approach that processes an entire video as input as opposed to previous models such as [58], which takes a segment video volume as input [57]. This strategy allows for the model to be scaled to large datasets since there are no preprocessing steps [57]. Moreover, C3D is based entirely on 3D convolution in all layers whereas prior approaches, such as [33] and [35] were built on 2D-convolutions with 3D convolutions used only in the Slow Fusion model [57]. The Inception based I3D architecture, first introduced in [19], is another widely used 3D-CNN architecture [55] that has achieved a state of the art performance [31]. These architectures have achieved a performance that is comparable to that of 2D-CNNs despite being relatively shallow in comparison [59]. It is expected that a superior performance will be achieved with deeper 3D-CNN architectures however these are difficult to train [59].

Architectures based on Residual Networks (ResNets) allow deep networks, with a large number of hidden layers, to be trained successfully [59]. It has been shown that increasing the number of layers does not necessarily result in a higher classification accuracy [20]. In-fact, it has been observed that a saturation point is reached where the classification accuracy remains constant despite increasing the depth of the network [20]. Further increasing the depth of the network beyond the saturation point results in a decrease of the classification accuracy [20]. This degradation has in part been attributed to the vanishing/exploding gradient problem [20, 60, 61], which is caused when the error between the output and training target is not backpropagated through all layers [43]. The issue of the vanishing gradient problem has been addressed with the addition of normalisation layers [20, 61–63] and shortcut connections, which are referred to as residual connections. A shortcut connection is a connection between two non-adjacent layers of the network that skips over one or more intermediate layers. These shortcut connections allow the error gradient to backpropagate from the later layers to the early layers [20, 55]. Figure 2.5 shows a shortcut connection, that skips over a single convolution layer, within a ResNet architecture.

ResNet3D is class of 3D-CNN architectures based on residual networks. These have been shown to achieve a superior performance to both C3D and I3D and as such are regarded as the state of the art [18]. There are a number of variants of ResNet architectures, with the most widely used being ResNet with *basic* and *bottleneck* blocks [20], pre-activation ResNet [64], wide ResNet (WRN) [65], ResNeXt [66] and DenseNet [67].

The ResNet basic block, an overview of which is shown in Figure 2.6, is a building block of the basic ResNet architecture, which consists of two 3D convolutional layers each followed by batch normalisation and a Rectified Linear Unit (ReLU) [55]. A shortcut connection connects the block input to the layer preceding the last ReLU in

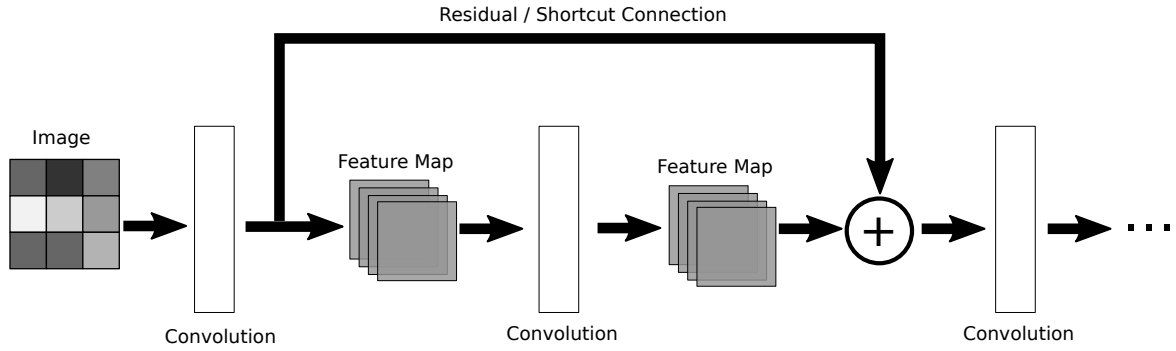


Figure 2.5 Diagram showing residual connections in ResNet architectures

the block, using a summation operation. The ResNet-18 and ResNet-34 architectures are based on basic blocks [55]. The ResNet bottleneck block, an overview of which is shown in Figure 2.7, consists of three convolutional layers, with the kernel size of the first layer being $1 \times 1 \times 1$ and the kernel size of the second layer being $3 \times 3 \times 3$ [55]. This block uses the same shortcut connection as the basic block. Bottleneck blocks form the basis of ResNet-50, ResNet-101, ResNet-152 and ResNet-200 [55].

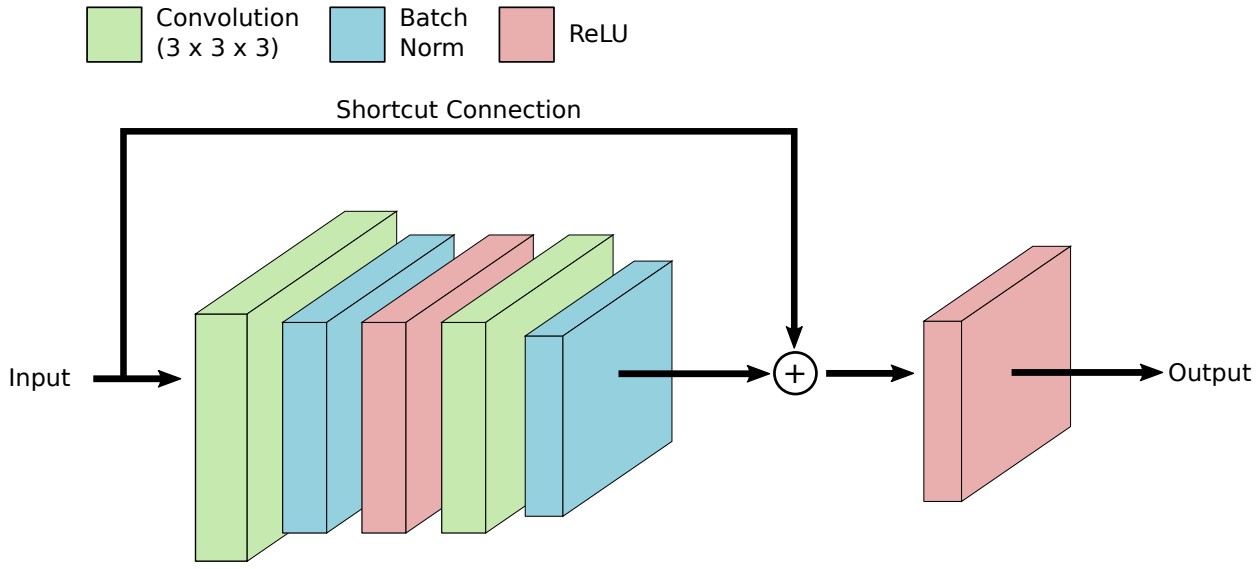


Figure 2.6 Structure of a ResNet basic block module

The pre-activation ResNet, the structure of which is shown in Figure 2.8, consist of the same components as the bottleneck block but arranged in a different order [55]. Rather than having a convolutional layer, followed by batch normalisation and ReLU, the batch normalisation comes first, which is followed by ReLU with the convolutional

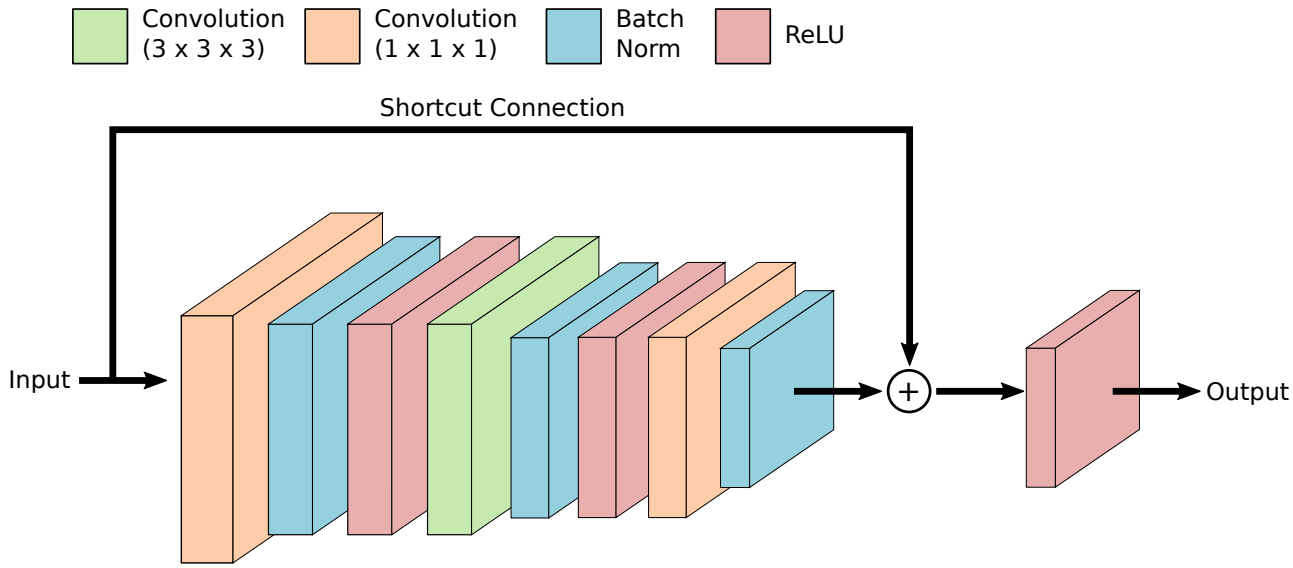


Figure 2.7 Structure of a ResNet bottleneck module

layer being the last layer in the block [55]. A shortcut connection connects the block input to the layer following the last convolutional layer in the block. This configuration was shown to facilitate the optimisation during training and reduce overfitting [55, 64].

The WRN architecture is a variation of the bottleneck block specifically designed to optimise training time when increasing the size of the network to increase its discriminative power [55]. The main difference between WRN and the bottleneck block is the number of feature maps for each convolutional layer. In WRN the number of feature maps are increased rather than increasing the number of layers [55]. This configuration has been shown to be efficient in parallel computing, particularly when using GPUs, since the convolution operations for each feature map can effectively be performed in parallel [55, 65].

The ResNeXt architecture extends the ResNet bottleneck block with the concept of cardinality [55]. Group convolutions are introduced which divide the input feature maps into small groups, with the cardinality being the number of middle convolutional layer groups in the bottleneck block [55]. It has been shown that increasing the cardinality is more effective than increasing the depth of ResNets based on basic and bottleneck blocks or increasing the width of WRNs [55, 66].

DenseNets are a variation of ResNets which differ in how the shortcut connections are made [55]. In the basic and bottleneck ResNets, the shortcut connections involve a summation operation whereas DenseNets connect the shortcut

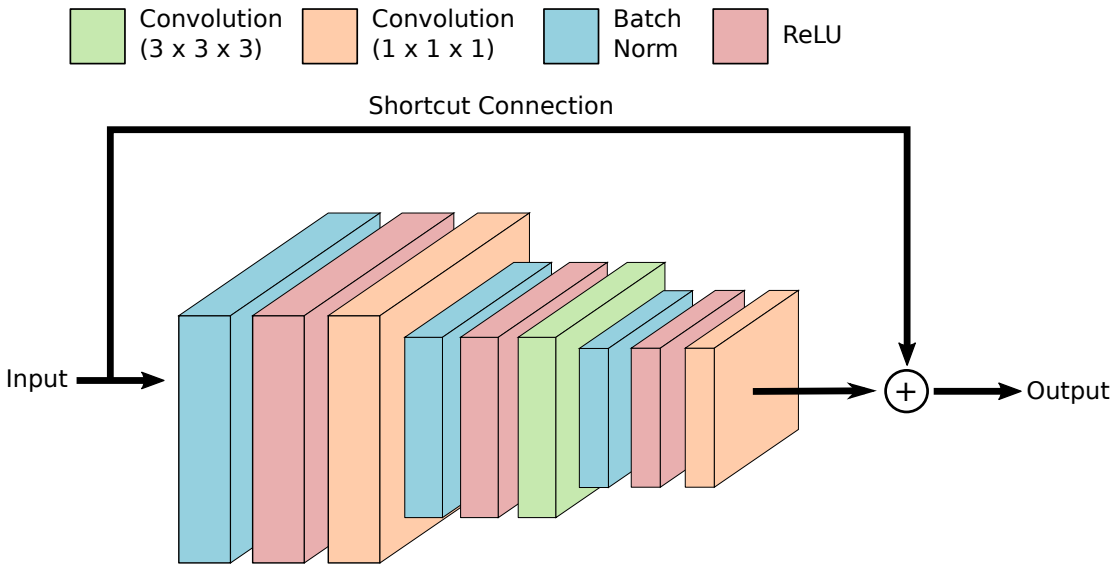


Figure 2.8 Diagram showing structure of a pre-activation ResNet

connections using a concatenation operation. Much like pre-activation ResNets, and unlike basic and bottleneck blocks, in DenseNets the convolutional layer follows the batch normalisation and ReLU with a shortcut connection from the input of the block to the output. This has been shown to achieve a higher classification accuracy with fewer parameters than other ResNet architectures [55, 67].

While 3D-CNNs have recently begun to outperform their 2D counterparts in action recognition tasks [55], and have even performed well in other computer vision tasks such as action detection [68], video captioning [69], action localisation [70] and video summation [71], training a 3D-CNN presents significant challenges [18]. The primary challenge is related to the large size of the network architecture, due to the 3D convolution kernels. As a result 3D-CNNs have considerably more parameters that have to be optimised during training, than those of 2D-CNNs [55]. Thus a larger dataset is required to successfully train a 3D-CNN than is required for training a 2D-CNN. The widely used HMDB-51 [72], containing 6766 action instances, and UCF-101 [73], containing 13320 action instances, action recognition datasets, which have been regarded as the most successful in the field, are too small for training 3D-CNNs from scratch [55, 59]. ActivityNet [74] is a comparatively larger dataset, containing 849 hours of video and 28000 action instances [55]. However, the results show that even a dataset of this size is too small to train a 3D-CNN from scratch [55]. Results have shown success with training 3D-CNNs from scratch, including deep architectures such as ResNet-152 which contains 152 layers, using the Kinetics-400

dataset [31], which contains more than 300000 trimmed videos covering 400 action categories [55]. Success has also been achieved using larger datasets such as Sports-1M [33] however their large size, which approaches 10 TB, makes training impractical [55].

Obtaining a sufficiently large dataset, such as Kinetics, is not always practical or even possible. Pretraining is a strategy used for reducing the training set size requirements. The idea behind pretraining is that the local feature descriptors required for action recognition, which are learned by the early layers in the network, generalise well to other datasets [56]. Therefore, the early layers can be trained once using a large dataset such as Kinetics, that doesn't have to be related to the dataset specific to the environment in which the HAR system is deployed. The pretrained model can then be distributed and further fine-tuned using smaller datasets that are specific to the environment in which the HAR system is deployed. Results of research [55, 56, 59] show that good results are achieved on the HMDB-51 and UCF-101 datasets using models pretrained on Kinetics-400 and Moments-in-Time [34].

Human action recognition approaches are not limited to RGB data only. Recent technological advances in human skeleton detection have allowed for the practical adoption of skeleton features in HAR [1]. Skeleton data, which consists of the coordinates of the joints comprising the human skeleton, is retrieved through vision sensors, including depth sensors, stereo cameras and motion capsules [1, 75]. The availability and affordability of commercial RGB-D (red, green, blue and depth) cameras such as Microsoft Kinect and Intel Realsense, coupled with human body skeleton-detection, have allowed for significant advances in skeleton-based HAR [1]. Early work in skeleton-based HAR focused on modelling geometric features based on the sequential and spatial characteristics of skeleton data using support vector machines, hidden Markov models and dynamic time-warping [1, 76]. However, hand-crafted features rely on heuristics which are limited by human domain knowledge [77, 78]. This knowledge may be helpful in specific settings but is not necessarily helpful in general environments. Moreover, only shallow features can be learned using hand-crafted approaches [79], which leads to an undermined performance particularly using unsupervised learning approaches [77]. Models based on hand-crafted features are only capable of recognising high-level features, such as *walking* and *running*, but fail to recognise context-aware activities such as *having coffee* [77, 80].

In recent years the trend in skeleton-based HAR is moving towards deep-learning based approaches, due to their success in image processing and video-based HAR [77]. This is due not only to their ability in learning higher level and more meaningful features than it is possible using hand-crafted features, but at the same time unsupervised and incremental learning is more feasible using a deep network structure [77]. Unsupervised learning techniques differ from supervised

learning techniques in that they do not required labelled training data. These techniques have a significant advantage over supervised learning techniques since annotating a large dataset is time consuming and labour intensive. Deep generative models are capable of learning action classes from a large amount of unlabelled data much more effectively than models based on hand-crafted features [77, 81].

Whilst CNNs have achieved success in image recognition and video-based HAR, there are additional considerations when adapting a CNN to skeleton based HAR. Most CNNs are designed to process images, whereas skeleton data consists of skeleton joint coordinates varying through time [77]. Each joint consists of three 1D vectors containing the readings for the x, y and z coordinates throughout time [77]. This data has to be transformed to a *virtual image* on which the CNN can be trained. There are two main strategy for adapting the skeleton joint coordinates, and those are the *model-driven* and *data-driven* strategies [77]. The data-driven strategy, adopted in [79, 82–86], involves treating each data dimension as a channel on which 1D convolution, followed by pooling is performed, after which the channels are flattened to unified DNN layers [77]. The advantage of this approach is that it is easy and simple to implement since the sensor data is effectively treated as a 1D image [77]. However, this approach fails to model the dependencies between the dimensions, which has a negative effect on its performance [77]. The *model-driven* strategy involves reshaping the input to a virtual 2D image so that 2D convolution can be performed [77]. One such approach involves simply combining all the dimensions into a single 2D image [87]. More complex approaches, such as the approach presented in [88], use a sliding window strategy to divide the time-series data into multiple overlapping segments that are then concatenated into a 2D image, while the model-based approach presented in [3], involves modality transformation to convert the sensor data to the visual domain using pressure distribution images. Model-driven approaches are more effective than data-driven approaches at modelling spatial and temporal dependencies between dimensions, however mapping time series data to a 2D image is not a trivial task and requires domain knowledge [77].

Autoencoders are a class of unsupervised learning techniques for skeleton based HAR, which learn a latent representation of the input data through hidden layers [77]. The architecture of an autoencoder is similar to that of a basic feed-forward neural network consisting of one input layer, one output layer and one or more hidden layers [89]. The autoencoder differs from a regular feed-forward neural network in that the number of neurons in the input and output layers is the same. Moreover, the goal of an autoencoder isn't to learn an action label but rather to reconstruct the input data at the output layer [89]. As a result, an autoencoder can be trained without annotated data, since it learns to reconstruct the input data rather than learning an action label. The autoencoder consists of an encoding step, which encodes the input to a latent

representation and a decoding step, which reconstructs the input from the latent representation [89]. The encoding step is formulated by (2.1), where x is the input, W is the weight matrix, b_1 is the bias vector and f is a nonlinear activation function [89]. The decoding step is formulated by (2.2), where b_2 is the bias vector for this step. The output $\tilde{x}$ should approximate the input x.

$$h = f(Wx + b_1) \tag{2.1}$$

$$\tilde{x} = f(W^T h + b_2) \tag{2.2}$$

The purpose of the autoencoder is to encode each skeleton to a latent representation that identifies the action, that is each instance of a given action a should produce the same latent representation. This is achieved by minimising the error between the input x and the reconstructed input $\tilde{x}$.

The stacked autoencoder (SAE) is an unsupervised learning architecture consisting of a stack of multiple autoencoders, where each layer of the SAE network is a basic autoencoder model [77]. After multiple rounds of training, using the same error minimisation strategy, the learned latent representations, at each autoencoder model, are stacked with labels to form a classifier [77]. SAE methods have the major advantage that they can be trained by unsupervised learning, which means annotated data is not required. However, the performance of SAE is highly dependent on the layers and activation functions of the individual autoencoders [77] and finding the most optimal architecture is not a trivial task [89].

The approaches to HAR that have been discussed up till this point are unimodal, which means they only consider a single data modality; either colour (RGB) data or skeleton data. This comes with a fundamental limitation since the particular data modality lacks certain information which may be beneficial to HAR [1]. RGB-based unimodal methods are limited by the lack of 3D structure provided by skeleton data [1]. On the other hand, skeleton-based methods are limited by the absence of texture and appearance features [1]. As a result these methods fail to distinguish between actions that have similar skeletal movements such as *typing* and *writing* [1]. To resolve these issues multimodal methods, which exploit data from multiple modalities, have been developed. Vision-based multimodal methods that incorporate skeleton and RGB modalities have been shown to achieve a higher performance compared to unimodal methods [1, 90]. The core task of multi-modal methods is data fusion [1]. Data fusion techniques can be classified as either data-level fusion, feature-level or decision-level fusion depending on where the fusion operation is performed [1]. Data-level fusion is performed at the input layer by concatenating the data modalities into a single input. This is rarely used in practice when the data modalities are intrinsically heterogeneous, which is the case with skeleton and RGB data [1]. Feature-level fusion involves

concatenating feature-level representations at the fully connected layers while decision-level fusion involves aggregating the results at the final layers, usually with a softmax function [1, 91–93]. Whilst multimodal methods, do offer a significant advantage over unimodal methods, in that they are able to exploit both the appearance information and 3D structure, they come with the disadvantage of having to process a larger volume of data, which increases the training time and resource consumption of the HAR system. Moreover, how to effectively fuse data from multiple modalities such that HAR accuracy is increased is still an open question without a clear solution that performs well in every situation [1].

3 Literature Review

This chapter contains an in-depth review of several recent state of the art human action recognition methods outlining the strengths and weaknesses of each approach. The chapter begins with a review of 3D-CNN methods and a comparison with 2D-CNN based methods. This is followed by a detailed review of three recent unsupervised approaches, which include a method based on subspace clustering, an autoencoder-based method and a contrastive-learning method. The chapter concludes with a review of a recent multi-modal HAR approach that learns features from both skeleton and colour data.

The performance of 3D-CNNs surpasses that of traditional 2D-CNNs in video-based human action recognition when the network is trained using a sufficiently sized training dataset. However, obtaining a sufficiently large dataset is not practical in some scenarios. The study by Hara *et al.* [55] aims to determine how large of a dataset is required to train a 3D-CNN, such that its performance surpasses that of 2D-CNNs, and whether existing HAR datasets are sufficiently large to train a 3D-CNN. Previous studies [31, 59] showed that the widely used UCF-101 [73], HMDB51 [72] and ActivityNet [74] datasets are not sufficiently large for training a 3D-CNN, with a high accuracy only being achieved when training using the Kinetics [31] dataset.

The first part of the study [55] focuses on the relatively shallow ResNet-18 architecture as a baseline to determine which datasets are suitable for training a 3D-CNN. This is based on the assumption that if overfitting is observed with ResNet-18 when trained on a given dataset, that dataset is too small for training deeper 3D-CNNs [55]. The remainder of the study [55] focuses on experimentation with deeper 3D CNNs, with the model depth varied from 18 to 200. The study performs experimentation with the ResNet basic block, ResNet bottleneck block, pre-activation ResNet, WRN, ResNeXt and the Dense ResNet architectures.

Prior to training the models, random training samples are generated from the training set for data augmentation [55] by a temporal sampling step followed by a spatial sampling step. The temporal sampling step involves selecting a temporal position in the video by uniform sampling, around which a 16-frame clip is generated. Videos shorter than 16 frames are repeated until they are of the required length. The spatial sampling step involves selecting a random spatial position from the four corners or centre of the video frames and a spatial scale of the sample in order to perform multi-scale cropping [55]. The scale is selected from the set $\left\{1, \frac{1}{2^{\frac{1}{4}}}, \frac{1}{\sqrt{2}}, \frac{1}{2^{\frac{3}{4}}}, \frac{1}{2}\right\}$ using the same procedure originally described in [94]. The scale is applied to the shorter side of the frame to determine the sample width and height. For example a scale of 1 means that the sample width and height are the same as the shorter side of the frame whilst a

scale of 0.5 means the sample size is half the size of the shorter side of the frame [55]. After spatial-temporal cropping using the selected temporal and spatial positions, scale and aspect ratio of 1, each sample is resized to 112×112 pixels. This results in an input matrix with a size of $3 \times 16 \times 112 \times 112$ representing 16 frames of 3 channel images each of a size of 112×112 pixels. Additionally, 50% of the samples are randomly selected and flipped horizontally. The final step in generating a training sample is mean subtraction which involves subtracting the mean pixel colour value of the sample from each channel. The generated samples retain the same action labels as the original videos.

The networks, explored in the study, are trained by stochastic gradient descent (SGD), with momentum, on the randomly generated training samples [55]. Training is achieved by minimising the cross-entropy losses of which the gradients are backpropagated from the output to the remaining layers of the network, with a weight decay of 0.001 and momentum of 0.9 [55]. When training from scratch, an initial learning rate of 0.1 is used which is then divided by 10 as training progresses. The initial learning rate for fine-tuning is 0.001 with a weight decay of $1\text{e}{-5}$ [55].

The study [55] adopts a different strategy for generating the data samples during inference. A sliding window scheme is used to generate the input clips, that is each video is split into non-overlapping 16-frame clips, rather than random temporal sampling. Instead of the random spatial cropping, adopted during training, each clip is cropped around the centre with a scale of 1. Each clip is fed into the network as input to compute the clip's scores for each action class. The scores are then averaged over all the clips of the video with the recognised action label being determined from the class with the maximum score.

Four datasets are considered in the study [55] namely UCF-101 [73], HMDB-51 [72], ActivityNet [74] and Kinetics-400 [31]. HMDB-51 and UCF-101 provide temporally trimmed samples where frames which do not depict the action are removed from the video [55]. On the other hand, the training samples provided by Kinetics and ActivityNet are not trimmed to the action.

The results of training ResNet-18, which is the shallowest network considered in the study, show that the validation losses on the UCF-101, HMDB-51 and ActivityNet datasets are significantly higher than the training losses, which indicates overfitting [55]. The validation set accuracies achieved on UCF-101, HMDB51 and ActivityNet, 40.1%, 16.2% and 26.8% respectively, are much lower than earlier methods [55], which indicates that these datasets are not sufficiently large to train even the shallowest 3D ResNet architecture let alone deeper architectures [55]. A classification accuracy of 54.2% was achieved on the Kinetics-400 validation set which is only slightly lower than the accuracy achieved on the training set. This indicates that Kinetics is sufficiently large for training a 3D ResNet [55].

The study [55] conducted experiments with deeper ResNet architectures, up to

ResNet-200, trained on Kinetics. An increase in the classification accuracy was observed, with an increase in network depth, up until ResNet-152, which achieved a top-1 classification accuracy of 63%. The accuracy of ResNet-200 was only 0.1% higher than the accuracy of ResNet-152, which is an indication that the training is beginning to overfit [55]. The results of the experiments with other ResNet architectures showed that the highest top-1 classification accuracy, 65.1% was achieved using ResNeXt-101. The study concludes that the Kinetics dataset is sufficiently large for training deep ResNet architectures.

Fine-tuning a 3D ResNet pretrained on Kinetics was explored in the study [55] as an alternative for training deep 3D ResNet architectures on UCF-101 and HMDB-51. The fine-tuning strategy adopted in the work involves training only the final convolution layer (conv5_x) and the fully connected layer [55]. The results of the experiments showed that a higher classification accuracy is achieved when pretraining is done using Kinetics followed by fine-tuning, with a classification accuracy of 89.6% being achieved on UCF-101 and 63.5% on HMDB-51 with ResNet-200. The highest classification accuracy, on both datasets, was achieved using ResNeXt-101, with a classification accuracy of 90.7% achieved on UCF-101 and 63.8% achieved on HMDB-51. Additionally, ResNeXt-101 also outperformed state of the art methods such as C3D [57], P3D [95], two-stream CNN [35], TDD [96]. With an increased input size, ResNeXt-101 also outperforms ST Multiplier Net [97] and TSN [46].

The work of Chen *et al.* [18] is a study that aims to provide a better understanding of the difference in performance and spatiotemporal behaviour between 2D-CNN and 3D-CNN HAR models. The authors identified that a significant weakness of existing studies is the lack of a standard evaluation dataset, unlike image processing where ImageNet [98] has been established as a standard. Additionally, the results are often skewed by differences in the network backbone. While Kinetics-400 [31] is the most popular dataset, which is used by around 60% of papers, it is known to be biased towards models which give a higher weight to spatial information over temporal information [18]. This makes the dataset inappropriate for validating a particular model's capability in learning temporal features [18]. Something-Something (V1 and V2) [99] is another widely used HAR dataset, which is used by around 50% of papers. Compared to Kinetics-400, this dataset is known to have significantly different temporal characteristics [18].

A number of popular HAR approaches are analysed, namely pure 3D-CNNs such as I3D [31], ResNet3D [59], and 2+1D models, which decompose a 3D convolutional filter into a 2D spatial filter followed by a 1D temporal filter, such as Separable 3D CNN (S3D) [100] and 2+1D ResNet (R(1+2)D) [101]. Pure 2D-CNN approaches such as TSN [46] and Temporal Aggregation Modules (TAM) [36] are also analysed and compared to 3D-CNNs. The study [18] focuses on these approaches

since they have achieved good results on popular large-scale datasets [18]. Additionally many other approaches, such as SlowFast [102] and CSN [32], are based on the models analysed in the study [18]. To facilitate the analysis the authors introduce a general framework for action recognition models which includes a spatiotemporal module followed by an optional temporal pooling module. This framework allows a HAR approach to be tested with different configurations, such as different backbones, temporal pooling and temporal aggregation strategies.

The HAR models are evaluated on the Kinetics-400 [31], Something-Something V2 (SSV2) [99], and Moments-in-time (MiT) [34] datasets. The study [55] also evaluates the model on a reduced version of each dataset, referred to as Mini-Kinetics, Mini-SSVV2 and Mini-MiT, by randomly selecting half of the action categories. The Mini-MiT is provided by [34] and contains one eighth of the full dataset. A two-step process is used for training the models, where first a *starter model* is trained on a selected number of frames, and then the model is fine-tuned using the remaining frames [18]. The starter models are either inflated with or initialised from their corresponding ImageNet pre-trained models [18]. Training is performed in four iterations where the starter model is trained during the first iteration with the remaining iterations serving to fine-tune the model. The maximum number of frames used for training is varied between experiments to determine the effect of the size of the input on the performance of the model. Clip-level accuracy is used as an evaluation metric, which is in contrast to the study by Hara *et al.* [55] where video-level accuracy was reported. The difference between the two is that clip-level accuracy is based on the prediction results of individual video clips, whereas video-level accuracy is based on the combined prediction results of multiple video clips [18]. As a result, the video-level accuracy is in most cases higher than the clip-level accuracy [18].

The study [18] reports a number of important observations from the results of the experiments. A higher classification accuracy, with a gain between 5% and 10%, is achieved without temporal pooling than with temporal pooling across all models except TSN, where a 20% increase in accuracy is reported [18]. These results show that temporal pooling is detrimental to the performance of the model as it counters the effectiveness of temporal modelling [18]. Additional gains in accuracy, of around 6%, were observed when replacing the InceptionV1 backbone, used by the I3D model, with ResNet50, which indicates that ResNet50 is a stronger backbone. A classification accuracy of 76.61% was achieved by ResNet50, without temporal pooling, on the Kinetics dataset which is on par with the classification accuracy achieved by SlowFast, 77%, on the same dataset. Similar results were achieved on the SSV2 dataset where an accuracy of 62.84% was achieved using ResNet50 and an accuracy of 63% was achieved using SlowFast.

The study [18] concludes with an analysis of the performance of the models in

transfer learning, which involves pretraining on a large general dataset followed by fine-tuning on a smaller dataset that is specific to a particular setting. The models that are explored are I3D-ResNet-50, SlowFast and TAM each with a ResNet-50 backbone. The models are pretrained on the Kinetics dataset and then fine-tuned on UCF-101 [73], HMDB-51 [72], Jester [103] and Mini-SSV2. Fine-tuning is performed for 45 epochs with a cosine annealing learning rate schedule starting with 0.01 [18]. An input size of 32 frames is used on which synchronised batch normalisation, with a batch size of 48, is used. A comparable performance is achieved with the three models that were explored. I3D with a ResNet-50 backbone achieved the highest classification accuracy, 97.12% on UCF-101, with TAM and SlowFast achieving an accuracy of 95.05% and 95.67%, respectively. SlowFast achieves the highest accuracy, 74.61% on HMDB-51, with I3D-ResNet-50 and TAM achieving 72.32% and 71.67%, respectively. On Mini-SSV2 TAM outperforms the other two models with an accuracy of 66.91% whereas I3D-ResNet and SlowFast achieved an accuracy of 65.86% and 63.93%, respectively. These results show that the performance of a given HAR model is highly influenced by the choice of backbone, since I3D, TAM and SlowFast achieved similar results when using a common backbone [18]. Moreover, these results, along with the results of the study by Hara *et al.* [55], demonstrate that pre-training followed by fine-tuning is an effective strategy for training a 3D-CNN when the dataset is too small for training from scratch.

It can be concluded from these studies [18, 55] that video-based approaches built on 2D-CNNs and 3D-CNNs have the advantage in that they are simple to use and set up, since they only require colour video input and come with few preprocessing steps. 3D-CNNs are particularly advantageous in this regard since the bulk of the processing is done by the network itself, which is given the entire video clip as input. However, these approaches require large training sets and deep CNN architectures which take a long time to train and have a large memory consumption. Despite, the potential advantage of 3D-CNNs, in superior temporal modelling, they are more challenging to train than their 2D counterparts whilst an increase in classification accuracy is not always observed. Besides these challenges, both 3D-CNN and 2D-CNN approaches are trained by fully supervised learning. Thus, every training sample requires an action label, which is provided by manually annotating the entire dataset. Given the large dataset requirements of such approaches, this is time-consuming and error prone [104]. Moreover, these approaches cannot be adopted in a big-data pipeline where new training data is received in real-time because each training sample has to be manually annotated by a human [104]. Unsupervised approaches to HAR offer a significant advantage over supervised approaches, in this regard, since they eliminate the burden of providing an annotated training set [104].

The work of Paoletta *et al.* [104] introduces a fully unsupervised approach to

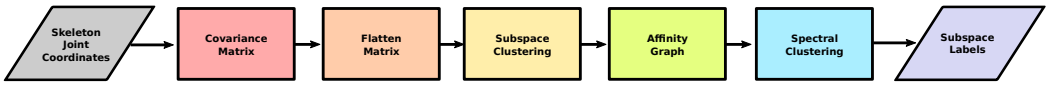


Figure 3.1 Pipeline of subspace-clustering HAR method

HAR based on subspace clustering. An overview of the method is shown in Figure 3.1. Subspace clustering is a technique for reducing the dimensionality of data by representing it as a union of subspaces, where each subspace has a lower dimensionality and simpler geometric structure [104]. It has found use in a wide range of computer vision tasks including image representation and compression [105], image segmentation [106], and the segmentation of dynamic moving objects [107, 108]. This technique is used to learn encodings from which an affinity matrix is constructed, which is then used to identify clusters in data according to the modelled similarities, or dissimilarities, between samples [104, 109]. The authors use subspace clustering on skeleton data to identify subspaces that correspond to action classes.

There are a number of different methods for performing subspace clustering on data. The most widely used approach involves learning an affinity matrix followed by a spectral clustering method such as low-rank representation [104, 110, 111]. Methods based on self-representation, which involve reconstructing a data sample from a linear combination of other samples [110, 112–114] have proven to be effective for high-dimensional data [104].

The basis of the method proposed in [104] is the construction of a covariance matrix representation of the input skeleton data. The covariance matrix representation has been used in image classification and object detection [115] and is an effective representation for skeleton-based HAR since it is capable of modelling second-order statistics [104]. The input video is represented as a matrix containing the coordinates of the skeleton joints through each timestamp. This matrix is converted to a covariance matrix, $\mathbf{\Lambda}$, and then flattened to a column vector, by taking only the elements along the diagonal and the upper triangle. This can be done since a covariance matrix is always symmetrical, which means $\mathbf{\Lambda} = \mathbf{\Lambda}^\top$ [104]. The flattened covariance matrix is then used as a single data point in the input to the subspace clustering algorithm.

Let $\mathbf{X}$ denote a $D \times N$ matrix where each column is a flattened covariance matrix with D data points. N is the number of flattened covariance matrices used to construct $\mathbf{X}$. The problem of subspace clustering is formulated as finding an $N \times N$ matrix $\mathbf{C}$ of coefficients such that

$$\mathbf{X} = \mathbf{XC} \tag{3.1}$$

with $\mathbf{C}$ subject to the constraint that the diagonal is equal to zero, $\mathtt{diag}(\mathbf{C}) = 0$, in order to avoid the trivial solution of $\mathbf{C}$ being the identity matrix [104].

The work [104] explores a number of solutions for finding a matrix **C** that satisfies these constraints, such as Least Squares Regression (SS-LSR) [114], which formulates the problem of finding **C** as a minimisation of an L^2 penalty, Sparse Subspace clustering via Alternating Direction Method of Multipliers (SSC-ADMM) [112, 116, 117], which formulates the problem as a maximisation of the sparsity of **C** and Sparse Subspace Clustering by Orthogonal Matching Pursuit (SSC-OMP) [118], which differs from SSC-ADMM only in the optimisation method.

Once the matrix of coefficients **C** is found, it is used to build an affinity graph matrix **W**, on which spectral clustering is performed to assign each datapoint into a subspace. The final step of the proposed HAR solution is to convert the subspace labels to action class labels. This is achieved by associating each subspace label to an action label using the Hungarian algorithm [104, 119], which selects an association between cluster labels and action labels such that the number of misclassified instances is minimised. Whilst this final step still requires annotated data, the annotation can be limited to a subset of the training data whereas a fully supervised approach requires annotating the entire training set. As a result unsupervised learning approaches such as this one offer significant methodological and computational advantages over supervised approaches [104], though often at a cost to classification accuracy.

The authors' implementation does, however, come with a significant limitation in that the subspace clustering problem is formulated as finding an $N \times N$ matrix of coefficients that satisfies (3.1), where N is effectively the number of training samples. This requires loading the entire training set in memory at once, whereas CNN based approaches are trained on batches of samples which can be loaded in memory individually. Thus, the size of the training set is severely limited by the available memory of the machine.

The autoencoder has become the basis of most state of the art unsupervised HAR approaches. Capsule network based methods learn representations of objects in terms of their constituent parts and their poses [120]. This representation takes the form of a set of transformation parameters applied on a part identity followed by an activation function [120]. In [121] the Motion Capsule Autoencoder (MCAE) unsupervised HAR approach is introduced. The approach builds on viewport-invariant capsule based representations for images [120, 122] by introducing an unsupervised capsule framework that learns the trajectory of the skeleton joints corresponding to a given action class [121]. This problem is solved in two steps, *snippet learning* and *segment learning*. Both steps involve learning the trajectory of points in temporally consecutive samples, referred to as snippets and segments, with the difference between the two being in the time span of the trajectory that is learned. Snippets cover movement in a narrow time-span whereas segments cover a longer-time span consisting of multiple snippets.

The Capsule Network [120] is a method for representing objects in images by their constituent parts and poses, which are discovered automatically [121]. This is achieved by modelling the object in terms of capsules, where each capsule represents a pose with a transformation applied to it. The pose is part of the identity information of a capsule, whereas the transformation parameters are learned by unsupervised learning. This strategy has an advantage over learning features from images directly by CNNs, in that the learned features are viewpoint-invariant, more compact, flexible and discriminative [121]. When first proposed, capsules were learnt by agreement-based routing mechanisms [123, 124], however recently this strategy has been superseded by the unsupervised stacked capsule autoencoder (SCAE) [122], which uses feed-forward encoders and decoders to learn capsule representations for images [121]. MCAE [121] extends capsule autoencoders to learn skeleton joint trajectories for skeleton-based HAR.

At its core MCAE learns a discriminative representation of the trajectory of a point in terms of a motion pattern that is subject to an arbitrary and unknown geometric transformation based on the input [121]. The trajectory of a d-dimensional point $\mathbf{x}_i \in \mathbb{R}^d$ is represented as $\mathbf{X} = \{\mathbf{x}_i \mid i = 1, \ldots, L\}$ where L is the number of time steps. The input trajectory $\mathbf{X}$ is divided into $\frac{L}{l}$ non-overlapping snippets, where l is the length of each snippet, which are mapped to a semantic-agnostic representation by a snippet autoencoder. The individual snippet representations, derived by the snippet autoencoder, are then combined into segments which are fed into a segment autoencoder to derive a representation of the full motion.

The snippet encoder encodes the trajectory of a skeleton joint $\mathbf{X} = \{\mathbf{x}_j \mid j = 1, \ldots, l\}$, through a time slice of l time-steps, into a weighted sum of N snippet capsules (SniCaps). The SniCap is defined as a transformation matrix $\mathbf{A} \in \mathbb{R}^{(d+1)\times(d+1)}$ applied on a motion template $\mathcal{T} = \{\mathbf{t}_j | j = 1, \ldots, l\}$, which identifies a basic motion pattern, followed by the snippet activation $\mu \in [0, 1]$, which controls the weight given to the SniCap. The template $\mathcal{T}$ is independent of the input, with each SniCap being assigned a unique template, and forms the identity information of the SniCap. The transformation parameter $\mathbf{A}$ and snippet activation μ are learnt from the input by unsupervised learning. An overview of the snippet encoder is shown in Figure 3.2.

The snippet encoder is followed by a decoding step reconstructs the original snippet $\mathbf{x}_{i:i+l}$ from the its SniCap representation. The original snippet is reconstructed as a weighted sum of snippet templates, with the weights determined by the learnt SniCap transformation parameters.

The segment autoencoder encodes a point trajectory over a longer time span, than that of a single snippet, to a representation that is expressed in terms of segment capsules (SegCaps). A segment is a point trajectory of length L that is made up of

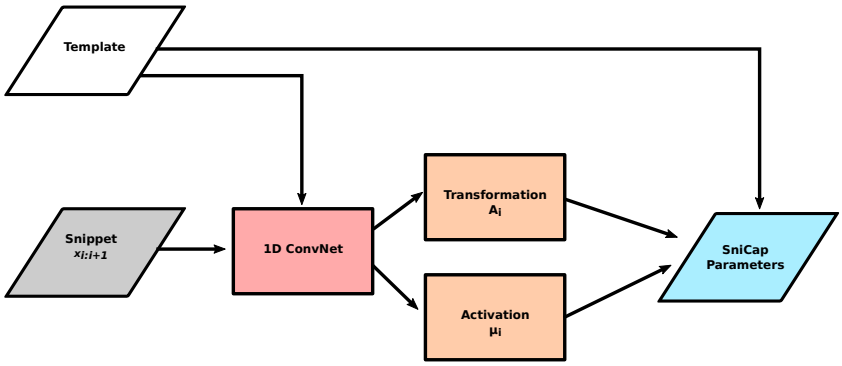


Figure 3.2 Pipeline of MCAE snippet encoder

$S = \frac{L}{l}$ consecutive and non-overlapping snippets. Similar to the SniCap, the SegCap $(\mathcal{P}, \mathbf{B}, \nu)$ consists of a template $\mathcal{P}$, a transformation parameter $\mathbf{B}$ and a segment activation ν. However, rather than the template encoding a reference motion trajectory, the SegCap template $\mathcal{P}$ is expressed in terms of N snippet templates. Similar to the SniCap, the transformation and activation parameters $(\mathbf{B}, \nu)$ are dependent on the input whereas the template forms the identity information of the SegCap.

Since the objective of the snippet autoencoder is to reconstruct the original input, it is trained by minimising the error between the original and reconstructed inputs. For a training sample $\mathbf{X} = \{x_i \mid i = 1, \ldots, L\}$, the loss function is defined as

$$\mathcal{L}_{\text{Rec}}^{\text{Sni}} = \sum_{i=1}^{L} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i)\|_2^2 \tag{3.2}$$

where $\hat{\mathbf{x}}_i$ is the i^{th} reconstructed snippet.

The objective of the segment autoencoder is to reconstruct the SniCap parameters, therefore it is trained by minimising the error between the original and reconstructed parameters. The loss function is defined as

$$\mathcal{L}_{\text{Rec}}^{\text{Seg}} = \sum_{i=1}^{S} \left\|(\hat{\mathbf{A}}_i - \mathbf{A}_i)\right\|_2^2 + \left\|(\hat{\boldsymbol{\mu}}_i - \boldsymbol{\mu})\right\|_2^2 \tag{3.3}$$

where $\hat{\mathbf{A}}_i$ and $\hat{\boldsymbol{\mu}}_i$ are the reconstructed transformation and activation parameters of the i^{th} SniCap.

The MCAE method described till this point is only capable of learning motion patterns of a single point. However, for HAR a method which can learn motion patterns consisting of multiple points is required. The authors propose an extension to MCAE, referred to as MCAE Multiple Point (MCAE-MP), which can learn motion

patterns from a set of points $\mathcal{X} = \{\mathbf{X}_i \| i = 1, \ldots, K\}$ where $\mathbf{X}_i$ is the trajectory of the i^{th} point, with K being the number of points. The extension is trained by processing each point separately, using MCAE, to produce K segment activation vectors $\{\boldsymbol{\nu}_i \mid i = 1, \ldots, K\}$, which are concatenated into a single representation $\boldsymbol{\nu} \in \mathbb{R}^{KM}$ on which unsupervised learning is performed.

The performance of MCAE-MP in skeleton-based HAR is evaluated using the NW-UCLA [125], NTU-60 [126] and NTU-120 [127] datasets. The NW-UCLA is split such that view 1 and 2 are used for training and view 3 is used for validation. The evaluation on NTU-60 and NTU-120 is performed using both the cross-subject (XSUB) and cross-view (XVIEW), referred to as cross-setting (XSET) in NTU-120, evaluation protocols. The XSUB protocol splits the dataset such that the action instances in the training set are performed by different actors, referred to as subjects, than those in the validation set. The XVIEW protocol splits the dataset such that the training set contains different views of the same action, performed by the same subject, from those in the validation set.

The performance of MCAE is compared to unsupervised methods based on skeleton and depth-data, such as the work of Luo *et al.* [128] and the work of Li *et al.* [129], unsupervised methods based on skeleton data only, SeBiReNet [130], LongT GAN [131], MS^{2}L [132], CAE+ [133], Predict and Cluster (P&C) [134] and two supervised methods, DropGraph [135], and JOLO-GCN [136]. Two variants of MCAE-MP are explored, MCAE-MP (SLP) which uses a single layer perceptron (SLP) as the auxiliary classifier, and MCAE-MP (1NN) which uses a 1-nearest-neighbour classifier instead of an SLP. The MCAE-MP (SLP) method achieved the highest classification accuracy, 65.6% on NTU-60 (XSUB) among the unsupervised approaches, which are based on skeleton data only, with the second highest accuracy, 58.5%, achieved by CAE+. However the method was outperformed by the work of Li *et al.* [129], which is based on both skeleton and depth data. MCAE-MP (SLP) achieved the highest accuracy among the unsupervised approaches on NTU-120 XSUB, 52.8%, and XSET, 54.7%. On the other hand, the highest classification accuracy among the unsupervised approaches was achieved with MCAE-MP (1NN) on NTU-60 (XVIEW), 82.4%, and NW-UCLA, 84.9%. Whilst these results indicate an improvement over the state of the art in unsupervised learning, the fully supervised approaches DropGraph and JOLO-GCN outperform MCAE-MP by a margin of 8% to 35% across all datasets, which indicates that fully-supervised approaches are superior in classification accuracy. A disadvantage of MCAE is that it requires an additional training step to learn the motion templates before the network can be trained on skeleton data to learn action classes. Additionally, a number of parameters, such as the spatiotemporal disturbance that is applied on the input samples, are dataset dependent.

The autoencoder is not limited to skeleton data only. In [137], an autoencoder

based action recognition method is proposed that learns actions from images. Prior to the autoencoder, a preprocessing step is used to reduce the size of the data by removing irrelevant data and noise. The purpose of the preprocessing step is to improve the recognition rate, that is to reduce the inference time, and to minimise memory usage [137]. The preprocessing step involves converting the frame images to greyscale and reducing their dimensions by downsampling them to size of 50×50 pixels. Aggregate Channel Features (ACF) [138] is then applied to detect the humans in the video. This removes the background noise and extracts the relevant region of interest from the video. To further remove the background clutter from the scene, the depth map is used as a mask to identify the foreground pixels, and remove the background from the image. Following the preprocessing step, the sequence of processed images is used to train a sparse autoencoder. As in other autoencoder based approaches, the autoencoder weights are learned by unsupervised learning where the objective of the autoencoder is to reproduce the input. Once trained, the filters of the autoencoder are used to derive a set of feature maps from the input image sequence. The histogram of oriented gradients (HOG) is used to extract the local object and shape features [137]. The histogram of oriented gradients is associated to an action label using the Modified Hausdorff Distance (MHD) [139]. The proposed method was evaluated on the Weizmann [140], CAD60 [141], and the MMU dataset, which is proposed in the same paper [137]. The reported results show an accuracy of 100% on the Weizmann dataset, 88.24% on CAD60 and 99.5% on MMU.

In [142] a Hierarchical Contrast (HiCo) for unsupervised skeleton-based action representation learning method is proposed. This method is based on contrastive-learning which is a variation of the autoencoder unsupervised learning method. Constrastive-learning for skeleton based action recognition, which has become the dominant unsupervised HAR approach in recent years due to its simplicity and superior performance [142–144], was first proposed in [133], based on the earlier Momentum Contrast (MoCo) [145] for unsupervised image representation learning. The method involves transforming a single skeleton instance into two instances, by random augmentation strategies, which are fed into a query and key encoder respectively to obtain instance-level features [133, 142]. The contrast between the outputs of the key and query encoders is conducted such that augmentations of the same skeleton, referred to as positive pairs, have more similar representations than those of different skeletons, referred to as negative pairs [133, 142]. Skeleton specific spatial and temporal augmentations were proposed in [146], which exploit the spatiotemporal invariances of similar action sequences [142, 146]. Most prior approaches to contrast learning-based HAR, do not exploit the hierarchical nature of the human skeleton [142]. Existing approaches that utilise hierarchical modelling, such as [147, 148] either only consider the spatial hierarchical structure or only consider the

temporal hierarchical structure [142]. The proposed HiCo method improves on these approaches by jointly modelling both the spatial and temporal hierarchical structure of the human skeleton.

The HiCo method [142] consists of two main components, namely a hierarchical encoder network and the hierarchical contrast for unsupervised learning. The hierarchical encoder network transforms the skeleton sequence into clip-level, part-level, domain-level, instance-level representations. The hierarchical contrast for unsupervised learning component performs action classification by minimising a loss-function over the clip-level, part-level, domain-level and instance-level representations.

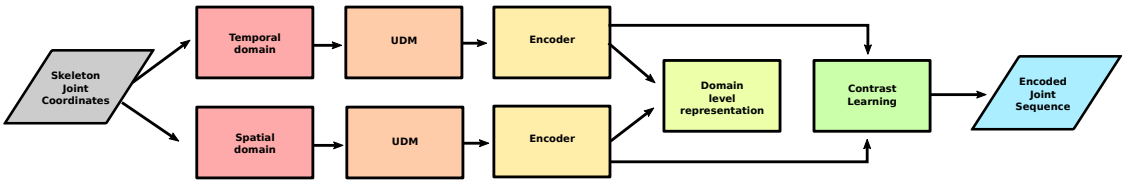


Figure 3.3 Pipeline of HiCo HAR method showing spatial and temporal domain branches

The hierarchical encoder network consists of two branches for the temporal and spatial domains, which is illustrated in Figure 3.3. The temporal branch encodes the skeleton sequence to a clip-level representation at multiple temporal granularities. Likewise, the spatial branch encodes the skeleton sequence into a part-level representation at multiple spatial granularities. The domain-level and instance-level representations are assembled out of the clip-level and part-level representations.

The clip-level and part-level encoders follow the same steps. The skeleton sequence is first reshaped to a time-majored domain, in the case of the clip-level representation, or a part-level domain, in the case of the part-level representation. The reshaped inputs are then projected to a C-dimensional feature space using two fully connected layers from which multiple clip-level and part-level sequences are generated using a unified downsampling module. UDM is defined by:

$$\text{UDM}(\cdot) = \text{MaxPool1D}(\text{LN}(\sigma(\text{Conv1D}(\cdot)))) \tag{3.4}$$

where Conv1D denotes a 1D convolution layer with a kernel size of 5 and stride of 1, LN denotes the Layer Norm and MaxPool1D denotes a 1D max pooling layer with a kernel size of 2. UDM is applied in series to derive clip-level and part-level sequences of larger temporal and spatial granularities.

The temporal domain-level representation is constructed by concatenating the clip-level feature vectors derived from successive UDM applications on the clip-level

representation. Similarly, the spatial domain representation is constructed by concatenating the part-level feature vectors derived from the successive UDM applications on the part-level representation. Finally the instance level representation, which represents the entire sample, is defined as the concatenation of the temporal and spatial domain representations.

The final step of the method proposed in [142] is contrast learning. The work adopts the MoCo approach [145] for contrast learning, which uses a query encoder and key encoder with a dynamic dictionary queue and moving averaged update mechanism [142]. The work also adopts a two-layer perceptron, originally described in [149], for feature projection before contrast learning. The classifier is trained in an unsupervised manner by minimising the total loss, defined as the sum of the instance-level, domain-level, clip-level and part-level contrast losses, which are computed on the instance level, temporal and spatial domain level, clip-level and part-level representations respectively.

The HiCo method is evaluated using four widely used skeleton-based HAR datasets, namely NTU-60 [126], NTU-120 [127], PKU-MMD I and PKU-MMD II [150]. Similar to other works such as MCAE [121], both the XSUB and XVIEW evaluation protocols are followed on NTU-60 and NTU-120, whilst only the XSUB protocol is used on PKU-MMD I and II. The method was compared to state of the art autoencoder based methods, namely LongT GAN [131], EnGAN-PoseRNN [151], P&C [134], SeBiReNet [130], H-Transformer [152], Skeleton Cloud Colorization [153], GL-Transformer [154]. The method was also compared to state of the art contrastive-learning based methods, namely AS-CAL [133], CrosSCLR [143], ISC [146], AimCLR [155], and hybrid methods such as MS^{2}L, PCRP [156].

The reported results of the experiments show that the HiCo method achieves a superior performance to other methods across all datasets. The highest top-1 classification accuracy, 89.4%, was achieved on the PKU-MMD I dataset, with the HiCo-LSTM method. This is followed by NTU-60 where an XSUB accuracy of 81.4% and XVIEW accuracy of 88.8% are achieved. An accuracy of 73.7% and 74.5% was achieved on NTU-120 with the XSUB and XSET evaluation protocols respectively. The lowest accuracy, 54.7%, was observed on PKU-MMD II. This was also observed among the remaining methods, where an even poorer performance was reported. The authors state that this is due to the dataset containing more noise, than the remaining datasets, which is caused by the view variation [142, 155]. The gain in classification accuracy by HiCo-LSTM, compared to the state of the art, on the NTU-60 and NTU-120 datasets ranges from 3.5% to 6.6%. The gain in classification accuracy is 2.2% on PKU-MMD I and 16.2% on the challenging PKU-MMD II dataset.

In [157], a skeleton-based contrastive learning HAR method, Clip-Driven Contrastive Learning (CdCLR), was proposed. Other contrastive learning methods treat

entire sequences as action instances, and therefore the contrast is only computed between action instances occurring in different sequences [157]. On the other hand, CdCLR treats individual time-slices, referred to as clips, within the same sequences as action sequences. This increases the amount of data which is available for training, which improves data utilisation [157]. The method was evaluated on the NTU-60 [126], NW-UCLA [125] and iMiGUE [158] datasets. The reported results [157], show a classification accuracy of 79.8% and 71.1% on the NTU-60 dataset, with the cross-view and cross-subject evaluation protocols, respectively. On NW-UCLA an accuracy of 87.5% was reported [157]. The lowest accuracy, 39.38% was reported [157].

Whilst the results achieved using skeleton-based HAR methods such as MCAE [121] and HiCo [142] are promising, HAR approaches based on skeleton data alone are fundamentally limited since they lack the texture information present in RGB video [1]. Similarly, HAR methods based on RGB video data alone lack the 3D structure provided by skeleton data [1]. Multimodal HAR approaches, which incorporate data from multiple modalities such as RGB video and skeleton data, have demonstrated an improved performance over unimodal methods [1]. A simple multi-modal HAR solution is to aggregate the results of a video-based HAR classifier, such as S3D [100], with a skeleton-based classifier. However, the performance of video-based methods, such as I3D and S3D, has been shown to be highly dependent on the background [1]. A high classification accuracy is observed when training I3D and S3D on datasets, such as the UCF-101 dataset [73], which contains actions that are performed in particular settings [1, 159, 160]. Examples of such actions are surfing and water-skiing, which are always performed in a beach setting. This is due to the background contributing to action classification [1, 159, 160]. However, it is not always the case that a particular action is performed in a given setting and, as a result, the same methods perform poorly when trained on datasets in which the background does not contribute to action classification, such as the NTU datasets [126, 127], which have a consistent background in all training and validation samples across all actions [1, 161, 162]. The lack of a distinguishing background in the NTU datasets has been shown to be particularly challenging for video-based HAR classifiers, such as I3D and S3D. Additionally these classifiers require vast computing and memory resources, for example a GPU cluster of 56 GPUs is required to successfully train I3D and S3D on NTU-60 and NTU-120, as well as taking a long time to converge [1].

The model-based multimodal network (MMNet) [1], is a recent multi-modal HAR approach which performs action classification whilst fusing the data from two modalities, namely skeleton data and RGB colour data. The network consists of three classifiers, two of which are dedicated to learning features from skeleton data with the third dedicated to learning features from RGB data. As in previous work [163, 164], the skeleton data is divided into skeleton joint and bone data, which is itself derived from

the skeleton joint coordinates, on which separate classifiers are trained. The RGB classifier is trained on a single image constructed from the spatiotemporal region of interest (ST-ROI) that captures the action in the video. This image is constructed by concatenating individual spatial regions of interest (ROIs), identified in multiple frames using the skeleton data, into a single image.

To summarise the MMNet architecture, let $\{J^{(i)}, B^{(i)}, V^{(i)}\}$ denote training sample i, with $i \in [1, N]$ where N is the number of training samples. $J^{(i)}$ denotes the skeleton joint input, $B^{(i)}$ denotes the skeleton bone input and $V^{(i)}$ denotes the RGB video input of the i^{th} training sample. The action label corresponding to the i^{th} training sample is denoted by $y^{(i)}$. The classification operation by MMNet is defined, at a high-level, by

$$\hat{y} = G_J(\mathbf{\Theta}_J, J) + G_B(\mathbf{\Theta}_B, B) + G_V(\mathbf{\Theta}_V, V) \tag{3.5}$$

where $\mathbf{\Theta}_J$, $\mathbf{\Theta}_B$ and $\mathbf{\Theta}_V$ are the joint, bone and RGB video model parameters and $\hat{y}$ is the predicted action label.

A 2D ResNet-18 model is used to learn features from the RGB data modality, which is trained on 2D images capturing the ST-ROI of each training sample. The ST-ROI itself is constructed based using the skeleton joint coordinates of the actors carrying out the action. To identify the skeleton joint coordinates, the authors used the OpenPose tool [2] since the coordinates retrieved by it are somewhat more accurate than those retrieved by the Microsoft Kinect v2 sensor [1].

L frames, at times $\boldsymbol{\tau} = \{\textit{interval} \times l \mid l = 1, \ldots, L, \textit{interval} = T/L\}$, are selected for constructing the ST-ROI $R^{(i)}$, which is constructed by concatenating L sub-ROIs horizontally, where each sub-ROI $R_t^{(i)}$ is the vertical concatenation of M' spatial ROIs selected at time t, with M' being the number of selected skeleton joints. This produces a grid containing $M' \times L$ spatial ROIs. For samples containing multiple actors, each sub-ROI $R_{tk}^{(i)}, k \in [1, K]$ for a given time t is concatenated horizontally, where K is the number of actors. This strategy drastically reduces the volume of RGB video data while still preserving the object's appearance and movement information of the actors [1].

A graph convolutional network (GCN) [163, 165] is used to learn features from the skeleton joint and bone data which is trained on the skeleton graph. The skeleton graph is constructed using the skeleton joint coordinates as the vertices and the skeleton bones, which connect the joints as the edges. Additionally the skeleton joints of temporally adjacent skeletons are connected by edges. Besides serving as the classifier for the skeleton joint and bone modalities, the GCN is also used to derive a weight function that indicates which joints are most important to action classification. Each spatial sub-ROI corresponding, to a single joint, is multiplied by the weight determined by the GCN. Effectively this produces an ST-ROI image where the spatial ROIs corresponding to joints which are important for the action have a higher intensity.

The performance of MMNet was evaluated on the NTU-60 [126], NTU-120 [127], PKU-MMD [166], NW-UCLA [125] and Toyota Smarthome [167] datasets. MMNet achieved a high classification accuracy, XSUB 96% and XVIEW 98.8%, on NTU-60 and outperformed all unimodal methods as well as several state of the art multimodal methods such as the Video Pose Network (VPN) [168], TSMF [169], Body Pose Evolution Map [170] and S-Res-LSTM [171]. However, it was outperformed by VPN++ Poses [172], which achieved an XSUB accuracy of 96.6% and XVIEW accuracy of 99.1%. Similarly, MMNet was also outperformed by VPN++ on the Toyota Smarthome dataset, where VPN++ achieved a classification accuracy of 71%, whereas MMNet achieved an accuracy of 70.1%. On the NTU-120 MMNet achieved the highest classification accuracy, XSUB 92.9% and XSET 94.4%, among all methods including VPN++. On NW-UCLA MMNet was outperformed by CTR-GCN [173], which is a skeleton-based HAR method.

Despite being outperformed by VPN++ in two datasets, the difference in classification accuracy between the two methods was at most 0.9%. However, MMNet has a significant advantage in that it is trained on ST-ROI images, whereas VPN is based on I3D which is trained on 64 frames per sample [1]. This not only requires a larger and deeper network, compared to MMNet, which uses the smaller ResNet18, but takes a longer time to train and has a considerably larger memory consumption [1]. MMNet has a significant advantage over 3D-CNN video-based methods in that it uses a shallower network which requires less data, time and resources to train. Additionally, a higher performance is achieved than skeleton-based methods such as HiCo [142], due to exploiting data from multiple modalities. The main disadvantage of MMNet is its relative complexity compared to other approaches, since it requires collecting colour data as well as skeleton data and training multiple classifiers. The method also has numerous preprocessing steps, including skeleton estimation using OpenPose [2], which increases the difficulty of deploying the method in real-world scenarios. Moreover, the parameters used to construct the ST-ROI, which include the skeleton joints that are selected and the size of the spatial ROIs, are dataset dependent. This presents an additional challenge to the practical adoption of the HAR method.

4 Motion Saliency

It has been identified in Chapters 2 and 3 that the size of the data associated with video presents a significant challenge to successful human action recognition, since it increases both the time required for training and the resource consumption, in terms of memory and energy, of the HAR system. Most of the data present in video consists of background clutter that is irrelevant to the classification of human actions [15]. If the data that is processed by the classifier can be limited, with low effort, to only those regions which are relevant to the action classification task, the total computational time and resource consumption of the HAR system can be reduced significantly [15]. Since every human action necessitates motion, the region of the video capturing the action can be identified by the presence of motion. Thus, the problem of identifying the relevant regions can be reduced to the problem of finding regions containing significant motion. One way to identify these relevant regions is through motion saliency detection methods. The first section of this chapter begins with a background and literature review of motion saliency techniques. The second section details the novel motion saliency detection solution that was developed, which is followed by a section detailing the evaluation of the solution and the results achieved. The final section concludes this chapter with a summary of the results.

4.1 Background and Literature Review

Saliency detection is the field of visual attention analysis which deals with identifying regions or objects in visual media that appeal to the human visual system [174]. Moreover, the aim of saliency detection is to identify those regions which stand out and capture human attention quickly [174]. This strategy is tremendously useful in understanding semantic content from video and images [174], and forms the basis of many computer vision tasks such as video segmentation [175], adaptive content delivery [176], and the widely used JPEG image compression standard [177].

Traditional image saliency techniques are based on feature selection from colour and texture features [174, 178]. The saliency detection problem is generally formulated in terms of local contrast analysis where the aim is to identify the regions with the highest local contrast [174, 176]. In other words, the goal of image saliency detection is to find regions or objects which are brighter than their surroundings and thus are likely the first to be noticed. The goal of saliency techniques is to reduce the size of the data to interesting or relevant regions [179]. This strategy imitates the strategy adopted by the human visual system in which only attention is dedicated only to certain objects or areas [179].

A common challenge faced by saliency techniques is the implicit requirement that the identified regions are relevant to the task at hand [180]. However, this is difficult to quantify whilst maintaining a general enough method for multiple applications [180]. In [180], Kadir, *et al.* proposed a novel saliency detection method, *Scale Saliency*, which is based on identifying regions that exhibit unpredictable characteristics simultaneously in a feature-space and over scale. The method is reported to be invariant to intensity shifts and robust to small changes in viewpoint, as well as offering a more general saliency model compared to conventional feature detection techniques. The authors did note, however, that the algorithm is sensitive to noise and comes with a large computational load where 160 seconds of computational time are spent on a single image [180].

In [179], Itti, *et al.* proposed a framework for identifying salient regions that is based on the identification of regions using multiple feature maps. The presented method is a bottom-up approach, which means salient regions are identified solely based on the properties that stimulate the visual system. Salient regions are identified in a multi-step process which begins with representing the input in iconic (appearance-based) topographic feature maps, followed by centre-surround computations in every feature at different spatial scales, and within feature spatial competition for activity [179]. The information obtained from these feature maps is combined into a single map, which represents the local saliency of any one location with respect to its neighbourhood [179]. The regions which are most salient are identified by the maximum of the feature maps. Finally, the saliency map is endowed with internal dynamics to allow the perceptive system to scan the visual input such that its different parts are visited by the focus of attention in the order of decreasing saliency, based on the salient regions identified in the previous step [179]. The reported results show that the proposed method is able to identify salient objects in cluttered outdoor scenes regardless of whether they are large objects covering a large portion of the image or small objects covering one hundredth of the image. With regards to computational time, the authors report a processing time of ten seconds for low-resolution images and 15 minutes for high resolution images [179]. With more recent computing hardware the computational time is likely to be reduced considerably.

Another approach to saliency detection is to formulate it as the problem of finding a suitable threshold for a given intensity function. In [181], a saliency detection method, named *Maximally Stable External Regions*, is proposed. In simple terms, the method finds the threshold that separates the salient regions from the background when applied on a greyscale intensity image. The process involves iterating through the possible thresholds ranging from the minimum threshold of 0, which excludes the entire image to the maximum threshold, which identifies the entire image as salient. The threshold which results in the smallest change to the resulting saliency map is

identified as the optimal threshold for identifying the salient regions [181]. The advantages of this method, compared to others, are that it is invariant to affine transformations of the image intensity and it is capable of multi-scale detection, that is it is able to identify both small and large salient regions [181].

While image based saliency techniques have achieved promising results it has been found that they are not suitable for saliency detection in video since moving objects usually capture significantly more attention than bright but stationary objects [174]. Furthermore, the region with the highest local contrast is not necessarily a region that captures a human action.

Saliency detection in video is based on motion discrimination rather than local contrast and thus identifies regions that contain significant motion [174]. These techniques are referred to as motion saliency detection. Additionally, background subtraction techniques, such as Gaussian Mixture Model (GMM) [182] background subtraction, solve the same problem and can, thus, be used for motion saliency detection [174]. Figure 4.1 demonstrates the difference between regions identified using traditional contrast based saliency techniques, which are shown in blue, and regions identified by motion saliency techniques, which are shown in red.

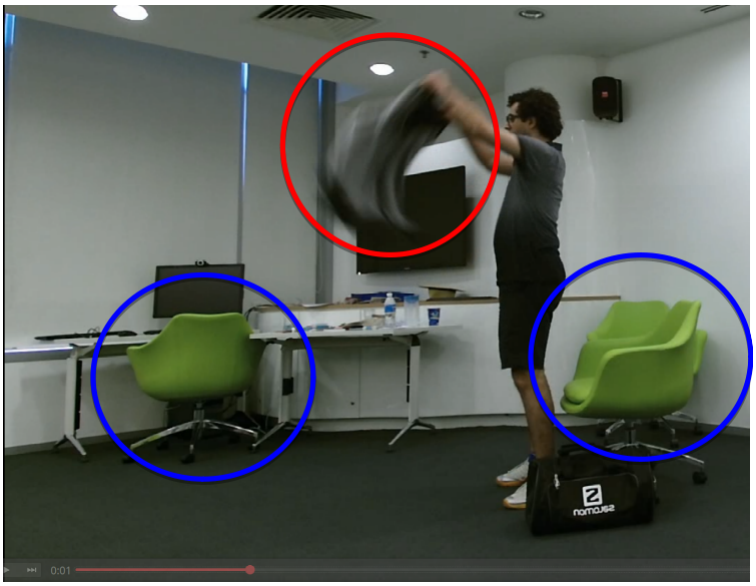


Figure 4.1 Comparison between contrast-based salient regions (shown in blue) and motion salient regions (show in red)

Early work in background subtraction involved building a probabilistic model of the background in order to classify pixels as either foreground or background based on the model [183]. Whilst basic probabilistic background subtraction methods such as GMM [182] and Kernel Density Estimation (KDE) [184] have achieved a good performance, they have been surpassed by more sophisticated approaches [183], such as SubSENSE [185], Pixel-based Adaptive Word Consensus Segmenter (PAWCS) [186]

and Sliding Window Change Detection (SWCD) [187]. In recent years, deep-learning based background subtraction methods, such as the Foreground Segmentation Network (FgSegNet) [188], have begun outperforming those based on traditional techniques [189, 190]. However, the performance of these methods has been found to be highly tuned to the videos on which they are trained [189]. Some of these methods, for example FgSegNet, are trained on a sequence of frames and then evaluated on another sequence of frames from the same video [189]. The performance of these methods degrades significantly when they are evaluated on different videos from the ones on which they are trained [189].

A motion saliency detection method was proposed by Xue *et al.* [174] that is based on Robust Principle Component Analysis (RPCA). The general idea of the method is to express a video, or a time-slice of a video, as a 3D tensor of 2D colour frame images stacked along a third time dimension, on which RPCA is performed to determine regions containing motion. Low-rank and sparse matrix decomposition is performed on the tensor input to find foreground moving objects. This method operates under the assumption that foreground motion objects occupy only a fraction of the frame image pixels and will thus be treated as sparse errors which are identified by the sparse matrix. Examples of such objects are cars and pedestrians.

The performance of the method [174] is compared to the frame difference method, GMM [182] and Temporal Spectrum Residual (TSR) [191]. The reported results show that the frame difference method is heavily affected by noise whilst GMM suffers from ghosting in the initial frames, which means background regions are falsely identified as salient, due to requiring multiple frames to build the background model. The results also showed that GMM fails entirely when the camera is in motion due to the changing background. The proposed method outperforms all four methods, including TSR, in all the experimental setups.

This method has the significant advantage of not requiring training data, which allows it to be adopted in a HAR pipeline without increasing the training set size requirements. However, the operation it's based on, namely RPCA, is a computationally expensive operation, which reduces the benefit of adopting the method in a HAR pipeline.

BSUV-Net 2.0 [183] is a recent background subtraction method that improves on the original BSUV-Net [189]. The building block forming the basis of BSUV-Net is the U-Net-based [192] CNN, which produces a probabilistic foreground estimate from the concatenation of three input images [183]. BSUV-Net takes as input two background images, referred to as "empty" and "recent", captured at different time scales and the current frame. The "empty" input is selected such that the frame does not contain any moving objects. The original BSUV-Net method applies a median filter over a large number of initial frames to suppress moving objects present in the

beginning of the video [189]. The “recent” input is obtained from the median of the previous 100 frames.

Each input image contains three colour channels, for the red, green and blue colours, as well as an additional channel containing a foreground probability map (FPM), which is estimated by semantic image segmentation using the DeepLabv3 which is the third version of the DeepLab [193] architecture. DeepLabv3 is trained on the ADE20K [194] semantic segmentation dataset, which consists of 150 class labels and 20,000 densely-annotated images. The class labels of this dataset are divided into two sets representing background and foreground objects. BSUV-Net regards the person, car, cushion, box, book, boat, bus, truck, bottle, van, bag and bicycle class labels as foreground objects whereas the remaining class labels are regarded as background objects.

Let $C = \{c_0, c_1, \ldots, c_{149}\}$ denote the set of class labels in the ADE20K dataset, and p_{c_j} denote the pixel-wise probabilities for given a class label $c_j \in C$, as determined by the SoftMax layer of DeepLabv3. The FPM for a given input frame $\mathbf{I}$ is computed by

$$\mathbf{S}[m,n] = \sum_{c_j \in F} p_{c_j}[m,n] \tag{4.1}$$

where $\mathbf{I}[m,n]$ is the input pixel value at the spatial location m, n, while $p_{c_j}[m,n]$ represents the probability that the pixel at the location m, n belongs to an object with class label c_j, $\mathbf{S}[m,n]$ is the output FPM value for the input pixel at m, n and F is the set of foreground classes.

The three input images are fed into a U-NET type fully connected CNN (FCNN), consisting of an encoder and decoder. The encoder part of the network reduces the spatial dimensions of the input progressively using 2×2 max-pooling operators. The decoder part of the network increases the dimensions to the dimensions of the input using up-convolutional layers with each layer consisting of a transposed convolution with a stride of 2. The layers comprising the encoder portion of the network are connected to the corresponding layers of the decoder with residual connections. This is done to help the network combine the low-level information in the initial layers with the high-level information of the deeper layers. To improve the performance of the network on unseen samples and increase its generalisation capacity, each convolutional layer is followed by a strong batch normalisation [63] and each max-pooling layer is preceded by a spatial dropout [195]. The final layer of the network is a sigmoid activation, which produces an output probability map for each pixel.

The operation of the BSUV-Net is defined as a non-linear map $\mathbf{G}(\mathbf{W}) : \mathbf{X} \rightarrow \hat{\mathbf{Y}}$ from the 12-channel input image $\mathbf{X} \in \mathbb{R}^{w \times h \times 12}$, which consists of the concatenation of the three input images with each input having a width of w, a height of h and four channels, to the output pixel-wise foreground probability $\hat{\mathbf{Y}} \in [0,1]^{w \times h}$. $\mathbf{W}$ denotes the

weights of the neural network **G**.

Given that the background pixels usually outnumber the foreground pixels by a significant factor, the commonly used cross-entropy and mean-squared error loss functions are not suitable for training BSUV-Net [189]. As a result the authors used the Jaccard index as the loss function, which is defined by

$$J_R(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{T + \sum_{m,n}(\mathbf{Y}[m,n]\hat{\mathbf{Y}}[m,n])}{T + \sum_{m,n}(\mathbf{Y}[m,n] + \hat{\mathbf{Y}}[m,n] - \mathbf{Y}[m,n]\hat{\mathbf{Y}}[m,n])} \tag{4.2}$$

where $\mathbf{Y} \in \{0,1\}^{w \times h}$ is the ground truth for the input sample **X**, T is a smoothing parameter and m, n are the spatial dimensions.

The authors [183] proposed a novel strategy for preventing overfitting by introducing illumination changes between the empty reference frame $\mathbf{R}_E$ and the current frame. The reference frame augmented with illumination changes $\hat{\mathbf{R}}_E$ is computed by

$$\hat{\mathbf{R}}_E[m,n,c] = \mathbf{R}_E[m,n,c] + \mathbf{d}[c] \qquad \text{for } c = 1,2,3 \tag{4.3}$$

where **d** represents a frame-specific global illumination model change that is randomly chosen for each training sample.

BSUV-Net 2.0 [183] improves on the original BSUV-Net [189] by adding spatiotemporal data augmentations, which modify the training dataset such that it includes samples which simulate challenging scenarios. A spatially-aligned crop is added to reduce all training samples to the same size. This is done to increase the training speed since multiple samples can be processed in parallel, by the GPU, if they are the same size. A random shift is added to simulate camera jitter, since few public datasets contain examples of camera jitter. Data augmentations are also defined to simulate camera zoom and a moving camera.

The performance of BSUV-Net 2.0 is evaluated on the CDNet-2014 dataset [196] and compared to the original BSUV-Net [189] and a number of state of the art background subtraction methods, namely IUTIS-5 [197], WisenetMD [198] and FgSegNet v2 [188] in terms of recall (Re), specificity (Sp), false positive rate (FPR), false negative rate (FNR), percentage of wrong classifications (PWC), precision (Pr) and F-score. BSUV-Net 2.0 achieved the highest F-Score of 0.8387 as well as the highest specificity and precision. Additionally BSUV-Net 2.0 achieved the lowest FPR, FNR and PWC of all methods that were analysed. However, the original BSUV-Net, as well as WisenetMD, achieved a higher recall than BSUV-Net 2.0. This indicates they are able to identify more of the foreground object pixels however at the same time the lower precision indicates that the error rate has increased.

While BSUV-Net 2.0 offers a high F-Score and a superior performance to other background subtraction methods, it is a computationally intensive method with a

frame rate of 2.5 on a video with a resolution of 640×480 pixels running on an Nvidia Tesla P100 GPU. The authors propose Fast-BSUV-Net 2.0, which is a variant of BSUV-Net 2.0 that does not use the FPM channel, which is identified as a bottleneck. Fast-BSUV-Net 2.0 takes a nine channel image as input, which consists of three images each containing three colour channels. Fast-BSUV-Net 2.0 achieves an F-Score of 0.8039 on CDNet-2014, which trails closely behind BSUV-Net 2.0 and still outperforms the remaining methods. The frame-rate of Fast BSUV-Net 2.0, on a video with a resolution of 640×480 is 13 which is approximately five times higher than the frame-rate of BSUV-Net 2.0. However, even with the improvements in speed offered by Fast-BSUV-Net 2.0, the frame rate is still too low for real-time processing of video which generally arrives at a frame rate of 25-30 frames per second. Moreover, this method requires a separate training step with a training dataset dedicated to background subtraction. This adds to the overall time required for training the HAR classifier.

4.2 Computationally Simple and Efficient Motion Saliency

Motion saliency detection offers a potential strategy for limiting the data that is processed by a HAR classifier to the regions which capture the action being carried out. This eliminates background clutter which reduces the size of the data that is processed and hence reduces the time and computing resources required for training. In order for this benefit to be realised it is essential that the motion saliency detection technique is computationally efficient so that the reduction in training time is not offset by the time consumed by motion saliency detection. The state of the art motion saliency detection and background subtraction methods that have been explored, namely the RPCA based method proposed by Xue *et al.* [174] and BSUV-Net 2.0 [189] are computationally expensive. Furthermore, the latter method is unsuitable for reducing the time spent by a HAR system since it requires training the background subtraction network in addition to training the HAR classifier. Due to the specific requirements and the limitations of existing solutions, a new motion saliency detection solution was developed, an overview of which is shown in Figure 4.2.

The difference of frames was identified as a computationally simple and efficient method for identifying regions in a video that have changed between frames. The idea behind this method is that regions containing motion change significantly between frames while those regions without motion remain the same throughout the frames of the video. Thus a high difference between frames is observed in regions with motion, while the difference is close to zero in regions without motion. The frame difference between adjacent frames will likely result in a low difference for slow-moving objects,

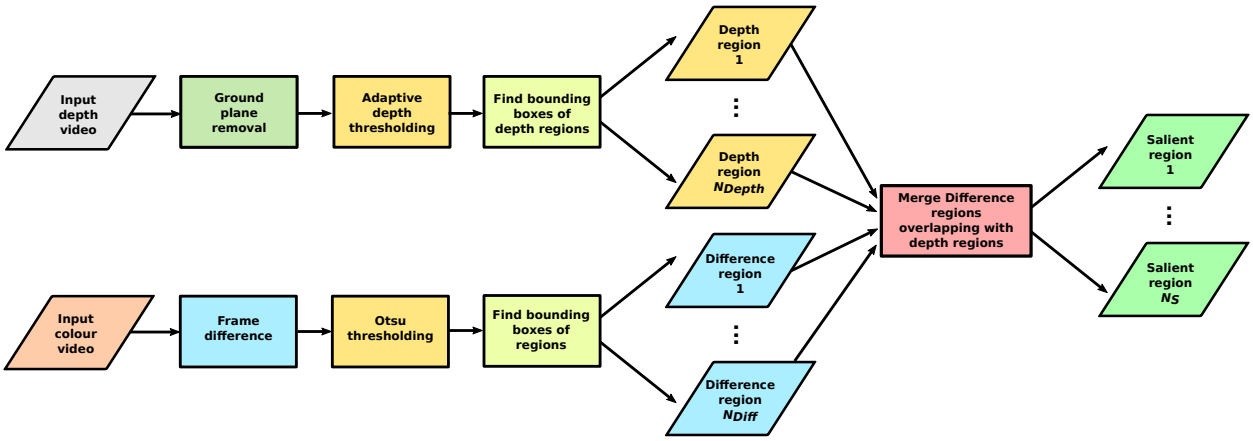


Figure 4.2 Pipeline of proposed computationally efficient motion saliency detection solution

and thus these regions will not be distinguishable from background regions. This becomes especially apparent in video captured at higher frame rates, greater than 25 frames per second, where the object motion between adjacent frames is minuscule. Thus, the difference across multiple frames, for example every five frames, is computed to increase the difference in the pixel intensity values of motion regions while maintaining a small frame difference in the background and near stationary objects.

The output of frame difference is a motion saliency map where bright regions indicate areas of significant motion, and dark regions indicate static non-moving regions. Otsu thresholding [199], which is a technique for automatically determining the threshold used to separate the foreground from the background in grey-scale images, is used to convert this saliency map to a binary mask where the pixels of motion regions have a value of 1 and the remaining pixels having a value of 0.

4.2.1 Using depth to reduce the effect of noise

The frame difference method alone has been shown to be heavily affected by noise and background clutter [174]. An example of the effect of noise on the results of frame difference is shown in Figure 4.3 on the sample `s01_e01` from the UTKinect [200] dataset. The method also picks up perturbations in the background clutter which are usually not relevant to the video processing task at hand. As a result, the saliency map produced by frame difference is only useful as a rough estimate. Frame difference alone is not suitable for identifying the regions that capture a human action.

RGB+D, red green blue plus depth, video provides additional depth information that can be exploited to limit the regions detected by frame difference to those, which represent actual motion and not noise. Each frame has an associated depth map where

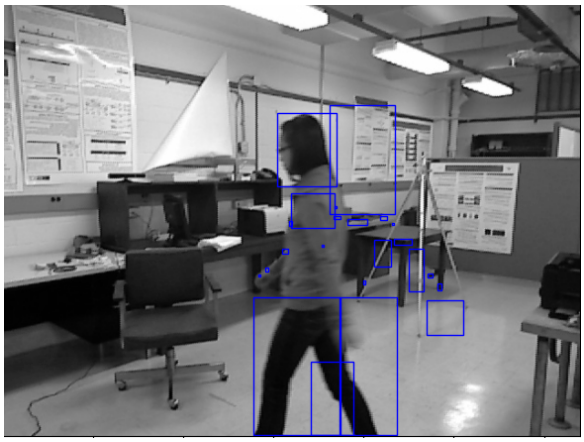


Figure 4.3 Noise regions detected using frame difference

the intensity value of each pixel represents the distance between the camera and the object on the scene. Besides offering a potential strategy for eliminating regions caused by noise, depth maps are invariant to illumination changes, thus they are effective in eliminating false motion regions caused by sudden illumination changes, such as shadows appearing due to the sun coming out from behind the clouds. The developed motion saliency method exploits the depth map to determine the bounding box of the foreground object. Since the object pixels are necessarily at a depth, which is within a certain range, a simple thresholding technique is effective in determining the region comprising the foreground object. This does not produce a mask which accurately covers the foreground object, but produces a rough bounding box, which is sufficient for reducing the size of the data that is processed by the HAR classifier.

Determining the depth threshold

Experimentation with a hard-coded threshold, chosen for a particular video, showed that this technique is effective for separating the foreground object from the background in most frames. However, a hard-coded threshold is scene dependent and has to be manually selected for each video, which is not practical in real-world applications. Ideally an optimal threshold is chosen for each frame automatically without any user input.

The multiple object tracking algorithm proposed by Yu *et al.* in [201] showed that an effective depth threshold, for separating the foreground from the background, can be selected automatically by analysing the histogram of the depth pixel intensity values. The idea behind the method is that each object occupies a specific range of values along the depth axis, with most of the object pixels falling at a single value. It is this value that can be determined by finding the locations of the local maxima in the histogram of depth pixel intensity values.

The set of local maximum V, referred to as the discontinuity points, is defined by

$$V = \{i \mid b_i \geq b_\mu, \forall \mu \in \{i-1, i+1\}\} \tag{4.4}$$

where b_i is the number of pixels with intensity values falling in bin i of the histogram.

The depth values are divided into N_V ranges, where N_V is the number of discontinuity points in V. Each range is defined by

$$[l_i, h_i) = \left[\frac{v_{i-1} + v_i}{2}, \frac{v_i + v_{i+1}}{2}\right) \tag{4.5}$$

where l_i and h_i are the lower and upper bounds, respectively, of the i^{th} depth range and v_i is the i^{th} discontinuity point. The number of ranges provides an indication of the number of objects in the scene, however this includes objects forming part of the background which are not of interest to motion saliency detection.

The regions identified by frame difference provide an approximate estimate of the areas in the video which contain motion. These regions can thus be used to determine which of the depth ranges pertain to an object of interest and which ranges are caused by noise and clutter. It is assumed that most of the non-zero pixels in the binary mask identified by frame difference belong to a moving object. Thus the depth-range for separating the moving object from the background, can be found by identifying the depth range $[l_t, h_t)$ that contains the median depth of the pixels in the regions identified by frame difference. The depth range containing the moving object is the range satisfying

$$l_t \leq d_m < h_t \tag{4.6}$$

where d_m is the median depth of the pixels within the regions identified by frame difference, l_t and h_t are the lower and upper bound of the depth range. This range is used to separate the foreground object from the background. This range is applied as a threshold on the depth map using

$$t_{x,y} = \begin{cases} 1, & \text{if } l \leq d_{x,y} < h \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

where $t_{x,y}$ is the pixel at (x, y) of the resulting thresholded image and $d_{x,y}$ is the value of the pixel at (x, y) of the depth map. Each contiguous group of non-zero pixels in the binary mask represents an object region.

The set of regions identified by frame difference is reduced to only those regions which intersect with an object region in the binary mask produced by thresholding using (4.7). The regions which do not intersect with any object region are discarded. Additionally the bounding boxes of the kept regions, which intersect with a

single common object region, are merged into their union box. The results of depth thresholding, in a single frame of sample `s01_e01` from the UTKinect dataset, are shown in Figure 4.4.

(a) Depth map (b) Thresholded depth map

Figure 4.4 Depth thresholding

This method was found to produce a single or small number of bounding boxes covering the regions with significant motion. However, the ground plane is often retained by the thresholding procedure, defined by (4.7), resulting in a large bounding rectangle covering the entire ground. To rectify this issue, the ground plane has to be removed before thresholding.

Removal of the ground plane

The ground plane is the region in the depth map which represents the ground or floor of the scene. The specific characteristics of the ground plane in depth images allows it to be identified and removed [202]. A solution for removing the ground plane from depth images, based on these characteristics, was proposed by Khan Hoa *et al.* in [202]. The solution is based on the observation that the pixels forming the ground plane have a gradient of zero along the x-axis and a non-zero gradient along the y axis. The ground plane is identified by searching for pixels which satisfy these conditions.

The first step of the solution is to compute the gradient of the depth image in the x and y directions, which produces two depth gradient maps representing the x and y gradients, respectively. The work [202] proposes a procedure for computing these gradient maps using the focal length and camera parameters of the capturing system. Let $\mathbf{D} \in \mathbb{R}^{h \times w}$ denote the depth image of dimensions $h \times w$ and $d_{i,j}$, where $i \in [0, h)$ and $j \in [0, w)$, denote the value of the pixel at the spatial location (i, j) within the depth map $\mathbf{D}$. Let $\mathbf{G}^{(x)} \in \mathbb{R}^{h \times w}$ and $\mathbf{G}^{(y)} \in \mathbb{R}^{h \times w}$ denote the x and y gradient depth maps, respectively, with $g^{(x)}_{i,j}$ and $g^{(y)}_{i,j}$ denoting the value of the pixel at the spatial location (i, j) within the gradient depth maps $\mathbf{G}^{(x)}$ and $\mathbf{G}^{(y)}$, respectively.

A binary mask $\mathbf{M} \in \mathbb{R}^{h \times w}$, where ground pixels have a value of 1 and non-ground

pixels have a value of 0, is constructed using

$$m_{i,j} = \begin{cases} 1 & \text{if } g_{i,j}^{(x)} = 0 \text{ and } g_{i,j}^{(y)} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

where $m_{i,j}$ denotes the value of the pixel at the spatial location (i, j) within **M**.

The results of this procedure alone are noisy with many ground pixels classified as non-ground. This creates holes in the ground plane. The work [202] proposes a block grouping procedure to filter out the non-ground pixels classified as ground and fill the holes in the ground plane. The mask **M** is divided into blocks of size $h_b \times w_b$ with $n_b = h_b \cdot w_b$. In each block, the number of ground pixels n_g is determined and the ratio R between the number of ground pixels and the total number of pixels in the block is computed using

$$R = \frac{n_g}{n_b} \tag{4.9}$$

All blocks with a ratio R greater than a threshold T_r are considered ground blocks. The values of the pixels within all ground blocks are set to 1. As a final step all contiguous regions of non-ground pixels that have less than T_s pixels are marked as ground. This fills small holes in the ground plane.

To adapt this procedure for motion saliency detection, some changes are necessary. The focal length and camera parameters, used to compute the depth gradient maps $\mathbf{G}^{(x)}$ and $\mathbf{G}^{(y)}$, are not always available and notable RGBD datasets, such as UTKinect [200] and NW-UCLA [125], do not provide these parameters. It was found that the Sobel edge detection operator [203] computes a suitable approximation for $\mathbf{G}^{(x)}$ and $\mathbf{G}^{(y)}$. $\mathbf{S}_x$, defined by (4.10), is convolved with the depth image **D** to produce $\mathbf{G}^{(x)}$, while $\mathbf{G}^{(y)}$ is produced by convolving **D** with $\mathbf{S}_y$, which is defined by (4.11).

$$\mathbf{S}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$\mathbf{S}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.11}$$

The depth gradient maps are, thus, computed using

$$\mathbf{G}^{(x)} = \mathbf{D} * \mathbf{S}_x \tag{4.12}$$

$$\mathbf{G}^{(y)} = \mathbf{D} * \mathbf{S}_y \tag{4.13}$$

where $*$ denotes the 2D linear convolution operator.

Through experimentation it was found that the optimal block size $b_h \times b_w$, for the block grouping procedure is 3×3. Larger block sizes result in regions corresponding to the foreground object being marked as ground, especially those that are close to the ground plane. Smaller block sizes result in most holes not being filled. The optimal threshold T_r for the ratio R was found to be 0.4. This results in an entire block being marked as ground if at least 40% of the block's pixels are marked as ground. Larger thresholds result in a significant number of unfilled holes, whereas smaller thresholds result in foreground object regions being marked as ground. The addition of a morphological closing and opening step prior to block grouping was found to significantly reduce the number of holes in the ground plane, particularly holes larger than the block size.

A value of 100 pixels was found to be optimal for the threshold size T_s. This means all contiguous regions of non-ground pixels consisting of less than 100 pixels are marked as ground. This fills remaining holes in the ground plane that are not identified by the block grouping procedure. This value was determined based on the UTKinect dataset, which has a resolution of 640×480 pixels. However, a larger threshold value may be required for larger video resolutions.

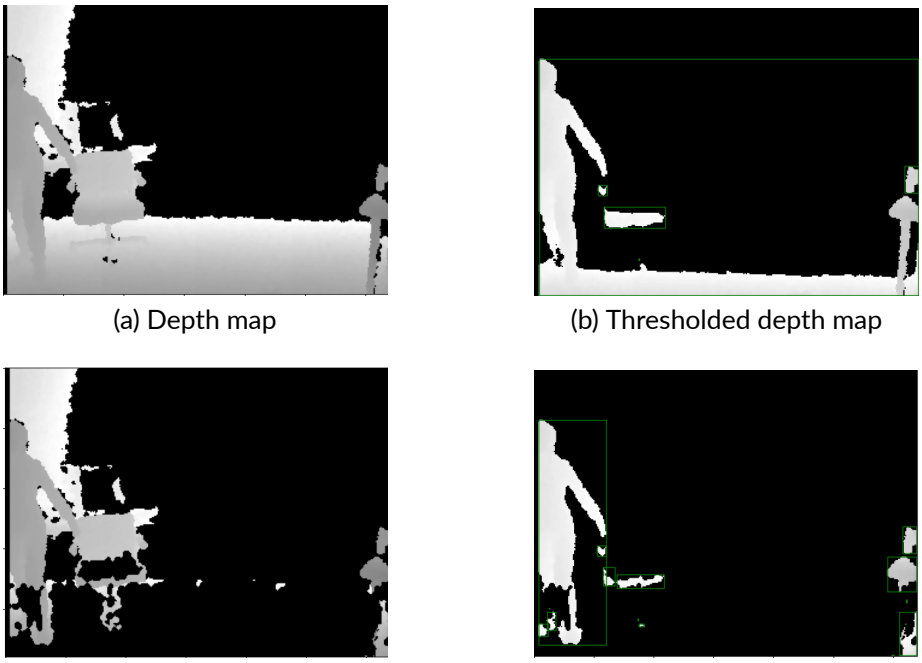
Ground plane removal significantly improves the results of depth thresholding. Without ground plane removal the entire ground plane is retained by the depth thresholding which results in a single object region covering the entire ground, as shown in Figure 4.5b. With ground plane removal, the object regions that are identified cover only the individual foreground objects, as shown in Figure 4.5d.

4.2.2 Improving performance with Kalman Filtering

The motion saliency solution was found to perform well in most cases. However, the presence of noise and defects in the depth map results in multiple small regions being identified that cover only parts of the foreground object. An example of this is shown in Figure 4.6. Ideally a single region that covers the entire object should be identified. This section explores the use of Kalman Filtering [204] to track salient regions through time and merge identified regions belonging to the same object into a single region.

Kalman Filtering

Kalman filtering is a method for estimating the true state of a process based on observable measurements [204, 205]. This is widely used in object tracking to estimate the position and the velocity of the object of interest through time. The Kalman filter arrives at an estimate of the process state using a two step process consisting of a *predict* step and an *update* step. The process state is modelled by a Gaussian



(a) Depth map

(b) Thresholded depth map

(c) Depth map after ground removal

(d) Thresholded depth map after ground removal

Figure 4.5 The effect of the ground plane on depth thresholding

Figure 4.6 Visual results using motion saliency showing multiple regions identified that pertain to the same object

distribution with mean at $\mathbf{x}_k$ which is the vector representing the process state at time k, and covariance $\mathbf{P}_k$. Given the process $\mathbf{x}_{k-1}, \mathbf{P}_{k-1}$ at time $k-1$, the process state at the next time step k is predicted using

$$\hat{\mathbf{x}}_k = \mathbf{F}\mathbf{x}_{k-1} \tag{4.14}$$

$$\hat{\mathbf{P}}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\top + \mathbf{Q} \tag{4.15}$$

where $\mathbf{F}$ is the state transition matrix, which describes how the process state is derived from its previous state based on a model of the process and $\mathbf{Q}$ is the process noise covariance matrix, which is based on a model of the process noise. $\hat{\mathbf{x}}_k$ is the predicted process state vector and $\hat{\mathbf{P}}_k$ is the covariance matrix of the prediction, which can be interpreted as the confidence.

The process state prediction is then corrected using a measurement of the state

$\mathbf{z}_k$ at time k and a model of the measurement noise. The *update* step is defined by

$$\mathbf{K} = \hat{\mathbf{P}}_k\mathbf{H}^\top(\mathbf{H}\hat{\mathbf{P}}_k\mathbf{H}^\top + \mathbf{R})^{-1} \tag{4.16}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k) \tag{4.17}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\hat{\mathbf{P}}_k \tag{4.18}$$

where **H** is the measurement matrix relating the measurement vector to a process state vector, **R** is the measurement noise covariance matrix and **I** is the identity matrix. **K** is the Kalman gain parameter, defined in terms of **Q** and **R**, which is a matrix that controls how much weight is given to the measurement $\mathbf{z}_k$ and how much weight is given to the predicted state $\hat{\mathbf{x}}_k$. A process noise **Q** that is higher than the measurement noise **R** results in more weight being given to the measurement. Conversely, a measurement noise **R** that is higher than the process noise **Q**, results in more weight being given to the predicted state.

In object tracking the process state is defined in terms of the position of the object of interest (x_k, y_k) and its velocity $(\dot{x}_k, \dot{y}_k)$ [205]. This is defined as

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^\top \tag{4.19}$$

Generally a constant velocity model perturbed by Gaussian noise is assumed, where the predicted object position at time k is defined as the sum of the object's position and velocity at time $k-1$, while the predicted velocity at time k is the same as the velocity at time $k-1$ [205]. The state transition matrix is, thus, defined by

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

The measurement of the object's position $\mathbf{z}_k$ is obtained by an object detection algorithm such as GMM background subtraction, while the measured velocity is the difference between the object's previous position and the measured position [205]. The measured state is mapped directly to the state vector, thus the measurement matrix **H** is the identity matrix.

This tracking algorithm can be extended to multiple object tracking by designating a Kalman filter to each object in the scene. The measurements are still obtained by an object detection algorithm, however each measurement has to be associated with the correct Kalman filter that is designated to tracking the object. This is achieved by minimising the total Euclidean distance between the prediction positions, of the Kalman filters, and the measured positions associated to the filters.

The aim is to find a mapping $f : \mathbf{P} \rightarrow \mathbf{M}$ between the set of predicted positions **P** to the set of measurements **M**, such that the following loss function is minimised

$$\mathcal{L}_{\text{distance}} = \sum_{i=1}^{N} \|p_i - f(p_i)\| \tag{4.21}$$

where N is the number of Kalman filters, $p_i \in \mathbf{P}$ is the position predicted by the i^{th} filter and $f(p_i)$ is the associated measurement. An optimal mapping f is found using the Hungarian algorithm [119].

Tracking salient regions with Kalman filtering

This section details the addition of Kalman filtering to the motion saliency detection solution to track the identified regions through time with the objective of grouping multiple regions that cover different parts of a human body into a single region covering the entire person. An overview of the Kalman filter tracking pipeline is shown in Figure 4.7 and the steps of the algorithm are shown in Algorithm 1.

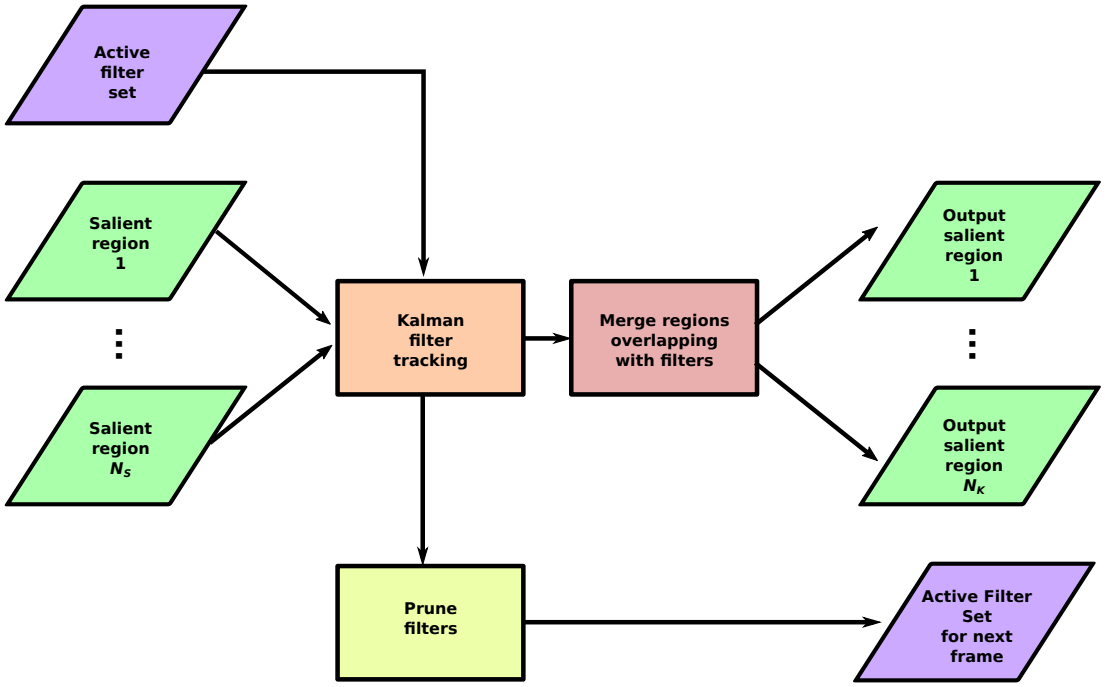


Figure 4.7 Pipeline showing tracking of salient regions using Kalman filtering

In the initial frame, the motion saliency detection solution is run to identify the salient regions. A new Kalman filter is created to track the position of the largest identified region with the filter state initialised to the position of the region. The strategy throughout the remaining frames is to use the regions identified by motion saliency detection as measurements, while using the Kalman filter to predict and track

Algorithm 1 Tracking Algorithm

1: **R** $\leftarrow$ Set of identified object regions
2: **F** $\leftarrow$ Set of active Kalman filters
3: **for all** $f \in$ **F do**
4: predict(f)
5: **end for**
6: **S** $\leftarrow$ assign(**R**, **F**) ▷ Map each region to a filter
7: **for all** $(r, f) \in$ **S do**
8: update(f, r) ▷ Update filter f using measurement r
9: **end for**
10: **U** $\leftarrow \{r \mid r \notin$ **S**$\}$ ▷ Set of regions not assigned to a filter
11: $r_l \leftarrow$ largest region in **U**
12: $f_l \leftarrow$ **new** filter(r_l) ▷ Create new filter using r_l
13: **F** $\leftarrow$ **F** $\cup \{f_l\}$ ▷ Add new filter to active filters set

the associated salient regions through time. The salient region, identified by motion saliency, that is closest to the position predicted by the Kalman filter is used as the measurement in the Kalman update step. Similar to the procedure in the first frame, a new Kalman filter is created to track the largest salient region that is not associated with any Kalman filter.

In subsequent frames the identified salient regions are associated to the active Kalman filters by minimising the total Euclidean distance, defined by (4.21), between the positions of the identified regions and the positions predicted by their associated Kalman filters. Once a filter is created, its window is set to the size of the associated salient region and kept constant throughout the video while the filter is active.

Motion saliency detection identifies one or more bounding rectangles surrounding the salient regions. However the Kalman state vector consists of a 2D point representing the position and a 2D vector representing the velocity. Thus, a single point has to be retrieved from the bounding rectangle of the salient region in order for it to be used as a measurement. The centre of the bounding rectangle is the simplest point to use, however this was found to be highly dependent on the accuracy of the identified salient region. A drift was observed, where the Kalman filter starts moving away from the object when the centre of the bounding rectangle does not fall exactly on the centre of the object. The centre of mass of the non-zero pixels with the region of the thresholded depth map bounded by the bounding rectangle, was found to be a much more robust measurement, that is not affected by inaccuracies in the identified region and sudden changes in the movement and size of the object. Thus, the centre of mass is used as the position measurement. An example from the UTKinect dataset demonstrating how the centre of mass lies significantly closer to the actual centre of the object, is shown in Figure 4.8.

It was observed that due to noise and defects in the depth map, salient regions

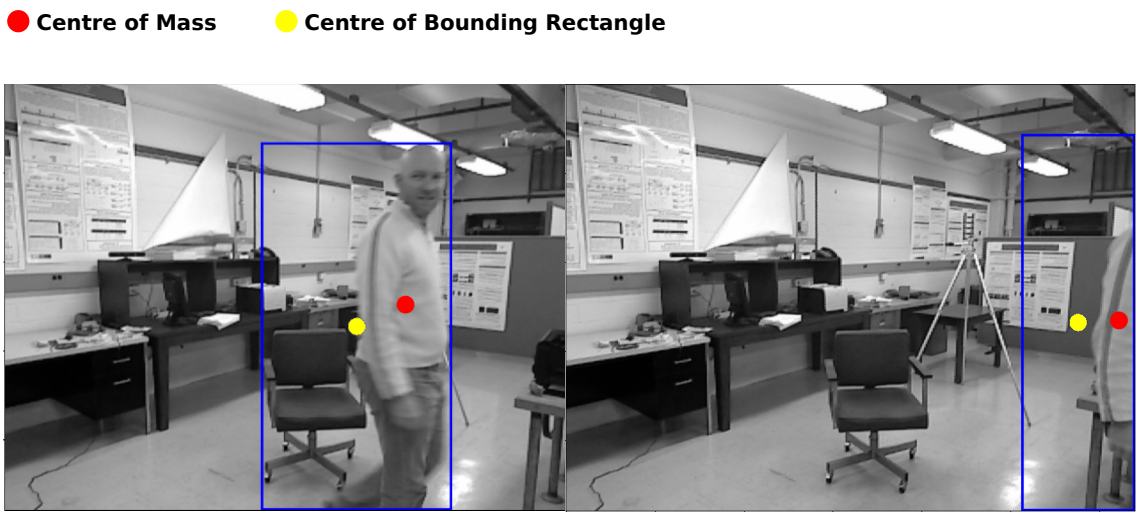


Figure 4.8 Comparison of bounding rectangle centre (yellow) with centre of mass (red) as position measurement

appear occasionally for a small number of frames and then disappear afterwards, particularly if multiple salient regions are identified that pertain to a single object. These filters have a negative affect on the results if they are not removed when no salient region is associated with them. On the other hand, filters which have been active for a longer time period should not be removed if no region is associated with them in a single frame, since the object may still be on the scene. To resolve these issues, each filter is given an age indicating the number of time steps for which it has been active. Filters with an age of less than a starting threshold t_s are removed from the active filter set if no region is assigned to them in a given frame. Furthermore, only filters with an age greater than the grouping threshold t_g are considered in the subsequent grouping step. Through experimentation it was found that the best performance is achieved with $t_s = 5$ and $t_g = 2$.

To avoid tracking objects which have left the scene, filters which overlap with the edge of the frame are removed. A filter is removed from the active set if its tracking window overlaps with the edge of the image and at least 50% of the pixels, of the thresholded depth map, are non-zero in the subregion making up half of the window in the direction of the edge. If no Kalman filters satisfy these constraints, the subsequent grouping step is not performed and the output of motion saliency detection is used as a fallback.

Once the update step is performed for each Kalman filter, using the measurements obtained by motion saliency detection, the state vector of each filter, which encodes the position and velocity of the tracked object, is used to group multiple salient regions pertaining to a single object into a single region covering the entire foreground object. For each Kalman filter, the set of salient regions which

overlap with the filter's windows is determined. The filter window is centred at the position encoded in the filter's state vector and, as mentioned earlier, it's size is kept constant for as long as the filter is active. The bounding rectangles of the regions that overlap with the filter window are merged into a single bounding rectangle covering their union box. This results in a single region covering more of the object instead of multiple regions covering only part of it. Figure 4.9 shows an example of the region obtained, which is shown in red, by grouping the salient regions, shown in blue, that overlap with the Kalman filter window. The figure also shows that merging all regions identified by motion saliency does not give equivalent results since not all identified regions belong to the same object.

Figure 4.9 Example of bounding rectangle (red) covering union box of regions (blue) that overlap with Kalman filter window

Experimentation with various values of the process **Q** and measurement noise **R** parameters showed that due to the unpredictable nature of the object motion, the best results are achieved when the process noise is greater than the measurement noise, that is $\mathbf{Q} > \mathbf{R}$. When the measurement noise is greater than the process noise, the filter was observed to lose track of the object during sudden changes in velocity. The best results were achieved with a discrete constant white noise model with **Q** having a Gaussian distribution with a variance of 100 and $\mathbf{R} = 10^{-5}\mathbf{I}$, where **I** is the identity matrix.

4.3 Evaluation and Results

The developed solution was evaluated on the UTKinect [200] dataset. This dataset was chosen since it is widely used in the evaluation of depth and skeleton-based human action recognition solutions, which is the application where the motion saliency

detection solution is intended to be used. The visual results using motion saliency detection (without Kalman filtering) on test sequence `s01_e01` are shown in Figure 4.10.

Figure 4.10 Visual motion saliency detection results on s01_e01 sequence of UTKinect dataset

The performance of the solution is evaluated in terms of the recall (Re), precision (Pr) and F-measure metrics that are described in [206] and used in the evaluation of [183, 189]. The recall gives an indication of how many of the true salient regions are identified relative to the total number of true salient regions. A high recall indicates that the solution is able to identify most of the actual salient regions. The precision gives an indication of ratio of the true salient regions to the total number of regions that are identified. A high precision indicates that the error rate of the solution is low and most of the identified regions are true salient regions, whereas a low precision indicates a high error rate. The F-measure combines the two metrics.

The recall is defined by

$$\mathrm{Re} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.22}$$

where TP is the number of true positives and FN is the number of false positives. A true positive is considered when the intersection over union (IOU) between the ground truth labels and identified salient regions is greater than 0.5.

The precision is defined by

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.23}$$

where FP is the number of false positives. A false positive is a salient region, identified with the solution, for which the IOU with the ground truth is less than 0.5.

The F-measure is defined by

$$\text{F-measure} = \frac{\mathrm{Pr} \cdot \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}} \tag{4.24}$$

The performance of the solution is compared to GMM [182] background subtraction, which is widely used due to its computational efficiency, and the fast

Table 4.1 Performance results of proposed motion saliency solution compared to BSUV-Net 2.0 and GMM

Method	Re	Pr	F-measure
Fast-BSUV-Net 2.0	0.84	0.73	0.78
GMM	0.34	0.77	0.47
Motion saliency alone	**0.88**	**0.78**	**0.83**
Motion saliency with Kalman filter	**0.90**	**0.78**	**0.83**

variant (Fast-BSUV-Net 2.0) of BSUV-Net 2.0 [183], which is the state of the art in background subtraction. The results, presented in Table 4.1, show that the proposed method, with and without Kalman filtering, outperforms both GMM and BSUV-Net 2.0 in all three evaluation metrics. The higher recall, compared to GMM and BSUV-Net 2.0, indicates that the proposed method is able to identify more salient regions, while the higher precision indicates that its error rate is lower. The higher F-measure highlights the overall superior performance of the proposed method.

The addition of the Kalman filter improves the recall of motion saliency detection from 0.88 without Kalman filtering to 0.9 with Kalman filtering. This indicates that motion saliency with Kalman filtering is able to identify salient regions which motion saliency alone is unable to. This becomes more significant when the IOU threshold, for an identified salient region to be considered a true positive, is higher, which indicates that on average motion saliency with Kalman filtering identifies a larger area of the salient regions. Figure 4.11 shows how the recall varies with the IOU threshold. From Figure 4.12, which shows how the recall varies with the precision, it is evident that with the Kalman filter addition a higher recall is observed for the same precision, which in turn indicates that the increase in recall does not result in an increased error rate. This presents an advantage when using motion saliency for reducing the size of the data that is processed by the HAR classifier, since the additional regions that motion saliency with Kalman filtering is able to identify may be relevant to the classification task and their successful indication could potentially result in increased classification accuracy. The visual results on four consecutive frames of test sequences `s02_e01`, `s05_e01`, and `s07_e01` are shown in Figure 4.13, with Figures 4.13(a), 4.13(c) and 4.13(e) showing the results obtained using the motion saliency detection algorithm alone, and Figures 4.13(b), 4.13(d) and 4.13(f) showing the results obtained using motion saliency with Kalman filtering. These results also show that with Kalman filtering a single region covering the entire foreground object is found whereas with motion saliency alone multiple smaller regions covering only parts of the foreground object are found.

The experiments were performed on an 8-core Intel Core i7-11800H with an

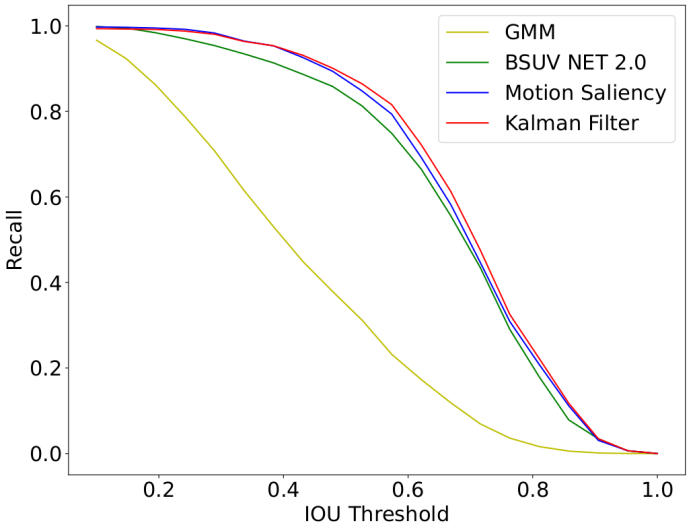


Figure 4.11 Graph showing how recall varies with IOU threshold

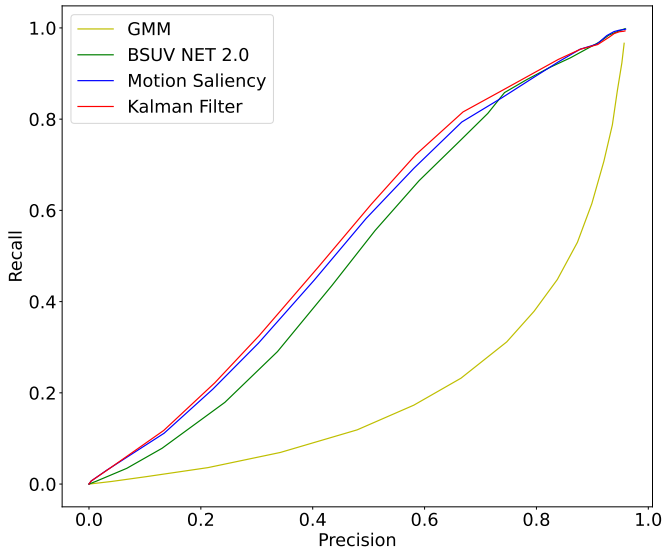
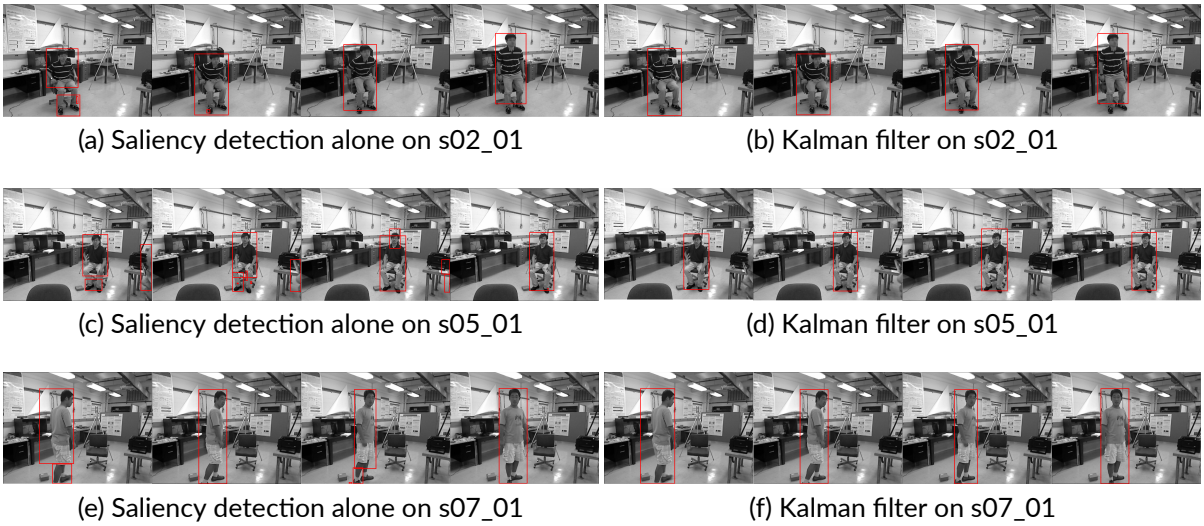


Figure 4.12 Graph showing how recall varies with precision



(a) Saliency detection alone on s02_01 (b) Kalman filter on s02_01

(c) Saliency detection alone on s05_01 (d) Kalman filter on s05_01

(e) Saliency detection alone on s07_01 (f) Kalman filter on s07_01

Figure 4.13 Visual results comparing motion saliency with and without Kalman filtering

Table 4.2 Execution time of proposed motion saliency solutions compared to BSUV-Net 2.0 and GMM

Method	Frames per second (FPS)	Time per frame (milliseconds)
Fast-BSUV-Net 2.0	9.79	102
GMM	12.89	78
Motion saliency alone	**25.78**	**39**
Motion saliency with Kalman Filter	**24.79**	**40**

Nvidia GeForce RTX 3060 graphics card and 32 GB of RAM running Ubuntu 20.04. The average execution times of the algorithms, evaluated on videos with a resolution of 640×480, are shown in Table 4.2. The proposed motion saliency method (without Kalman filtering) has an average frame rate of 25.78 frames per second which is significantly higher than the frame rates of both BSUV-Net 2.0 and GMM, which are 9.79 and 12.89 frames per second, respectively. The addition of Kalman filtering introduces a small computational penalty, with the processing time per frame increasing by 1 millisecond. However, this cost is negligible as the frame rate of motion saliency with Kalman filtering, 24.79 is still drastically higher than the frame rates of both BSUV-Net 2.0 and GMM.

4.4 Discussion

The long time required for training a HAR classifier, due to the large training set requirements, was identified as a significant challenge to deploying a HAR system in real-world applications. Motion saliency detection was explored as a strategy for

limiting the data that is processed by the HAR classifier to only those regions which capture the action being carried out. Existing motion saliency solutions were found to be too computationally intensive, which offsets any reduction in computational time achieved by adopting the solution in a HAR pipeline.

A new motion saliency detection solution was developed, which is based on the computationally simple and efficient difference of frames method. Depth data is exploited to improve its accuracy and limit the regions identified by frame difference to only those regions that represent motion in a foreground object. Spurious regions caused by noise are removed. The results, shown in Tables 4.1 and 4.2, show that the method is able to achieve a performance that is superior to the state of the art, while requiring much less computational time. Furthermore, this method does not require a separate training step thus it can easily be incorporated in a HAR pipeline without increasing the training data set size requirements. The results of this research, along with a detailed explanation of the motion saliency method, were published in [4]. A second paper [5] details the addition of Kalman filtering to motion saliency and the results achieved.

5 Reducing HAR Training Time with Motion Saliency

In this chapter the use of the novel motion saliency detection solution, detailed in Section 4.2, for reducing the training time of a HAR system is explored. The first section explores its application in video based HAR methods, while the second section explores the application of motion saliency in the MMNet [1] method.

5.1 Motion Saliency in 3D CNNs

Human action recognition methods based on 3D-CNNs have achieved promising results and have recently begun outperforming their 2D-CNN based counterparts [18]. However, it has been shown that these methods are more challenging to train due to the large dataset requirements, which in turn leads to a long time being spent in training and a large consumption of memory and energy resources [18]. Motion saliency has been identified as a possible strategy for limiting the video data to the relevant regions of the video, which generally reduces its size and therefore, in turn reduces the training time. However, for this to be realised the time taken by the motion saliency detection component should not exceed the time saved by reducing the size of the data.

A variety of 3D-CNN HAR models are available, however the study by Chen *et al.* [18] showed that the network architecture has the largest effect on the classification accuracy. ResNet3D has been shown to be the strongest architecture [18]. The architecture is available in many depths with ResNet3D-18 being the shallowest and ResNet3D-200 being the deepest. ResNet3D-50 is the most widely used architecture since it strikes a balance between classification accuracy and training time requirements [18]. Furthermore, as reported by the study [55], the gain in classification accuracy using deeper architectures is only 0.3% to 2.5%. This section explores the addition of motion saliency detection, for reducing the size of the data that is processed, to the bottleneck ResNet3D-50 [55] HAR classifier.

5.1.1 Training Algorithm

The training algorithm, as detailed by the study [55], for the bottleneck ResNet3D-50 model, shown in Algorithm 2, receives as input a list of N_s training video samples S. This list is divided into batches, with each batch containing N_b samples, where N_b is a parameter provided when launching the training algorithm. Batching is primarily required due to the batch normalisation layers, which serve to reduce overfitting and improve the generalisation ability of the model. However, batching also serves a

secondary purpose of minimising the number of times that data has to be transferred between CPU and GPU memory. Furthermore, when an entire batch is loaded into GPU memory, the model can be trained on all N_b samples in the batch in parallel. This reduces the training time since the training procedure is run $\frac{N_s}{N_b}$ times, whereas without batching the training procedure would have to be run N_s times. However, batching also increases the GPU memory requirements of the system. The larger the batch size the more GPU memory is required. Given that GPU memory is significantly more expensive than CPU memory, while being available in considerably smaller sizes, this limits the batch size that can be used in practice. The study [55], used a batch size of 128, which is specified in the implementation published with the study, for fine-tuning on the UCF-101 [73] and HMDB-51 [72] datasets.

Algorithm 2 I3D-ResNet50 Training Algorithm

```
S = List of training video samples
N_s = Number of training samples
N_b = Batch size
N_c = Number of frames to select from each video

for 1, 2, ..., N_s / N_b do
    B = RandSelect(S, N_b)                    ▷ Randomly select N_b samples from S
    TrainBatch = N_b × N_c × 3 × h × w matrix

    for s_i ∈ 0, 1, ..., N_b do
        s = B[s_i]
        frames = RandSelectFrames(s, N_c) ▷ Randomly select N_c frames from sample

        for f ∈ 0, 1, ..., N_c do
            frame = frames[f]
            frame = RandCrop(frame)    ▷ Crop frame around random spatial location
            TrainBatch[s_i, f] = frame
        end for
    end for

    Train(TrainBatch)                         ▷ Train network using current batch
end for
```

A batch is constructed by randomly selecting N_b samples. The selected samples are exclusive to the batch, that is once they are selected in a given batch, they are not selected in any other batch during the current training epoch. Given that I3D-ResNet50 is a 3D-CNN, the input during training is a 3D tensor constructed by stacking the 2D frames of the training video sample along a third time dimension. N_c frames, referred to as the clip size, are selected from each sample. There are two strategies for selecting the frames, namely sampling N_c consecutive frames or sampling N_c frames from random temporal locations in the video. The latter strategy is adopted

during training, while the former is adopted during inference. Once N_c frames are sampled, each frame is cropped around a random spatial location to a size of $h \times w$ which is common to all samples in the batch. After temporal sampling and spatial cropping an $N_c \times 3 \times h \times w$ tensor is constructed, which represents a single training sample. The N_b training samples are stacked along a fifth dimension to produce the batch input which is an $N_b \times N_c \times 3 \times h \times w$ tensor. In the study [55], 16 frames were selected from each training sample, and each frame was cropped around a random spatial location to a size of 112×112 pixels. Therefore, the batch input is a tensor of size $128 \times 16 \times 3 \times 112 \times 112$.

5.1.2 Motion Saliency Detection Module

As mentioned earlier the input to the network is a tensor containing the randomly sampled frame images of the video clip stacked along a time dimension. Motion saliency detection has to be performed on each of the sampled frames to limit the area to the subject carrying out the action. This presents a number of challenges since motion saliency detection cannot be applied selectively on the sampled frames in isolation, that is motion saliency detection requires processing the frames of the video sequentially from start to finish.

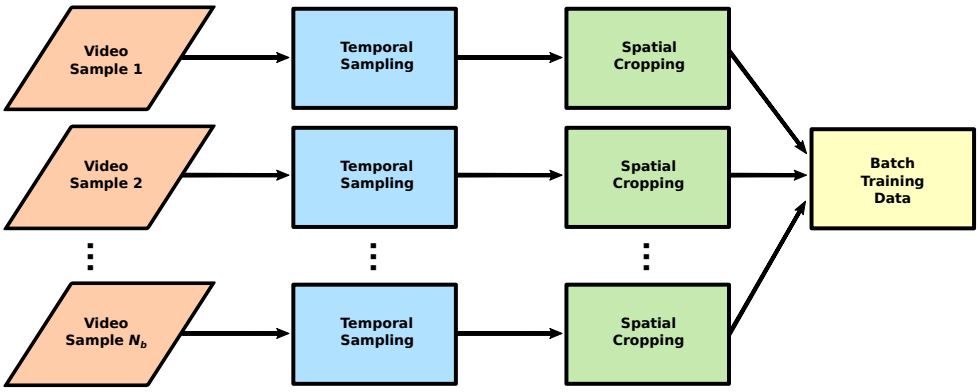


Figure 5.1 3D-CNN batch training data preparation pipeline without motion saliency

Given these constraints, motion saliency detection has to be applied directly after loading the video sample and prior to temporal sampling. Figure 5.2 shows the data preparation pipeline for a single batch, with the position of the motion saliency detection module in red. For comparison, Figure 5.1 shows the data preparation pipeline without motion saliency. These figures also demonstrate that motion saliency detection adds two additional steps to the data preparation pipeline, namely loading the depth data and the motion saliency detection component. As mentioned earlier it is essential that the time taken during these two steps does not exceed the reduction in time achieved by reducing the size of the data that is processed.

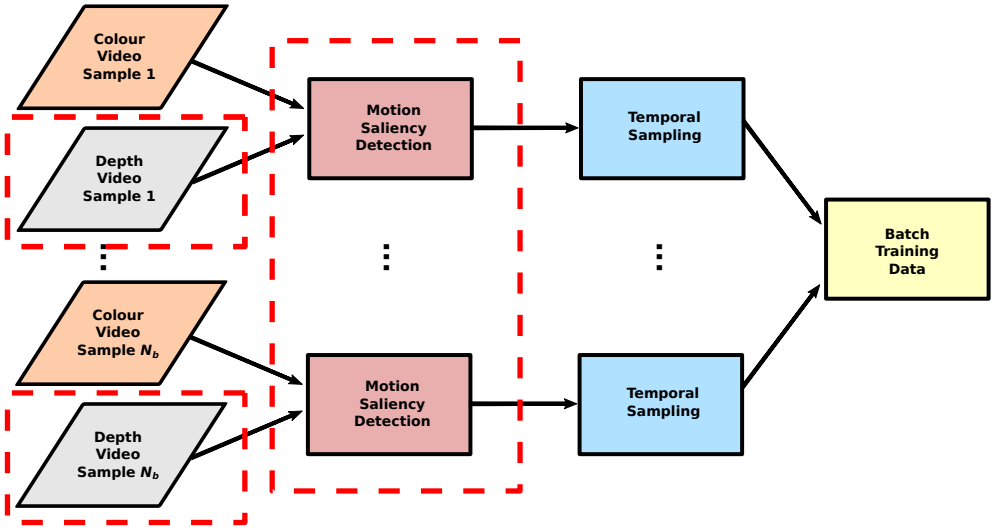


Figure 5.2 3D-CNN batch training data preparation pipeline with motion saliency

The spatial cropping modules are removed since motion saliency detection performs spatial cropping to the relevant regions of the video and thus further cropping is not needed. Given that the training input to the network is a 3D matrix with the frames of the video stacked along a time dimension, the frame images must have the same dimensions. Cropping each frame to its motion salient regions cannot be done since the salient regions are not necessarily of the same size in each frame. Scaling each cropped region to a common size will introduce artefacts caused by upsampling and downsampling, which vary between frames. Moreover, the main benefit of 3D-CNNs lies in their ability to learn temporal features specific to the action class. An example of such a feature is the change in position of the subject during a walking action. Some of these features are lost when cropping each frame to the subject carrying out the action. To preserve these features and produce a list of frames of the same size, each frame is cropped to the union box of all the salient regions identified in all frames, as shown in Figure 5.3. Effectively, this procedure identifies a spatiotemporal salient region in the video that captures the action taking place. The regions outside this spatiotemporal salient region are irrelevant and can be discarded since the action does not take place in these regions.

Batching presents an additional challenge since every training sample in a batch must be of the same size, whereas the cropped spatiotemporal salient region differs in dimensions between training samples. Likewise, scaling this region to a common size will introduce upsampling and downsampling artefacts, which vary between the training samples. To resolve these issues, each sample in the batch is cropped to the union box of the spatiotemporal salient regions, identified by motion saliency, of all samples.

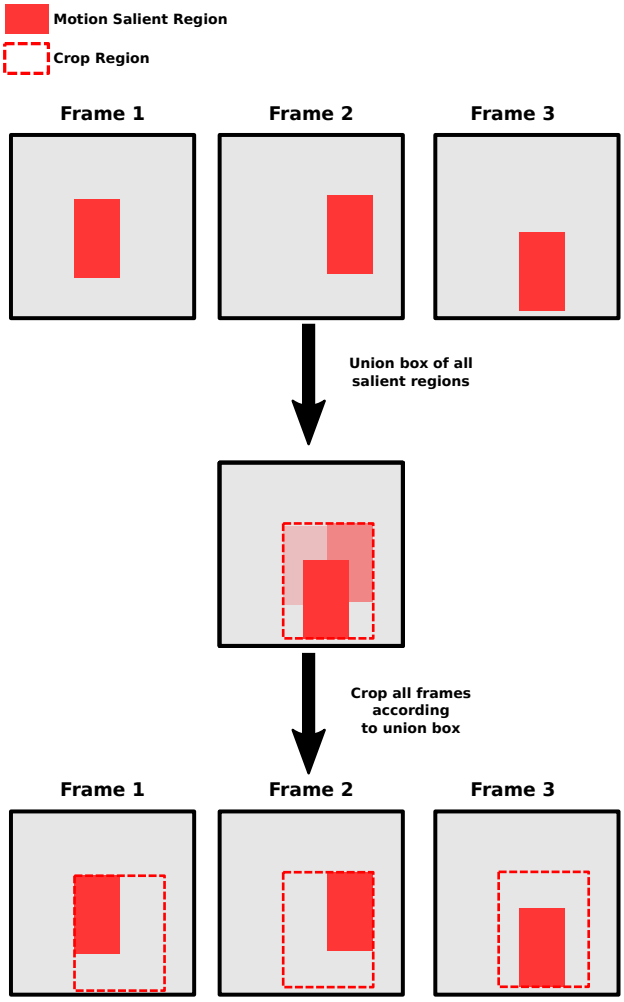


Figure 5.3 Diagram illustrating how the crop region is derived from the identified motion salient regions

5.1.3 Ablation Study

This section details the experimentation done to determine the optimal parameters and configuration of the motion saliency detection module. The objective of the experiments is to determine the effect of the motion saliency detection module on the classification accuracy and training time of the HAR system.

Before experimentation with the motion saliency detection module, it is necessary to establish a baseline performance to determine the classification accuracy that can be expected for a given training set size. The study [55] reports a top-1 classification accuracy of 89.3% on UCF-101 [73] and 61.7% on HMDB-51 [72] after fine-tuning for 200 epochs a ResNet3D-50 model that was pretrained on the Kinetics-400 [31] action recognition dataset. The aim is to replicate this performance using the implementation published with the study [55]. However, it is immediately apparent that the batch size of 128 training samples requires an enormous amount of

Table 5.1 Classification accuracy of fine-tuning Kinetics-400 pretrained ResNet3D model on UCF-101 and HMDB-51 datasets with varying training set sizes

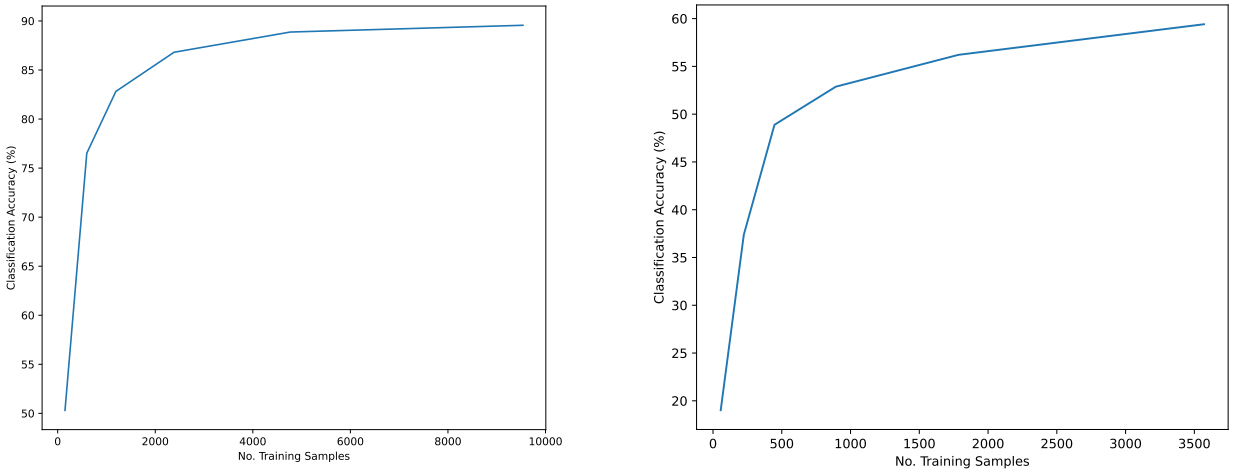
Dataset	Training Set Size	Validation Set Size	Top-1 Accuracy (%)
UCF-101	9537	3783	89.56
UCF-101	4769	3783	88.87
UCF-101	2835	3783	86.81
UCF-101	1193	3783	82.82
UCF-101	597	3783	76.53
UCF-101	150	3783	50.30
HMDB-51	3570	1530	59.41
HMDB-51	1785	1530	56.21
HMDB-51	893	1530	52.88
HMDB-51	447	1530	48.89
HMDB-51	224	1530	37.39
HMDB-51	56	1530	18.20

GPU memory, rendering training unfeasible for most applications. Given the hardware constraints the batch size was reduced to 16 training samples, which is the maximum batch size permitted by the available GPU memory.

The results of fine-tuning for 200 epochs a ResNet3D-50 model, pretrained on Kinetics-400 [31], using varying training set sizes are shown in Table 5.1. The classification accuracy, 89.56%, achieved on UCF-101 using the full training set size of 9537 training samples is higher using the experimental setup with a batch size of 16 training samples than the accuracy reported by the original study [55]. However the accuracy achieved on HMDB-51 59.41%, using the full training set consisting of 3570 samples, is lower than the accuracy reported in the study [55] by 2.29%.

The results also show that the classification accuracy increases logarithmically with the training set size, where a drastic increase is observed initially. With larger training sets the gain in classification accuracy becomes less significant even when doubling the number of training samples. For example, with 4769 training samples a classification accuracy of 88.87% was achieved on UCF-101. However, with approximately double the training samples, 9537 samples, the classification accuracy only increased to 89.56%, which is a gain of less than 1%. This is clearly illustrated in Figure 5.4, which shows that initially the classification accuracy increases rapidly but then the rate of increase gradually decreases.

To evaluate the effect of the motion saliency detection module on the training time and classification accuracy, the original ResNet3D-50 model, without motion saliency detection, is fine-tuned on the UTKinect dataset with a training set consisting



(a) UCF-101 (b) HMDB-51

Figure 5.4 Classification Accuracy vs Training Set Size using ResNet3D-50

Table 5.2 Comparison between classification accuracy with (+MS) and without motion saliency, where (P) indicates the parallelised implementation of the motion saliency module and (PC) indicates parallelisation with caching

Dataset	Method	Batch Size	Top-1 Accuracy	Training Time (hours)
UTKinect	ResNet3D-50	16	30.51	7
UTKinect	ResNet3D-50+MS	16	37.29	22
NTU-60 (300)	ResNet3D-50	16	11.11	142
NTU-60 (300)	ResNet3D-50+MS	16	10.00	288
UTKinect	ResNet3D-50	1	20.34	10
UTKinect	ResNet3D-50+MS	1	27.19	24
UTKinect	ResNet3D-50+MS (P)	1	27.19	9
UTKinect	ResNet3D-50+MS (PC)	1	27.19	4
NTU-60 (300)	ResNet3D-50	1	7.88	452
NTU-60 (300)	ResNet3D-50+MS	1	16.67	211
NTU-60 (300)	ResNet3D-50+MS (PC)	1	16.67	128

of 140 samples and a validation set consisting of 59. The model was also fine-tuned using a reduced subset of the NTU-60 dataset consisting of 300 training samples. These datasets were chosen, since they provide depth data which is required for motion saliency detection. This is referred to as NTU-60 (300). It is then compared to the model with the motion saliency detection module (ResNet3D-50+MS). A low classification accuracy, in the range 18.20% to 50.30% is expected, due to the small training set, however this is sufficient for determining the effect on training time as well as obtaining an initial estimate of the impact on the classification accuracy. The experiments were run on an 8-core Intel Core i7-11800H with an Nvidia GeForce RTX 3060 graphics card and 32 GB of RAM running Ubuntu 20.04.

The results, shown in Table 5.2, reveal that the motion saliency detection module improves the classification accuracy by 7% resulting in a classification accuracy of 37.29%, using a batch size of 16, compared to 30.51% without motion saliency detection. However, the addition of the motion saliency module resulted in a training time that is three times longer, 22 minutes, than the original model without motion saliency detection, which took 7 minutes to train. Given that each sample in a batch is cropped to the union of the spatiotemporal salient regions of all the samples in the batch, it is likely that this limits the actual reduction in the size of the data that is processed, especially if the actions are carried out in different locations across the training samples. A second set of experiments were carried out without batch normalisation, that is a batch size of 1. The same trends are observed in the results of these experiments as were observed in the experiments with a batch size of 16. A higher classification accuracy, 27.19%, was observed with the motion saliency detection module compared to without the module, however the training time is still higher, 24 minutes, compared to without motion saliency detection, 10 minutes. It is evident, that the training time is now 2.4 times higher, whereas with a batch size of 16 it was 3.1 times higher.

Optimisation

These results showed that despite the motion saliency detection solution being significantly faster than other solutions such as BSUV-Net 2.0 [189] and GMM [182], it is still too slow for a reduction in training time to be realised. However, the implementation as it is does not exploit the parallel processing capabilities of modern CPUs and GPUs.

Given that the ground plane detection algorithm, which forms part of the motion saliency detection module, processes the depth map image in individual blocks, with the results of processing one block independent from the results of processing the remaining blocks, it can be parallelised significantly by shifting the implementation to

the GPU. The ground plane detection algorithm is shown in Algorithm 3. The loop spanning lines 7–16 is shifted to the GPU, such that multiple iterations are executed simultaneously in parallel, where each iteration processes different pixels of the gradient maps. Similarly, given that motion saliency detection in one frame does not depend on the results of the previous frames, the frames can be processed independently in parallel using multiple threads. However, this precludes the use of the Kalman filter addition to motion saliency, that was detailed in Section 4.2.2, since the Kalman filter depends on the results from the previous frames.

Algorithm 3 Ground Plane Detection Algorithm

```
1:  gx = X Gradient Map Image
2:  gy = Y Gradient Map Image
3:  out = Output Ground Plane Mask
4:  h = Image height
5:  w = Image width
6:
7:  for y ∈ 0, 1, . . . , h do
8:      for x ∈ 0, 1, . . . , w do
9:          if gx[y,x] == 0 and gy[y,x] ≠ 0 then
10:             out[y,x] = 1
11:         else if gx[y,x] == 0 and gy[y,x] == 0 and y ≥ h ∗ 3/4 then
12:             out[y,x] = 1
13:         else
14:             out[y,x] = 0
15:         end if
16:     end for
17: end for
```

The results, shown in Table 5.2, using the optimised motion saliency detection module, referred to as ResNet3D-50+MS (P), show an improvement in that the training time on the UTKinect dataset is reduced from 24 minutes, with the unoptimised motion saliency detection module, down to 9 minutes. The training time is now less than the time taken, 10 minutes, to train the original model without motion saliency detection. However, the reduction in training time, 1 minute, is minimal compared to the increase in computational complexity and resource consumption.

It was observed that the bottleneck lies in the loading of the depth map data from storage to memory. This data cannot be loaded in its entirety and retained in memory throughout the training procedure since it's size is larger than the available CPU memory of most machines. However, the identified spatiotemporal salient regions can be cached effectively since they only consist of four coordinates. The memory consumed by the cache is $4 \times 2 \times N$ where N is the number of training samples, thus the spatial complexity is $\mathbf{O}(N)$. With caching, refereed to as ResNet50-3D+MS (PC), the training time is reduced to 4 minutes on UTKinect and 128

Table 5.3 Classification Accuracy and Training time with and without motion saliency on the full NTU-60 XSUB training set

Method	Batch Size	Top-1 Accuracy (%)	Training Time (hours)
ResNet3D-50	16	10	819
ResNet3D-50	1	7	888
ResNet3D-50+MS (PC)	1	22	467

minutes, down from 142 minutes, on NTU-60 (300).

Evaluation on Full NTU-60 Dataset

ResNet50+MS (PC), which is the ResNet-50 with the optimised motion saliency detection module, is evaluated on the full NTU-60 dataset, which consists of 56,880 samples, and compared to the original ResNet-50 model. The results, presented in Table 5.3, show that a significant reduction in training time is achieved. The original ResNet3D-50 model takes 819 hours to complete, 888 hours without batching, whilst training ResNet3D-50+MS (PC) takes 467 hours. However, the classification accuracy achieved with all methods, including the original ResNet3D-50, is low, and even falls below the classification accuracy achieved with the reduced training set size. Whilst the motion saliency detection module increased the classification accuracy from 10% to 22%, it is still far too low to be considered for practical applications. These models have been observed to perform well in outdoor settings where the background contributes to action recognition however they perform poorly in indoor settings where the background is consistent across all action classes [1], such as the NTU-60 dataset. It is likely that in both cases, with and without the motion saliency detection module, the ResNet3D-50 classifier is learning features from the background rather than features relevant to action classification and thus fails to distinguish between different action classes.

5.2 Motion Saliency in MMNet

The MMNet [1] HAR method has achieved a state of the art performance particularly in indoor environments where 3D-CNN based methods perform poorly. In particular MMNet is reported to have achieved a high classification accuracy on the NTU-60 [126] and NTU-120 [127] datasets, which consist of videos taking place in indoor settings with a consistent background across all action classes. The method consists of three classifiers; one for the skeleton joint, one for the bone and one for the RGB data

modality. The RGB classifier is based on a ResNet-18 network that is trained on 2D images capturing a spatiotemporal region of interest (ST-ROI) within the video. The ST-ROI is constructed by concatenating multiple spatial ROIs from different frames into a single image, which are identified by the position of the skeleton joints. This procedure introduces considerable computational complexity since the skeleton joint coordinates are estimated using the OpenPose tool [2] during a preprocessing step. This is a costly step both in terms of computational time, memory and energy resources. Moreover, the performance of the method is highly dependent on the choice of skeleton joints that are used to identify the spatial ROIs. The exact joints that are selected is dataset dependent, which introduces additional methodological complexity and hinders the practical adoption of the method in real-world applications. In this section, the use of motion saliency detection as an alternative method for identifying an ST-ROI is explored.

5.2.1 Identifying ST-ROI using Motion Saliency Detection

Incorporating motion saliency detecting in the MMNet HAR classifier poses different challenges from those faced when adopting motion saliency detection in a 3D-CNN based HAR classifier, described in Section 5.1. Unlike the 3D-CNN, which takes an entire video clip as input, the MMNet RGB modality classifier is trained on 2D images capturing a spatiotemporal region of interest. This means the training data provided to MMNet has already been reduced to the regions that are relevant to action recognition. In order for a reduction in training time to be achieved, it is essential that motion saliency detection completes in less time than MMNet requires to identify the ST-ROI.

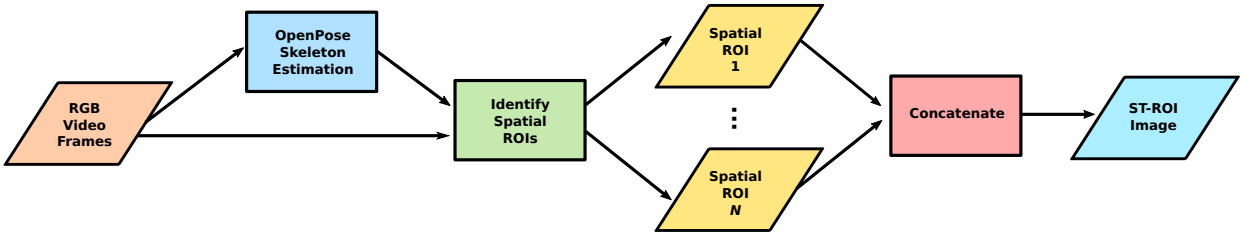


Figure 5.5 Pipeline showing the identification and construction of ST-ROI image in the original MMNet RGB classifier

Figure 5.5 shows the pipeline of the strategy adopted by MMNet from the identification of the spatial regions of interest to the construction of the ST-ROI image that forms the training input to the MMNet RGB classifier. The first step is the estimation of the skeleton joint coordinates in each frame of the video using the OpenPose tool [2], which is a tool for detecting keypoints comprising the human body. The estimated skeleton joint coordinates are then used to identify a number of spatial

ROIs surrounding the skeleton joints in each frame.

Five skeleton joints, corresponding to the head, wrists and ankles, are selected around which a spatial ROI is identified. Each spatial ROI is constructed by cropping the square region of the frame image that is centred on the skeleton joint. The size of this square varies with the dataset, for example a 48×48 square is selected for the NTU-60 [126] dataset whereas a 24×24 square is selected for the NW-UCLA [125] dataset. Additionally, the square crop region is shifted by an offset on the NW-UCLA dataset while no shift exists on the remaining evaluation datasets.

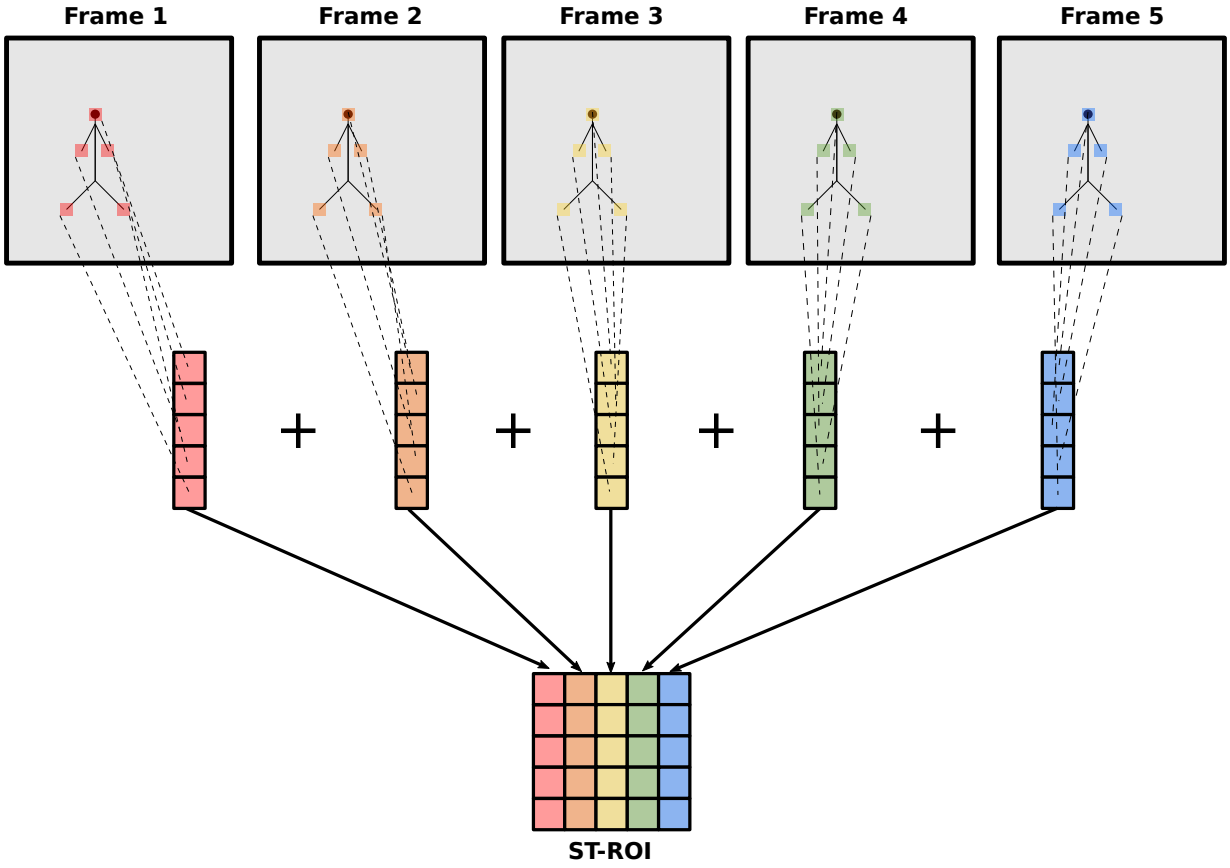


Figure 5.6 Diagram illustrating how the ST-ROI is constructed using MMNet

The identified individual spatial ROIs are concatenated vertically in a column to produce the spatial region for a given frame. The spatial regions of five frames are concatenated horizontally to produce the ST-ROI which consists of a 5×5 grid of spatial ROIs. This is illustrated in Figure 5.6.

The bottleneck of this procedure lies in the skeleton joint coordinate estimation using the OpenPose tool. This tool not only adds methodological complexity, since it, itself, requires training in addition to training the RGB modality classifier, but is also a costly step in terms of computational time. The reason given for why OpenPose is used to estimate the skeleton joint coordinates, as opposed to using the skeleton data retrieved through depth sensors such as Microsoft Kinect and Intel RealSense, is that it

is somewhat more accurate [1]. Furthermore, the skeleton data is not always provided in camera space coordinates. Motion saliency detection does not require skeleton data and thus this step can be eliminated by replacing MMNet's strategy for identifying an ST-ROI with motion saliency detection.

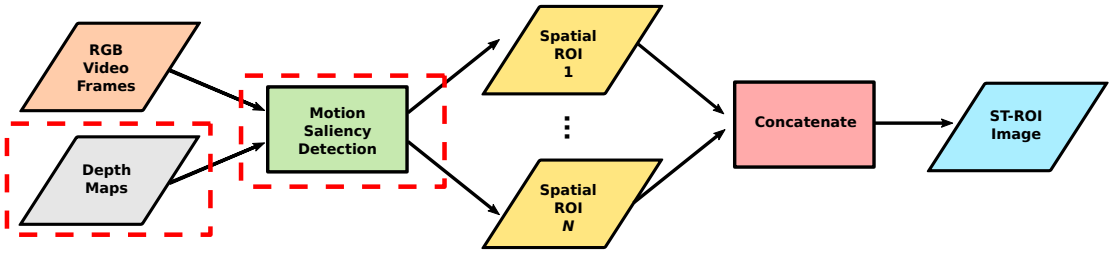


Figure 5.7 Pipeline showing the identification and construction of the ST-ROI image using motion saliency detection

Motion saliency detection identifies a set of regions in each frame of the video. The strategy is to use this set of regions to crop each frame to the region that is relevant to human action recognition, after which, the cropped frames are concatenated to form an ST-ROI image. Figure 5.7 shows the pipeline to construct an ST-ROI image using motion saliency detection. The OpenPose skeleton estimation step has been removed along with the identification of the spatial ROIs using the skeleton joint information. These steps have been replaced by motion saliency detection to identify the spatial ROIs, which are then concatenated to form a single image.

A single spatial ROI per frame is identified that covers a larger area than the five spatial ROIs identified by MMNet. The spatial ROI is constructed from the union box of the salient regions identified in a given frame. Let N_i denote the number of salient regions identified in a given frame i, and r_j, where $j \in [1, N_i]$, be the j^{th} salient region that was identified. The spatial ROI, S_i, for the i^{th} frame is defined by

$$x_{S_i} = \min\{x_{r_j} \mid j \in [1, N_i]\} \tag{5.1}$$

$$y_{S_i} = \min\{y_{r_j} \mid j \in [1, N_i]\} \tag{5.2}$$

$$w_{S_i} = \min\{w_{r_j} \mid j \in [1, N_i]\} \tag{5.3}$$

$$h_{S_i} = \min\{h_{r_j} \mid j \in [1, N_i]\} \tag{5.4}$$

where x_{r_j} and y_{r_j} are the x and y coordinates, respectively, of the top-left corner of the salient region r_j while w_{r_j} and h_{r_j} are the width and height, respectively, of the salient region r_j. Equations (5.1) and (5.2) define the top-left corner of the spatial ROI for the i^{th} frame while (5.3) and (5.4) define its width and height.

In order to concatenate the spatial ROIs into a single image covering the ST-ROI of the video, the spatial ROIs are required to be of the same size. Scaling each spatial

ROI to a common size will lead to upsampling and downsampling artefacts that vary between the frames. The strategy detailed in Section 5.1.2, where the union box of the spatial ROIs of all frames is used as the crop region in each frame, is not suitable for constructing an ST-ROI since it will lead to each spatial ROI covering an area that is larger than the relevant region in the frame. To arrive at a common size for the spatial ROIs, the average size of the spatial ROIs is used. This prevents the issue of choosing a size that is too large or too small for the action.

The common size of the spatial ROIs is defined by

$$w'_S = \frac{\sum_{n=1}^{K} w_{S_n}}{K} \tag{5.5}$$

$$h'_S = \frac{\sum_{n=1}^{K} h_{S_n}}{K} \tag{5.6}$$

where K is the number of frames. The bounding rectangle of each identified spatial ROI is resized to a width and height defined by (5.5) and (5.6), respectively, while its centre is preserved. The top-left corner of the resized spatial ROI for frame i is defined by

$$x'_{S_i} = (x_{S_i} + \frac{w_{S_i}}{2}) - \frac{w'_S}{2} \tag{5.7}$$

$$y'_{S_i} = (y_{S_i} + \frac{h_{S_i}}{2}) - \frac{h'_S}{2} \tag{5.8}$$

where x'_{S_i} is the x coordinate and y'_{S_i} is the y coordinate. Each frame is cropped to the spatial ROI identified by this bounding rectangle.

Similar to the strategy adopted by MMNet, the spatial ROI images are concatenated into a single image covering the ST-ROI of the video. There are various possibilities for the configuration in which the images are concatenated. One such possibility is to concatenate the images into a square grid, which creates an ST-ROI image that is similar to the ST-ROI images generated by MMNet. Alternatively the images can be concatenated horizontally in a row, as illustrated in Figure 5.8.

5.2.2 Ablation Study

This section details the experiments performed to determine the optimal parameters, which include the number of frames that are selected for the ST-ROI, the configuration in which they are concatenated and the size to which the individual spatial ROIs are scaled. The objective of these experiments is to find values of parameters such that the maximum classification accuracy is achieved in the minimum training time.

The first experiments focus on determining the optimal number of spatial ROIs to include in the ST-ROI and the configuration in which they are concatenated to form

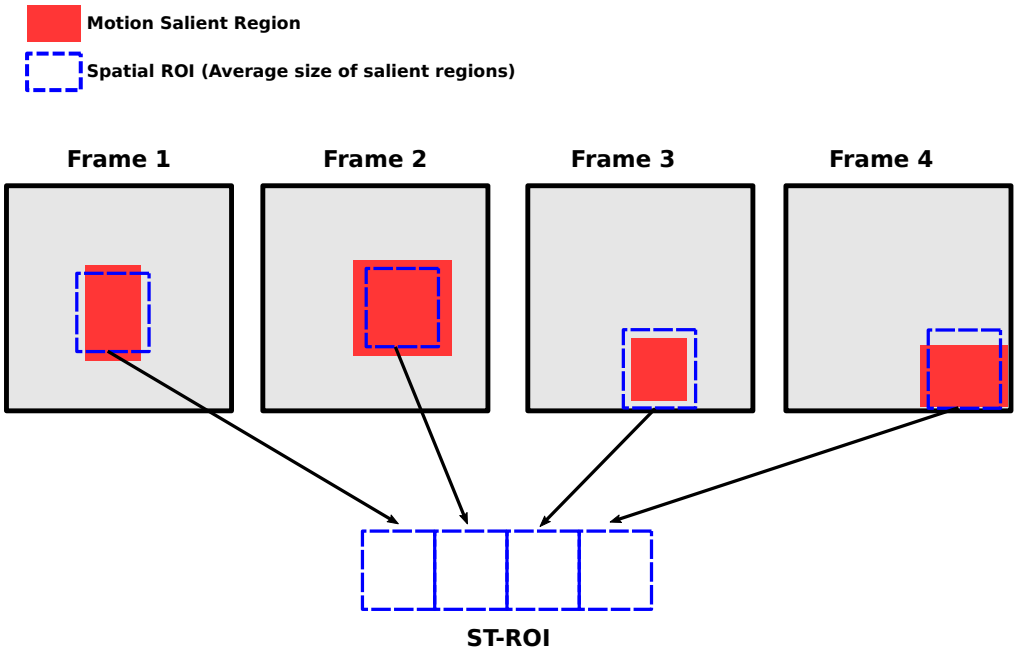


Figure 5.8 Diagram illustrating the construction of the ST-ROI when using motion saliency

a single image. As mentioned earlier the spatial ROIs can be concatenated either in a square grid, of which an example on a sample from the NTU-60 dataset is shown in Figure 5.9, or a horizontal row, an example of which, involving two actors, is shown in Figure 5.10. For comparison the ST-ROI images constructed by MMNet for the same samples are shown in Figures 5.12 and 5.11.

The number of spatial ROIs, that is the number of frames, used to construct the ST-ROI was varied from 4 to 16. Given that the size of the spatial ROIs differs between samples, the height of the spatial ROIs is fixed for a fair comparison, with the width scaled proportionally to preserve the aspect ratio. The height was varied from 60 to 480 pixels, which is the height of the ST-ROI images produced by MMNet. The same network model, a 2D ResNet-18 pretrained on ImageNet [26], and training strategy that were used by MMNet, are used in these experiments. The experiments are performed on the full NTU-60 dataset using the XSUB evaluation protocol. This dataset was chosen since a good performance is expected given its a large RGB+D dataset and a good performance was achieved on it using MMNet. The experiments were performed on a machine having an Intel i7-11800H with 8 cores, an Nvidia GeForce GTX 3060 graphics card and 32 GB of RAM.

In the initial experiments, a square grid configuration was used, where the spatial ROIs are arranged in a 2×2, 3×3 and 4×4 grid. The height of the spatial ROIs was varied, from 120 for the 4×4 configuration to 240 for the 2×2 configuration, such that the height of the ST-ROI image is 480 pixels, which is the height of the

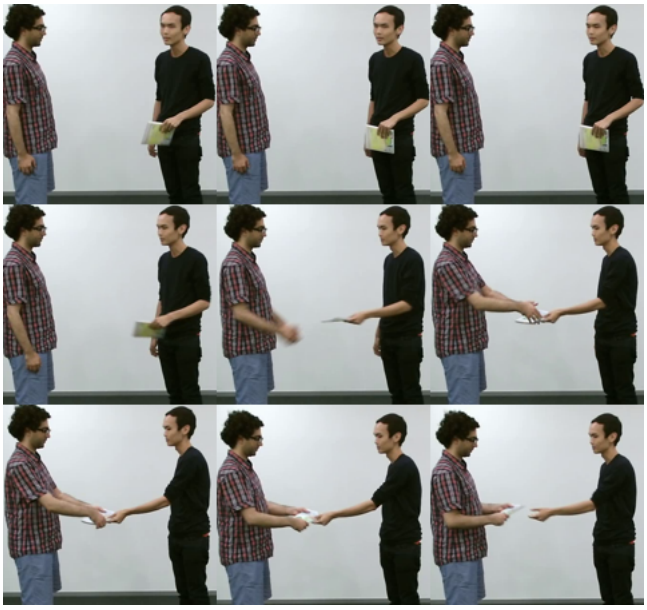


Figure 5.9 Example of motion saliency ST-ROI with 9 frames concatenated in a 3×3 grid capturing the action “giving object” involving two actors

Figure 5.10 Example of motion saliency ST-ROI with 16 frames concatenated horizontally (1×16) capturing the action “pick up”

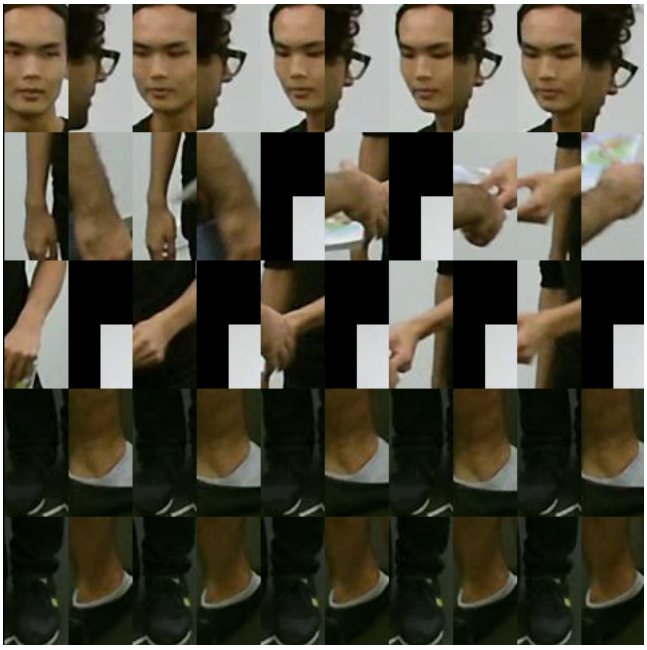


Figure 5.11 Example of ST-ROI constructed by MMNet capturing the action “giving object” involving two actors

Figure 5.12 Example of ST-ROI constructed by MMNet capturing the action “pick up”

Table 5.4 RGB modality classification accuracy using varying ST-ROI configurations and sizes on NTU-60 (XSUB) dataset

Configuration	Height (px)	Top-1 Accuracy (%)	Top-5 Accuracy (%)
2×2	240	57.90	84.67
3×3	160	64.08	88.17
4×4	120	59.44	84.97
1×4	480	65.63	89.64
1×9	480	63.53	87.86
1×16	480	70.39	91.56
1×16	120	69.51	91.75
1×16	60	67.67	90.31
1×32	480	68.34	92.07

Table 5.5 RGB modality training and inference times using varying ST-ROI configurations and sizes on NTU-60 (XSUB) dataset

Configuration	Height (px)	Training Time (hours)	Inference Time (milliseconds)
2×2	240	9.69	10
3×3	160	10.00	11
4×4	120	10.01	11
1×4	480	27.29	18
1×9	480	29.17	29
1×16	480	32.05	33
1×16	120	30.98	25
1×16	60	3.79	4
1×32	480	97.17	78

ST-ROI image produced by MMNet. The results, presented in Table 5.4, show that a low classification accuracy is observed ranging from 57.90% to 64.08%. The lowest classification accuracy was observed with the 2×2 configuration, which contains the lowest number of frames. However, the highest classification accuracy was not observed with the 4×4 configuration, which has the most frames, but with the 3×3 configuration. A possible explanation for these results is that, while the classification accuracy increases with the number of frames, the 4×4 configuration also uses the smallest spatial ROI size, which negatively affects the classification accuracy. The training times, presented in Table 5.5, do not show significant variation between the configurations. The shortest training time was 9.69 hours for the 2×2 configuration, and the longest training time was 10.01 hours for the 4×4 configuration. Given that the height of the ST-ROI images is consistent, 480 pixels, throughout all grid configurations, the size of the images, and hence the training time, does not vary significantly. The inference time per sample, also presented in Table 5.5, show even less variation between the experiments, with inference taking one millisecond longer using the 3×3 and 4×4 configurations than with the 2×2 configuration.

The remaining experiments focused on configurations where the spatial ROIs are concatenated in a horizontal row. The spatial ROI height is initially fixed at 480 pixels, to maintain the same height as the MMNet ST-ROI images, and then progressively reduced in subsequent experiments. The results, presented in Table 5.4, show a significant increase in classification accuracy over the grid configurations. A higher classification accuracy was observed, 65.63%, for the row configuration with the lowest number of frames, 1×4, than the accuracy observed, 59.44%, for the grid configuration containing the most frames, 4×4. However, it is also observed, from the training and inference times shown in Table 5.5, that the 1×4 configuration took 27.29

Table 5.6 Fused model classification accuracy using 1×16 ST-ROI configuration with varying height on NTU-60 (XSUB) dataset

ST-ROI Height (px)	Top-1 Accuracy (%)	Top-5 Accuracy (%)
480	89.90	91.19
120	90.05	92.99
60	89.97	91.25

hours to train, which is nearly three times longer than the time required to train the grid configuration with the same number of frames, 2×2. Whilst the ST-ROI height used in the 1×4 configuration is the same as that used in the 2×2 configuration, more images are concatenated in the horizontal dimension, which increases the width of the ST-ROI image and hence increases the training time.

The highest classification accuracy, 70.39%, was observed using the 1×16 configuration, which consists of 16 frames concatenated horizontally. This configuration also produced a top-5 classification accuracy of 91.56%. Training the classifier using this ST-ROI configuration, took 32.05 hours which is three times longer than the training time, 10.01 hours, using the 4×4 configuration, which contains the same number of frames. Reducing the size of the spatial ROIs, that are used to construct the 1×16 ST-ROI image, to 120 pixels decreased the classification accuracy to 69.51%, which is a drop of less than 1%. However, the training time only decreased to 30.98 hours, which is a reduction by less than 2 hours and is still significantly longer than the time required for training using the 4×4 configuration. Reducing the ST-ROI height further to 60 pixels, resulted in the classification accuracy decreasing to 67.67%. However, this classification accuracy is higher than the accuracy observed using the 3×3 configuration, 65.63%, while the training time, 3.79 hours is drastically lower than 10 hours which is the training time observed using that configuration. In-fact, this is the lowest training time observed across all experiments. Similarly, the lowest inference time, 4 milliseconds per sample, was observed using this configuration. The impact on the top-5 classification accuracy is less significant, with the top-5 accuracy only decreasing by 1.25% from 91.56%, using a height of 480 pixels, to 90.31%, using a height of 60 pixels. Increasing the number of frames to 32 did not result in an increase in the classification accuracy. The classification accuracy, 68.34%, observed with the 1×32 configuration is actually lower than the classification accuracy observed with the 1×16 configuration. However, the training time with the 1×32 configuration is significantly longer. Given these results, it is deduced that the best configuration is 1×16 with a height of 60 pixels.

The objective of the final experiments of the ablation study is to determine the classification accuracy and training time using the full fused model, which fuses the

Table 5.7 Fused model training time and inference times using 1×16 ST-ROI configuration with varying height on NTU-60 (XSUB) dataset

ST-ROI Height (px)	Training Time (hours)	Inference Time (milliseconds)
480	72.65	76
120	33.79	36
60	10.98	11

Table 5.8 Fused model power consumption using 1×16 ST-ROI configuration with varying height on NTU-60 (XSUB) dataset

ST-ROI Height (px)	Average Consumption (W)	Total Consumption (kWh)
480	69.39	5.04
120	92.18	2.77
60	167.15	1.84

results of the skeleton joint, bone and RGB classifiers. The experiments used the same models and training strategy for the skeleton joint and bone classifiers, while the RGB classifier was trained on the ST-ROI images constructed using spatial ROIs identified by motion saliency. The results, presented in Table 5.6, show that the highest top-1 and top-5 classification accuracies, 90.05% and 92.99%, were observed with an ST-ROI height of 120 pixels. The results also show a higher classification accuracy using an ST-ROI height of 60 pixels, top-1 89.97% and top-5 91.25%, than with an ST-ROI height of 480 pixels, where a top-1 accuracy of 89.90% and top-5 accuracy of 91.19% were observed.

The results for the training times taken with each ST-ROI height, presented in Table 5.7, show that the shortest training time, 10.98 hours was observed using an ST-ROI height of 60 pixels, which is three times lower than the training time using a height of 120 pixels, 33.79 hours, where the highest classification accuracy was observed, and nearly seven times lower than the training time, 72.65 hours, using a height of 480 pixels.

The power consumption of the GPU was measured throughout the entire training process. Two metrics were recorded, namely the power consumption at each second, measured in Watts (W), which is used to compute the average power consumption, and the total power consumed during the training process, measured in kilo Watt hours (kWh). The power consumption results, presented in Table 5.8, show a different trend from the results for the training time. The highest average power consumption, 167.15 W, was observed using an ST-ROI height of 60 pixels while the lowest was observed using an ST-ROI of 480 pixels. However, the total power consumption, of the training process, was lowest, 1.84 kWh, using an ST-ROI height of

60 pixels and highest, 5.04 kWh, using an ST-ROI height of 480 pixels. This is due, to the drastically lower training time using a height of 60 pixels. Thus, even though the average power consumption increases considerably, the total power consumed is still less due to the shorter time spent in training. Interestingly, it is observed that the average power consumption increases when the ST-ROI height is decreased. A possible explanation for this is that with a larger ST-ROI height, and hence larger ST-ROI image size, more time is spent, per second, transferring the data to GPU memory and, thus, less time is spent in computations.

5.2.3 Discussion

The results of the ablation study in Section 5.2.2, showed that a good classification accuracy can be achieved by constructing an ST-ROI out of spatial ROIs identified by motion saliency detection. These results also showed that the highest classification accuracy is achieved with an ST-ROI constructed out of 16 spatial ROIs identified in 16 frames. Reducing the ST-ROI height to 60 pixels, resulted in a comparable classification accuracy, however with a drastically lower training time and power consumption. Thus it is determined that an ST-ROI constructed from regions identified using motion saliency detection, with these parameters, can potentially replace the skeleton-based ST-ROI strategy adopted by MMNet, which involves a costly OpenPose skeleton estimation step. A paper detailing this research and the results was accepted for publication [6] by the IEEE International Conference on Intelligent Computer Communication and Processing.

6 Evaluation

This chapter details the experiments done to evaluate the performance of the HAR method described in Section 5.2, which builds on MMNet [1] by replacing the strategy for identifying an ST-ROI based on skeleton data estimated using OpenPose with motion saliency. This proposed method will be referred to as MMNet-MS throughout this chapter. The optimal motion saliency parameters as determined in the ablation study in Section 5.2.2 are used. The configuration 1×16 is used, that is the ST-ROI is constructed out of 16 frames concatenated horizontally in a single row with an ST-ROI height of 60 pixels.

There a number of performance metrics that need to be evaluated. The top-1 classification accuracy measures the probability that the action class predicted by the method is the correct class for the action depicted in the input video. On the other hand, the top-N accuracy measures the probability that the actual action class label is in the top-N labels predicted by the method. This is useful for applications where a specific set of actions are being looked for rather than requiring to know the exact action. The training time measures the length of time required to train the HAR model using a dataset of a given size, which gives an indication of the practicality of deploying the HAR system while adapting it to specific environments. Whilst training time does give an indication of the resource usage of the HAR system, the power consumption is measured separately since a lower training time does not necessarily imply a lower power consumption. Finally, the inference time measures the length of time required by the HAR system to determine the class label of the action in the input video. This gives an indication of the system's real-time performance.

The performance of MMNet-MS is evaluated, in terms of classification accuracy, training time, power consumption and inference time, and compared to that of the original MMNet method, which uses skeleton data estimated by the OpenPose tool to identify the ST-ROI. The methods were evaluated on the NTU-60 [126], NTU-120 [127], Northwestern-UCLA (NW-UCLA) [125], TST fall detection [207], HWU-USP [208], UTD-MHAD [209] and UTKinect (UTK) [200] datasets. The NTU-60 [126] dataset is a widely used large-scale RGB+D dataset consisting of general action categories such as "stand up", "pick up" and "throw", with 60 class labels in total. The NTU-120 [127] dataset extends NTU-60 with another 60 class labels representing actions that are more challenging to recognise. Both NTU-60 and NTU-120 contain actions performed by a single actor and actions involving two actors, such as "hugging", "kicking" and "walking towards". The Northwestern-UCLA (NW-UCLA) dataset [125] was used to evaluate MMNet-MS in order to assess its performance on a smaller scale dataset, which contains far fewer samples than NTU-60. Additionally, this dataset was

Table 6.1 Size of training and validation sets of datasets used in the experiments

Dataset	Setting	No. Training Samples	No. Validation Samples	No. Action Classes
NTU-60 (XSUB)	Public	40091	16487	60
NTU-60 (XVIEW)	Public	37646	18932	60
NTU-120 (XSUB)	Public	63026	50919	120
NTU-120 (XSET)	Public	54468	59477	120
NW-UCLA	Public	1027	467	10
UTD	Public	430	431	27
UTK	Public	129	60	10
HWU-USP	Private	99	45	9
TST	Private	96	88	8

also used to evaluate the original MMNet method, thus it serves to provide an additional comparison. TST fall detection consists of samples capturing human falls and actions related to daily living. This dataset is also distinguished from the remaining datasets in that it is privacy preserving and does not provide colour video data, but only provides depth and skeleton data. Thus, this dataset serves to evaluate the performance of the solution in private settings. HWU-USP is another privacy preserving dataset but rather than omitting colour data altogether, the faces of the subjects are blurred. This dataset provides samples of actions related to daily living captured over longer untrimmed sequences. The UTD-MHAD (UTD) [209] and UTKinect (UTK) [200] datasets are further smaller scale RGB+D which are widely used in the evaluation of skeleton based solutions. In addition to these datasets, MMNet-MS is also evaluated on a privacy preserving version of NW-UCLA where the colour data is removed and only the depth and skeleton data are used during the experiments. This dataset is referred to as NW-UCLA (depth). The sizes of the datasets used in the experiments is shown in Table 6.1.

Two sets of experiments are performed using MMNet and the proposed solution. The first set uses only the RGB modality, which is referred to as MMNet (RGB) and MMNet-MS (RGB), whilst the second set uses the full fused model, referred to as MMNet (fused) and MMNet-MS (fused). The fused model experiments are not performed on the TST fall detection and HWU-USP datasets since the skeleton data provided by the former was found to be incomplete while the latter does not provide skeleton data at all. The same architecture, training strategy and number of training epochs, 80, is used during the experiments with the proposed solution, similar to that

Table 6.2 Top-1 classification accuracy (%) results

Method	Dataset									
	NTU-60 (XSUB)	NTU-60 (XVIEW)	NTU-120 (XSUB)	NTU-120 (XSET)	NW-UCLA	NW-UCLA (depth)	TST	HWU-USP	UTD	UTK
MCAE [121]	65.60	82.40	52.80	54.70	83.60	–	–	–	–	–
HiCo [142]	81.40	88.80	72.80	74.10	–	–	–	–	–	81.36
SSC [104]	18.80	20.10	–	–	–	–	–	–	–	74.37
MMNet (RGB)	74.42	84.12	64.66	59.80	28.05	11.35	29.55	11.11	10.90	13.33
MMNet (fused) retrained	94.82	96.46	85.60	86.21	80.30	79.01	–	–	29.23	61.67
MMNet (fused) reported results [1]	96.00	98.88	92.90	94.40	88.60	–	–	–	–	–
MMNet-MS (RGB)	67.67	73.05	62.08	53.54	34.34	17.06	55.69	22.22	6.73	16.67
MMNet-MS (fused)	89.97	95.83	84.39	84.89	80.35	79.27	–	–	30.85	61.67

used with the original MMNet model. However, due to GPU memory constraints the batch size of the fused model is reduced from 64 to 16 samples for both the proposed solution and MMNet. The models are trained on a machine with an Nvidia GeForce RTX 3060 GPU with four threads dedicated to the training process.

6.1 Classification Accuracy

The top-1 accuracy results of the experiments are presented in Table 6.2. Two sets of results are shown for the original MMNet model, namely the original results reported in the paper [1], which are denoted by “MMNet (fused) reported results” and the results of retraining the MMNet models with the batch size reduced from 64 to 16 samples, which are denoted by “MMNet (fused) retrained”. The reduction in batch size resulted in a reduction in classification accuracy of of 2–8%. To avoid a bias in the results caused by the different batch sizes, the results of MMNet-MS are compared to the results of retraining the original MMNet model with the reduced batch size. The results show that MMNet-MS (RGB) achieves a comparable, but lower, classification accuracy, on the NTU-60 dataset compared to MMNet (RGB). The largest decrease in classification accuracy, 9.35%, is observed on NTU-60 with the XVIEW evaluation protocol, where an accuracy of 82.4% was observed for the original MMNet (RGB) while an accuracy of 73.05% was observed for MMNet-MS (RGB). The difference in classification accuracy is less pronounced with the fused models where an accuracy of 96.46% was observed for MMNet (fused) while an accuracy of 95.83% was observed for MMNet-MS (fused) on the NTU-60 (XVIEW) dataset. Thus, the classification accuracy decreased by less than 1%. Similar results are observed on NTU-60 (XSUB) where the accuracy observed for MMNet-MS (fused) is 4.85% lower than the accuracy observed

Table 6.3 Top-5 classification accuracy (%) results

Method	Dataset									
	NTU-60 (XSUB)	NTU-60 (XVIEW)	NTU-120 (XSUB)	NTU-120 (XSET)	NW-UCLA	NW-UCLA (depth)	TST	HWU-USP	UTD	UTK
MMNet (RGB)	95.57	98.64	91.12	85.45	81.58	49.04	79.55	55.56	28.77	50.00
MMNet (fused)	96.74	99.82	98.07	97.90	96.57	90.15	–	–	30.40	76.67
MMNet-MS (RGB)	90.31	91.90	80.03	82.32	82.51	69.55	93.18	66.67	25.99	53.33
MMNet-MS (fused)	91.25	99.79	97.85	97.59	96.68	96.90	–	–	37.35	85.00

for the original MMNet (fused). The results for NTU-120 show less of a difference where the decrease in classification accuracy of MMNet-MS (fused) compared to the original MMNet (fused) is less than 2% for both evaluation protocols, XSUB and XSET.

The results for the remaining datasets show a low classification accuracy with both the models, MMNet (RGB) and MMNet-MS (RGB), that use colour data only. This shows that colour data alone is not sufficient for successful action recognition using either of the models. However at the same time, the classification accuracy achieved using motion saliency to identify the ST-ROI is consistently higher than the original MMNet RGB classifier, across all datasets with the exception of UTD-MHAD. The most notable difference is in the TST fall detection dataset, where an accuracy of 29.55% was observed for the original MMNet (RGB) while an accuracy of 55.69% was observed for MMNet-MS (RGB). A possible explanation for this is that OpenPose fails to estimate accurate skeleton joint coordinates from depth map images. Additionally the skeleton-based ST-ROI likely does not capture enough information from the depth map. The results for the fused models show an equivalent classification accuracy, for MMNet and MMNet-MS, across all datasets. The lowest classification accuracy was observed on UTD-MHAD, 20.19% for MMNet and 20.89% for MMNet-MS. The UTD-MHAD dataset consists of 27 action classes of which 14 are performed using only the arms. It is likely that these models are unsuitable for recognising actions performed using a single body part.

The top-5 accuracy results, presented in Table 6.3, show that the performance decreases by 5%–7% on the NTU-60 dataset, with both evaluation protocols, using MMNet-MS (RGB) compared to the original MMNet (RGB). The same trend is observed using for the fused models. However, the results for the remaining datasets show that MMNet-MS consistently outperforms MMNet, both with the fused and RGB only models, across all datasets except UTKinect and UTD-MHAD. The most notable difference in performance is observed, on NW-UCLA (depth) where a top-5 classification accuracy of 49.04% is observed for MMNet (RGB) while a top-5 accuracy of 69.55% is observed for MMNet-MS (RGB). A similar trend was observed for fused

models where the top-5 accuracy observed for MMNet-MS (fused) is 6.75% higher than that observed for the original MMNet (fused). The MMNet-MS models also achieved a significantly higher top-5 accuracy, by 11%–14% on TST fall detection and HWU-USP, which are privacy preserving datasets in assisted living settings. While, a higher top-5 accuracy was observed for the original MMNet (RGB) model compared to MMNet-MS (RGB) on the UTKinect and UTD-MHAD datasets, the top-5 accuracy achieved by MMNet-MS on the fused model is higher than that of the original MMNet (fused). The top-5 accuracy observed on the NW-UCLA dataset is equivalent for both MMNet and MMNet-MS.

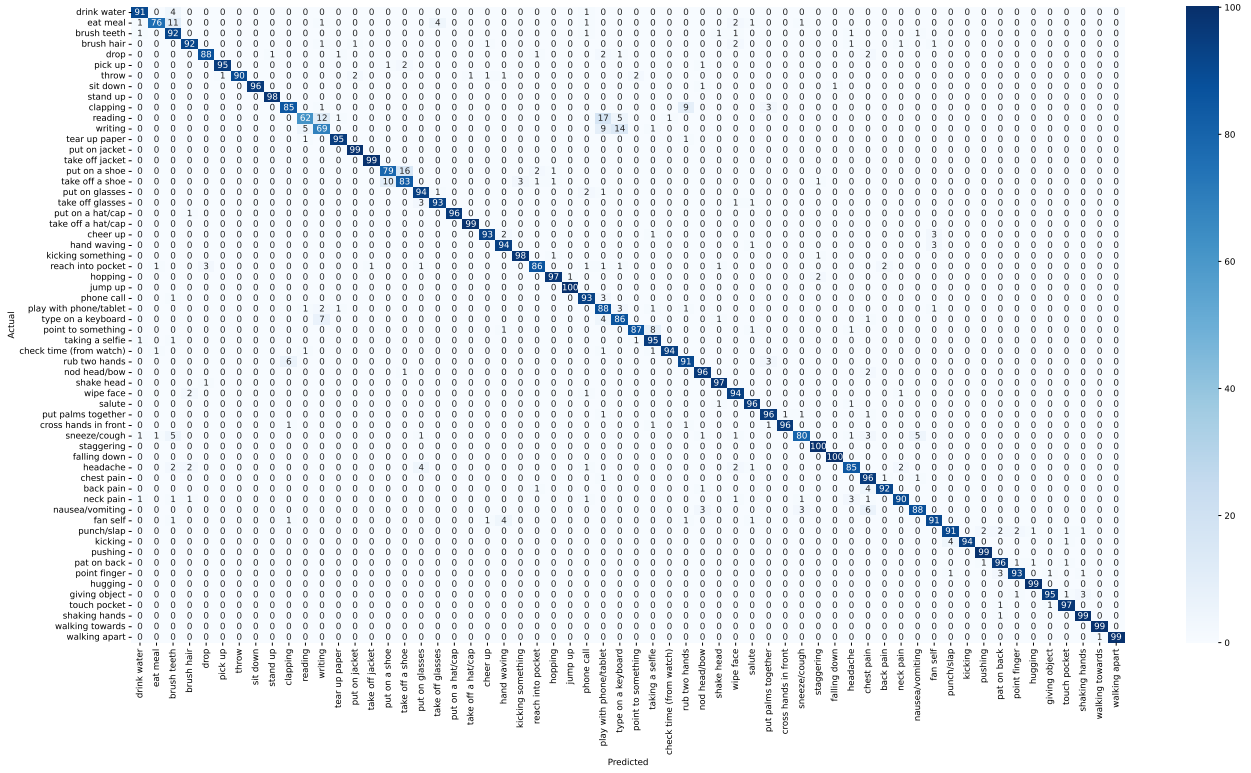


Figure 6.1 Confusion matrix for original MMNet (fused) on NTU-60 (XSUB)

The performance of MMNet-MS was also compared to three HAR methods, namely Hierarchical Contrast (HiCo) [142], Multiple Capsule Autoencoder (MCAE) [121] and Subspace Clustering (SSC) [104]. The accuracy observed for MMNet-MS (fused) on NTU-60 was 7.0% higher on XSUB and 8.57% higher XVIEW compared to the accuracy observed for HiCo. The difference in classification accuracy was more pronounced on the NTU-120 dataset where the accuracy was 11.59% higher on XSUB and 10.79% higher on XSET compared to HiCo. However, HiCo and SSC outperformed both MMNet-MS (fused) and the original MMNet fused model by approximately 10–20%. The accuracy observed for MCAE on the NEW-UCLA dataset was 3% higher than the accuracy observed for both MMNet-MS (fused) and the original MMNet

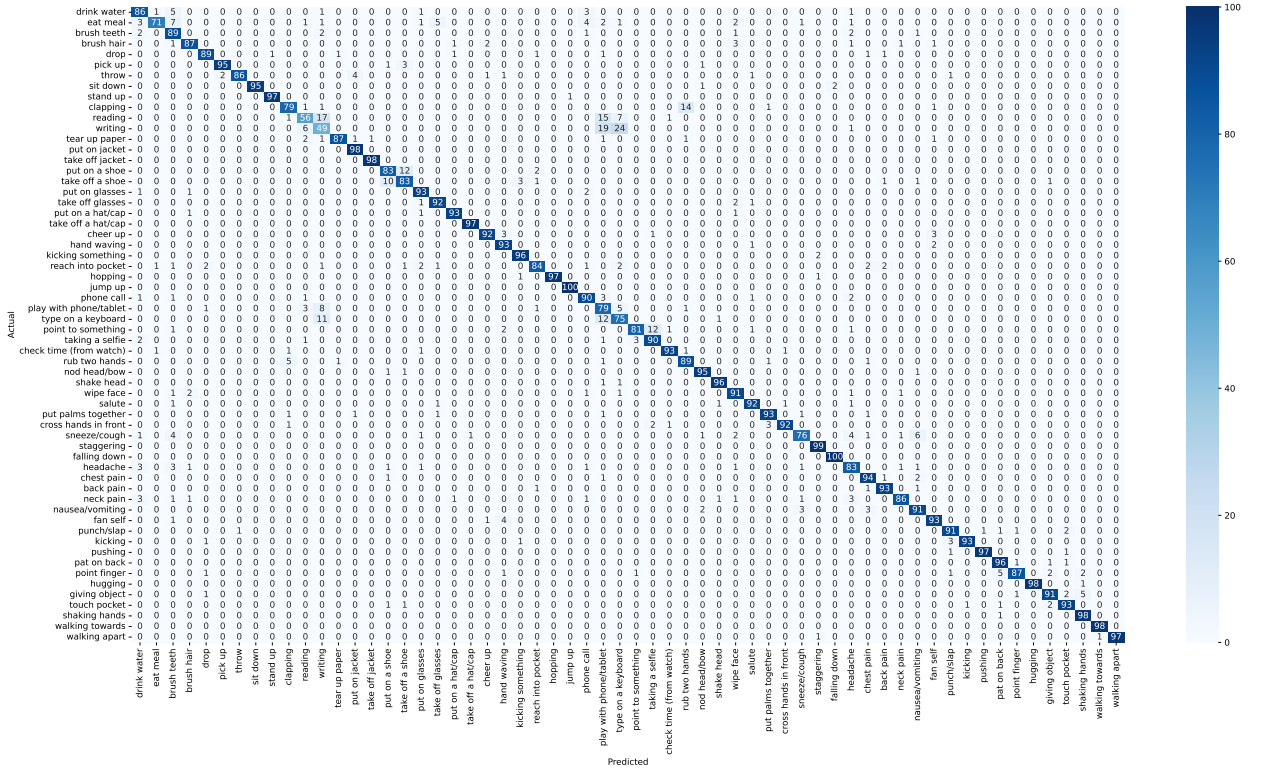


Figure 6.2 Confusion matrix for MMNet-MS (fused) on NTU-60 (XSUB)

fused model. However, the accuracy observed for MCAE on NTU-60 (XSUB) and NTU-120, with both evaluation protocols, was significantly lower than that observed for MMNet-MS (fused). Specifically, the accuracy observed for MMNet-MS (fused) on NTU-120 (XSUB) is 21.59% lower than MMNet-MS (fused) while the accuracy observed on NTU-120 (XSET) is 30.19% lower than MMNet-MS (fused). These results indicate that HiCo, SSC and MCAE have a superior performance to MMNet-MS (fused) on small datasets, however the performance of MMNet-MS (fused) is superior when trained on a sufficiently large dataset. Additionally, these results indicate that the performance of MMNet-MS (fused) is comparable, in terms of classification accuracy to the original MMNet fused model across datasets.

To determine the class of actions for which MMNet-MS performs better or worse than the original MMNet, the confusion matrices are analysed. The NTU-60 (XSUB) confusion matrices, presented in Figures 6.1 and 6.2, show that MMNet-MS (fused) is more accurate in classifying the actions “drop”, “put on a shoe”, “back pain”, “nausea/vomiting” and “fan self” than the original model without motion saliency. The classification accuracy for these action classes is shown in Table 6.4. The lowest classification accuracy observed for MMNet-MS was on the actions “writing”, “reading”, “eat meal”, “type on a keyboard” and “play with phone/tablet”, with a higher classification accuracy, shown in Table 6.5, observed using the original MMNet (fused)

Table 6.4 Classification accuracy (%) for NTU-60 (XSUB) actions where MMNet-MS (fused) outperforms MMNet (fused)

Action	MMNet (fused)	MMNet-MS (fused)
drop	88	89
put on a shoe	79	83
back pain	92	93
nausea/vomiting	88	91
fan self	91	93

Table 6.5 Classification accuracy (%) for NTU-60 (XSUB) actions where lowest accuracy was observed for MMNet-MS (fused)

Action	MMNet (fused)	MMNet-MS (fused)
writing	69	49
reading	62	56
eat meal	76	71
type on a keyboard	86	75
play with phone/tablet	88	79

model for these actions.

Similar results are observed in the NTU-60 (XVIEW) confusion matrices, shown in Figures 6.3 and 6.4. However, a higher classification accuracy, shown in Table 6.6, was also observed for the actions “throw”, “clapping”, “reading”, “take off a shoe”, “rub two hands”, “hopping”, “play with phone/tablet”, “salute”, “pushing” and “touch pocket” with MMNet-MS compared to MMNet. It is also observed that MMNet-MS no longer outperforms the original model on the actions “back pain”, “nausea/vomiting” and “fan self”. From these results it can be deduced that the original model performs better than MMNet-MS on actions involving a small number of body parts, such as reading and writing, particularly when the actor performing the actions is different from the actor on which the model was trained. On the other hand MMNet-MS outperforms MMNet on actions which involve a body particular pose such as putting on a shoe.

The NW-UCLA confusion matrices, presented in Figures 6.5 and 6.6, show that MMNet-MS (fused) outperforms the original model without motion saliency on 5 out of 10 action classes namely “pick up with one hand”, “walk around”, “stand up”, “donning” (putting on a jacket) and “doffing” (taking off a jacket). On the other hand the original model outperforms MMNet-MS on “pick up with two hands”, “drop trash” and “carry”. These results show that MMNet-MS outperforms MMNet on actions involving multiple body parts, such as walking around, standing up, putting on and taking off a jacket. On the other hand, the original model is more capable of distinguishing

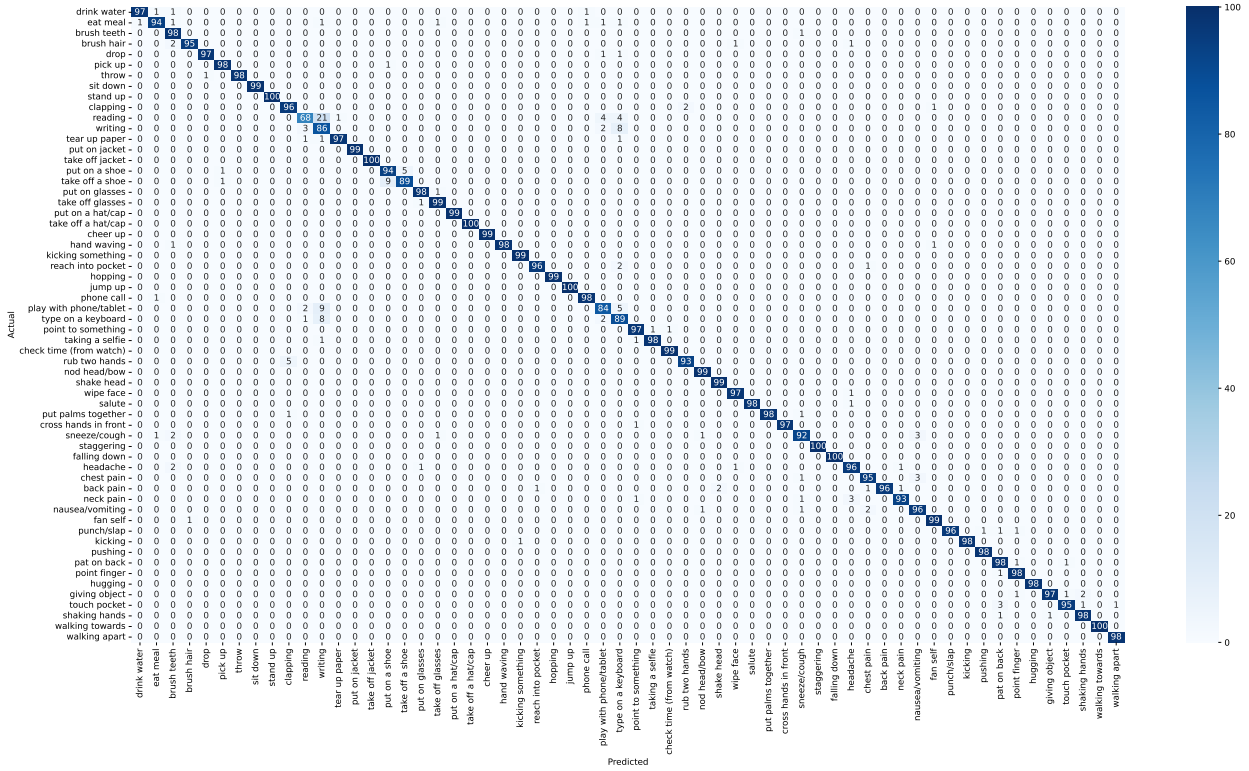


Figure 6.3 Confusion matrix for original MMNet (fused) on NTU-60 (XVIEW)

between the same action class performed with one and two hands.

The confusion matrices, presented in Figures 6.7 and 6.8, for the depth only NW-UCLA, which represents a private setting, show that MMNet-MS (fused) is more accurate than the original model at classifying the actions “drop trash”, “sit down” and “doffing”, while the original model is more accurate at recognising the actions “pick up with one hand”, “pick up with two hands”, “walk around”, “throw” and “carry”. While the results of the previous experiments showed that MMNet-MS is superior at recognising actions that involve either the full body, such as sitting down, or large movements in multiple body parts, such as donning, the results show that this is not guaranteed in the general case, since the original model outperforms MMNet-MS in some action categories involving the entire body, such as walking.

As mentioned earlier, a poor performance was observed on the UTD-MHAD with both the original model and MMNet-MS. The UTD-MHAD confusion matrix for the original MMNet fused model, presented in Figure 6.9, shows that only three action classes, namely “sit to stand”, “stand to sit” and “squat” are accurately classified, while an additional two categories, “two hand front clap” and “arm curl” are recognised somewhat accurately with an accuracy of 69%. 88% of the instances of the “two hand push” action are classified as “squat” actions. This is likely due to both actions have similar arm and upper body movements with the two actions only being

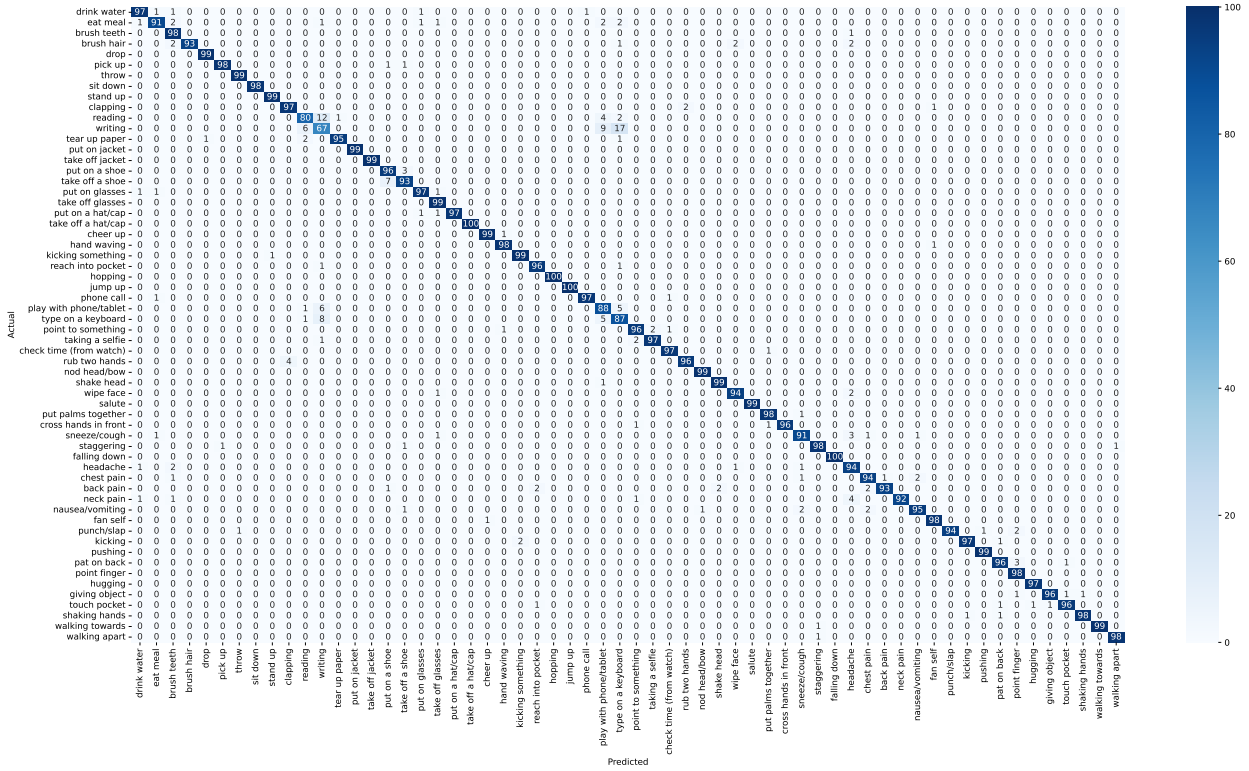


Figure 6.4 Confusion matrix for MMNet-MS (fused) on NTU-60 (XVIEW)

distinguishable in the movement of the legs. Similarly, “tennis serve” is confused with “arm curl”, “front boxing” is confused with “squat”, and “cross arms” is confused with “two hand front clap”. From these results it can be concluded that the low classification accuracy is due to the model being unable to distinguish between actions composed out of similar movements while differing only in the movements of a single body part.

The UTD-MHAD confusion matrix for the MMNet-MS (fused) model, presented in Figure 6.10, shows a slightly improved performance over the original model. MMNet-MS recognised the “squat”, “forward lunge”, “stand to sit”, “sit to stand”, “walk in place” and “two hand front clap” actions with an accuracy greater than or equal to 75%, while the “arm curl” and “arm right swipe” were classified with an accuracy of 56% and 69%, respectively. However, a similar trend is observed where actions that differ only in the movement of a single body part are confused with each other. Examples of such are “two hand push”, which is classified as “squat”, “front boxing”, which is classified as “squat”, “draw triangle”, “draw circle (counter clockwise)” and “hand wave”, which are all classified as “arm right swipe”, and “cross arms”, which is classified as “two hand front clap”.

A low classification accuracy was also observed on the UTKinect, although it is higher than the classification accuracy observed on the UTD-MHAD dataset. The UTKinect confusion matrix for the original MMNet fused model, presented in

Table 6.6 Classification accuracy (%) for NTU-60 (XVIEW) actions where MMNet-MS (fused) outperforms MMNet (fused)

Action	MMNet (fused)	MMNet-MS (fused)
drop	97	99
throw	98	99
clapping	96	97
reading	68	80
put on a shoe	94	96
take off a shoe	89	93
hopping	99	100
play with phone/tablet	84	88
rub two hands	93	96
salute	98	99
pushing	98	99
touch pocket	95	96

Figure 6.11, shows that the model is able to recognise the “clap hands” and “wave hands” actions with 100% accuracy, while the “carry” and “push” actions are both classified with 83% accuracy. Additionally, it is able to recognise the “walk”, “sit down” and “stand up” actions somewhat accurately with an accuracy of 67%. However, it is unable to classify the “pick up” action at all, that is 0% accuracy was observed, while the accuracy observed for the “throw” and “pull” actions is less than 50%. The confusion matrix for MMNet-MS (fused), presented in Figure 6.11, shows a similar trend in that the model is only able to classify the “walk”, “sit down”, “carry”, “push”, “wave hands” and “clap hands” actions. However, a significantly higher classification accuracy was observed for the “walk” action, 83% compared to 67% for the original model without motion saliency, while a significantly lower accuracy was observed for the “stand up” action, 50% compared to 67% for the original model without motion saliency. Overall, the high classification accuracy for certain class of actions, in both the UTKinect and UTD-MHAD datasets, while a near-zero classification accuracy for other classes is an indication of overfitting due to the small size of the training set.

6.2 Time and Power Consumption

As mentioned in previous sections, the aim of replacing the OpenPose skeleton based strategy for determining an ST-ROI with motion saliency is to reduce the total time spent in training and inference as well as to reduce the power consumption of the HAR system. The time spent in training was recorded for each experiment.

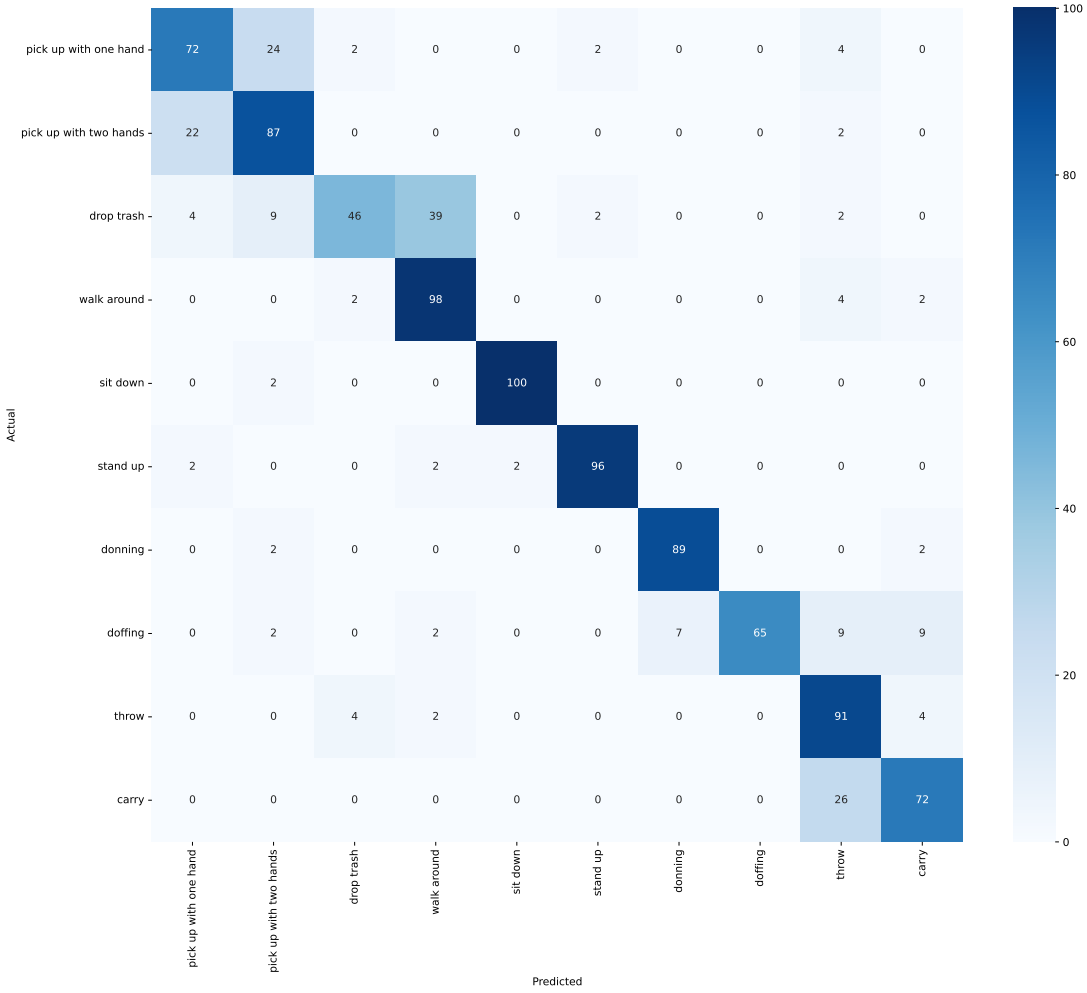


Figure 6.5 Confusion matrix for original MMNet (fused) on NW-UCLA

The training time of each experiment, presented in Table 6.7, shows that the training time is significantly reduced when using the ST-ROIs identified by motion saliency detection instead of those identified by the OpenPose skeleton data. MMNet-MS consistently takes less time to train than the original MMNet model across all datasets. The most notable difference in training time is observed on the NTU-60 dataset, with both training protocols. The MMNet-MS RGB model, that is trained on ST-ROI images only, completes training, on the NTU-60 (XSUB) dataset, in 3.75 hours while the original MMNet RGB model takes 19.51 hours, which is approximately five times longer. Similar results are observed with the fused models where MMNet-MS (fused) completes training, on the NTU-60 (XSUB) dataset, in 10.98 hours while the original fused model takes 43.44 hours, which is nearly four times longer. Less of a reduction in training time was observed on the NTU-120 dataset. However, the reduction in training time is still significant, where MMNet-MS (fused) took 47.79 hours to train on the NTU-120 (XSUB) dataset while the original fused model took 73.63 hours, which is 1.5 times longer.

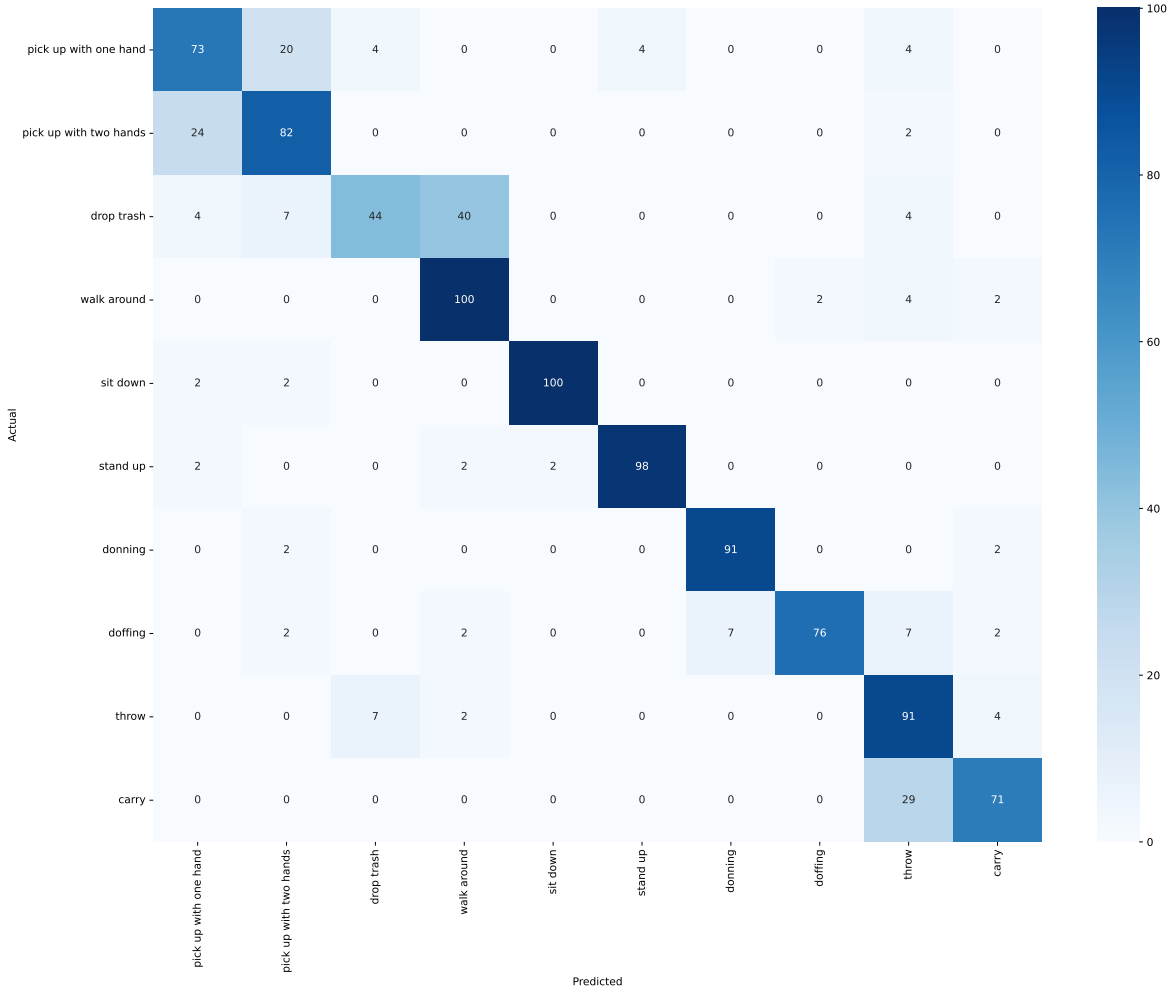


Figure 6.6 Confusion matrix for MMNet-MS (fused) on NW-UCLA

The total training time shows the amount of time taken to reach the optimal model parameters. However, an accuracy close to the maximum is generally reached early in the training process while the remainder of the epochs perform small adjustments to the model parameters which don't affect the accuracy significantly. Figures 6.13 and 6.14 show the relation between classification accuracy and training time for the MMNet RGB models and fused models, respectively. The plots show that the MMNet-MS RGB model begins converging towards its maximum classification accuracy in less than one hour while the original MMNet RGB model begins converging towards its maximum accuracy in approximately three hours. A similar trend is observed for the fused models.

The average inference time per sample gives an indication of the real-time performance of a HAR method. A lower training time may imply a lower inference time as well however this is not a given. The average sample inference times recorded during the experiments for each dataset are shown in Table 6.8. The results show that motion saliency reduces the inference time of the MMNet model compared to using

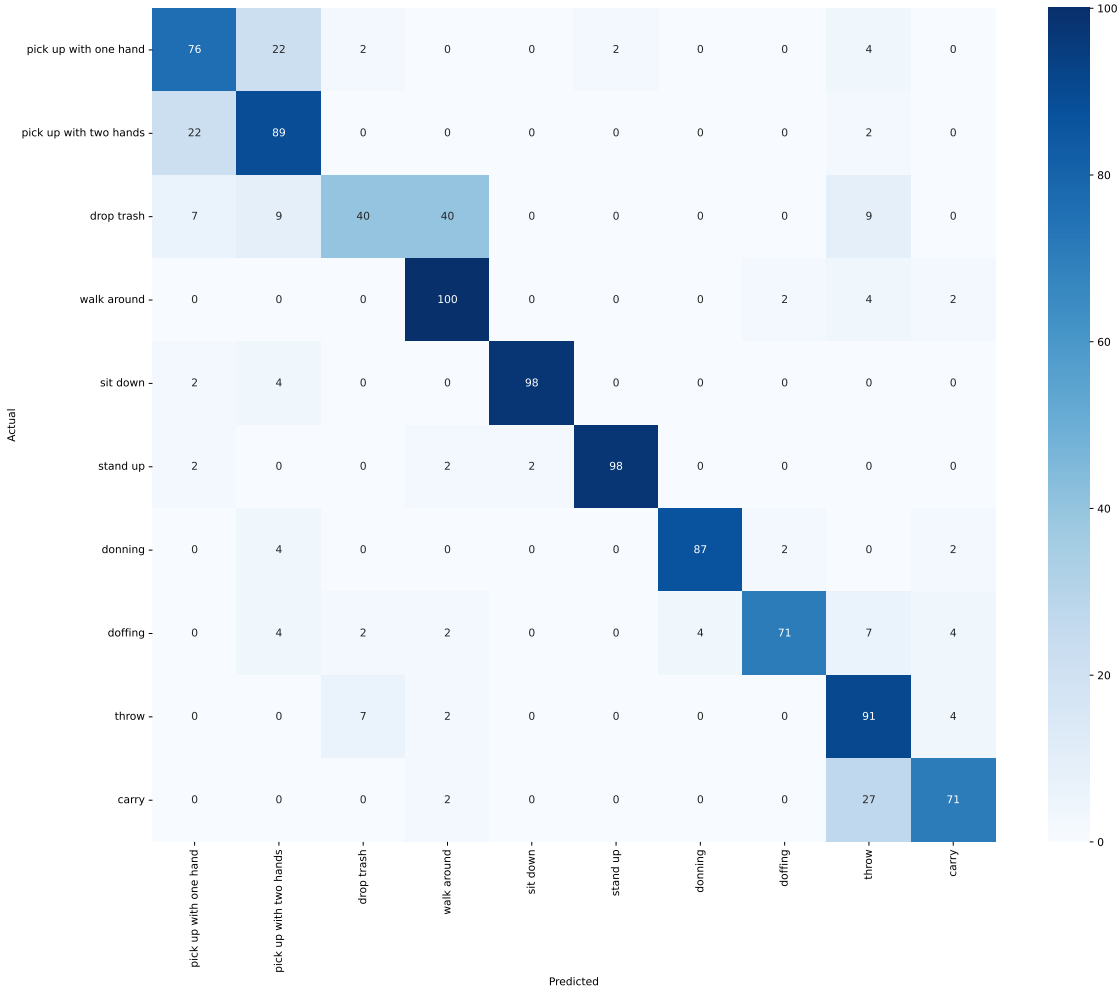


Figure 6.7 Confusion matrix for original MMNet (fused) on NW-UCLA (depth)

the ST-ROI identified by OpenPose skeleton data. MMNet-MS delivers an action classification on average 1.5 to 5 times faster than the original model. The most pronounced difference in the inference times of the RGB models was observed on the NTU-60 (XSUB) dataset where MMNet-MS takes 4 milliseconds to classify a video while the original model takes 19 milliseconds. The largest difference in the inference times of the fused models was observed on the NTU-120 (XSUB) dataset where MMNet-MS takes 9 milliseconds to classify a video while the original fused model takes 45 milliseconds, which is 5 times longer.

The times taken to run OpenPose skeleton estimation and motion saliency are shown in Table 6.9 for each dataset used in the experiments. The results show that motion saliency detection is between 2 to 12 times faster than OpenPose skeleton estimation. The most notable difference was observed on NTU-60 and NTU-120 where motion saliency detection took 8.74 and 17.81 hours on NTU-60 and NTU-120, respectively, for the entire dataset while OpenPose skeleton estimation took 99.17 and 217.80 hours, respectively. The smallest difference was observed on UTKinect where

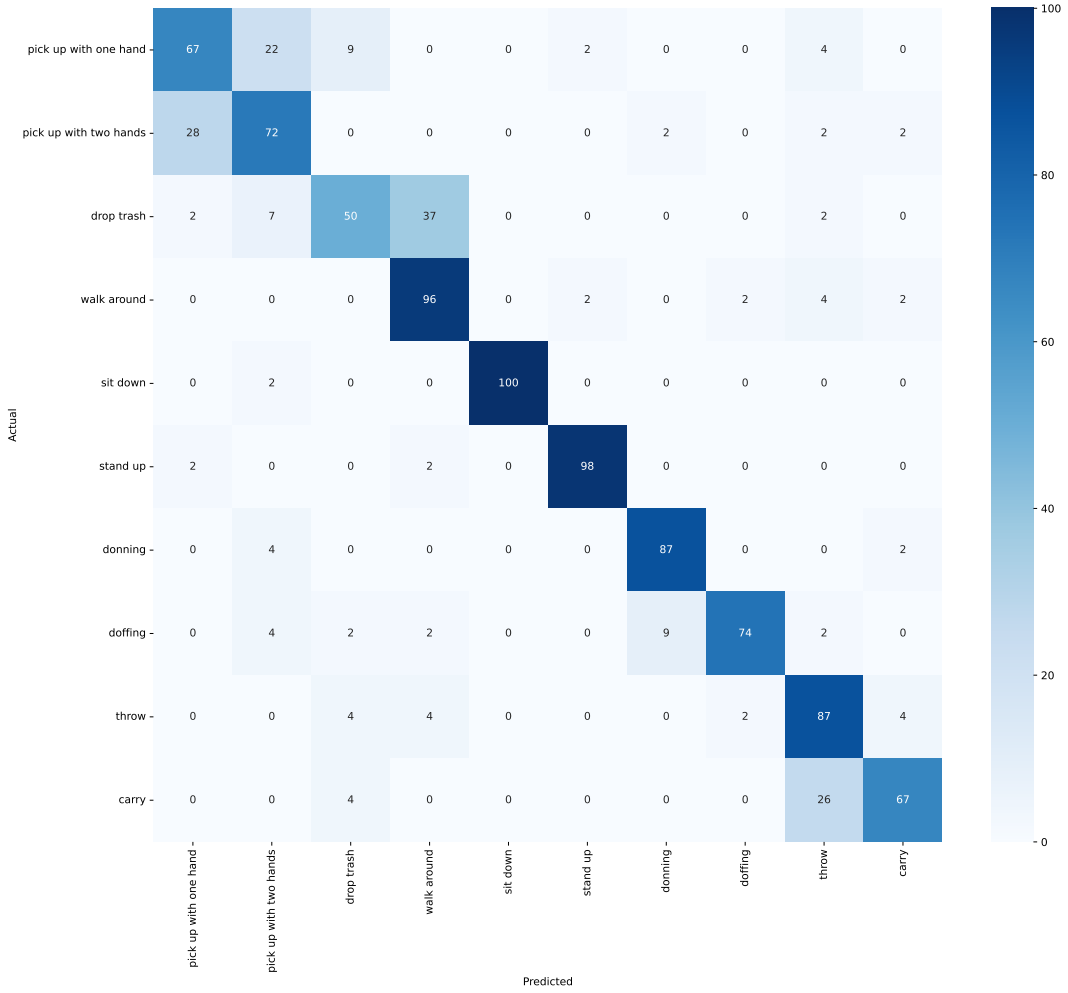


Figure 6.8 Confusion matrix for MMNet-MS (fused) on NW-UCLA (depth)

motion saliency detection took 0.02 hours, i.e 1.2 minutes, while OpenPose skeleton estimation took 0.04 hours, i.e. 2.4 minutes. This is due to UTKinect being the smallest dataset consisting of videos with a low frame rate, which decrease the computational time of both OpenPose skeleton estimation and motion saliency detection.

These results show that OpenPose skeleton estimation is a computationally expensive step that significantly increases both the training and inference time of the MMNet model. Motion saliency detection has been shown to be significantly faster than OpenPose skeleton, while the impact on the classification accuracy of the model, discussed in Section 6.1, is minimal. Thus motion saliency detection is a viable alternative to OpenPose skeleton estimation for identifying the ST-ROI in applications where a lower training and inference time is required.

The recorded average GPU power consumption, measured in Watts, during training using each dataset is shown in Table 6.10. It is evident that MMNet-MS has a higher power consumption, in all datasets, than the original model, which indicates that it consumes more energy per second. A possible reason for this is that given the size of

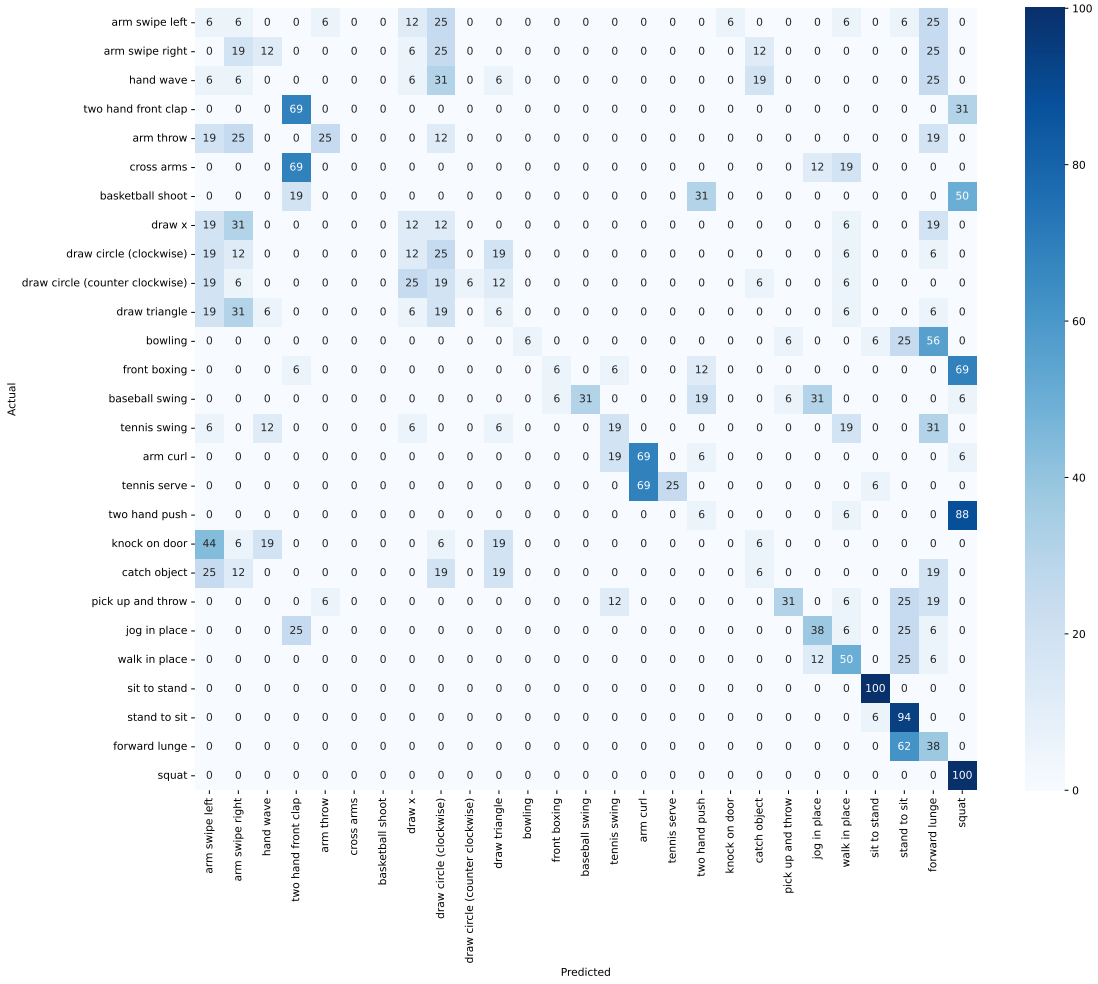


Figure 6.9 Confusion matrix for original MMNet (fused) on UTD-MHAD

the data that is being processed is reduced, less time is being spent transferring data to and from GPU memory while more time is being spent in computation. On the other hand the total power consumed during training, which is shown in Table 6.11, is less for MMNet-MS than the original model across most of the datasets. This is due to the training time being significantly decreased. The exception is the NTU-120 dataset where the total power consumption of MMNet-MS is higher than MMNet. In this case the decrease in training time is not sufficient to offset the increase in average power consumption. These results show that while motion saliency detection has the potential to reduce the total power consumption of a HAR model, this is not guaranteed and is dependent on the reduction of the training time that is achieved.

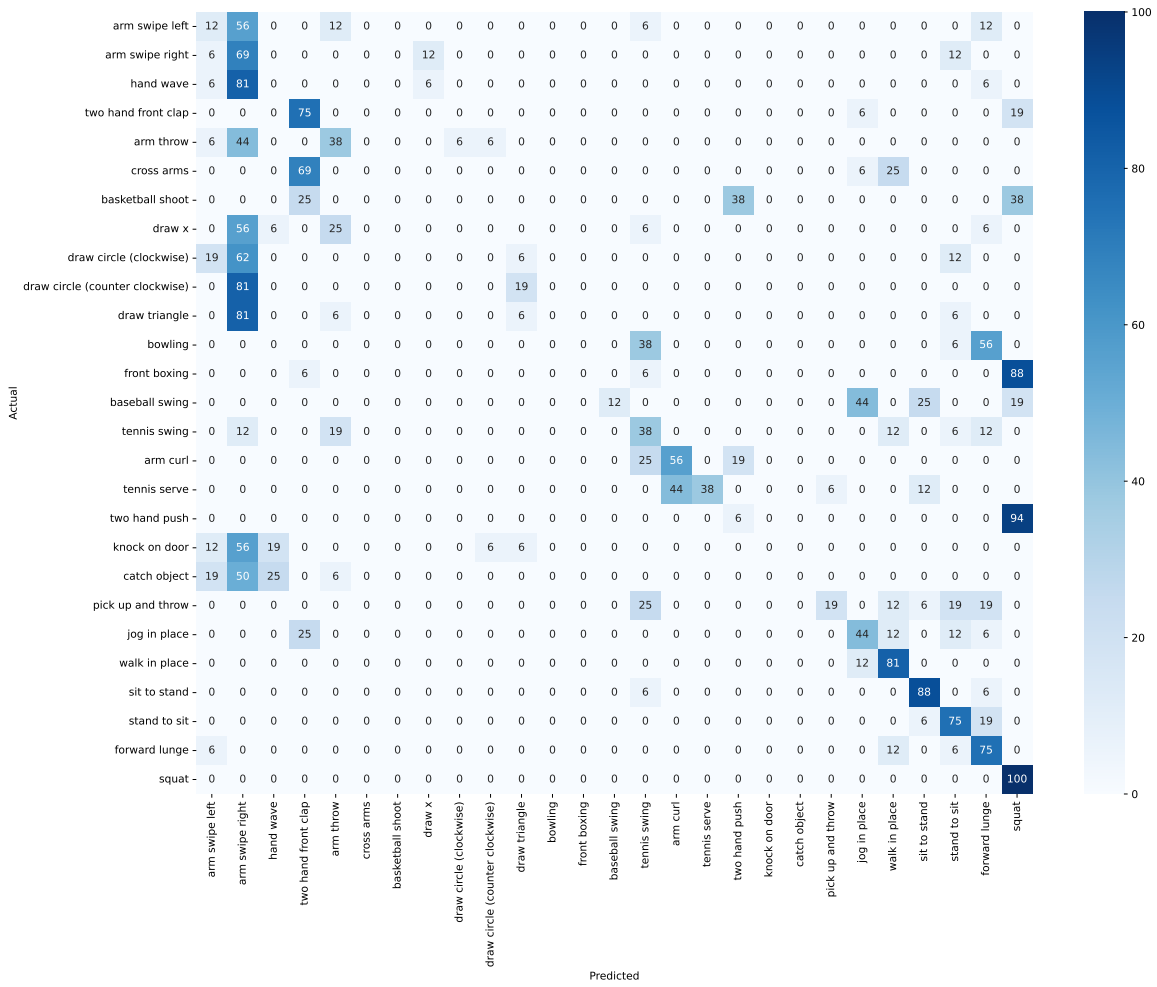


Figure 6.10 Confusion matrix for MMNet-MS (fused) on UTD-MHAD

Table 6.7 Time (hours) spent in training during experiments

Method	Dataset									
	NTU-60 (XSUB)	NTU-60 (XVIEW)	NTU-120 (XSUB)	NTU-120 (XSET)	NW-UCLA	NW-UCLA (depth)	TST	HWU-USP	UTD	UTK
MMNet (RGB)	19.51	23.52	48.37	45.12	0.15	0.14	0.27	0.05	0.08	0.03
MMNet-MS (RGB)	**3.75**	**8.07**	**19.87**	**20.31**	**0.10**	**0.09**	**0.19**	**0.03**	**0.05**	**0.02**
MMNet (fused)	43.44	40.39	73.63	71.05	0.20	0.19	–	–	0.09	0.05
MMNet-MS (fused)	**10.98**	**10.34**	**47.79**	**49.89**	**0.15**	**0.15**	–	–	**0.06**	**0.03**

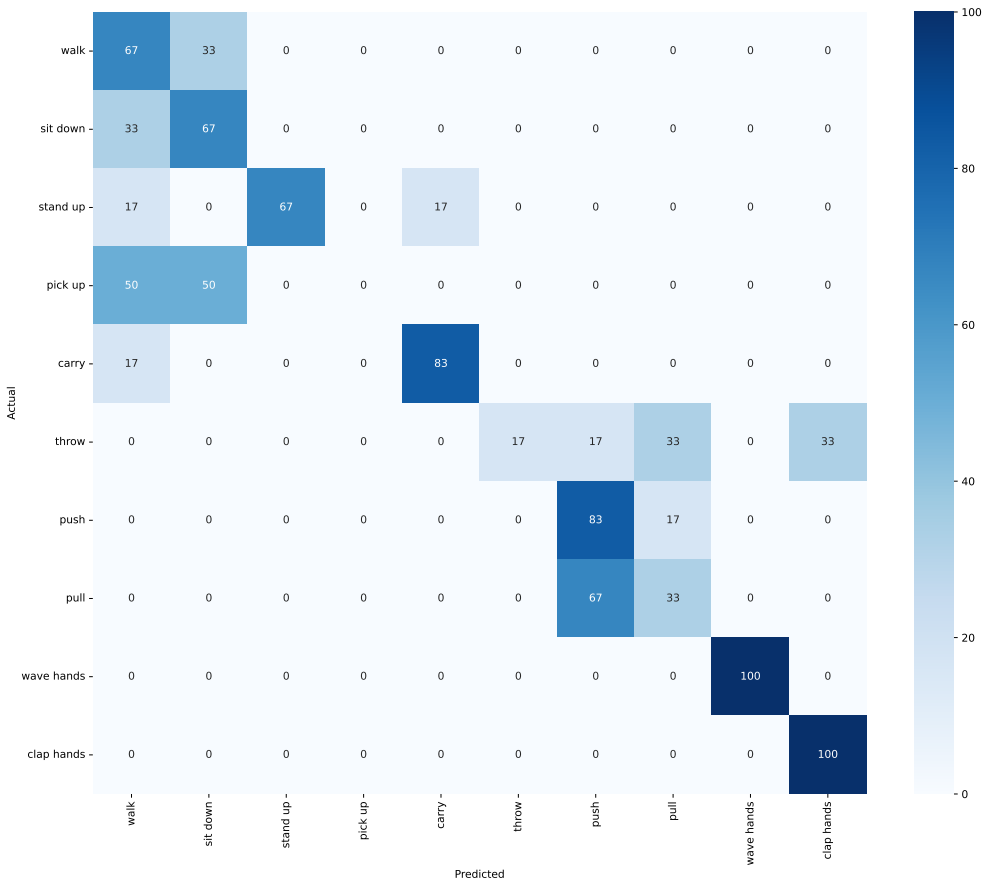


Figure 6.11 Confusion matrix for original MMNet (fused) on UTKinect

Table 6.8 Average inference time (milliseconds) per sample

Method	Dataset									
	NTU-60 (XSUB)	NTU-60 (XVIEW)	NTU-120 (XSUB)	NTU-120 (XSET)	NW-UCLA	NW-UCLA (depth)	TST	HWU-USP	UTD	UTK
MMNet (RGB)	19	22	21	20	6	6	17	14	7	12.5
MMNet-MS (RGB)	**4**	**6**	**5**	**7**	**4**	**4**	**7**	**6**	**3**	**8**
MMNet (fused)	45	44	45	40	8	8	–	–	8	17
MMNet-MS (fused)	**11**	**10**	**9**	**11**	**6**	**6**	–	–	**4**	**11**

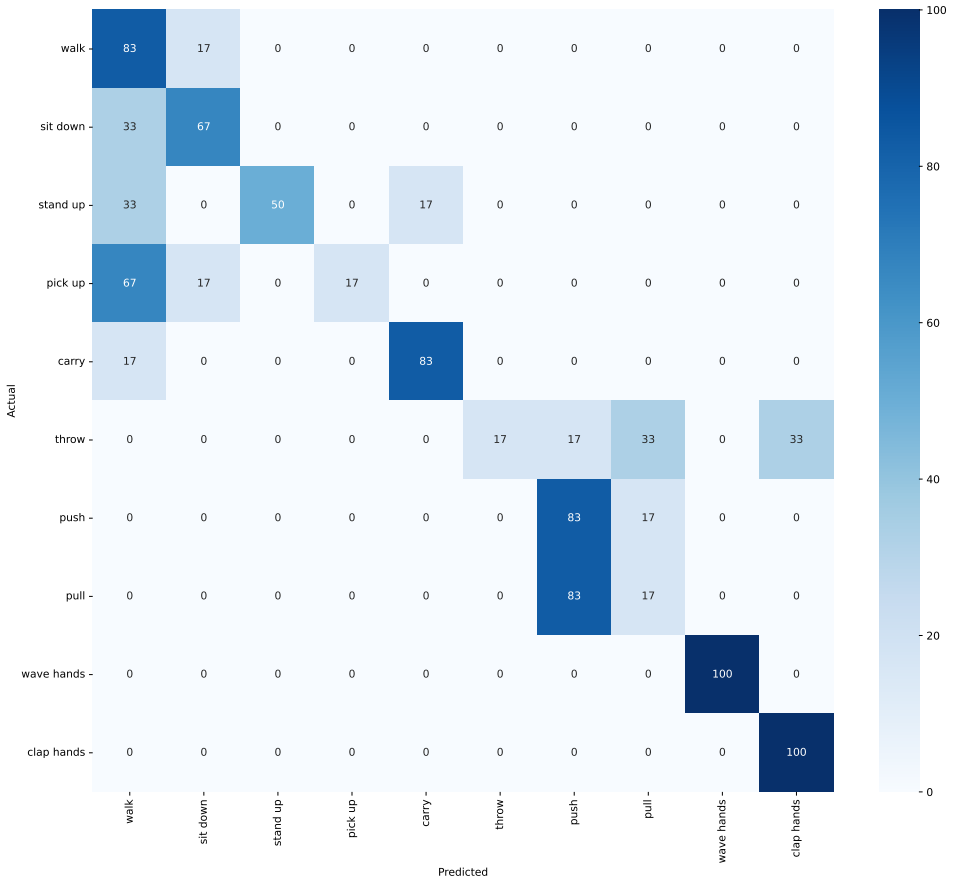


Figure 6.12 Confusion matrix for MMNet-MS (fused) on UTKinect

Table 6.9 Comparison of time (hours) taken by OpenPose and Motion Saliency Detection

Method	Dataset							
	NTU-60	NTU-120	NW-UCLA	NW-UCLA (depth)	TST	HWU-USP	UTD	UTK
OpenPose	99.17	217.80	0.83	0.83	0.98	0.48	0.35	0.04
Motion Saliency	**8.74**	**17.81**	**0.10**	**0.12**	**0.13**	**0.06**	**0.05**	**0.02**

Table 6.10 Average GPU power (W) consumption during training

Method	Dataset							
	NTU-60 (XSUB)	NTU-60 (XVIEW)	NTU-120 (XSUB)	NTU-120 (XSET)	NW-UCLA	NW-UCLA (depth)	UTD	UTK
MMNet (fused)	**80.740**	**82.880**	**78.360**	**79.150**	**87.450**	**87.440**	**77.470**	**66.960**
MMNet-MS (fused)	167.15	169.37	147.23	150.90	98.51	100.14	91.56	75.90

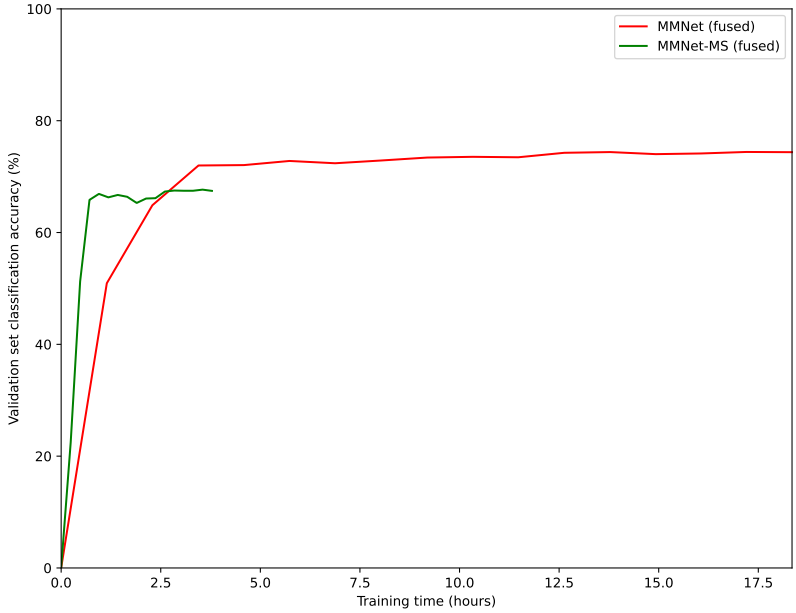


Figure 6.13 Classification accuracy vs training time for RGB models on NTU-60 (XSUB)

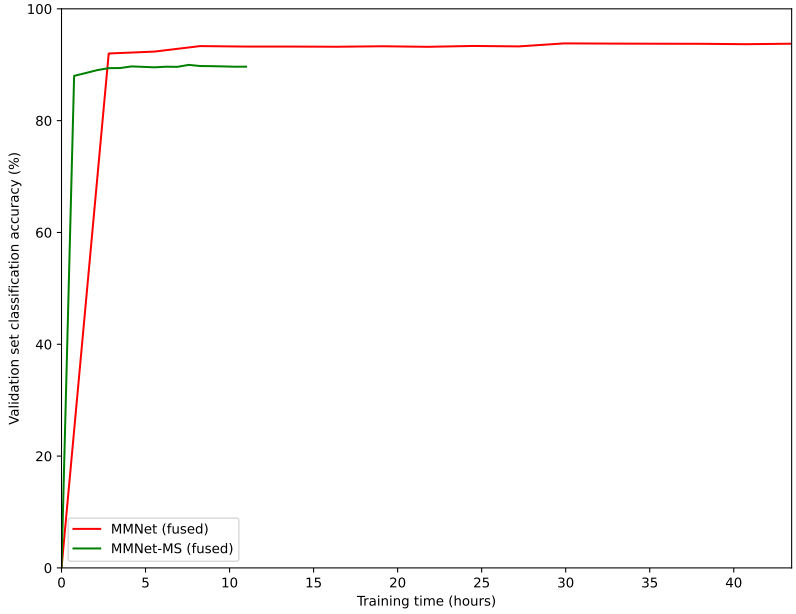


Figure 6.14 Classification accuracy vs training time for fused models on NTU-60 (XSUB)

Table 6.11 Total GPU power (kWh) consumption during training

Method	Dataset							
	NTU-60 (XSUB)	NTU-60 (XVIEW)	NTU-120 (XSUB)	NTU-120 (XSET)	NW-UCLA	NW-UCLA (depth)	UTD	UTK
MMNet (fused)	3.521	3.347	**5.770**	**5.623**	0.017	0.017	0.007	0.003
MMNet-MS (fused)	**1.840**	**1.750**	7.036	7.528	**0.015**	**0.015**	**0.005**	**0.002**

7 Conclusion

In this work research was done into the state of the art of human action recognition. It was identified that the existing solutions that were discussed, all require enormous datasets for training and expensive hardware. This results in a long training time which reduces the practically of adapting a HAR method to different environments. This long training time is the result of the large amount of data contained in video. Most of this data consists of background clutter, which is not relevant to successful human action classification and in some cases was actually found to negatively affect the classification accuracy of HAR methods, such as those based on 3D-CNNs. Thus, an opportunity was identified for significantly reducing the time spent in training and inference by limiting the data that is processed by the classifier to only those regions that capture the action taking place. The aim of this thesis was to develop a new HAR method that requires less time for training and inference whilst achieving a comparable accuracy to state of the art methods.

Through research, it was found that the first objective, namely developing a solution for identifying the regions where the action is taking place, can be addressed using motion saliency detection techniques, which deal with identifying salient regions in video based on the presence of motion. Existing state of the art motion saliency detection solutions, such as BSUV-Net 2.0 [183], were found to be far too computationally intensive for any benefit, in terms of reduced training time, to be realised. Furthermore, such methods require training in addition to the training of the main HAR classifier. The difference of frames method was identified as a computationally efficient method for identifying regions containing motion, however this method alone is vulnerable to noise. Given that RGB+D video is becoming increasingly accessible with commercial RGB+D cameras such as Microsoft Kinect and Intel RealSense available, the use of depth information to limit the regions identified by frame difference to only those regions capturing actual motion was explored. A new motion saliency solution based on this strategy was developed. Through experimentation, detailed in Section 4.3, it was shown that the developed motion saliency solution is both faster and more accurate than existing solutions such as BSUV-Net 2.0 [183] and GMM [182]. An improvement to the developed motion saliency solution, that tracks salient regions between frames using Kalman filtering, was also developed. Experimentation showed a small improvement in accuracy with a negligible increase in computational time. Two papers were published. The first paper [4] details the motion saliency detection method while the second paper [5] details the improved method using Kalman filtering. The developed motion saliency solution, discussed in 4.2, addresses the objective of identifying the regions that capture human

actions in video.

The second objective, using the developed saliency solution for identifying action regions in a HAR pipeline to reduce the size of the data that is processed by the classifier, was addressed by exploring the application of motion saliency in various HAR methods. The application of motion saliency detection in 3D-CNN HAR methods, namely ResNet3D-50 since it has been identified as being the most widely used 3D-CNN model, was explored. Motion saliency was applied on the frames to limit the 3D tensor input, which consists of 2D colour images stacked along a third time dimension, to only the region capturing the action taking place. The developed strategy involves using the salient regions, identified in each frame, to identify a spatiotemporal salient region throughout the entire video. For a reduction in training time to be realised, the implementation of the motion saliency detection solution was optimised using parallelisation across multiple CPU threads and the GPU. The experimental results showed a significant reduction in training time however it also showed a low classification accuracy on both the UTKinect and NTU-60 datasets. The addition of motion saliency resulted in an increase in classification accuracy, however the accuracy is still too low for the method to be considered for practical applications. It was deduced that the low classification accuracy is due to the datasets consisting of actions taking place in indoor settings, in which these methods have been identified, in previous studies [1], to perform poorly. Thus, the application of motion saliency in other HAR methods was explored.

The application of motion saliency in the MMNet [1] method, which is a recent state of the art multimodal HAR method, was explored since the method has been reported to achieve a superior performance in indoor settings which are particularly relevant to assisted-living and surveillance applications. Motion saliency was used as an alternative strategy for identifying an ST-ROI with the objective of eliminating the costly OpenPose [2] skeleton estimation step. The developed strategy involves identifying salient regions in each frame, which are then concatenated together into a single frame image capturing the ST-ROI of the video. This was done to replace the existing strategy which involves estimating skeleton joint coordinates with the OpenPose tool to identify spatial regions of interest surrounding the joints, which are then concatenated into a single ST-ROI image. An ablation study was conducted to determine the optimal configuration, in which the salient regions are concatenated to form a single image. The results of the study showed that the highest classification accuracy was achieved with an ST-ROI constructed out of sixteen salient regions concatenated horizontally in a row. Additionally, it was found that an ST-ROI with a fixed height of 60 pixels results in the greatest reduction to training time with a minimal impact on classification accuracy. This addresses the objectives of optimising the HAR pipeline such that the computational time and power consumption is reduced

and finding the optimal parameters such that the resulting classification accuracy is comparable to existing HAR solutions.

To address the final objective, namely evaluating the performance of the solution and comparing it to the state of the art, the performance of the new method, referred to as MMNet-MS, which uses motion saliency to identify an ST-ROI, was evaluated in terms of classification accuracy, training time, inference time and power consumption while being compared to the original MMNet method as well as the HiCo [142], MCAE [121] and SSC [104] methods. The experimental results showed that MMNet-MS achieves a comparable classification accuracy to MMNet across all datasets, both when using colour information only and with the fused model that uses colour and skeleton data. Additionally, MMNet-MS was observed to significantly outperform MMNet on the TST fall detection [3] dataset, which is a privacy preserving dataset that only provides depth information without colour video. With regards to training, MMNet-MS was observed to be 1.5 to 6 times faster than MMNet across all datasets. Similarly, it was observed that MMNet-MS delivers an action classification for a video 2 to 9 times faster than MMNet. Finally, with regards to GPU power consumption, MMNet-MS was observed to consume less total power during training than MMNet in all evaluation datasets except NTU-120 [127]. A paper detailing the MMNet-MS method was published [6].

To summarise the main contributions of this work, a new motion saliency solution was developed and its application in human action recognition, with the objective of reducing the time and power consumption of HAR systems, was explored. This solution was used to develop a new HAR method MMNet-MS, which is based on the MMNet method. The results showed a significant reduction in training and inference time while maintaining a comparable classification accuracy. Given that a considerable increase in classification accuracy was observed on the TST fall detection [3] dataset, which contains samples depicting human falls and actions related to daily living in a private setting, there is a significant opportunity for future work in assisted living applications. In particular, the proposed MMNet-MS method can be incorporated in a larger assisted living system for detecting human falls and accidents. The addition of action localisation in time, such that MMNet-MS only processes those frames that capture the action is a potential avenue for future research, the results of which could lead to a more robust HAR system. Other avenues include research into further improving the classification accuracy of MMNet-MS particularly on smaller datasets, where a poor performance was observed. Additionally the large amount of energy consumed by training machine learning models is a concern, therefore further research intro reducing power consumption is needed in order for machine learning to be sustainable.

References

[1] B. X. Yu, Y. Liu, X. Zhang, S.-h. Zhong, and K. C. Chan, "Mmnet: A model-based multimodal network for human action recognition in rgb-d videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2022, ISSN: 1939-3539. DOI: 10.1109/tpami.2022.3177813.

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.143.

[3] M. S. Singh, V. Pondenkandath, B. Zhou, P. Lukowicz, and M. Liwickit, "Transforming sensor data to the image domain for deep learning — an application to footstep detection," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, May 2017. DOI: 10.1109/ijcnn.2017.7966182.

[4] A. Gutev and C. J. Debono, "Motion saliency detection using depth information for human action recognition applications," in *2023 International Symposium ELMAR*, IEEE, Sep. 2023. DOI: 10.1109/elmar59410.2023.10253928.

[5] A. Gutev and C. Debono, "Kalman filter aided depth-based motion saliency detection in human activity recognition applications," in *2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)*, IEEE, Jun. 2024. DOI: 10.1109/melecon56669.2024.10608651.

[6] A. Gutev and C. J. Debono, "Motion saliency based human action recognition in RGBD videos," in *2024 IEEE 20th International Conference on Intelligent Computer Communication and Processing (ICCP)*, IEEE, Oct. 2024, pp. 1–6. DOI: 10.1109/iccp63557.2024.10793009.

[7] W. Ma, X. Zhu, W. Xiang, W. Li, and Z. Wang, "Human behavior recognition and detection technology based on video stream," in *2023 International Conference on Intelligent Media, Big Data and Knowledge Mining (IMBDKM)*, vol. 42, IEEE, Mar. 2023, pp. 69–75. DOI: 10.1109/imbdkm57416.2023.00020.

[8] C. J. Debono, M. Sacco, and J. Ellul, "Monitoring indoor living spaces using depth information," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, IEEE, Nov. 2020. DOI: 10.1109/icce-berlin50680.2020.9352158.

[9] Z. Ahmad and N. Khan, "Human action recognition using deep multilevel multimodal (M^2) fusion of depth and inertial sensors," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1445–1455, Feb. 2020, ISSN: 2379-9153. DOI: 10.1109/jsen.2019.2947446.

[10] R. Poppe, “A survey on vision-based human action recognition,” *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, Jun. 2010, ISSN: 0262-8856. DOI: 10.1016/j.imavis.2009.11.014.

[11] F. Zhao, Z. Cao, Y. Xiao, J. Mao, and J. Yuan, “Real-time detection of fall from bed using a single depth camera,” *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1018–1032, Jul. 2019, ISSN: 1558-3783. DOI: 10.1109/tase.2018.2861382.

[12] K. Host and M. Ivašić-Kos, “An overview of human action recognition in sports based on computer vision,” *Heliyon*, vol. 8, no. 6, e09633, Jun. 2022, ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2022.e09633.

[13] H. Zhang, F. Zhou, W. Zhang, X. Yuan, and Z. Chen, “Real-time action recognition based on a modified deep belief network model,” in *2014 IEEE International Conference on Information and Automation (ICIA)*, IEEE, Jul. 2014, pp. 225–228. DOI: 10.1109/icinfa.2014.6932657.

[14] Y. Wang, S. He, X. Wei, and S. A. George, “Research on an effective human action recognition model based on 3d cnn,” in *2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, IEEE, Nov. 2022, pp. 1–6. DOI: 10.1109/cisp-bmei56279.2022.9980092.

[15] Y. Kong, Y. Wang, and A. Li, “Spatiotemporal saliency representation learning for video action recognition,” *IEEE Transactions on Multimedia*, vol. 24, pp. 1515–1528, 2022, ISSN: 1941-0077. DOI: 10.1109/tmm.2021.3066775.

[16] Y. L. Cun *et al.*, “Handwritten digit recognition with a back-propagation network,” *Advances in Neural Information Processing Systems 2*, pp. 396–404, 1990.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, ISSN: 1557-7317. DOI: 10.1145/3065386.

[18] C.-F. R. Chen *et al.*, “Deep analysis of cnn-based spatio-temporal representations for action recognition,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021. DOI: 10.1109/cvpr46437.2021.00610.

[19] C. Szegedy *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298594.

[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.90.

[21] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2012. DOI: 10.1109/cvpr.2012.6248110.

[22] Y. LeCun, F. J. Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, IEEE. DOI: 10.1109/cvpr.2004.1315150.

[23] F. Li, R. Fergus, P. Perona, and D. Zekrifa, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[24] N. Pinto, D. D. Cox, and J. J. DiCarlo, “Why is real-world visual object recognition hard?” *PLoS Computational Biology*, vol. 4, no. 1, K. J. Friston, Ed., e27, Jan. 2008, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.0040027.

[25] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, “Labelme: A database and web-based tool for image annotation,” *International Journal of Computer Vision*, vol. 77, no. 1–3, pp. 157–173, Oct. 2007, ISSN: 1573-1405. DOI: 10.1007/s11263-007-0090-8.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2009. DOI: 10.1109/cvpr.2009.5206848.

[27] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?” In *2009 IEEE 12th International Conference on Computer Vision*, IEEE, Sep. 2009. DOI: 10.1109/iccv.2009.5459469.

[28] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML ’09, ACM, Jun. 2009. DOI: 10.1145/1553374.1553453.

[29] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox, "A high-throughput screening approach to discovering good forms of biologically inspired visual representation," *PLoS Computational Biology*, vol. 5, no. 11, K. J. Friston, Ed., e1000579, Nov. 2009, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000579.

[30] S. C. Turaga *et al.*, "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural Computation*, vol. 22, no. 2, pp. 511–538, Feb. 2010, ISSN: 1530-888X. DOI: 10.1162/neco.2009.10-08-881.

[31] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.502.

[32] D. Tran, H. Wang, M. Feiszli, and L. Torresani, "Video classification with channel-separated convolutional networks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00565.

[33] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," Columbus, OH, USA: IEEE, Jun. 2014, pp. 1725–1732, ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.223.

[34] M. Monfort *et al.*, "Moments in time dataset: One million videos for event understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 502–508, 2 Feb. 1, 2020, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2901464.

[35] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 1, Montreal, Canada, 2014, pp. 568–576.

[36] Q. Fan, R. C. Chun-Fu, H. Kuehne, M. Pistoia, and D. Cox, "More is less: Learning efficient video representations by big-little network and depth wise temporal aggregation," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Dec. 2019, pp. 2264–2273.

[37] G. Cheron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.368.

[38] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2008. DOI: 10.1109/cvpr.2008.4587756.

[39] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local svm approach,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, 2004. DOI: 10.1109/icpr.2004.1334462.

[40] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *2013 IEEE International Conference on Computer Vision*, IEEE, Dec. 2013. DOI: 10.1109/iccv.2013.441.

[41] G. Gkioxari and J. Malik, “Finding action tubes,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298676.

[42] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7299101.

[43] J. Donahue *et al.*, “Long-term recurrent convolutional networks for visual recognition and description,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298878.

[44] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised learning of video representations using LSTMs,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, Lille, France, 2015, pp. 843–852.

[45] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to sequence – video to text,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.515.

[46] L. Wang *et al.*, “Temporal segment networks: Towards good practices for deep action recognition,” in *European Conference on Computer Vision (ECCV)*, Springer International Publishing, 2016, ISBN: 9783319464848. DOI: 10.1007/978-3-319-46484-8_2.

[47] L. Wang *et al.*, “Temporal segment networks for action recognition in videos,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2740–2755, Nov. 2019, ISSN: 1939-3539. DOI: 10.1109/tpami.2018.2868668.

[48] J. Lin, C. Gan, and S. Han, “Tsm: Temporal shift module for efficient video understanding,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00718.

[49] B. Zhou, A. Andonian, and A. Torralba, “Temporal relational reasoning in videos,” in *European Conference on Computer Vision (ECCV)*, 2018, pp. 803–818.

[50] B. Fernando, E. Gavves, M. Jose Oramas, A. Ghodrati, and T. Tuytelaars, “Modeling video evolution for action recognition,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7299176.

[51] Z. Xu, Y. Yang, and A. G. Hauptmann, “A discriminative cnn video representation for event detection,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298789.

[52] X. Peng, C. Zou, Y. Qiao, and P. Qiang, “Action recognition with stacked fisher vectors,” in *European Conference on Computer Vision (ECCV)*, Springer International Publishing, 2014, pp. 581–595.

[53] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, “Actionvlad: Learning spatio-temporal aggregation for action classification,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.337.

[54] R. Girdhar, J. Joao Carreira, C. Doersch, and A. Zisserman, “Video action transformer network,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019. DOI: 10.1109/cvpr.2019.00033.

[55] K. Hara, H. Kataoka, and Y. Satoh, “Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?” In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00685.

[56] K. Hara, H. Kataoka, and Y. Satoh, “Towards good practice for action recognition with spatiotemporal 3d convolutions,” in *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8546325.

[57] 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, Dec. 2015. DOI: 10.1109/iccv.2015.510.

[58] 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, Jan. 2013, ISSN: 2160-9292. DOI: 10.1109/tpami.2012.59.

[59] K. Hara, H. Kataoka, and Y. Satoh, “Learning spatio-temporal features with 3d residual networks for action recognition,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, IEEE, Oct. 2017. DOI: 10.1109/iccvw.2017.373.

[60] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994, ISSN: 1941-0093. DOI: 10.1109/72.279181.

[61] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *International Conference on Artificial Intelligence and Statistics*, 2010.

[62] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.123.

[63] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, Lille, France, 2015, pp. 448–456.

[64] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 630–645, ISBN: 9783319464930. DOI: 10.1007/978-3-319-46493-0_38.

[65] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in *Proceedings of the British Machine Vision Conference*, May 2016.

[66] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.634.

[67] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.243.

[68] Z. Shou, D. Wang, and S.-F. Chang, “Temporal action localization in untrimmed videos via multi-stage cnns,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.119.

[69] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, “Jointly modeling embedding and translation to bridge video and language,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.497.

[70] S. Paul, S. Roy, and A. K. Roy-Chowdhury, “W-TALC: Weakly-supervised temporal activitylocalization and classification,” in *European Conference on Computer Vision (ECCV)*, 2018, pp. 563–579.

[71] R. Panda and A. K. Roy-Chowdhury, “Collaborative summarization of topic-related videos,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.455.

[72] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: A large video database for human motion recognition,” in *2011 International Conference on Computer Vision*, IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126543.

[73] K. Soomro, A. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” *CRCV-TR*, Dec. 2012.

[74] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, “ActivityNet: A large-scale video benchmark for human activity understanding,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298698.

[75] J. K. Aggarwal and L. Xia, “Human activity recognition from 3D data: A review,” *Pattern Recognition Letters*, vol. 48, pp. 70–80, 2014.

[76] J. Aggarwal and M. Ryoo, “Human activity analysis: A review,” *ACM Computing Surveys*, vol. 43, no. 3, pp. 1–43, Apr. 2011, ISSN: 1557-7341. DOI: 10.1145/1922649.1922653.

[77] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: A survey,” *Pattern Recognition Letters*, vol. 119, pp. 3–11, Mar. 2019, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2018.02.010.

[78] Y. Bengio, “Deep learning of representations: Looking forward,” in *International Conference on Statistical Language and Speech Processing*, Springer, Jul. 2013, pp. 1–37.

[79] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, “Deep convolutional neural networks on multichannel time series for human activity recognition,” in *International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina, 2015, pp. 25–31.

[80] Q. Yang, “Activity recognition: Linking low-level sensors to high-level intelligence,” in *International Joint Conference on Artificial Intelligence*, 2009, pp. 20–25.

[81] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, ISSN: 1530-888X. DOI: 10.1162/neco.2006.18.7.1527.

[82] M. Zeng *et al.*, “Convolutional neural networks for human activity recognition using mobile sensors,” in *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*, ser. MobiCASE, ICST, 2014. DOI: 10.4108/icst.mobicase.2014.257786.

[83] Y. Chen and Y. Xue, “A deep learning approach to human activity recognition based on single accelerometer,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, Oct. 2015. DOI: 10.1109/smc.2015.263.

[84] N. Y. Hammerla, S. Halloran, and T. Ploetz, “Deep, convolutional, and recurrent models for human activity recognition using wearables,” in *International Joint Conference on Artificial Intelligence*, 2016.

[85] A. Sathyanarayana *et al.*, “Sleep quality prediction from wearable data using deep learning,” *JMIR mHealth and uHealth*, vol. 4, no. 4, e125, Nov. 2016, ISSN: 2291-5222. DOI: 10.2196/mhealth.6562.

[86] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, “Deep convolutional neural networks and learning ecg features for screening paroxysmal atrial fibrillation patients,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 12, pp. 2095–2104, Dec. 2018, ISSN: 2168-2232. DOI: 10.1109/tsmc.2017.2705582.

[87] S. Ha, J.-M. Yun, and S. Choi, “Multi-modal convolutional neural networks for activity recognition,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, Oct. 2015. DOI: 10.1109/smc.2015.525.

[88] W. Jiang and Z. Yin, “Human activity recognition using wearable sensors by deep convolutional neural networks,” in *Proceedings of the 23rd ACM international conference on Multimedia*, ser. MM ’15, ACM, Oct. 2015. DOI: 10.1145/2733373.2806333.

[89] B. S. M. Almaslukh, A. M. Artoli, and J. Al-Muhtadi, “An effective deep autoencoder approach for online smartphone-based human activity recognitio,” *Interna-tional Journal of Computer Science and Network Security*, 2017.

[90] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, “Skeleton-based action recognition using spatio-temporal lstm network with trust gates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 3007–3021, Dec. 2018, ISSN: 1939-3539. DOI: 10.1109/tpami.2017.2771306.

[91] P. Wei, Y. Zhao, N. Zheng, and S.-C. Zhu, "Modeling 4d human-object interactions for joint event segmentation, recognition, and object localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1165–1179, Jun. 2017, ISSN: 2160-9292. DOI: 10.1109/tpami.2016.2574712.

[92] D. Wu *et al.*, "Deep dynamic neural networks for multimodal gesture segmentation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1583–1597, Aug. 2016, ISSN: 1939-3539. DOI: 10.1109/tpami.2016.2537340.

[93] B. Pan, J. Sun, W. Lin, L. Wang, and W. Lin, "Cross-stream selective networks for action recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jun. 2019. DOI: 10.1109/cvprw.2019.00059.

[94] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, Mar. 2013, ISSN: 1573-1405. DOI: 10.1007/s11263-012-0594-8.

[95] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.590.

[96] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7299059.

[97] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.787.

[98] M. Ebrahim, M. Al-Ayyoub, and M. A. Alsmirat, "Will transfer learning enhance imagenet classification accuracy using imagenet-pretrained models?," Irbid, Jordan: IEEE, 2019, pp. 211–216, ISBN: 978-1-7281-0046-3. DOI: 10.1109/IACS.2019.8809114.

[99] R. Goyal *et al.*, "The "something something" video database for learning and evaluating visual common sense," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.622.

[100] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *European Conference on Computer Vision (ECCV)*, Springer International Publishing, Sep. 2018, pp. 318–335. DOI: 10.1007/978-3-030-01267-0_19.

[101] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00675.

[102] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00630.

[103] J. Materzynska, G. Berger, I. Bax, and R. Memisevic, "The jester dataset: A large-scale video dataset of human gestures," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, Oct. 2019. DOI: 10.1109/iccvw.2019.00349.

[104] G. Paoletti, J. Cavazza, C. Beyan, and A. Del Bue, "Subspace clustering for action recognition with covariance representations and temporal pruning," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, Jan. 2021. DOI: 10.1109/icpr48806.2021.9412060.

[105] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multiscale hybrid linear models for lossy image representation," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3655–3671, Dec. 2006, ISSN: 1057-7149. DOI: 10.1109/tip.2006.882016.

[106] A. Y. Yang, J. Wright, Y. Ma, and S. S. Sastry, "Unsupervised segmentation of natural images via lossy data compressio," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[107] C. W. Gear, *International Journal of Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998, ISSN: 0920-5691. DOI: 10.1023/a:1008026310903.

[108] J. P. Costeira and T. Kanade, *International Journal of Computer Vision*, vol. 29, no. 3, pp. 159–179, 1998, ISSN: 0920-5691. DOI: 10.1023/a:1008000628999.

[109] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, Mar. 2011, ISSN: 1053-5888. DOI: 10.1109/msp.2010.939739.

[110] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, Jan. 2013, ISSN: 2160-9292. DOI: 10.1109/tpami.2012.88.

[111] C. Lu, J. Feng, Z. Lin, T. Mei, and S. Yan, "Subspace clustering by block diagonal representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 487–501, Feb. 2019, ISSN: 1939-3539. DOI: 10.1109/tpami.2018.2794348.

[112] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013, ISSN: 2160-9292. DOI: 10.1109/tpami.2013.57.

[113] H. Hu, Z. Lin, J. Feng, and J. Zhou, "Smooth representation clustering," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2014. DOI: 10.1109/cvpr.2014.484.

[114] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Computer Vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 347–360, ISBN: 9783642337864. DOI: 10.1007/978-3-642-33786-4_26.

[115] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 589–600, ISBN: 9783540338352. DOI: 10.1007/11744047_45.

[116] C. You, C.-G. Li, D. P. Robinson, and R. Vidal, "Oracle based active set algorithm for scalable elastic net subspace clustering," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.426.

[117] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. D. Reid, "Deep subspace clustering networks," *Advances in Neural Information Processing Systems*, pp. 24–33, 2017.

[118] C. You, D. P. Robinson, and R. Vidal, "Scalable sparse subspace clustering by orthogonal matching pursuit," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.425.

[119] H. W. Kuhn, "The hungarian method for the assignment problem," *NavalResearch Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[120] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proceedings of the 21th International Conference on Artificial Neural Networks (ICANN)*, vol. 1, 2011, pp. 44–51.

[121] Z. Xu, X. Shen, Y. Wong, and M. S. Kankanhalli, "Unsupervised motion representation learning with capsule autoencoders," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021, pp. 3205–3217.

[122] A. Kosiorek, S. Sabour, Y. Teh, and G. Hinton, "Stacked capsule autoencoders," in *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2019, pp. 15 512–15 522.

[123] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 3859–3869.

[124] G. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with em routing," in *International Conference on Learning Representations (ICLR)*, 2018.

[125] J. Wang, X. Nie, Y. Xia, Y. Wu, and S.-C. Zhu, "Cross-view action modeling, learning, and recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2014. DOI: 10.1109/cvpr.2014.339.

[126] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.115.

[127] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2684–2701, Oct. 2020, ISSN: 1939-3539. DOI: 10.1109/tpami.2019.2916873.

[128] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.751.

[129] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Unsupervised learning of view-invariant action representations," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Montreal, Canada, 2018, pp. 1262–1272.

[130] Q. Nie, Z. Liu, and Y. Liu, "Unsupervised 3d human pose representation with viewpoint and pose disentanglement," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 102–118.

[131] N. Zheng, J. Wen, R. Liu, L. Long, J. Dai, and Z. Gong, "Unsupervised representation learning with long-term dynamics for skeleton based action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, ISSN: 2159-5399. DOI: 10.1609/aaai.v32i1.11853.

[132] L. Lin, S. Song, W. Yang, and J. Liu, "Ms2l: Multi-task self-supervised learning for skeleton based action recognition," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 2490–2498.

[133] H. Rao, S. Xu, X. Hu, J. Cheng, and B. Hu, "Augmented skeleton based contrastive action learning with momentum lstm for unsupervised action recognition," *Information Sciences*, vol. 569, pp. 90–109, Aug. 2021, ISSN: 0020-0255. DOI: 10.1016/j.ins.2021.04.023.

[134] K. Su, X. Liu, and E. Shlizerman, "Predict & cluster: Unsupervised skeleton based action recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020. DOI: 10.1109/cvpr42600.2020.00965.

[135] K. Cheng, Y. Zhang, C. Cao, L. Shi, J. Cheng, and H. Lu, "Decoupling GCN with dropgraph module for skeleton-based action recognition," in *European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 536–553.

[136] J. Cai, N. Jiang, X. Han, K. Jia, and J. Lu, "Jolo-gcn: Mining joint-centered light-weight information for skeleton-based action recognition," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Jan. 2021. DOI: 10.1109/wacv48630.2021.00278.

[137] P. S. Tan, K. M. Lim, and C. P. Lee, "Human action recognition with sparse autoencoder and histogram of oriented gradients," in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, IEEE, Sep. 2020, pp. 1–5. DOI: 10.1109/iicaiet49801.2020.9257863.

[138] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014, ISSN: 2160-9292. DOI: 10.1109/tpami.2014.2300479.

[139] M.-P. Dubuisson and A. K. Jain, "A modified hausdorff distance for object matching," in *Proceedings of 12th International Conference on Pattern Recognition*, ser. ICPR-94, vol. 1, IEEE Comput. Soc. Press, 2002, pp. 566–568. DOI: 10.1109/icpr.1994.576361.

[140] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE, 2005, 1395–1402 Vol. 2. DOI: 10.1109/iccv.2005.28.

[141] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Human activity detection from rgbd images," in *Proceedings of the 16th AAAI Conference on Plan, Activity, and Intent Recognition*, AAAI Press, 2011, pp. 47–55.

[142] J. Dong, S. Sun, Z. Liu, S. Chen, B. Liu, and X. Wang, "Hierarchical contrast for unsupervised skeleton-based action representation learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, pp. 525–533, Jun. 2023, ISSN: 2159-5399. DOI: 10.1609/aaai.v37i1.25127.

[143] L. Li, M. Wang, B. Ni, H. Wang, J. Yang, and W. Zhang, "3d human action representation learning via cross-view consistency pursuit," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021. DOI: 10.1109/cvpr46437.2021.00471.

[144] T. Guo, H. Liu, Z. Chen, M. Liu, T. Wang, and R. Ding, "Contrastive learning from extremely augmented skeleton sequences for self-supervised action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2021, pp. 762–770.

[145] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020. DOI: 10.1109/cvpr42600.2020.00975.

[146] F. M. Thoker, H. Doughty, and C. G. M. Snoek, "Skeleton-contrastive 3d action representation learning," in *Proceedings of the 29th ACM International Conference on Multimedia*, ser. MM '21, ACM, Oct. 2021. DOI: 10.1145/3474085.3475307.

[147] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298714.

[148] Y. Chen *et al.*, "Hierarchically self-supervised transformer for human skeleton representation learning," in *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 185–202, ISBN: 9783031198090. DOI: 10.1007/978-3-031-19809-0_11.

[149] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 1697–1607.

[150] J. Liu, S. Song, C. Liu, Y. Li, and Y. Hu, "A benchmark dataset and comparison study for multi-modal human action analytics," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 16, no. 2, pp. 1–24, May 2020, ISSN: 1551-6865. DOI: 10.1145/3365212.

[151] J. N. Kundu, M. Gor, P. K. Uppala, and V. B. Radhakrishnan, "Unsupervised feature learning of human actions as trajectories in pose embedding manifold," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Jan. 2019. DOI: 10.1109/wacv.2019.00160.

[152] Y.-B. Cheng, X. Chen, J. Chen, P. Wei, D. Zhang, and L. Lin, "Hierarchical transformer: Unsupervised representation learning for skeleton-based human action recognition," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, Jul. 2021. DOI: 10.1109/icme51207.2021.9428459.

[153] S. Yang, J. Liu, S. Lu, M. H. Er, and A. C. Kot, "Skeleton cloud colorization for unsupervised 3d action representation learning," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01317.

[154] B. Kim, H. Chang, J. Kim, and J. Choi, "Global-local motion transformer for unsupervised skeleton-based action learning," in *European Conference on Computer Vision (ECCV)*, 2022, pp. 209–225.

[155] T. Guo, H. Liu, Z. Chen, M. Liu, T. Wang, and R. Ding, "Contrastive learning from extremely augmented skeleton sequences for self-supervised action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, pp. 762–770, Jun. 2022, ISSN: 2159-5399. DOI: 10.1609/aaai.v36i1.19957.

[156] S. Xu, H. Rao, X. Hu, J. Cheng, and B. Hu, "Prototypical contrast and reverse prediction: Unsupervised skeleton based action recognition," *IEEE Transactions on Multimedia*, vol. 25, pp. 624–634, 2023, ISSN: 1941-0077. DOI: 10.1109/tmm.2021.3129616.

[157] R. Gao, X. Liu, J. Yang, and H. Yue, "Cdclr: Clip-driven contrastive learning for skeleton-based action recognition," in *2022 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, IEEE, Dec. 2022, pp. 1–5. DOI: 10.1109/vcip56404.2022.10008837.

[158] X. Liu, H. Shi, H. Chen, Z. Yu, X. Li, and G. Zhao, "Imigue: An identity-free video dataset for micro-gesture understanding and emotion analysis," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 10 626–10 637. DOI: 10.1109/cvpr46437.2021.01049.

[159] B. M. Martinez, D. Modolo, Y. Xiong, and J. Tighe, "Action recognition with spatial-temporal discriminative filter banks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00558.

[160] J. Weng *et al.*, "Temporal distinct representation learning for action recognition," in *European Conference on Computer Vision (ECCV)*, Jul. 2020, pp. 5481–5490.

[161] F. Baradel, C. Wolf, J. Mille, and G. W. Taylor, "Glimpse clouds: Human activity recognition from unstructured feature points," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00056.

[162] Z. Luo, J.-T. Hsieh, L. Jiang, J. C. Niebles, and L. Fei-Fei, "Graph distillation for action detection with privileged modalities," in *European Conference on Computer Vision (ECCV)*, Springer International Publishing, 2018, pp. 174–192.

[163] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019. DOI: 10.1109/cvpr.2019.01230.

[164] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, "Disentangling and unifying graph convolutions for skeleton-based action recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020. DOI: 10.1109/cvpr42600.2020.00022.

[165] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of the 32nd AAAI conference on Artificial Intelligence*, 2018, pp. 7444–7452.

[166] C. Liu, Y. Hu, Y. Li, S. Song, and J. Liu, "Pku-mmd: A large scale benchmark for skeleton-based human action understanding," in *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities*, ser. MM '17, ACM, Oct. 2017. DOI: 10.1145/3132734.3132739.

[167] S. Das *et al.*, "Toyota smarthome: Real-world activities of daily living," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00092.

[168] S. Das, S. Sharma, R. Dai, F. Brémond, and M. Thonnat, "VPN: Learning video-pose embedding for activities of daily living," in *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020, pp. 72–90.

[169] B. X. Yu, Y. Liu, and K. C. Chan, "Multimodal fusion via teacher-student network for indoor action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, pp. 3199–3207, May 2021, ISSN: 2159-5399. DOI: 10.1609/aaai.v35i4.16430.

[170] M. Liu and J. Yuan, "Recognizing human actions as the evolution of pose estimation maps," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00127.

[171] S. Song, J. Liu, Y. Li, and Z. Guo, "Modality compensation network: Cross-modal adaptation for action recognition," *IEEE Transactions on Image Processing*, vol. 29, pp. 3957–3969, 2020, ISSN: 1941-0042. DOI: 10.1109/tip.2020.2967577.

[172] S. Das, R. Dai, D. Yang, and F. Bremond, "Vpn++: Rethinking video-pose embeddings for understanding activities of daily living," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9703–9717, Dec. 2022, ISSN: 1939-3539. DOI: 10.1109/tpami.2021.3127885.

[173] Y. Chen, Z. Zhang, C. Yuan, B. Li, Y. Deng, and W. Hu, "Channel-wise topology refinement graph convolution for skeleton-based action recognition," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01311.

[174] Y. Xue, X. Guo, and X. Cao, "Motion saliency detection using low-rank and sparse decomposition," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Mar. 2012. DOI: 10.1109/icassp.2012.6288171.

[175] K. Fukuchi, K. Miyazato, A. Kimura, S. Takagi, and J. Yamato, "Saliency-based video segmentation with graph cuts and sequentially updated priors," in *2009 IEEE International Conference on Multimedia and Expo*, IEEE, Jun. 2009. DOI: 10.1109/icme.2009.5202577.

[176] Y.-F. Ma and H.-J. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *Proceedings of the eleventh ACM international conference on Multimedia*, ser. MM03, ACM, Nov. 2003. DOI: 10.1145/957013.957094.

[177] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The jpeg2000 still image coding system: An overview," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, 2000, ISSN: 0098-3063. DOI: 10.1109/30.920468.

[178] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998, ISSN: 0162-8828. DOI: 10.1109/34.730558.

[179] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, no. 10–12, pp. 1489–1506, Jun. 2000, ISSN: 0042-6989. DOI: 10.1016/s0042-6989(99)00163-7.

[180] T. Kadir, "Scale saliency: A novel approach to salient feature and scale selection," in *International Conference on Visual Information Engineering (VIE 2003). Ideas, Applications, Experience*, vol. 2003, IEE, 2003, pp. 25–28. DOI: 10.1049/cp:20030478.

[181] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, Sep. 2004, ISSN: 0262-8856. DOI: 10.1016/j.imavis.2004.02.006.

[182] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, ser. CVPR-99, IEEE Comput. Soc, 1999, pp. 246–252. DOI: 10.1109/cvpr.1999.784637.

[183] M. O. Tezcan, P. Ishwar, and J. Konrad, "Bsuv-net 2.0: Spatio-temporal data augmentations for video-agnostic supervised background subtraction," *IEEE Access*, vol. 9, pp. 53 849–53 860, 2021, ISSN: 2169-3536. DOI: 10.1109/access.2021.3071163.

[184] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, Jul. 2002, ISSN: 0018-9219. DOI: 10.1109/jproc.2002.801448.

[185] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "Subsense: A universal change detection method with local adaptive sensitivity," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 359–373, Jan. 2015, ISSN: 1941-0042. DOI: 10.1109/tip.2014.2378053.

[186] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "A self-adjusting approach to change detection based on background word consensus," in *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, Jan. 2015. DOI: 10.1109/wacv.2015.137.

[187] Ş. Işık, K. Özkan, S. Günal, and Ö. N. Gerek, "Swcd: A sliding window and self-regulated learning-based background updating method for change detection in videos," *Journal of Electronic Imaging*, vol. 27, no. 02, p. 1, Mar. 2018, ISSN: 1017-9909. DOI: 10.1117/1.jei.27.2.023002.

[188] L. A. Lim and H. Yalim Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognition Letters*, vol. 112, pp. 256–262, 2018, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865518303702.

[189] M. O. Tezcan, P. Ishwar, and J. Konrad, "Bsuv-net: A fully-convolutional neural network for background subtraction of unseen videos," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Mar. 2020. DOI: 10.1109/wacv45572.2020.9093464.

[190] T. Bouwmans, S. Javed, M. Sultana, and S. K. Jung, “Deep neural network concepts for background subtraction:a systematic review and comparative evaluation,” *Neural Networks*, vol. 117, pp. 8–66, Sep. 2019, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.04.024.

[191] X. Cui, Q. Liu, and D. Metaxas, “Temporal spectral residual: Fast motion saliency detection,” in *Proceedings of the 17th ACM international conference on Multimedia*, ser. MM09, ACM, Oct. 2009. DOI: 10.1145/1631272.1631370.

[192] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical Image Computing and Computer-assisted Intervention*, Springer, 2015, pp. 234–241.

[193] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, Apr. 2018, ISSN: 2160-9292. DOI: 10.1109/tpami.2017.2699184.

[194] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.544.

[195] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298664.

[196] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, “A novel video dataset for change detection benchmarking,” *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4663–4679, Nov. 2014, ISSN: 1941-0042. DOI: 10.1109/tip.2014.2346013.

[197] S. Bianco, G. Ciocca, and R. Schettini, “How far can you get by combining change detection algorithms?” In *International Conference on Image Analysis and Processing*, Springer, 2017, pp. 96–107.

[198] S.-h. Lee, G.-c. Lee, J. Yoo, and S. Kwon, “Wisenetmd: Motion detection using dynamic background region analysis,” *Symmetry*, vol. 11, no. 5, p. 621, May 2019, ISSN: 2073-8994. DOI: 10.3390/sym11050621.

[199] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979, ISSN: 2168-2909. DOI: 10.1109/tsmc.1979.4310076.

[200] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, Jun. 2012. DOI: 10.1109/cvprw.2012.6239233.

[201] H. Yu, L. Qin, Q. Huang, H. Yao, and L. Li, "Online multi-target tracking via depth range segmentation," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, Nov. 2017. DOI: 10.1109/globalsip.2017.8309048.

[202] D. Khanh Hoa, P. Cuong, and T. Nguyen, "An effective ground plane extraction using depth map estimation from a kinect device," *Journal of Science and Technology*, pp. 19–25, Jan. 2017.

[203] N. Kanopoulos, N. Vasanthavada, and R. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, Apr. 1988, ISSN: 0018-9200. DOI: 10.1109/4.996.

[204] R. E. Kalman, "A new approach to linear filtering and predictionproblems," *Transaction of the ASME Journal of Basic Engineering*, pp. 35–45, Mar. 1960.

[205] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, May 2003, ISSN: 0162-8828. DOI: 10.1109/tpami.2003.1195991.

[206] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, Jun. 2012. DOI: 10.1109/cvprw.2012.6238919.

[207] Y. Yi-fei, H. Qing-chun, D. Ming-ming, Y. Hong-ri, and L. Jin-yang, "Design and implementation of fall detection system based on video," in *2020 International Conference on Computer Engineering and Application (ICCEA)*, vol. 40, IEEE, Mar. 2020, pp. 893–899. DOI: 10.1109/iccea50009.2020.00196.

[208] C. M. Ranieri, S. MacLeod, M. Dragone, P. A. Vargas, and R. A. F. Romero, "Activity recognition for ambient assisted living with videos, inertial units and ambient sensors," *Sensors*, vol. 21, no. 3, p. 768, Jan. 2021, ISSN: 1424-8220. DOI: 10.3390/s21030768.

[209] C. Chen, R. Jafari, and N. Kehtarnavaz, "Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," in *2015 IEEE International Conference on Image Processing (ICIP)*, IEEE, Sep. 2015. DOI: 10.1109/icip.2015.7350781.

Appendix A Publications

This appendix contains the papers that were published during the Ph.D. programme.

Motion Saliency Detection using Depth Information for Human Action Recognition Applications

Alexander Gutev[1], Carl James Debono[1]
[1] University of Malta
Msida, Malta
alexander.gutev.15@um.edu.mt

***Abstract*—Video processing tasks require the analysis of an enormous amount of data. However, in most practical applications this analysis is not required on the whole frame or video but is limited to the regions where some action is taking place. If these regions can be identified with low effort, the total computation time required to process and analyze such videos can be drastically reduced. In this work a new motion saliency detection algorithm intended to be incorporated in a human action recognition pipeline is presented. The approach uses depth data to improve the performance of the difference of frames method which is computationally simple and efficient. Results show that the algorithm achieves a performance which is comparable to the state of the art while requiring a much lower computational time.**

***Keywords*—motion saliency; RGBD video; human action recognition**

I. Introduction

Human action recognition, which is the task of classifying human actions in video, is an emerging field with applications ranging from surveillance and security to assisted living systems [1], [2]. A human action classifier has to process a large set of data resulting in long training and inference times. However, most of this data consists of background clutter that is irrelevant to the classification task [3]. The relevant regions are limited to those which capture the action being carried out. If these regions can be identified with low effort, the total computation time can be significantly reduced [3]. A possible solution to identify these regions is to apply motion saliency techniques.

Motion saliency detection is a class of methods for finding salient regions in video based on the presence of motion, which differs from traditional saliency detection methods that are focused on finding regions within an image that capture the attention of the human visual system [4].

This paper presents a novel motion saliency detection algorithm, for RGBD video. It is designed to be computationally simple and accurate in determining the regions of motion in videos containing human activity. The algorithm is based on the differences of frames method and depth segmentation using adaptive thresholding. Experimental results show that the solution achieved a performance that is comparable to the state of the art in much less time, making it a good candidate for use in a human action recognition pipeline.

The rest of the paper is organized as follows. The second section provides a summary of related work in the area and the computational issue of these solutions. The third section details our work with the fourth section providing an evaluation of its performance. The final section concludes this paper with a summary of our work and its achievements.

II. Related Work

Background subtraction techniques, such as Gaussian Mixture Model (GMM) background subtraction [5], [6], solve the same problem as motion saliency detection [4]. However, the state of the art background subtraction methods are based on Convolutional Neural Networks (CNNs), such as BSUV-NET 2.0, as they achieve a superior performance to GMMs [7]. BSUV-NET 2.0, which is based on a UNET-type CNN, is reported to have achieved good results on unseen videos [7], that is videos which are different from those that the network is trained on. However, this performance comes at the cost of long training and inference times. Furthermore, this method requires training a CNN classifier in addition to the main human action classifier. This adds additional training data requirements which renders this method unsuitable for reducing the training set size requirements and the total computational time required for training a human action recognition classifier.

The motion saliency detection method proposed by Xue, *et al*. in [4] reports a superior accuracy compared to GMMs and the difference of frames, and does not require training. However, the Robust Principle Component Analysis (RPCA) operation, which forms the basis of the method, is an expensive operation in terms of computational speed. Furthermore, the method requires analyzing the entire time slice, which makes it unsuitable for a real-time human action recognition pipeline such as an assistive living system.

Our work solves these issues by proposing a computationally efficient, yet accurate, approach that can be incorporated in a human action recognition pipeline.

III. The Motion Saliency Detection Algorithm

Our approach uses the difference of the color frame images to identify the regions in the video that contain motion and depth map segmentation to limit the output to only those regions which contain relevant motion of a foreground object. The remainder of this section details each step of the approach, a block diagram of which is shown in Fig. 1.

979-8-3503-2512-6/23/$31.00 ©2023 IEEE

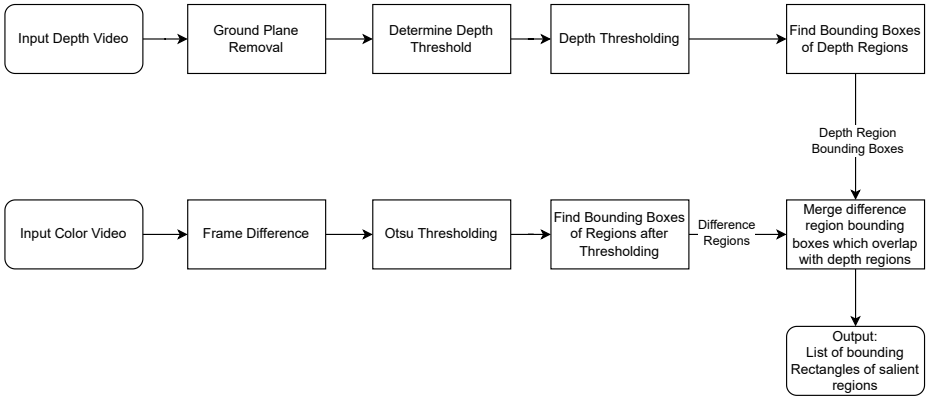


Figure 1. Pipeline of our approach

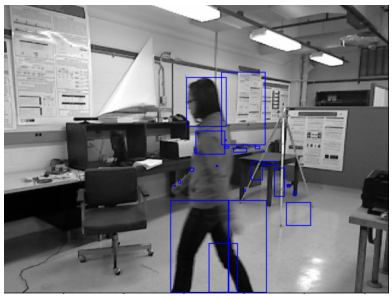


Figure 2. The impact of noise and clutter result in multiple regions, including regions that do not contain motion.

A. *Frame difference*

The proposed solution is based on the difference of the color frame images. In regions containing motion the difference between the frames is relatively high, whereas in regions without motion this difference is low. To detect relevant motion, especially in video with high frame rates, the difference across multiple frames, e.g. across every five frames, is taken such that there is a larger change in the motion regions. This simplifies the detection of motion areas due to their larger differences compared to the very small differences of stationary and near-stationary objects.

The output of this step is a motion saliency map where bright regions indicate areas containing motion. Otsu thresholding [8], a technique for determining the threshold used to separate the foreground from the background in gray-scale images, is used to convert the saliency map to a binary mask that identifies the regions which contain motion.

B. *Using depth to limit saliency to interesting regions*

The frame difference method, by itself, is heavily affected by noise and background clutter. An example is shown in Fig. 2 on a video from the UTKinect dataset [9]. This results in a performance which is inferior to the state of the art [4].

The depth map in RGBD video provides the distance between the camera and the objects in the scene. This additional information can be used to limit the detected regions to only those which represent actual motion in relevant regions. Depth maps are additionally invariant to illumination changes, thus they are effective in eliminating false motion regions caused by sudden illumination changes, such as shadows appearing due to the sun coming out from behind the clouds.

In a human action recognition system the relevant regions are those which contain the subjects carrying out the actions. A subject occupies a specific range of values along the depth axis. The regions containing the subjects can thus be found by thresholding the depth map.

1) Determining the depth threshold: The depth threshold is scene dependent and needs to be automatically determined. Our work uses the depth-range based segmentation component of the multiple object tracking algorithm proposed by Yu, *et al.* [10]. The depth map is segmented by determining the range of depth values of each object. The number of local maxima in the histogram of depth values gives the number of objects in the scene and the range of depth values of each object is determined using:

$$V = \{i \mid b_i \geq b_\mu, \forall \mu \in \{i-1, i+1\}\} \tag{1}$$

where V denotes the set of discontinuity points and b_i denotes the number of pixels in bin i of the depth histogram.

The depth values are divided into ranges, where each range is the depth range of an object, using:

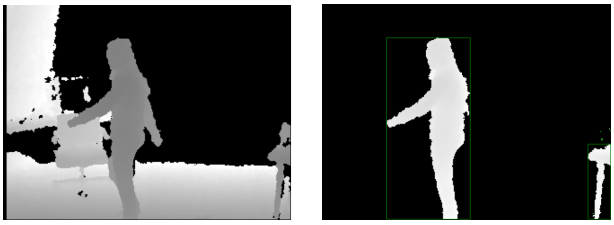
$$[l_i, h_i) = \left[\frac{v_{i-1} + v_i}{2}, \frac{v_i + v_{i+1}}{2}\right) \tag{2}$$

where l_i and h_i are the lower and upper bounds of the depth range, respectively, and v_i is the i^{th} discontinuity point.

Most of the regions identified by the frame difference method represent regions of motion. In our approach the depth range of the moving object is found by determining the range which contains the median depth d_m of the pixels in these regions. Thus, for the depth range $[l_k, h_k)$:

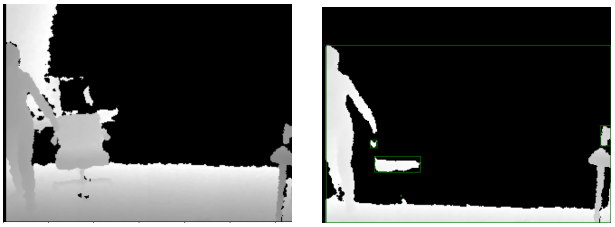
$$l_k \leq d_m < h_k \tag{3}$$

The foreground object is separated from the background object using:

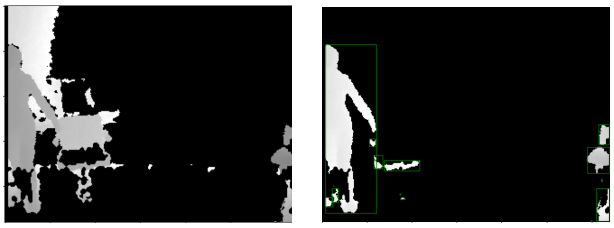


(a) Depth map (b) Thresholded depth map

Figure 3. Depth thresholding



(a) Original depth map (b) Thresholded depth map picking up ground plane



(c) Depth map with ground plane removed (d) Final thresholded depth map

Figure 4. Effects of ground plane on depth thresholding

$$t_{x,y} = \begin{cases} 1, & \text{if } l_k \leq d_{x,y} < h_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $t_{x,y}$ is the pixel at (x, y) of the resulting thresholded image and $d_{x,y}$ is the pixel at (x, y) of the depth map.

Only the bounding rectangles, found by the frame difference of the color images that intersect with a depth region, after thresholding using (4), are retained and the rest are discarded.

The output of this step is a bounding box per depth region, which is the union box of all the salient regions that intersect with the region. The result of depth thresholding in a single frame of a video from the UTKinect dataset is shown in Fig. 3.

The ground plane is often segmented as part of the foreground object, resulting in a large bounding rectangle covering the entire ground rather than just the object of interest. An example of this is shown in Fig. 4b. This is fixed by removing the ground plane before thresholding.

2) Removing the ground plane: Khanh Hoa, *et al*. proposed an approach for detecting the ground plane in depth images in [11]. The method computes the gradients of the depth map in the width, x, and height, y, directions of the frame. In [11], the gradients are computed based on the focal length and camera parameters of the video capturing system. However, our work uses the Sobel operator [12] to compute an approximation since these parameters are not always readily available.

The Sobel operator [12], defined by (5) and (6), is applied directly on the depth map to compute an approximation of the x, and y gradient. The pixels comprising the ground plane have a gradient of 0 along the x-axis and a non-zero gradient along the y-axis.

$$\mathbf{S_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\mathbf{S_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{6}$$

This step results in a large number of non-ground pixels being classified as ground, and a large number of holes in the ground plane due to noise. The work [11] describes a block grouping procedure to filter out the non-ground pixels classified as ground, and fill in the holes in the ground plane. It was found through experimentation that a block size of 3×3 and ground to non-ground ratio of 0.4 achieves the best results. Our approach adds a morphological closing and opening step to reduce the number of the holes in the ground plane.

All remaining holes in the ground plane that are smaller than a threshold are marked as ground. A threshold value of 100 pixels was used, however a larger value will be necessary for larger video resolutions.

The identified ground pixels are set to zero prior to the segmentation step, as shown in Fig. 4c. The results after ground plane removal, shown in Fig. 4d, are significantly improved, with the ground and other connected objects no longer being marked as salient regions.

IV. Evaluation and Results

The solution developed in this paper was evaluated using the UTKinect [9] dataset. This dataset was chosen since it is widely used in the evaluation of depth-based human action recognition approaches. The visual results on three consecutive frames of test sequence s01_e01 are shown in Fig. 5.

The performance of the algorithm is compared with the widely used Gaussian Mixture Model (GMM) background subtraction approach [5], [6], and BSUV-NET 2.0 [7], which is a state of the art background subtraction method for video segmentation and for detecting motion in video.

The metrics for evaluating motion saliency approaches that are described in [13], namely the recall (Re), precision (Pr), and F-measure, are used in the evaluation. The recall, precision and F-measure are computed using (7), (8) and (9) respectively, where TP is the number of true positives, FN is the number of false negatives and FP is the number of false positives. A true positive is considered when the intersection of the union (IOU) between the ground truth labels and salient regions found using this approach is greater than 0.5.

Figure 5. Visual motion saliency detection results on three frames of the s01_e01 sequence

$$\text{Re} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

$$\text{Pr} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

$$\text{F-measure} = 2\frac{\text{Pr} \cdot \text{Re}}{\text{Pr} + \text{Re}} \tag{9}$$

TABLE I. Performance results

Method	**Re**	**Pr**	**F-measure**
BSUV-NET 2.0	0.84	0.73	0.78
GMM	0.34	0.77	0.47
Our method	0.88	0.78	0.83

TABLE II. Execution time

Method	**Frames / second**	**Time per frame (milliseconds)**
BSUV-NET 2.0	9.79	102
GMM	12.89	78
Our method	25.78	39

The results presented in Table I show that the proposed method has both a high recall (Re) and a high precision (Pr). This means that most of the regions found by the algorithm are actual salient regions and most of the salient regions are found. Our method achieves a superior performance to both GMM and BSUV-NET 2.0 in all three metrics.

The experiments were carried out on an Intel Core i7-11800H with 8 cores, with an Nvidia GeForce RTX 3060 graphics card and 32 GB of RAM running Ubuntu 20.04. The average execution times of the three algorithms on videos having a resolution of 640×480 are shown in Table II. Our method achieves an average execution time of 25.78 frames per second which is much faster than the 9.79 and 12.89 frames per second achieved by the BSUV NET 2.0 and GMM methods, respectively.

V. Conclusion

In this paper a new motion saliency detection algorithm was presented. The algorithm is based on the computationally simple and efficient difference of frames method, with depth data being incorporated to improve the performance.

The proposed method achieves a superior performance to both the BSUV-NET 2.0 and GMM approaches while requiring a much lower computation time. This makes the method suitable for adoption in a human action recognition pipeline where only a small portion of the processing time can be allocated to motion saliency detection.

References

[1] C. J. Debono, M. Sacco, and J. Ellul, "Monitoring indoor living spaces using depth information," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, pp. 1–5, 2020.

[2] Z. Ahmad and N. Khan, "Human action recognition using deep multilevel multimodal (M^2) fusion of depth and inertial sensors," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1445–1455, 2020.

[3] Y. Kong, Y. Wang, and A. Li, "Spatiotemporal saliency representation learning for video action recognition," *IEEE Transactions on Multimedia*, vol. 24, pp. 1515–1528, 2022.

[4] Y. Xue, X. Guo, and X. Cao, "Motion saliency detection using low-rank and sparse decomposition," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1485–1488, 2012.

[5] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, pp. 28–31, 2004.

[6] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, pp. 773–780, may 2006.

[7] M. Tezcan, P. Ishwar, and J. Konrad, "Bsuv-net 2.0: Spatio-temporal data augmentations for video-agnostic supervised background subtraction," *IEEE Access*, vol. 9, pp. 53849–53860, Apr. 2021.

[8] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[9] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27, 2012.

[10] H. Yu, L. Qin, Q. Huang, H. Yao, and L. Li, "Online multi-target tracking via depth range segmentation," in *Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 691–695, 2017.

[11] D. Khanh Hoa, P. Cuong, and T. Nguyen, "An effective ground plane extraction using depth map estimation from a kinect device," *Journal of Science and Technology*, pp. 19–25, Jan. 2017.

[12] N. Kanopoulos, N. Vasanthavada, and R. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988.

[13] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "changedetection.net: A new change detection benchmark dataset," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012.

Kalman Filter Aided Depth-based Motion Saliency Detection in Human Activity Recognition Applications

Alexander Gutev
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
alexander.gutev.15@um.edu.mt

Carl Debono
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
c.debono@ieee.org

***Abstract*—The computational resources required by video processing tasks in human activity recognition (HAR) applications can be reduced if the input data is limited to only those regions which contain motion. This demands accurate identification of these regions to maximize the computational savings while maintaining similar performances of the video processing tasks. This work explores the use of Kalman filtering in combination with depth-based motion saliency to find the regions of human activity within video content. The Kalman filter tracks salient regions throughout the RGBD (Red, Green, Blue plus Depth) video and facilitates merging of regions pertaining to the same object. Experimental results show that a superior performance is achieved when compared to the state of the art BSUV-Net 2.0 algorithm.**

***Index Terms*—motion saliency, RGBD video, video analysis**

I. Introduction

Most video processing tasks require that an enormous amount of data is processed. One method to reduce computation time and hardware requirements is to process only the data which is meaningful to the task at hand. Motion saliency detection can be used to identify the relevant data in various video processing tasks, such as video compression [1], providing summaries of surveillance video [2] and human action recognition (HAR) [3], [4]. To be effective, it is essential that the motion saliency detection algorithm is computationally simple and efficient, such that the overall computation of the video processing pipeline is reduced [4].

In this paper we combine Kalman filtering with an RGBD (Red, Green, Blue plus Depth) based motion saliency detector [5] to determine the areas of human activity within videos. Motion saliency detection is used to identify the regions in a video which contain significant motion, while the Kalman filter is used to track the identified regions through time. Results, show that the proposed solution is able to identify more of the salient regions than motion saliency detection alone with a minor increase in computational time.

The rest of the paper is organized as follows. The second section gives a summary of related work with a motivation for this work and how it differs from other solutions. The third section gives an overview of Kalman filtering and its application to object tracking. The fourth section details our work with the fifth section providing an evaluation of its performance. The final section concludes this paper.

II. Related Work

Background subtraction techniques, such as Gaussian Mixture Model (GMM) background subtraction [6], [7], are examples of early methods that solve the same problem as motion saliency detection [1]. State of the art background subtraction methods based on Convolutional Neural Networks (CNNs) achieve better performances than GMMs [8]. BSUV-Net 2.0 is a background subtraction approach based on a UNET-type CNN that has achieved promising results [8]. However, this performance comes at the cost of long inference times.

A motion saliency detection approach for RGBD video based on the difference of frames method was proposed in [5]. The regions containing motion in the video are found by the difference of the color frame images. To reduce the effect of noise and background clutter, the depth map is segmented to determine the regions containing actual objects of interest. The regions identified by frame difference which do not intersect with any object of interest are discarded, thus limiting the results to relevant regions only. The approach achieved a superior performance to BSUV-Net 2.0 with a significantly lower per-frame processing time. Furthermore, this approach does not require training a classifier making it better suited for applications where labeled training samples are not abundant, such as in HAR applications [5].

Notwithstanding the positive results, it was found that the motion saliency detection approach in [5] often results in multiple regions covering parts of a moving object being detected instead of a single region covering the entire object. This work proposes an improvement to [5] by adding Kalman filtering to track the moving objects through time and determine which regions belong to the same moving object.

III. Kalman filtering in object tracking

Kalman filtering is a method for estimating the true state of a process based on observable measurements [9], [10].

In object tracking this is used to estimate the position and velocity of the object of interest. A prediction of the new state, based on a model of the process, is computed using (1) and (2). The output of this step is an n-dimensional Gaussian distribution with mean $\hat{\mathbf{x}}_k$, which is the predicted state vector, and covariance $\hat{\mathbf{P}}_k$, which is interpreted as the confidence in the prediction [9], [10]. The prediction is then corrected with a measurement of the state $\mathbf{z}_k$ by (3), (4) and (5). The output of this step is an n-dimensional Gaussian distribution with mean $\mathbf{x}_k$ and covariance $\mathbf{P}_k$. This gives the final process state estimate for time k.

$$\hat{\mathbf{x}}_k = \mathbf{F}\mathbf{x}_{k-1} \tag{1}$$

$$\hat{\mathbf{P}}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{2}$$

$$\mathbf{K} = \hat{\mathbf{P}}_k\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R})^{-1} \tag{3}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k) \tag{4}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\hat{\mathbf{P}}_k \tag{5}$$

$\mathbf{F}$ and $\mathbf{H}$ are the state transition and measurement matrices. $\mathbf{Q}$ and $\mathbf{R}$ are the process and measurement noise covariance matrices, and $\mathbf{I}$ is the identity matrix. The Kalman gain parameter $\mathbf{K}$, computed by (3), controls how much weight is given to the measurement $\mathbf{z}_k$ and how much weight is given to the predicted state $\mathbf{x}_k$ [9], [10].

The process state $\mathbf{x}_k$, at time k, is defined by the object's position (x_k, y_k) and velocity $(\dot{x}_k, \dot{y}_k)$ [9]:

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T \tag{6}$$

A constant velocity model perturbed by Gaussian noise is assumed, where the predicted object position at time k is given by the sum of the object's position and velocity at time $k-1$ [9]. This is defined by the state transition matrix:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The measurement $\mathbf{z}_k$ is obtained using an object detection algorithm such as GMM background subtraction [9]. The measurement matrix $\mathbf{H}$ maps the measurement to the state vector. An identity matrix is used to map the measured position and velocity directly to the state vector.

The noise model parameters $\mathbf{Q}$ and $\mathbf{R}$ control the weight given to the prediction and measurement when computing the final state estimate [9]. Our work achieved the best performance using a discrete constant white noise model with variance 100 for $\mathbf{Q}$, and $\mathbf{R} = 10^{-5} \cdot \mathbf{I}$.

IV. The tracking algorithm

This section describes the proposed Kalman filter addition in detail. A summary of the steps defining our algorithm is shown in Algorithm 1.

Kalman filtering is used to track the regions identified by the motion saliency detection algorithm with time. This information is further used to group multiple regions that cover different parts of a human body into a single region covering the entire person. The pipeline in Fig. 1 shows the relevant steps of the existing motion saliency detection approach [5] in white and the proposed Kalman filtering in orange.

A Kalman filter is initialized to track each object that appears on the scene. The regions, identified by the depth map segmentation step, detailed in [5], provide the number of objects and a measurement of their position. However, this step also identifies regions belonging to stationary objects which are not of interest. To reduce the set of regions to those containing moving objects, only those regions which have an intersection over union greater than 0.5 with the regions identified by motion saliency detection are considered. Each identified object region is assigned to a Kalman filter using the Hungarian algorithm [11], which minimizes the total Euclidean distance between each filter's prediction and the position of the assigned depth region.

Algorithm 1 Tracking Algorithm

$\mathbf{R} \leftarrow$ Set of identified object regions
$\mathbf{F} \leftarrow$ Set of active Kalman filters
for all $f \in \mathbf{F}$ **do**
    $\text{predict}(f)$
end for
$\mathbf{S} \leftarrow \text{assign}(\mathbf{R}, \mathbf{F})$ ▷ Map each region to a filter
for all $(r, f) \in \mathbf{S}$ **do**
    $\text{update}(f, r)$ ▷ Update filter f using measurement r
end for
$\mathbf{U} \leftarrow \{r \mid r \notin \mathbf{S}\}$ ▷ Set of regions not assigned to a filter
$r_l \leftarrow$ largest region in $\mathbf{U}$
$f_l \leftarrow$ **new** $\text{filter}(r_l)$ ▷ Create new filter using r_l
$\mathbf{F} \leftarrow \mathbf{F} \cup \{f_l\}$ ▷ Add new filter to active filters set

A new filter is created with its state initialized to the position of the largest unassigned object region, if there are any. The tracking window of a filter is set to the size of the region with which the filter is initialized and this size is kept constant throughout the video for as long as the filter is active.

A. Object position measurement

The center of mass of the non-zero depth pixels within the depth region, was found to be more robust to sudden changes in movement or object size than other locations. Thus, this is used as the measurement of the object's position.

B. Filter aging

Each filter is given an age, a, equal to the number of time steps for which it has been active. To mitigate the impact of filters initialized due to noise, all filters with $a < 5$ time steps are removed from the active filter set if a depth region is not assigned to them. Furthermore, only filters with $a \geq 2$ time steps are used in the subsequent grouping step.

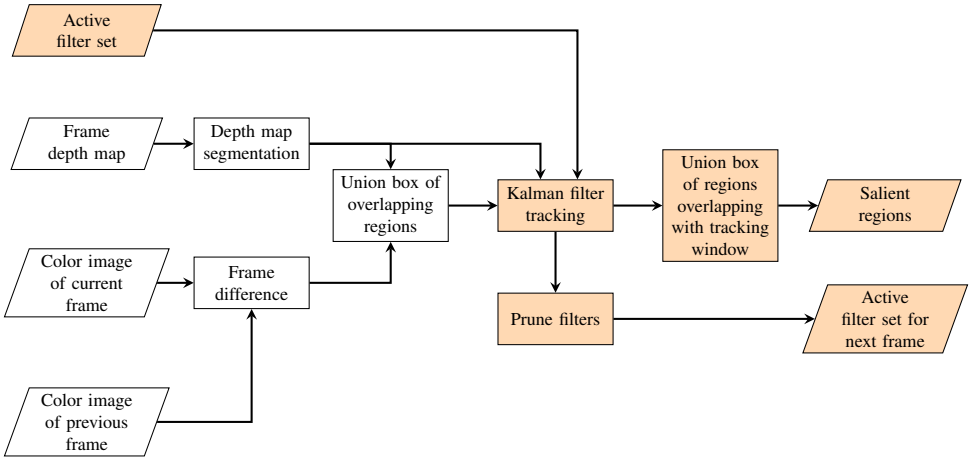
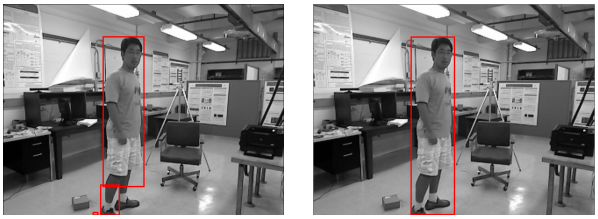


Fig. 1. Pipeline of motion saliency detection algorithm (white) with the Kalman filter addition (orange)



(a) Motion saliency detection alone

(b) Motion saliency detection with Kalman filtering

Fig. 2. Results showing Kalman filter covering more of the person than the motion saliency detection algorithm alone

To avoid tracking objects which are no longer in the scene, filters which overlap with the edge of the frame are removed. A filter is removed from the active set if its tracking window overlaps with the edge of the image and at least 50% of the depth pixels are non-zero in the subregion making up half of the window in the direction of the edge.

C. Grouping salient regions

The regions identified by motion saliency detection which overlap with a filter's tracking window are replaced with their union box. This results in a region covering more of the object instead of multiple regions covering only parts of it. This step is only performed for filters with $a \geq 2$ time steps.

The output is a single bounding rectangle per Kalman filter, as shown in Fig. 2. If there are no Kalman filters that satisfy the constraints above, the output of saliency detection is used as a fallback.

V. Evaluation and Results

The developed solution was evaluated using the UTKinect [12] dataset, since it is widely used in depth-based HAR systems. The visual results on four consecutive frames of test sequences s02_e01, s05_e01, and s07_e01 are shown in Fig. 3, with Figures 3(a), 3(c) and 3(e) showing the results obtained using the motion saliency detection algorithm alone, and Figures 3(b), 3(d) and 3(f) showing the results obtained using the proposed solution.

The performance of the proposed solution is compared with the state of the art BSUV-Net 2.0 [8], and the motion saliency detection approach [5] on which this work builds. The performance is also compared with the widely used Gaussian Mixture Model (GMM) background subtraction approach [6], [7] due to its computational efficiency.

The algorithms are evaluated using the recall (Re) computed using (8), precision (Pr) computed using (9), and F-measure computed using (10) metrics described in [13]. TP is the number of true positives, that is where the intersection over union (IOU) between the ground truth and identified regions is greater than 0.5, and FP and FN are the number of false positives and false negatives, respectively. Additionally, similar to the evaluation technique used in [14], the results with varying IOU thresholds are plotted, to ensure that the performance achieved, relative to the other methods, isn't due to the particular IOU threshold that is chosen.

$$\text{Re} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

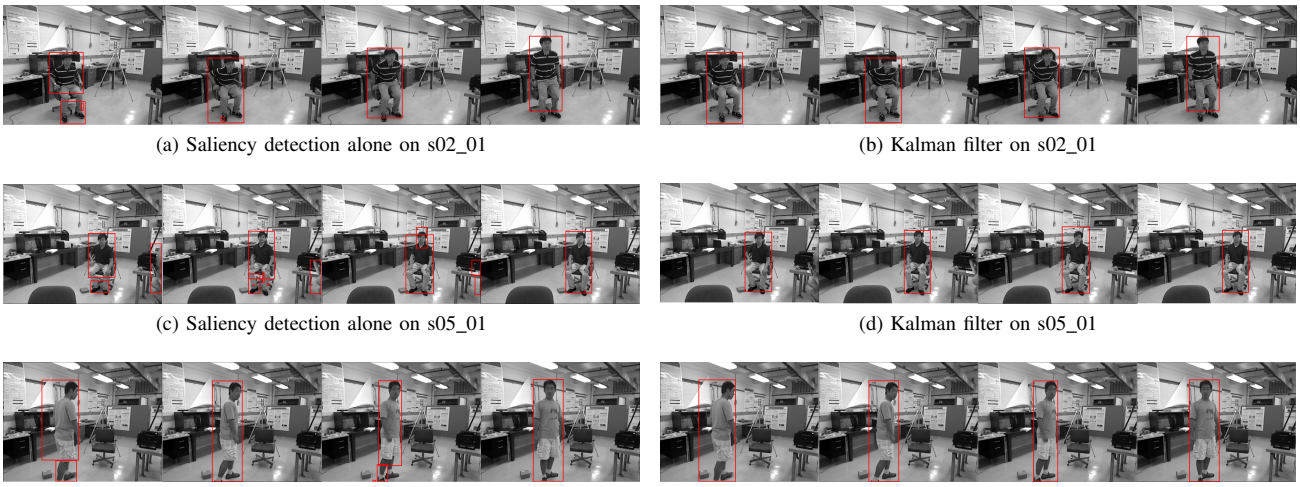
$$\text{Pr} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{F-measure} = 2\frac{\text{Pr} \cdot \text{Re}}{\text{Pr} + \text{Re}} \tag{10}$$

TABLE I
Performance Results

Method	Re	Pr	F-measure	FPS
BSUV-Net 2.0	0.84	0.73	0.78	9.79
GMM	0.34	0.77	0.47	12.89
Saliency alone	0.88	0.78	0.83	25.78
Saliency with Kalman Filter	0.90	0.78	0.83	24.79

The results presented in Table I show that the proposed solution gives the best recall rate of 0.9. This indicates that



(a) Saliency detection alone on s02_01

(b) Kalman filter on s02_01

(c) Saliency detection alone on s05_01

(d) Kalman filter on s05_01

(e) Saliency detection alone on s07_01

(f) Kalman filter on s07_01

Fig. 3. Visual motion saliency detection results

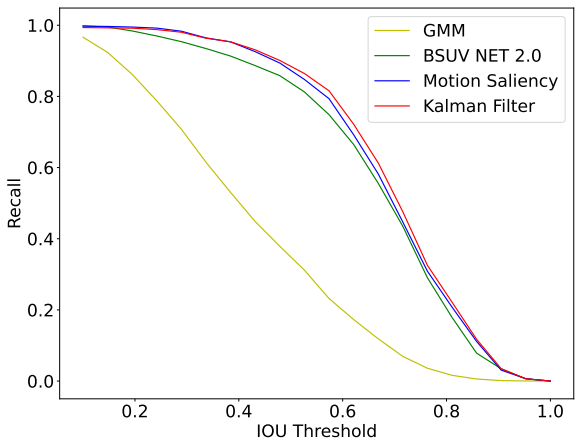


Fig. 4. Graph showing how recall varies with IOU threshold

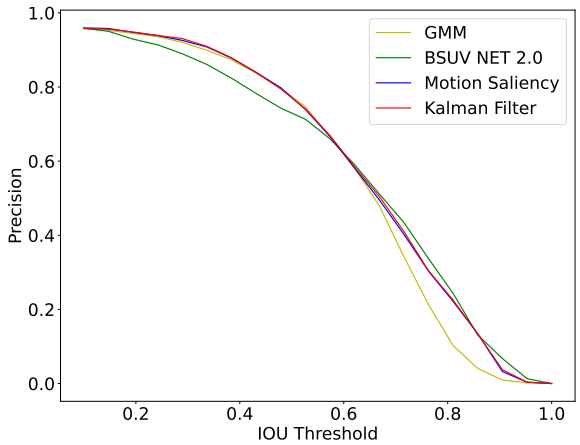


Fig. 5. Graph showing how precision varies with IOU threshold

more of the salient regions are identified with the addition of the Kalman filter than with motion saliency alone. This is further confirmed by Fig. 4, which shows that the recall of our solution remains higher than motion saliency alone with IOU thresholds between 0.5 and 0.8, indicating that the Kalman filter is able to identify a larger area of the salient regions. The precision and resulting F-measure of our solution match the motion saliency alone, which shows that the increase in recall does not come at the cost of increasing the number of non-salient regions erroneously identified as salient. This is confirmed by Fig. 5, which shows that this holds true even for larger IOU thresholds.

These results show that the proposed solution has an advantage over the motion saliency solution alone when used for reducing the size of the data that is processed by a HAR classifier. The increase in recall suggests that the proposed solution is capable of identifying regions containing motion that the motion saliency solution alone is unable to identify. These regions may be relevant to the classification task and their successful identification could potentially result in increased classification accuracy. At the same time, the precision is maintained which shows that this advantage does not come at the cost of increasing the portion of irrelevant data that is included for processing by the HAR classifier.

The experiments were carried out on an Intel Core i7-11800H with 8 cores, with an Nvidia GeForce RTX 3060 graphics card and 32 GB of RAM running Ubuntu 20.04. The average frame rates in frames per second (FPS) of the four algorithms on videos having a resolution of 640×480 are also presented in Table I. The Kalman filter has minimal

impact on the processing time and still achieves a much higher frame rate of 24.79 FPS than the BSUV-Net 2.0 and GMM with 9.79 FPS and 12.89 FPS, respectively.

VI. Conclusion

This paper presented a solution to determine areas of human activity within RGBD video content. Kalman filtering is added to motion saliency detection in order to track salient regions throughout the video. This addition achieved improved results with a minimal impact on processing time. The computational times are still much lower than the BSUV-Net 2.0 and GMM algorithms. This makes the proposed solution suitable for adoption in a HAR pipeline where limited processing time can be allocated for motion saliency detection. The regions of interest are then fed to the HAR algorithm that will need less computational resources than processing of the whole frames.

Potential avenues for future research involve incorporating the Kalman filter component in other saliency detection approaches in order to determine whether a similar increase in performance can be obtained.

References

[1] Y. Xue, X. Guo, and X. Cao, "Motion saliency detection using low-rank and sparse decomposition," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1485–1488.

[2] M. M. Salehin and M. Paul, "Summarizing surveillance video by saliency transition and moving object information," in *Proceedings of the 2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2015, pp. 1–8.

[3] Y. Kong, Y. Wang, and A. Li, "Spatiotemporal saliency representation learning for video action recognition," *IEEE Transactions on Multimedia*, vol. 24, pp. 1515–1528, 2022.

[4] Z. Zheng, G. An, and Q. Ruan, "Motion guided feature-augmented network for action recognition," in *2020 15th IEEE International Conference on Signal Processing (ICSP)*, vol. 1, 2020, pp. 391–394.

[5] A. Gutev and C. J. Debono, "Motion saliency detection using depth information for human action recognition applications," in *2023 International Symposium ELMAR*, 2023, pp. 67–70.

[6] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, 2004, pp. 28–31 Vol.2.

[7] Z. Zivkovic and F. v. d. Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, May 2006.

[8] M. Tezcan, P. Ishwar, and J. Konrad, "Bsuv-net 2.0: Spatio-temporal data augmentations for video-agnostic supervised background subtraction," *IEEE Access*, vol. 9, pp. 53 849–53 860, Apr. 2021.

[9] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, May 2003.

[10] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASME Journal of Basic Engineering*, pp. 35–45, Mar. 1960.

[11] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics*, vol. 2, no. 1, pp. 83–97, Mar. 1955.

[12] L. Xia, C. C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 20–27.

[13] N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "changedetection.net: A new change detection benchmark dataset," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2012.

[14] Y. Zhang, K. Liu, and T. Wang, "End-to-end visual object tracking with motion saliency guidance," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 6566–6571.

Motion Saliency Based Human Action Recognition in RGBD Videos

Alexander Gutev
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
alexander.gutev.15@um.edu.mt

Carl James Debono
Department of Communications and Computer Engineering
University of Malta
Msida, Malta
c.debono@ieee.org

***Abstract*—Human action recognition (HAR) systems need to process large volumes of data posing several challenges including, but not limited to, accurately identifying the actions and classifying them in near real time. Most of the state of the art solutions rely on machine learning techniques where a substantial amount of time is required both for the training and the inference phases. Existing solutions try to reduce the size of the data that is processed by the HAR classifier by constructing a spatio-temporal region of interest (ST-ROI) based on skeleton data, such as the model-based multimodal network (MMNet). However, the performance of these solutions depends on the accuracy of the skeleton data. This paper presents a novel solution to reduce the size of the data that is processed during classification by constructing an ST-ROI using regions identified as motion salient. The results show that the classification accuracy of the proposed solution is comparable to existing solutions such as MMNet but with a training time that is drastically lower.**

***Index Terms*—Human action recognition, motion saliency detection, region of interest, RGBD video**

I. Introduction

Human action recognition (HAR), which is the task of classifying human actions in video, is an emerging field with applications ranging from surveillance and security to assisted living systems [1], [2]. Human action classifiers have to process a large amount of data, most of which consists of background clutter that is not only irrelevant to the action classification task [3], but can also have a negative effect on the classification accuracy [4].

The relevant regions within the video are limited to those areas which capture the action being carried out by the human. Multimodal HAR solutions, such as the Model-based Multimodal Network (MMNet) [4] exploit skeleton data to identify a spatio-temporal region that captures the subject carrying out an action within the video. Whilst this method has a lower computational cost than video-based models, such as the Video-Pose Network (VPN) [5], it requires the estimation of skeleton data using the OpenPose tool [6], which is a computationally expensive step.

Given that the human is performing an action over a number of frames, the areas that are relevant to action recognition can be identified using a motion saliency detection technique [3], [7], [8]. In this paper we exploit this feature and explore the use of motion saliency, focusing on the motion saliency detection approach developed in [9], as a replacement for skeleton data in the identification of an ST-ROI. We further identify the benefits that can be obtained with regards to reduction in time spent on training and inference as well as the impact on classification accuracy.

The paper is organized as follows: the next section details the problem and existing solutions with a motivation for this work. The third section provides an overview of our work, detailing how motion saliency detection is applied in a human action recognition pipeline. The fourth section provides the details of the experiments that were performed in this study with the results achieved. The final section concludes this paper with a summary of our work and its achievements.

979-8-3315-3997-9/24/$31.00

II. Background and Related Work

The large amount of RGB video data that a human action classifier has to process poses a significant challenge to successful human action recognition. State of the art solutions rely on machine learning techniques where this large volume of data negatively impacts the training time and hardware requirements [4]. For example, a cluster of 56 GPUs is required to train a HAR classifier on the NTU RGB+D [10], [11] dataset [4].

An additional challenge to HAR is the large amount of background clutter present in most video scenes [4]. In some scenarios that contain actions that are performed in particular settings, such as water-skiing and surfing, the features in the background can contribute to recognition [4], [12], [13]. This results in a high classification accuracy on the UCF-101 dataset [14] using an inflated 3D convolutional neural network (I3D) and a separable 3D convolutional neural network (S3D) [4], [12], [13]. However, this is not always the case and the same methods do not perform well in other settings like indoor settings. Such an example is the NTU dataset [10], [11] that has a consistent background which does not contribute to action recognition [4], [15], [16].

In [4] a multi-modal framework (MMNet) for human action recognition was presented. This framework comprises two action classifiers which are trained on the skeleton and the RGB color data, respectively. A third classifier then fuses

the results of the two classifiers to deliver the final action classification.

To limit the volume of the data processed by the RGB modality classifier, MMNet exploits the skeleton information to identify a sequence of spatial regions of interest (ROI) surrounding each skeleton joint. The identified regions are used to crop the data only to the regions surrounding specific skeleton bones. This is followed by temporal sampling to select only the regions from specific frames. An overview of this process is shown in Fig. 1.

A brief summary of the strategy employed by MMNet [4] for identifying the spatial ROIs follows. Given a dataset of N video samples, $V = \{V^{(i)} \mid i = 1, \ldots, N\}$, each action sequence can be represented as $V^{(i)} = (f_1^{(i)}, \ldots, f_T^{(i)})$ for the time interval $[1, T]$ [4]. The skeleton joint information is used to identify the spatial ROI, which is represented by the following transformation function for a given sample i:

$$R_{tj}^{(i)} = g(f_t^{(i)}, o_{tj}^{(i)}), j \in (m_1, \ldots, m_{M'_O}), M'_O < M_O \quad (1)$$

where $o_{tj}^{(i)}$ is the j^{th} joint of the skeleton at time t, m_1 to $m_{M'_O}$ are the M'_O indices of the skeleton joints that are considered, which are less than the total number of skeleton joints M_O [4]. Temporal sampling is performed by selecting L frames at time $\tau = \{\textit{interval} \times l \mid l = 1, \ldots, L; \textit{interval} = \frac{T}{L}\}$ [4].

After identifying the spatial ROIs and temporal sampling, the ROIs are concatenated in a single $M'_O \times L$ image forming the spatio-temporal region of interest (ST-ROI), on which the classifier is trained [4]. Actions consisting of two subjects result in an ST-ROI where each spatial ROI is the horizontal concatenation of the spatial ROI of both subjects.

MMNet uses the OpenPose tool [6] to estimate the skeleton information, rather than using the skeleton data that is retrieved through depth sensors such as Microsoft Kinect and Intel RealSense. The reason given is that the skeleton data estimated by OpenPose is somewhat more accurate than that retrieved by depth sensors [4]. This indicates that the performance of the this method is highly dependent on the accuracy of the skeleton information which is generally not immediately available and is obtained through a post-process of the captured data. Furthermore, running skeleton estimation using OpenPose is a computationally expensive step which adds to the overall time required for training and inference.

III. Using motion saliency to identify relevant regions

Our work was inspired by and builds on the human action recognition pipeline presented in [4]. We explore the application of motion saliency detection as an alternative strategy to the skeleton information for reducing the size of the RGB modality data. The aim was to produce a classification accuracy that matches [4] but in a reduced computational time and without relying on high quality skeleton data or OpenPose skeleton estimation. An overview of the steps of our proposed pipeline is shown in Fig. 2, with the main steps that differ from MMNet shown highlighted in orange.

In [9] a motion saliency detection approach which makes use of the depth information provided by RGBD video to improve the accuracy of the computationally simple and efficient difference of frames method was developed. This approach demonstrated a superior performance to other approaches that were considered, such as BSUV-Net 2.0 [17], in identifying salient regions whilst at the same time requiring much less computational time. Hence, this motion saliency detection method was adopted in our HAR pipeline.

The basis of the saliency detection approach in [9] is the difference of frames, which is used to roughly determine the regions in the video which capture motion. These alone are highly susceptible to noise [9] and hence the depth data of RGBD video is used to determine the regions which pertain to a subject carrying out an action. The depth map is segmented using an adaptive threshold determined from the depth value histogram. The segmented depth map is then used to determine which regions identified by the difference of frames intersect with a subject carrying out an action. Only those regions which intersect with a subject region are retained with the remaining regions discarded. The output of saliency detection is the set of salient regions per subject, where each region is the union box of the regions identified by difference of frames that intersect with the bounding box of the subject.

Let N be the number of salient regions identified in a single frame of a video, and r_j, where $j \in [1, N]$, be the j^{th} region identified. The spatial ROI for frame i, denoted by S_i, is the union box of every r_j identified in frame i, which is defined by the following:

$$x_{S_i} = \min\{x_{r_j} \mid j \in [1, N]\} \quad (2)$$

$$y_{S_i} = \min\{y_{r_j} \mid j \in [1, N]\} \quad (3)$$

$$w_{S_i} = \max\{w_{r_j} \mid j \in [1, N]\} \quad (4)$$

$$h_{S_i} = \max\{h_{r_j} \mid j \in [1, N]\} \quad (5)$$

where x_{r_j} and y_{r_j} are the x and y coordinates, respectively, of the salient region r_j, while w_{r_j} and h_{r_j} are the width and height of region r_j, respectively.

The equations (2) and (3) define the x and y coordinates, respectively, of the top left corner of the spatial ROI for frame i with the width and height of the spatial ROI defined by (4) and (5), respectively.

To determine a spatio-temporal region of interest (ST-ROI), the spatial ROIs for each frame are resized to a common size that is the average size of the spatial ROIs for all frames. This is defined by the following:

$$w'_{S_i} = \frac{\sum_{n=1}^{K} w_{S_n}}{K} \quad (6)$$

$$h'_{S_i} = \frac{\sum_{n=1}^{K} h_{S_n}}{K} \quad (7)$$

where K represents the number of frames. The spatial ROI for each frame is resized to a size with the width defined by (6) and the height defined by (7). Each spatial ROI is resized such that its center is preserved.

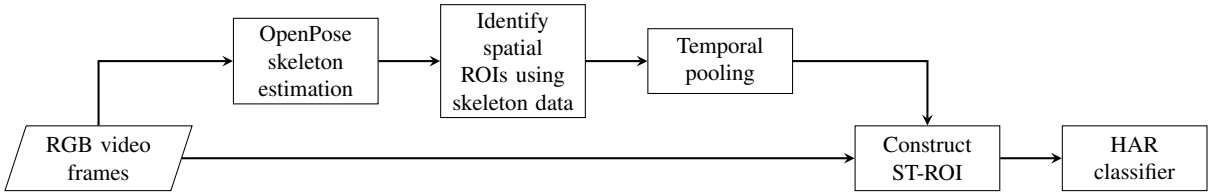


Fig. 1. Overview of the main steps in the MMNet RGB modality pipeline

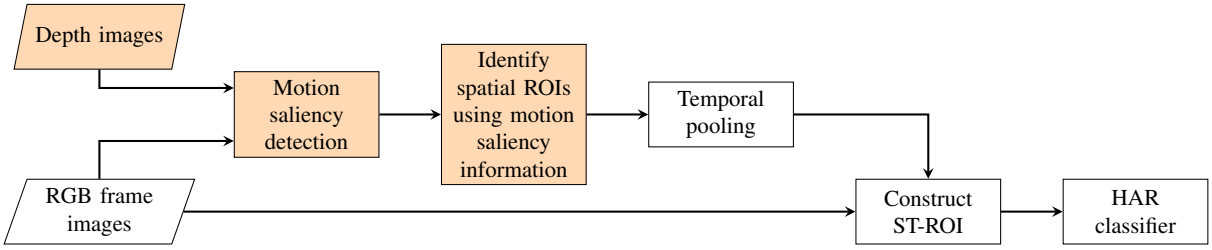


Fig. 2. Overview of the main steps in the proposed pipeline highlighting the differences from MMNet

To construct an ST-ROI that covers an entire video, the spatial ROIs for all frames are concatenated into a single image, which captures the ST-ROI size of the video. Such images are then fed into the classifier for training and inference.

IV. Experiments and Results

As stated earlier, the main goal of this work is to produce a classifier for the RGB modality with a performance that is comparable to the RGB classifier of MMNet [4] but requiring a shorter time for training. To evaluate this work, the performance of the classifier trained on an ST-ROI constructed using motion saliency detection is compared to the performance of the RGB modality classifier from MMNet, which is trained on an ST-ROI constructed out of spatial ROIs identified using OpenPose skeleton data.

A. Determining optimal motion saliency parameters

Multiple experiments were performed on the NTU-60 [10] RGB+D dataset, which provides depth data in addition to color video, with varying parameters of the ST-ROI to determine the optimal parameter configuration. The parameters that are varied are the number of spatial ROIs included in the ST-ROI, i.e. the number of frames, the configuration in which they are concatenated into a single image, and the size of the ROIs. The size of the ROIs varies with each sample depending on the number of actors and the type of action. For a fair comparison, the height of the spatial ROIs is fixed in the experiments, with the width scaled proportionally to preserve the aspect ratio.

The number of spatial ROIs included in the ST-ROI was varied from 4 to 16, with the height varying from 60 to 480 pixels, which is the height of the MMNet ST-ROI images. The spatial ROIs can either be concatenated in a square grid, an example of which is shown in Fig. 3, or in a horizontal row, an example of which is shown in Fig. 4. For comparison, an example of an ST-ROI image produced by MMNet is shown in Fig. 5.

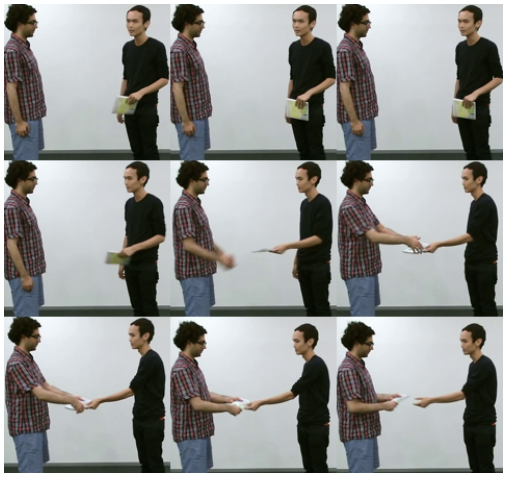


Fig. 3. Example of motion saliency ST-ROI with 9 frames concatenated in a 3×3 grid for the action "giving object" involving two actors

This work adopts the same classifier as MMNet, a ResNet with 18 layers illustrated in Fig. 6 and trained for 80 epochs with an initial learning rate of 0.1. The input to the classifier both during training and validation is a single ST-ROI image per video sample. The classifier outputs a vector of size 60, for the 60 possible action classes in NTU-60, where each element is the probability that the input contains an action of a particular class.

The experiments were performed on a machine having an Intel i7-11800H with 8 cores, an Nvidia GeForce GTX 3060 graphics card and 32 GB of RAM. The results of the experiments presented in Table I, along with the training time and the average inference time per sample, show a few general trends. A higher classification accuracy is achieved by concatenating the spatial ROIs horizontally, but the grid configuration requires a lower training and inference time. It

Fig. 4. Example of motion saliency ST-ROI with 16 frames concatenated horizontally in a row, 1×16 for the action "pick up"

TABLE I
ACCURACY, TRAINING AND INFERENCE TIME FOR VARIOUS MOTION SALIENCY CONFIGURATIONS

Configuration	Height (px)	Top-1 Accuracy (%)	Top-5 Accuracy (%)	Training Time (hours)	Inference Time (milliseconds)
2×2	240	57.90	84.67	9.69	10
3×3	160	64.08	88.17	10.00	11
4×4	120	59.44	84.97	10.01	11
1×4	480	65.63	89.64	27.29	18
1×9	480	63.53	87.68	29.17	29
1×16	480	70.39	91.56	32.05	33
1×16	120	69.51	91.75	30.98	25
1×16	60	67.67	90.31	3.79	4

TABLE II
ACCURACY, TRAINING AND INFERENCE TIME OF MOTION SALIENCY WITH LEAST TRAINING TIME CONFIGURATION COMPARED TO MMNET

Method	Height (px)	Top-1 Accuracy (%)	Top-5 Accuracy (%)	Training Time (hours)	Inference Time (milliseconds)
Motion Saliency (1×16)	60	67.67	90.31	3.79	4
MMNet	480	74.41	95.57	19.51	19

Fig. 5. Example of a ST-ROI produced by MMNet using OpenPose skeleton information for the action "drinking"

TABLE III
COMPARISON OF EXECUTION TIMES FOR DIFFERENT MOTION SALIENCY CONFIGURATIONS WITH OPENPOSE SKELETON ESTIMATION.

Configuration	Height (px)	Execution time (hours)
1×16	480	9.07
1×16	120	8.82
1×16	60	8.74
OpenPose	480	99.17

can also be observed that the more spatial ROIs are included in the ST-ROI, the higher the classification accuracy and the longer the training time.

The highest classification accuracy, 70.39%, was achieved using a configuration of 1×16 ROIs with an ST-ROI height of 480 pixels. The same configuration also required the longest training time, 32.05 hours, as well as the longest inference time, 33 milliseconds. The lowest training time, 3.79 hours, was achieved with the configuration 1×16 and an ST-ROI height of 60 pixels with a top-1 classification accuracy of 67.67%, which represents a loss of 2.72% accuracy. This configuration also resulted in the lowest inference time, 4 milliseconds. An interesting observation is that the top-5 classification accuracy only decreased by 1.25%. From the results, it can be deduced that the best motion saliency configuration, taking into account classification accuracy, training time and inference time, is an ST-ROI with a height of 60 pixels consisting of 16 images concatenated in a horizontal row.

B. Comparison to other methods

Table II compares the results obtained using motion saliency with the most optimal configuration, compared to the MMNet RGB classifier. The MMNet RGB modality classifier achieved a classification accuracy of 74.41% but with a training time

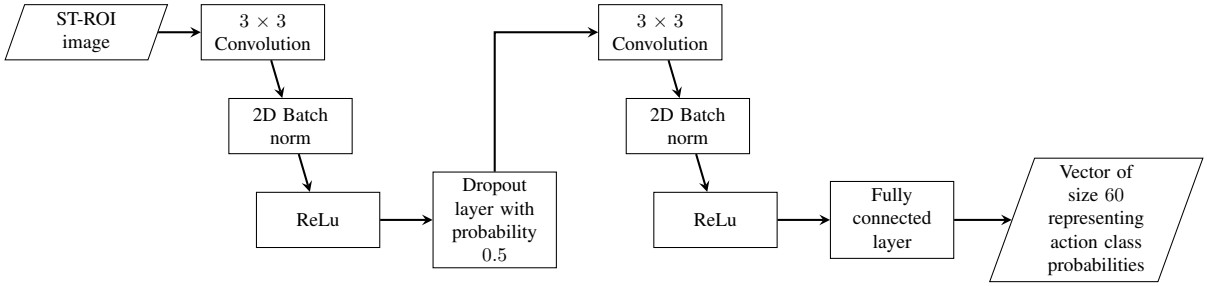


Fig. 6. Diagram showing the layers of the RGB ST-ROI classifier

of 19.51 hours, which is roughly five times longer than the proposed solution. This is also reflected in the average sample inference time, which is nearly five times longer for the MMNet RGB modality classifier, 19 ms, compared to the proposed solution, which took on average 4 ms per sample.

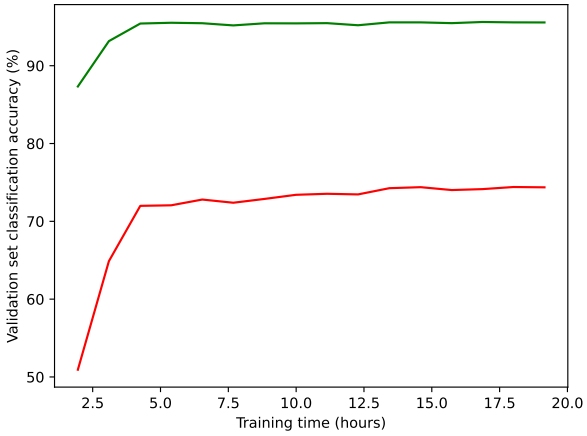


Fig. 7. MMNet validation set classification accuracy vs training time. Red graph represents the accuracy while the green graph represents the top-5 accuracy

The graphs presented in Fig. 7 and Fig. 8 show the classification accuracy achieved with training time for the MMNet RGB classifier and our work using the 1×16 ST-ROI configuration with a height of 60 pixels, respectively. These reveal that a classification accuracy close to the maximum is already reached at roughly 0.7 hours with the motion saliency ST-ROI, whereas the MMNet RGB classifier requires 4 hours to reach a classification accuracy close to the maximum.

In addition to requiring a longer time for training compared to the motion saliency ST-ROI configuration 1×16 with a height of 60 pixels, MMNet also requires a skeleton data estimation step using the OpenPose tool. This presents a high computational cost in comparison to motion saliency detection, as shown in Table III. The OpenPose skeleton estimation step took 99.17 hours to complete for the entire NTU-60 dataset. For comparison motion saliency detection using the ST-ROI configuration 1×16 with a height of 480 pixels, which resulted

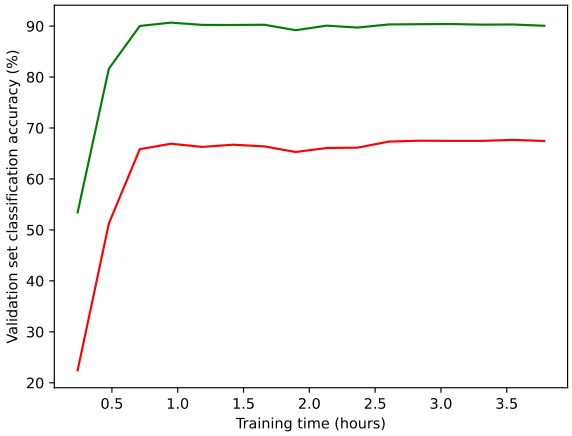


Fig. 8. Validation set classification accuracy vs training time using ST-ROI with 1×16 configuration and a height of 60 pixels. Red graph represents the accuracy while the green graph represents the top-5 accuracy

in the longest training time, took 9.07 hours to complete for the entire NTU-60 dataset. This means that even with the slowest motion saliency configuration, which produced the highest classification accuracy, 70.39%, our pipeline is still considerably faster, during training, than the MMNet RGB modality pipeline. The total computational times required, including the preprocessing steps and training, for various motion saliency configurations compared to the MMNet RGB modality pipeline are shown in Table IV.

TABLE IV
COMPARISON OF TOTAL COMPUTATIONAL TIME (INCLUDING TRAINING AND PREPROCESSING STEPS) FOR DIFFERENT MOTION SALIENCY CONFIGURATIONS WITH MMNET.

Configuration	Height (px)	Execution time (hours)
1×16	480	41.12
1×16	120	39.80
1×16	60	12.53
MMNet	480	118.68

These results show that the proposed solution can achieve results which are comparable to MMNet, but within a much shorter training time. This also means that the hardware consumes much less energy during training.

V. Conclusion

In this paper we explored the use of motion saliency detection in human action recognition. Experimental results showed that a considerable reduction in training time can be achieved by using motion saliency detection to limit the data that is processed to the regions which capture the human action.

The results showed that the minimum training time was achieved with an ST-ROI configured as 1×16 with a height of 60 pixels. This gave an accuracy which is close to that of MMNet [4], but with a training time which is five times shorter and without requiring the OpenPose skeleton estimation step. This indicates that the ST-ROI constructed using information from motion saliency detection can be used to drastically reduce the training time and energy consumption without significantly compromising classification accuracy.

Future areas of research involve exploring the use of this work in a multi-model human action recognition system and identifying ways that can improve the accuracy with minimal impact on the training time.

References

[1] C. J. Debono, M. Sacco, and J. Ellul, "Monitoring indoor living spaces using depth information," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, 2020, pp. 1–5.

[2] Z. Ahmad and N. Khan, "Human action recognition using deep multilevel multimodal (M^2) fusion of depth and inertial sensors," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1445–1455, 2020.

[3] Y. Kong, Y. Wang, and A. Li, "Spatiotemporal saliency representation learning for video action recognition," *IEEE Transactions on Multimedia*, vol. 24, pp. 1515–1528, 2022.

[4] B. X. Yu, Y. Liu, X. Zhang, S.-h. Zhong, and K. C. Chan, "Mmnet: A model-based multimodal network for human action recognition in rgb-d videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3522–3538, 2023.

[5] S. Das, S. Sharma, R. Dai, F. Brémond, and M. Thonnat, "Vpn: Learning video-pose embedding for activities of daily living," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 72–90.

[6] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.

[7] Y. Xue, X. Guo, and X. Cao, "Motion saliency detection using low-rank and sparse decomposition," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1485–1488.

[8] Z. Zheng, G. An, and Q. Ruan, "Motion guided feature-augmented network for action recognition," in *2020 15th IEEE International Conference on Signal Processing (ICSP)*, vol. 1, 2020, pp. 391–394.

[9] A. Gutev and C. J. Debono, "Motion saliency detection using depth information for human action recognition applications," in *2023 International Symposium ELMAR*, 2023, pp. 67–70.

[10] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2684–2701, 2020.

[11] ——, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2684–2701, 2020.

[12] B. M. Martínez, D. Modolo, Y. Xiong, and J. Tighe, "Action recognition with spatial-temporal discriminative filter banks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 5481–5490.

[13] A. Shahroudy, T.-T. Ng, Y. Gong, and G. Wang, "Deep multimodal feature analysis for action recognition in rgb+d videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1045–1058, 2018.

[14] K. Soomro, A. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, 12 2012.

[15] F. Baradel, C. Wolf, J. Mille, and G. W. Taylor, "Glimpse clouds: Human activity recognition from unstructured feature points," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 469–478.

[16] Z. Luo, J.-T. Hsieh, L. Jiang, J. C. Niebles, and L. Fei-Fei, "Graph distillation for action detection with privileged modalities," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 174–192.

[17] M. Tezcan, P. Ishwar, and J. Konrad, "Bsuv-net 2.0: Spatio-temporal data augmentations for video-agnostic supervised background subtraction," *IEEE Access*, vol. 9, pp. 53 849–53 860, Apr. 2021.