Detecting periodic action patterns in videos

Slesa Adhikari

Follow this and additional works at: https://louis.uah.edu/uah-theses

Recommended Citation
https://louis.uah.edu/uah-theses/323

This Thesis is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Theses by an authorized administrator of LOUIS.
DETECTING PERIODIC ACTION PATTERNS IN VIDEOS

by

SLESA ADHIKARI

A THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in
The Department of Computer Science
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2020
In presenting this thesis in partial fulfillment of the requirements for a master’s de-
gree from The University of Alabama in Huntsville, I agree that the Library of this
University shall make it freely available for inspection. I further agree that permis-
sion for extensive copying for scholarly purposes may be granted by my advisor or,
in his/her absence, by the Chair of the Department or the Dean of the School of
Graduate Studies. It is also understood that due recognition shall be given to me
and to The University of Alabama in Huntsville in any scholarly use which may be
made of any material in this thesis.

Slesa Adhikari

4/16/2020

(date)
THESIS APPROVAL FORM

Submitted by Slesa Adhikari in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in Computer Science and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in Computer Science.

Dr. Ramazun Aygun 04/13/2020  
Committee Chair

Dr. Haeyong Chung 04/15/2020  

Dr. Huaming Zhang 04/13/2020  

Department Chair

Dr. Heggere S. Ranganath 4/13/2020  
Department Chair

Dr. John Christy 04.15.2020  
College Dean

David Berkowitz 04/13/2020  
Graduate Dean

Dr. David Berkowitz (Date)
ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Masters of Science
College/Dept. Science/Computer in Computer Science

Name of Candidate Slesa Adhikari

Title Detecting Periodic Action Patterns in Videos

Finding periodic segments in videos has a wide range of applications like recognizing and classifying actions in a video. In this thesis, we present a solution to the problem of identifying repetitive segments in a video and finding the number of periodic actions appearing in these repetitive segments in an unsupervised manner.

The proposed method generates time-series data from the distance matrix of frames in a video. The time-series data is then analyzed to first determine the intervals where repetitions occur and then compute the number of periodic actions in these segments.

Our method was evaluated using the MHAD202-v dataset. The experimental results show that the repetitive actions were detected with precision around 0.95 and an F1-score greater than 0.91. The error rate for the count of periodic segments was just around 15.8%.

Abstract Approval: Committee Chair

Dr. Ramazan Aygun

Department Chair

Dr. Hegere S. Ranganath

Graduate Dean

David Berkowitz
ACKNOWLEDGMENTS

I wish to express my deepest gratitude to Dr. Ramazan Aygun, who took the time to provide constant suggestions in helping to refine my work, without which this thesis would not have been possible. It is whole-heartedly appreciated that your great advice for my research proved monumental towards the success of this study. I would also like to extend my sincere gratitude towards Dr. Haeyong Chung and Dr. Huaming Zhang for providing valuable suggestions to improve my thesis.

Finally, I would also like to thank my family and friends who were a constant source of support and encouragement throughout the duration of my thesis. I couldn’t have done it without them.
TABLE OF CONTENTS

List of Figures viii

List of Tables ix

Chapter

1 Introduction 1

1.1 Motivation ................................. 1

1.2 Problem Statement ............................ 2

1.3 Our Approach ............................... 4

1.4 Research Questions and Our Contribution ............... 6

1.5 Thesis Organization ............................ 7

2 Related Works 8

2.1 Mapping to 1-Dimensional Time Series Data ............... 9

2.2 Flow Techniques ............................ 11

2.3 Supervised Approaches .......................... 14

2.4 Dynamic Time Warping and Other Approaches ............. 16

2.5 Summary ................................ 18

3 Methodology 20

3.1 Feature Generation from Video ........................ 23
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Sample distance matrix for a video with repetitive actions</td>
</tr>
<tr>
<td>3.1</td>
<td>System Block Diagram</td>
</tr>
<tr>
<td>3.2</td>
<td>Illustration of a repetitive action (jumping jacks repeated 3 times)</td>
</tr>
<tr>
<td>3.3</td>
<td>The architecture of VGG-16.</td>
</tr>
<tr>
<td>3.4</td>
<td>Distance matrix with repetitive patterns shown</td>
</tr>
<tr>
<td>3.5</td>
<td>(a) distance matrix generated from gray-scale pixel intensities (b) distance matrix generated from deep convolutional features</td>
</tr>
<tr>
<td>3.6</td>
<td>Distance matrix after gaussian blurring</td>
</tr>
<tr>
<td>3.7</td>
<td>Sample time-series data generated from a distance matrix</td>
</tr>
<tr>
<td>3.8</td>
<td>Sample sub-actions in a video. First row: bending down and up; second row: stretching back and forth. The last frame in the first sub-action is shown as the first-frame in the second sub-action to show the relationship between them</td>
</tr>
<tr>
<td>3.9</td>
<td>Detection of the three periodic segments (red, magenta, black colored squares in the diagonal) by our approach</td>
</tr>
<tr>
<td>3.10</td>
<td>The first periodic segment (black highlight) detected by our algorithm is a false positive.</td>
</tr>
<tr>
<td>3.11</td>
<td>Pruning a candidate repetitive action due to the presence of few peaks.</td>
</tr>
<tr>
<td>4.1</td>
<td>Sample frames of 11 actions from the MHAD datasets. [1]</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Comparison of detecting frames appearing in repetitive actions</td>
</tr>
<tr>
<td>4.2</td>
<td>Performance for detecting the number of periodic actions in a repetitive action</td>
</tr>
<tr>
<td>4.3</td>
<td>Statistics for videos having periodic actions and the performance of our method for each segment</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Motivation

Over the past few decades, there has been a huge surge in the quantity of video data that is uploaded to the world wide web and online social networks. Nevertheless, methods to analyze these videos in an automated manner have not been developed at the same pace. Earlier content-based retrieval methods relied on feature extraction and indexing by dividing videos into smaller clips through shot detection [2] [3] [4]. With the advancement of deep learning methods, the presence of an action in video clips can be recognized [5] [6] [7] [8]. However, for better video content analysis and retrieval, methods that analyze beyond shot level splitting or detecting presence of actions are needed.

Many videos captured in real life consist of actions that have certain motions that are repeated multiple times. In this thesis, we aim to detect periodic continuous actions. Periodicity means that an action re-occurs multiple times at almost similar time intervals. Periodic motions are very common in nature and our daily lives [9] [10]. Many day-to-day actions of people are repetitive in nature. For example, walking, running, exercising, chopping vegetables/fruits, and brushing are sample
repetitive actions people do in their daily lives. Moreover, workout videos consist of sets of repetitions of a particular exercise; and dancing is also comprised of periodic locomotive motions of legs and arms. Thus, the automatic detection of these periodic patterns in videos has important applications as mentioned below.

1. **Action Localization:** Most of the videos in the wild contain actions that are repetitive in nature. Detecting and localizing periodic segments of a video can be extended to localize multiple actions in a long video since a repetitive segment is just a single action that has been performed multiple times.

2. **Action Detection:** Action detection in videos is usually a machine learning problem by learning features that distinguish actions or by comparing an action with similar actions. Detecting the periodic nature of an action could be an indication of the kind of action it is and can be used to detect those actions.

3. **Object Tracking:** Certain objects in this world exhibit periodic patterns. The motions of wheels, windmills, swings, pendulums all have a periodic nature [11]. Identifying periodic segments in the video might give some indication about the type of an object determining the periodic action. Then, it can be used for object tracking.

1.2 **Problem Statement**

In this thesis, we provide a novel approach to determine the temporal extent of each of the multiple periodic actions performed in the video. An action is a sequence of frames in a video where a well defined motion with a semantic meaning occurs.
In case of repetitive action, a single action unit is a sequence of frames with action that is repeated in the consecutive frame sequences. These frames sequences are of similar length and are visually similar to each other. This detection is achieved in a completely unsupervised manner without using any prior information about the presence of periodic actions, the number of periodic actions or the scenario in which the video is taken. Note that our goal is not to determine whether a video has a repetitive action and to label it. Rather, our main goal is to determine when those repetitive actions occur in a video, the period of the repetition and how many times the action has been repeated. Since a video may have multiple types of actions, the goal is to find each repetitive action in the video.

More specifically, given an input video that may or may not consist of periodic segments, the goal of our work is to analyze the time-series data obtained from the distance matrix of frames in a video to

1. determine if there are any periodic actions in the video,

2. identify all the frames where repetitive periodic motion is observed, and

3. temporally localize the segments that contain repetitive actions, i.e., for each periodic action in the video, identify the point in time where the action starts and ends.

4. use autocorrelation to estimate the periodicity of the repetitive periodic segment and use it to determine the repetition count.
Our method does not use any prior knowledge about the nature of the periodic action, the number of periodic actions, or the semantics of the observed scene.

1.3 Our Approach

Our proposed method analyzes time-series data extracted from videos. However, the representation of time-series data is challenging. When we used consecutive frame differences using the features of frames, the time-series data was found to be not good enough to identify repetitive actions. Instead, our algorithm operates on a distance matrix of the video which is a matrix containing the pairwise distances of frames in the video. For a video with $N$ frames, a distance matrix has a dimension of $N \times N$, where the value in the matrix at $i^{th}$ row and $j^{th}$ column indicates the distance between the corresponding $i^{th}$ and $j^{th}$ frames. Figure 1.1 shows an example of a distance matrix built in this way, highlighting the sections of repeated action. Evidently, the repeated pattern that is exhibited in the distance matrix (in the red square along the main diagonal) corresponds to the repetitive periodic actions in the video.

Rather than using this distance matrix, we generate time-series data from the distance matrix. We utilize consecutive frame differences but not from the original features of frames. We used gray-scale values and deep neural features obtained from a deep learning algorithm as two sets of features. We used the cosine distance measure to generate the distance matrix. Then each frame is represented as an N-dimensional feature vector indicating its distance to every frame in the sequence. The differences of frames are calculated based on row features in the matrix using Euclidean distance.
Figure 1.1: Sample distance matrix for a video with repetitive actions

Hence, for $N \times N$ distance matrix, we obtain $N - 1$ values where each value is a distance between two N-dimensional vectors. These $N - 1$ values are treated as time-series data. In the time-series data, the patterns emerge in the form of consecutive peak values (local maxima) with similar amplitude spanning a time interval. Then, our algorithm clusters these segments of time-series data containing peaks of similar heights to curate a list of candidate periodic segments. A pruning algorithm is used to discard the non-periodic segments from the list of candidate periodic segments. It analyzes only the portion of the time series for the candidate segment and applies smoothing to the sequence. If after smoothing, there are no or very few sharp changes in the series, the segment is marked as non-periodic.
Finally, we use autocorrelation on the detected periodic segments to estimate the period of repetition. Consequently, the number of repetitions is simply the total number of frames in the segment divided by the period of the segment.

Although the method proposed to determine the periodicity is generic - it does not involve any human focused kinematics like bone/joint movements - and is not restricted to any certain kind of motion, the experimental evaluation has been performed on a human action dataset only.

1.4 Research Questions and Our Contribution

This research aims to answer the following research questions:

1. Can time-series data obtained from the distance matrix created from the frame features of a video be used to detect periodic repetitive patterns in a video?

2. Can time-series data obtained from the distance matrix created from the frame features of a video be used to count the number of repetitions in a periodic action pattern?

The contributions that this research study makes are:

1. Detection of the segments in a video where there are repetitive periodic actions by analysing the patterns in the time-series obtained from the distance matrix of frame features of the video.

2. Estimation of the count of the number of repetitions in each of the periodic segments of a video by using correlation on time-series of the periodic segment.
1.5 Thesis Organization

The remainder of this thesis has been organized as follows. Chapter 2 provides the related work relevant to periodic action detection. Chapter 3 describes our methodology in detail by presenting stages of finding periodic actions. Chapter 4 explains the experiments performed in the MHAD202-v video dataset and presents performance evaluations by comparing with an existing periodic segment detection technique. Chapter 5 concludes the work that has been achieved and points out the possible future directions.
CHAPTER 2

RELATED WORKS

Detecting periodic actions has many applications in high throughput biological experiments, activity monitoring, sports, and gaming [12]. There have been recent studies on finding periodic actions in videos. Specifically, the same problem in the time-series domain has had a lot of attention. Consequently, one common strategy for detecting periodicity in video is to derive a 1D function to act as a surrogate for its dynamics, and then to use either frequency domain (Fourier transform) or time domain (auto-correlation, peak finding) techniques [12]. This approach has been popular in the literature [13] [11] [14]. A second popular approach utilizes the concept of “flowing” pixels [15] [16]. Similarly, there are a few studies that utilize supervised approaches that employ deep neural networks to solve the problem [12] [17]. Some techniques involve the analysis of self-similarity/distance matrices [14] [17] [18] and others use the dynamic time warping approach [19]. A brief overview of these these techniques is provided in the following sections.
2.1 Mapping to 1-Dimensional Time Series Data

One of the common strategies to detect the periodicity of a video signal is to first map it to a 1D signal such that it preserves the dynamics of the video and then apply either time series analysis or frequency domain analysis or some other method on the signal to detect the periodicity.

An example of this approach is used by Karvounas et al. [13]. They present an unsupervised method to localize the periodicity in a video that also contains non-periodic segments. They employ a method called particle swarm optimization (PSO) to minimize an objective function for the time-series data. To convert any video into time-series data, they consider the periodic fluctuations of image brightness intensities over time. A series of preprocessing is done by blurring the image frames using a Gaussian filter, followed by removal of background and then splitting the frames into 30x30 tiles. Only the tiles that show a large variation in the intensity (indicating significant motions) are taken and a time series is defined for each of these tiles to show the progression of their average intensities. While this method gives a pretty good estimation of the locality of periodic action and its period, the videos that it operates on are highly restrictive. The assumption is that the video should have only one action that is repeated and has non-periodic prefix and tail.

Another approach which adopts time series analysis is employed by Briassouli and Ahuja [11]. In their approach, the pixel intensities for each image frame in the video are projected onto the x and y axes, thus giving two one dimensional signals for each frame. These projections over time capture periodic motions along these
axes. Based on their velocities in each direction, a frequency modulated (FM) signal can be created (which is periodic in nature). Standard power spectrum analysis of these FM signals helps detect the frequencies present in these signals. Then periods of motion for individual objects can be extracted from these signals. Consequently, this approach helps identify multiple periodic motions in the same video sequence. Based on their experimental results, their videos seem to contain one type of motion but with multiple objects (e.g., pendulum, walking, dribbling). In our case, a video may have different types and number of periodic actions.

Another work on utilization of frequency analysis is done by Cutler and Davis [14]. They propose an algorithm for periodicity detection and analysis that consists of two parts. First, the motion is segmented and the object is tracked in the foreground. Then each object is aligned along the temporal axis (using the object’s tracking results) and the object’s self-similarity is computed as it evolves in time. The self-similarity metric is periodic for periodic motions. Time-frequency analysis is applied to detect and characterize the periodicity. The 2D lattice structures inherent in similarity matrices can be utilized to analyze the periodicity robustly. Based on their experimental results, it was observed that their videos consisted of a single type of periodic action (e.g., walking person, running dog). As mentioned before, our approach is not limited by the number of periodic segments in the video.

Approaches based on the frequency spectrum like these tend to have the limitation that the action frequency should be almost constant and it would emerge as a discernible peak at a time frequency graph. However, the amount of variation in ap-
pearance between repetitions and the variation in action length means that in certain cases, no such clear peak may be identifiable [12].

Our approach also utilizes this approach of mapping the video to a 1D time-series. Nevertheless, the novelty of our approach lies in the process of this mapping. To the best of our knowledge, our method is the first to utilize patterns exhibited by the self-distance matrix of the video as the basis for mapping to a time series for detecting periodic actions.

2.2 Flow Techniques

Another class of techniques considers the displacement/flow of pixels with respect to time to detect the repetitions. This “flow” is mapped into some form of data - time series or otherwise - and that mapping is used for further processing to detect periodicity.

One such technique is proposed by Polana and Nelson [15]. They detect the “flow” of pixels as time progresses in the video to detect the motion. Differential methods are used to compute the magnitude of flow at each pixel between any two consecutive image frames. Using thresholding over this flow magnitude, the parts of the video where there is maximum motion can be detected. These parts correspond to the “actor” who is moving in between the frames. The centroid of the actor can then be determined in each frame and the motion of this centroid between the frames can be used to estimate the mean velocity of the actor. Reference curves are drawn in the spatio-temporal space that depicts the motion of the centroids and the motion signals are extracted along these curves. These signals are now fed to a time-series
analysis algorithm. Fourier transform helps determine the superior frequency and the corresponding periodicity for each individual signal. The overall frequency of the image sequence is estimated as the frequency which has the maximum corresponding average periodicity (average period of each pixel involved in the motion). This method is successful in identifying quite complex periodic motions as well. However, the assumption is that the motion detected in the video belongs to a set of known set of periodic activities. Our approach, in contrast, does not have any restriction on the type of periodic motion.

A unique approach that does not utilize time series analysis but rather incorporates the 2D nature of video frames is presented by Runia et al. [16]. This new approach also addresses the imperfect periodicity in real life videos. First of all, the wavelet transformation is used to address the issues of non-static video dynamics. Secondly, they define the concept of 3D flow field \(F_t(x)\) which defines the flow (motion) of an object at spatial coordinate \(x\) and at time \(t\). An object has a periodic motion if a periodic 3D flow field can be tied to the object as \(F_t(x) = F_{t+T}(x + S)\) where \(T\) is the time period and \(S\) is the period over space. Based on differential geometry, the flow field is periodic if it satisfies one of these conditions:

\[
\nabla F_t(x) = \nabla F_{t+T}(x + \epsilon)
\]

\[
\nabla \times F_t(x) = \nabla \times F_{t+T}(x + \epsilon)
\]

\[
\nabla \cdot F_t(x) = \nabla \cdot F_{t+T}(x + \epsilon)
\]
where the notations define the gradient, curl and divergence of the flow field respectively.

These equations correspond to constant, intermittent and oscillating motion. The types of motions of the objects can also be identified based on the following properties: i) The translation is given by zero curl and zero divergence, ii) the rotation is given by non-zero curl but zero divergence, and iii) the scale is given by zero curl but non-zero divergence.

To project the idea of a 3D flow field into the 2D view of a video, the authors define the 2D periodicity property of the 3D flow field as $F_t(x) = F_{t+T}(\sigma(x+s))$ where $s$ denotes the displacement, $\sigma$ denotes the “observational scale” (camera zoom). The principle behind this is that the object has the same period in the 2D view as in the 3D view for the time domain for all the intrinsic properties. Finally, based on camera views (front, side), the type of motion (translation, rotation, and scale) and the type of periodicity (constant, intermittent and oscillating), 18 different cases are considered by the authors. The overall estimation of repetition then involves a three-stage approach. First, the target object is detected using image segmentation techniques. Then the flow of the target object is estimated using flow signals and decomposed into a time-frequency spectrum using wavelet transformation. The maximum of the wavelet spectrums is chosen to give a local frequency value for each time frame. Then the local frequencies are integrated over time to estimate the repetition count. Since multiple flow signals can be estimated, the most discriminative signal is chosen based on local regularity in the time-frequency space. Their results are quite impressive and show that they are able to handle complex movements and camera motions in
real-life videos. Their method requires that the wavelet spectrum would yield one dominant repetitive action [16].

2.3 Supervised Approaches

Some of the more recent methods use machine learning/deep learning techniques to identify periodicity and localize periodic segments in videos. Deep learning has achieved significant results for image processing and computer vision. Various RNN and LSTM techniques have also been successful on video analysis. A few studies in this field have also been proposed for periodicity detection in videos [12] [17].

Giorgos et. al. [17] have proposed a method using a convolutional neural network called ReActNet to classify whether or not a given frame belongs to a segment of repetition. The input to the network is a matrix of pairwise distances between frames (distance matrix), where each frame is represented as a feature vector obtained from a convolutional neural network. The target of the training samples is a matrix \( A \) such that the value at \( A_{ij} \) is 1 if the frames \( i \) and \( j \) belong to the same segment of repetition and 0 otherwise. Sub-blocks of the matrices are reused to create thousands of training samples from a few sets of videos. The architecture of ReActNet is built using a stack of hourglass modules [20], each of which is basically an autoencoder [21] with skip connections [22]. The encoder part is made up of convolutional and max-pooling layers and the decoder part is made up of corresponding deconvolutional and upsampling layers. Binary cross-entropy is used as the loss function for training. Although this approach to repetitive action recognition yields impressive results on synthetic as well as real world videos, from a computational point of view, it cannot
be used in real-time or online applications due to its feature extraction [17]. It obtains an overall F1-score of 88.5% on the QUVA [16] dataset with a model trained on PERTUBE [18] dataset, while our approach yields an F1-score of 91.5%, although it is not fair to compare these results obtained on two completely different datasets. Additionally, ReActNet is unable to count the number of repetitions of the action either.

Among the supervised techniques, Levi and Wolf [12] propose a method that can count repetitions online live video. However, the video is assumed to contain repetition of approximately the same action which may not always be true in real videos. It uses a shifting windows approach to evaluate the cycle length for each window. It poses this problem as a multiclass classification problem (using a Convolutional Neural Network). Then the cycle lengths are integrated over time. The entropy of the network’s predictions is used in order to automatically start and stop the repetition counter and to select the appropriate time scale. The approach is noteworthy in that it uses no real data to train its supervised model - it does so using synthetic data.

Methods based on supervised techniques have been very popular in recent times and they have been very successful in the field of video analysis as well. Machine Learning/Deep Learning techniques require a lot of training data to build a model with predictive capabilities. On the other hand, our approach does not require any training dataset. Although we use features extracted by VGG16 network, the algorithm itself does not rely on these features. This is shown by almost similar accuracy using grayscale frame features instead. Similarly, Machine/Deep learning approaches require a lot of computational power for training, while our approach...
can be done in a fairly less expensive manner. Moreover, the results by supervised methods may be biased towards the data in the training set.

\section*{2.4 Dynamic Time Warping and Other Approaches}

One of the popular methods to compare the subsequences of time-series to detect patterns is using the concept of Dynamic Time Warping (DTW). If $X$ and $Y$ are two finite-length sequences, each of length $n$, then DTW between them is defined as follows:

\[
DTW(X, Y) = d(x_0, y_0) + \min \begin{cases}
    DTW(X, \tilde{Y}) \\
    DTW(\tilde{X}, Y) \\
    DTW(\tilde{X}, \tilde{Y})
\end{cases}
\]

where $\tilde{X}$ is the subsequence of $X$ obtained by removing the first element of $X$. $d(x_i, y_j)$ is the distance between the elements $x_i$ and $y_j$: 0 if the two values are the same and 1 otherwise. If the two sequences are arranged in a grid with each sequence along an axis, a warping path is a path from the cell $[0, 0]$ to the cell $[n - 1, n - 1]$ corresponding to one way of alignment. DTW, in principle, finds the shortest warping path and can be calculated using dynamic programming. Elfeky et al. [19] propose a time-warping algorithm called WARP that can be used for time-series data. It builds on the concept of DTW to estimate the periodicity of a time series in the presence of noise. The central idea here is to find the minimum warping distance between a time series $T$ and its shifted versions. This minimum warping distance is used to
estimate the periodicity of the series. Note that the DTW matrix consists of zeros in the diagonal and if a candidate period of repetition is \( p \), the \( p^{th} \) subdiagonal would also contain many zeros. First of all, to avoid DTW path converging towards the diagonal, the diagonal elements are replaced by infinity (or large values). Secondly, to avoid dragging adjacent minimum warping paths to the \( p^{th} \) subdiagonal, only local minima of the warping cost values are considered. This approach gives one of the best noise-resilient results. In their experimental results, they do not cover cases where multiple repetitive actions occur in a video.

A more efficient alternative to DTW is given by Panagiotakis et al. in their paper [3]. They build upon their own previous work [23] which presents an algorithm called MUCOS with time complexity of \( O(N^2) \) as opposed to time complexity of \( O(N^6) \) using DTW. MUCOS approaches this problem by first computing a distance matrix based on dissimilarity between the image frames and then using the graph search (where an objective function is minimized) to find the shortest path. The authors show that if MUCOS is applied in a suitable fashion to the input video, it is able to detect periodic segments of the input video. They call this new approach P-MUCOS. P-MUCOS is also successful in computing the period of each segment containing repetitions. They do so by tracking the paths of the identified commonalities. Although they claim to get a significant improvement in period detection over the baseline power spectrum method, the experimental results have not been provided.
2.5 Summary

A large portion of literature review on periodicity/repetition detection involves mapping the data into a time series and applying standard time series analysis techniques for trend detection and periodicity estimation. Frequency domain approaches like Fourier transform and wavelet transform are extensively used, however, they are not robust against varying periods and non-static/ non-stationary videos or dealing with multiple periodic patterns in the same signal. Another class of algorithms belong to the field of supervised learning where a training set (inputs and their corresponding outputs) is employed to train a machine learning model and this trained model predicts the output for unseen inputs. These supervised techniques have been quite successful in periodicity detection as well but they have the disadvantage of taking a lot of computational power and requiring training datasets. Other techniques involve the concept of “flow” of pixels. The flow is mapped to a time-series and time-frequency analysis like Fourier and wavelet transform are applied. Miscellaneous techniques include employing dynamic time warping, correlation or detecting and analyzing patterns on a distance matrix of pairwise distances of frames in the video.

Our method takes the idea of mapping the video to a time series. It does so, however in a novel way, by taking the row of the distance vector as feature vectors for corresponding frames. The time series is then formed by computing the distance between consecutive pairs of frames. Doing so gives better cues to detect and separate the periodic patterns compared to using the frame features directly. Additionally, our
method can detect multiple periodic action patterns in the video and estimate each of their periods and quantify the repetition counts for each.

Our research matches most closely to the P-MUCOS approach in that we also use the distance matrix. However, instead of directly trying to detect the patterns in the distance matrix like they do, we project the matrix to a one-dimensional time series and use time-series analysis to find repetitions. Thus, we take the research done in [18] as our baseline to benchmark our experimental results.
CHAPTER 3

METHODOLOGY

In this chapter, we explain our framework to detect periodic actions in videos. The first step in our framework is to preprocess the video file and convert it into a representation that is easy to use for finding periodic actions. Our method utilizes distance matrix of frames appearing in the video. Once the distance matrix is constructed, the data is analyzed as a time series by analyzing the distance between consecutive rows of the distance matrix. Using this time series, the candidate periodic segments of the video are clustered. Finally, the non-periodic segments are eliminated from the set of candidate segments leaving only the periodic segments of the video. After that, the period and repetition count of each of the periodic segments are estimated by performing autocorrelation of the time-series for that periodic portion. The framework of our method is shown in Figure 3.1.

The focus of this research is to mathematically analyze videos to detect repeated patterns and express such an analysis as an algorithm. We will first formally define some terms which are of primary concern to our problem domain to assist us in the following subsections.
Video A video is an ordered sequence of image frames. Formally, a video $V$ of length $L$ can be written as a sequence $V = (F_1, F_2, F_3, \ldots, F_L)$ where each $F_i$ is an image of size $M \times N$. A primary concern when dealing with videos is the representation of each image frame. The representation technique can greatly affect the algorithm being used for video analysis. We explore two ways to represent a frame: gray-scale
intensity values and deep neural features extracted from a deep learning architecture. These representations are discussed in the next section (Feature Generation from Video).

**Action** An action $A = (I_j, I_{j+1}, \ldots, I_{j+k-1})$ of length $K \leq L$ is a subsequence in a video $V$, of length $L$, such that the visual information in frame $I_j$ flows continuously to $I_{j+k-1}$, forming one unit of information for the observer.

**Repetitive Action** An action $A = (J_1, J_2, \ldots, J_N)$, where $J_i$ is a frame of the video, is a repetitive action if it can be evenly divided into $P$ subsequences, each of similar size $S$ such that each subsequence is visually similar to another. In other words, if $A$ is repetitive, then we can define:

$$
A_1 = (J_1, J_2, \ldots, J_{S_1})
$$

$$
A_2 = (J_{S_1+1}, J_{S_1+2}, \ldots, J_{S_1+S_2}), \ldots
$$

$$
A_P = (J_{S_{(P-1)}+1}, J_{S_{(P-1)}+2}, \ldots, J_N)
$$

where $|A_1| \simeq |A_2| \simeq \ldots \simeq |A_P|$ and $\sum |A_i| = N$.

We call each $A_i$ a subaction or an action unit. An example is shown in Figure 3.2. In the example, the model is doing jumping jacks. Essentially a unit action in this sample representation of a video is the set of first two images. These unit actions are repeated multiple times in the video.

Note that the definition of repetitive action in reality is vague. A repetitive action can be composed of multiple subactions. It is not always clear whether these subactions make a single action or if they are different components of a repetitive
action. This can be judged based on the gap between subactions or semantics of actions.

### 3.1 Feature Generation from Video

The first step in our method is to map data in a form that is suitable for identifying repetitive actions. The input data in our research is a video that consists of a set of actions where each set of actions can be repetitive or non-repetitive in nature. A video is a sequence of images or frames. Each frame is composed of pixels which have three channels in our experiments. The first goal is to determine how similar or how different frames are in a video. Thus, a video should be represented in a format to compute similarities among frames.

For that purpose, we need to quantify the “similarity” between two frames. Mathematically, similarity between two objects can be computed by calculating their distance in space. Therefore, the first step is to define the space that the frames lie in. In our research, we experiment with two different representations: gray-scale intensities and deep neural features. This is done with a goal to have a comparison of the results obtained from two different representation and use the result to con-
clude which representation is better suited for detection of periodic action patterns in videos.

**Gray-scale Representation.** Traditionally, an image of size $U \times V$ is represented as a $U \times V$ matrix with the intensities of the pixels as the matrix values. Most likely, color information is not needed to detect repetitive actions. Hence the color channels may be mapped to gray-scale values to represent each pixel with a single value. These intensities can be normalized to values in the range of $[0, 1]$ where 0 indicates black and 1 indicates white colors. This matrix can be flattened into an 1-dimensional array. Although it loses locality properties, it is easier to process them for finding similarities since the relative ordering of values is maintained across all frames. Since original videos may have different sizes, we resize each image to the same fixed size of resolution of $224 \times 224$. The advantage of this approach is that using the actual pixel values retain significant information from the image.

**Deep Neural Features.** Although gray-scale representation might be an accurate representation, it will generate a feature vector of size $U \times V$ for an image of $U \times V$. Such a representation actually maintains the complete frame. Alternatively, images may be encoded into feature vectors using either image processing methods or deep learning techniques. With the immense success of the deep neural networks to extract features from images into vectors in the past decade, we utilize deep neural features. This also enables us to evaluate whether deep neural features are suitable for finding repetitive actions or not. In our research, we make use of features that are computed by a deep convolutional neural network, specifically the last max-pooling layer output of the VGG16 convnet. VGG is a class of convolutional network archi-
tecture developed by Visual Geometry Group at the University of Oxford to win the ImageNet Challenge 2014. VGG-16 has 16 layers that include convolutional layers, max-pooling layers, activation layers, and fully connected layers as seen in Figure 3.3.

![Figure 3.3: The architecture of VGG-16.](image)

The input to the VGG-16 layer is of size $224 \times 224 \times 3$. Thus, we resize our input video frames to this size for feature extraction. The frame features are extracted from the output of the last max pooling layer, just before passing them to the fully connected layers. The function of the fully connected layers is to utilize the features input and process them for image classification tasks. However, we are simply interested in the features, thus, we skip those layers.

VGG-16 network has been found to be very accurate when it comes to image classification. It is trained on the ImageNet dataset which has over 14 million images and 1000 classes. Each of these images is quality controlled and human-annotated. As such, the database is very reliable and has been used extensively for computer vision
experiments. The VGG-16 network achieves 92.7% top-5 accuracy on the ImageNet dataset. Evidently, the network should be able to extract the most distinguishing features of an image. The VGG-16 network that we use is pre-trained on the ImageNet dataset.

The importance of using this kind of features is two-fold. First, a neural network pre-trained on a large database of images (ImageNet) captures the essential or more important/distinguishing features of each frame. Second, such a representation of a frame largely reduces the space needed for storage and consequently the processing time of the subsequent algorithms. Instead of 50176 * 3 (for each of the RGB color channels) features for a frame of size, the features extracted by VGG16 consists of only 25088 features for a single frame.

From the extracted features, a video $V$ can be represented as an array of length $N$ (number of frames) and each frame is represented as a $L$-dimensional vector (feature length) as follows:

$$V = [F_1, ..., F_N]$$
$$F_k = [f_1, ..., f_L]; k = 1 \ldots N$$

where $F_k$ is the feature vector of frame $k$ and $f_i$ is the $i^{th}$ element of a frame feature vector.

3.2 Extracting Time Series Data from Video

Our research uses time series analysis on the given video to reveal the repetitive actions. Up till this point, a video is represented as a time series of feature vectors as
$V = [F_1, F_2, \ldots, F_N]$, where each $F_i$ is a vector. Each frame in the video is represented as a feature vector. Our next goal is to generate 1D time-series from the video where each frame (or frame difference) is represented with a single value. 1D time-series is easier to analyse, enables us to write simpler and efficient algorithms and allows us to adopt several time-series algorithms from fields such as the music industry which already deals with several problems involving 1D time-series.

More formally, a video $V = [F_1, F_2, \ldots, F_N]$ is converted to a time series $T = [T_1, T_2, \ldots, T_{N-1}]$ such that each $T_i$ is a real number. Furthermore, it is desirable that the information regarding repetitive actions in the video $V$ is preserved during the transformation. In other words, if an action $A = [F_i, \ldots, F_j]$ in the video $V$ where $i \geq 0$ and $j \leq N$ is repetitive, then the subseries $S = [T_i, \ldots, T_j]$ of the time series $T$ are expected to have a similar repetitive pattern.

### 3.2.1 Distance Matrix

The first step of converting a video into a 1D time-series is to extract the information about periodic patterns from the feature vectors. We have found that a simple way to reveal the periodic patterns in a video is using the distance matrix computed from these features. A distance matrix is a symmetric matrix where each value in the matrix represents a pairwise difference between two corresponding vectors. In our case, each value in the distance matrix represents the pairwise distance between two corresponding frames. The distance is computed based on the frame features. Distance matrix or similarity matrix can be used to identify the repetitive patterns in
videos. There are already methods developed to process distance/similarity matrix for video analysis [18] [23] [13] [17] [14].

Mathematically, for a video \( V = [F_1, F_2, \ldots, F_N] \) with \( N \) frames, a distance matrix \( M \) is defined as a square matrix of size \( N \times N \) where the entry \( M_{ij} \) quantifies the distance between frames \( F_i \) and \( F_j \) with a real number. Since the distance metric is symmetric, so is the matrix \( M \). It follows that all the main diagonal elements of the distance matrix are 0, since the distance of a frame to itself is 0.

\[
M_{ij} = \text{distance}(F_i, F_j)
\]

In our experiments, we make use of the \textbf{cosine distance} as the distance metric. The cosine distance for two vectors \( F_i \) and \( F_j \) is the complement of cosine similarity of the two vectors and is defined as:

\[
\text{cosine\_distance}(F_i, F_j) = 1 - \text{cosine\_similarity}(F_i, F_j)
\]

\[
= 1 - \cos(F_i, F_j)
\]

\[
= 1 - \frac{\text{dot}(F_i, F_j)}{(\|F_i\|\|F_j\|)}
\]

The cosine similarity measures how similar the two vectors are based on the angle between them in the high-dimensional space. More accurately, it is the cosine of the angle between the two vectors in the high dimensional space. Since all values of the feature vector are non-negative, the range of cosine distance becomes \([0..1]\): 0 means there is no correlation and 1 means they are oriented in the same direction.
Observation on various videos and their distance matrix pairs shows that if there are some temporally repetitive patterns present in the video, then the matrix exhibits particular structures. Figure 3.4 shows a distance matrix generated from a video using the cosine distance metric, where the repetitive structures are clearly visible. The distance matrix is displayed as a heat map where a pure blue color indicates the most similar frames (distance = 0) and a pure red color indicates the most dissimilar frames (distance = 1).

![Distance matrix with repetitive patterns shown](image)

**Figure 3.4:** Distance matrix with repetitive patterns shown

It is apparent that if an action $A = [F_i, \ldots, F_j]$ is repetitive in a video $V$, then a square of repetitive patterns is formed in the square submatrix $Q[i : j; i : j]$ of the distance matrix $M$. 

29
Detecting those patterns from the distance matrix could give insights on the actual periodic patterns in the video. However, recognition of patterns in 2D image is found to be more difficult than in 1D, complicating an already difficult problem. We propose a way to construct a 1D time-series that reflects these periodic patterns and helps us detect the regions of periodic patterns in a simple way.

### 3.2.2 Preprocessing the Distance Matrix

Distance matrix depends on the features representing the frames and distance metric used to compute distances between frames. Hence, distance matrices vary for the same dataset. In this work, we use the same distance metric. However, we evaluate two types of features: gray-scale intensity and deep neural features. Distance matrices obtained from the two types of feature vectors (grayscale intensities and deep neural features) are shown in Figure 3.5.

**Figure 3.5:** (a) distance matrix generated from gray-scale pixel intensities (b) distance matrix generated from deep convolutional features

---

30
For pixel intensities, we can see that the boundaries of regions and patterns are clearer than the distance matrix for deep neural features. In this video, there are three regions to be analyzed for repetitive actions that can be recognized by square regions with patterns along the diagonal. The difference between distance features is visible in the bottom-right square region with patterns. The first top-left blue square in Figure 3.5 in the bottom-right square (a) is divided into two parts in Figure 3.5 (b). This may complicate whether to consider that portion as a single action or two actions. We should note that VGG-16 is designed to recognize objects. So it has properties such as translation invariance. However, translation variance could be important for detecting frame differences. To overcome this problem, we apply a low-pass filter to the distance matrix generated using deep neural features.

Gaussian blurring is the process of applying a gaussian filter function, which is a low pass filter, to the image. Gaussian blur smooths the input image by reducing noise and the details. Mathematically, applying a Gaussian blur to an image corresponds to convolving the image with a Gaussian function. In the frequency domain, Gaussian blur reduces the image’s high-frequency components. Figure 3.6 shows the distance matrix after applying Gaussian filter to the distance matrix in Figure 3.5 (b).

### 3.2.3 Creating Time-series from Distance Matrix

To construct a time-series from the distance matrix, while preserving the information about its periodic structures, we propose the following method. Consider each row of the distance matrix as a feature vector. Each row currently represents a
frame in the original video and the goal is to get a single real number to represent it from this vector. We calculate the distance of each row, except the last row, to its succeeding row and obtain a series of \( N - 1 \) values from the \( N \times N \) distance matrix. These \( N - 1 \) values form the 1D time-series for analyzing repetitive actions. Mathematically, the time series \( T = [T_1, T_2, \ldots, T_{N-1}] \) is obtained from the distance matrix \( M \) using the following equation:

\[
T_i = \text{distance}(\text{Row}(M, i), \text{Row}(M, i + 1)); 1 \leq i < N
\]

where \( \text{Row}(M, k) \) is the vector formed from the values of the \( k_{th} \) row of matrix \( M \).

We use the Euclidean distance to compute frame differences. The Euclidean distance is the straight-line distance between two points in Euclidean space. In general, for an \( n \)-dimensional space, the distance between vectors for \( p \) and \( q \) is computed
as:

\[ d(p, q) = d(q, p) = \sqrt{\sum (q_i - p_i)^2}; \text{ where } i \in [1, n] \]

Figure 3.7: Sample time-series data generated from a distance matrix

An example of the time-series graph with the repeating pattern is shown in Figure 3.7. This time series graph is obtained using the distance matrix provided in Figure 3.5 (a). The corresponding video has three candidate repetitive periodic segments. In the time-series data, the heights of the peak values for each of the periodic segments are almost similar, and one periodic segment is separated from another periodic segment by huge peaks. It is noticeable how the repeating patterns in the distance matrix also appear in the time-series in the form of small consecutive segments consisting of similar transitions. At the boundary of each subaction, there is a large difference between the two frames and this can be seen as large peaks in the graph of our time-series data. If the subaction is repeating itself, the consecutive
peaks in such a region are similar in height since the transitions that occur between similar sub-actions are also similar.

It may be suggested that time-series data can be generated based on original features of frames. We should not that consecutive frames in a video are very similar in general. Our experiments have shown that such method does give a time-series with some patterns but those patterns are relatively more complex and harder to detect in the 1D time-series. Creating the time-series using the distance matrix as described in this section gives clearer distinction between repetitive action segments.

3.3 Detecting Sub-actions and Repetitions from the Time-series Data

We analyze the heights of peaks for finding repetitive actions after detecting the peaks in the time-series data. The similarity of heights in the consecutive peaks of any segment in the time-series indicate a candidate repetitive action and the number of such peaks gives an indication of the number of sub-actions present.

We should note that a repetitive action is not necessarily repetition of a single pattern. We are going to denote the repeated patterns in a repetitive action as a sub-action. If two types of sub-actions always come together and are then repeating over a period of time then we observe a group of two peaks of different values repeating a number of times. For the sake of terminology, we call the first kind of repetition to have “one sub-action” and the second kind of repetition to have “two sub-actions”. Obviously, we can have more than two sub-actions.

For example, consider a video sequence where a person bends his body all the way down and back and then keeps repeating themself. The person starts from
an intermediate position (default), then bends their body forward going all the way down and then returning to the default position followed by throwing their arms all the way back and finally, gain, returning to the default position (Figure 3.8). In this case, the number of subactions is two:

- **First sub-action**: going from default position to all the way down and back to the default position

- **Second sub-action**: going from default position to all the way up and back to the default position

![Sample sub-actions in a video](image)

*Figure 3.8*: Sample sub-actions in a video. First row: bending down and up; second row: stretching back and forth. The last frame in the first sub-action is shown as the first-frame in the second sub-action to show the relationship between them

From the time-series data, all the peaks (local maximas) are extracted. Looping through these peaks, two consecutive peaks are compared for their height difference. If the height difference is below a certain empirically determined threshold, the two peaks are taken as a part of a same repetitive action. However, if the difference
is higher, it is taken as a boundary between two actions. This helps separate the
distinct repetitive sections in the video. This method detects the repetitive actions
with a single sub-action. For periodic actions with multiple (n) sub-actions, instead
of two consecutive peaks, we compare the height difference between the pair of a peak
and the \( n^{th} \) peak after it, and this is done for each of the peaks.

Formally, for a time-series \( T = [T_1, T_2, \ldots, T_{N-1}] \), the algorithm to detect the
repetitive actions with up to ‘S’ sub-actions is as follows.

1. Find the list \( P \) of peaks in the time-series \( T \), indicating the points in \( T \) where
there are local maxima.

\[
P_k = \{ i \mid 0 < i < N - 1, T_i > T_{i+1}, T_i > T_{i-1} \}
\]

\( P_k > P_{k-1} \) for \( k > 0 \)

2. Calculate the height differences between each pair of consecutive peaks in \( P \)
and store it in a list \( DH \).

\[
DH_k = T_i - T_j \text{ where } i = P_k, j = P_{k+1} \text{ for } 1 \leq k < |P|
\]

3. Let \( R = \{ \} \) be the set of candidate repetitive actions in the video \( V \) which is
set to empty, initially. Process each height difference in \( DH \) as follows:

(a) Let \( k = 0 \). \( k \) is an index to iterate on the \( DH \) list (height differences).
(b) If \( k \geq |DH| \), then skip the following part and go to step 4. Else create a candidate repetitive action \( C = \langle \text{start} :, \text{end} :, \text{subactions} = ? \rangle \)

(c) If \( |DH_k - DH_{k+1}| < \epsilon \), then this is part of a potential repeated sub-action with 1 sub-action. Set \( C.\text{subactions} = 1 \).

i. Set \( C.\text{start} = P_k \), the frame index where the first peak of this repetitive segment occurs.

ii. Increment \( k \) by 1 while \( |DH_k - DH_{k+1}| < \epsilon \).

iii. Set \( C.\text{end} = P_{k+1} \), the frame index where the last peak of this repetitive segment occurs.

iv. Add \( C \) to the set \( R \), increment \( k \) by 1 and go to step (b).

(d) If \( |DH_k - DH_{k+2}| < \epsilon \) and \( |DH_{k+1} - DH_{k+3}| < \epsilon \), then this is part of a potential repetitive action with 2 sub-actions. Set \( C.\text{subactions} = 2 \).

i. Set \( C.\text{start} = P_k \), the frame index where the first peak of this repetitive segment occurs.

ii. Increment \( k \) by 1 while \( |DH_k - DH_{k+2}| < \epsilon \) and \( |DH_{k+1} - DH_{k+3}| < \epsilon \).

iii. Set \( C.\text{end} = P_{k+1} \), the frame index where the last peak of this repetitive segment occurs.

iv. Add \( C \) to the set \( R \), increment \( k \) by 1 and go to step (b).

(e) Similarly, for \( j = 3 \) to \( S \):

Define a condition \( X = |DH_k - DH_{k+j}| < E \) and \( |DH_{k+1} - DH_{k+j+1}| < \epsilon \) and ... and \( |DH_{k+j-1} - DH_{k+2j-1}| < \epsilon \) for a given \( k \). If \( X \) is satisfied
then this is part of a potential repetitive action with $j$ sub-actions. Set $C.subactions = j$.

i. Set $C.start = P_k$, the frame index where the first peak of this repetitive segment occurs.

ii. Increment $k$ by 1 while $X$ satisfies.

iii. Set $C.end = P_{k+1}$, the frame index where the last peak of this repetitive segment occurs.

iv. Add $C$ to the set $R$, increment $k$ by 1 and go to step (b).

(f) If this step is reached, no sub-action starts from this $k$, increment $k$ by 1 and repeat from b.

4. The algorithm has scanned all the peaks and their height differences, with $R$ containing all the candidate repetitive actions. The algorithm ends here.

Figure 3.9 shows the detection of three candidate repetitive actions using this approach. The detected segments are highlighted in the distance matrix to visualize together with the periodic structures. It can be seen that our algorithm is able to detect the repetitive pattern in the matrix.

Our algorithm detects candidate repetitive actions in videos. It is possible to obtain a false positive as shown in Figure 3.10. A simple pruning algorithm can help filter out these false positives to improve results. The pruning algorithm works on the basis of the following idea. A candidate segment $C$ in the set $R$ is a true positive if the time-series corresponding to the sub-matrix $Q$ in the distance matrix $D$ contains a good number of sharp transitions. The algorithm is expressed as follows:
Figure 3.9: Detection of the three periodic segments (red, magenta, black colored squares in the diagonal) by our approach

1. For each candidate repetitive action $C$ in $R$:

   (a) Take subseries $TQ[TC.start, \ldots, TC.end]$ of the time-series $T$.

   (b) Smooth out the time-series using the Gaussian filter, whose value of variance is determined experimentally.

   \[
   TQ' = \text{Gaussian}(TQ, \text{mean} = 0, \text{variance} = \sigma^2)\]
Figure 3.10: The first periodic segment (black highlight) detected by our algorithm is a false positive.

(c) If the smoothed out time-series contain less than a few sharp changes, mark the candidate segment as a false positive and discard it.

\[ peaks_k = \{i|0 < i < \|TQ\| - 1, TQ_i > TQ_{i+1}, TQ_i > TQ_{i-1}\} \]

\[ peaks_k > peaks_{k-1}; \text{ for } k > 0 \]

\[ \text{if } |peaks| < \text{threshold} : R = R - \{C\} \]

In our experiments, we set the threshold for the number of peaks to be 2.

An example of the submatrix for a false positive candidate and the corresponding smoothed out time-series is shown in Figure 3.11.
The first candidate periodic segment obtained from the distance matrix in Figure 3.10

(a) The first candidate periodic segment obtained from the distance matrix in Figure 3.10

(b) Its time-series after smoothing using gaussian filter. Note that there are no peaks in this graph, thus this segment is discarded as a false positive.

Figure 3.11: Pruning a candidate repetitive action due to the presence of few peaks.

3.4 Detecting the Number of Repetitions

At the end of the pruning algorithm, each candidate $C$ in the set of repeated actions $R$ has a *subactions* parameter indicating the number of subactions in each repetition segment. This number however is found to be less accurate. While the previous algorithm serves well in detecting the repetitive actions, its performance for finding the number of repetitive actions was not satisfactory. Therefore, we present another approach to calculate the number of subactions for each repetitive action $C$ in the set $R$.

The algorithm is based on a popular time-series analysis utility called autocorrelation. The fundamental idea behind this approach is that if the period of a series is $T$, then when the series is lagged by $T$ frames and correlated with the original
series, we will get a higher value than when the series is lagged by any other number of frames. Formally, autocorrelation of a signal is the perceived similarity between the signal and a delayed version of itself as a function of delay. Autocorrelation is a popular mathematical tool for signal analysis. It has been extensively used to find repeating patterns, such as detecting periodicity of a signal, finding the fundamental frequency, etc.

The autocorrelation of any series \( X = [x_1, x_2, \ldots, x_n] \) is a series \( Y \) calculated using Pearson correlation formula as follows:

\[
Y_i = E[(X - \mu(X))(X^i - \mu(X^i))]/(\sigma(X)\sigma(X^i))
\]

where \( E \) is the expectation, \( \mu \) is the mean, \( \sigma \) is the standard deviation and \( X^i \) is the lagged version of \( X \) by \( i \) units, defined as:

\[
X^i_j = X_{j+i} \text{ if } j \geq i \text{ and } 0 \text{ otherwise}; 0 \leq j \leq |X|
\]

We present the following algorithm based on using autocorrelation to find the period:

1. For each repetitive action \( A \) in \( R \):
   
   (a) Take subseries \( TQ = [T_{A.start}, \ldots, T_{A.end}] \) of the time-series \( T \).
(b) Calculate the autocorrelation of the subseries $TQ$ for overlapping content after lagging (or shifting):

$$AC_i = Correlate(TQ, TQ^i), \text{ where } i \in [0, \frac{|TQ|}{2}]$$

where $TQ^i$ is the shifted version of $TQ$ by $i$ units defined as:

$$TQ^i_j = TQ_{j+i} \text{ if } j \geq i \text{ and } 0 \text{ otherwise } ; 0 \leq j \leq |TQ|$$

(c) Discard the first $M$ values of the auto-correlation sequence $AC$, because sub-actions have at least some duration and repeated at these intervals.

$$AC' = \{AC_{M+1}, AC_{M+2}, \ldots, AC_{|AC|}\} \text{ where } M < |AC| \text{ is determined empirically and set to 10 in our experiments.}$$

(d) Get the index corresponding to the highest value in the auto-correlation sequence $AC'$. This index corresponds to the periodicity of the repetition or its time since the beginning of the repetitive action.

$$period = \arg \max_i (AC')$$

(e) Find the number of repetitions by dividing the total frames in the repetitive action by its period.

$$repetition = \frac{|AC|}{period}$$
3.5 Summary

In this chapter, we explained our methodology to locate repetitive actions and their count. Our main idea is that we extract time-series data based on differences of rows in the distance matrix. The similarity of repeating peaks in time-series data is analyzed for determining repetitive actions. False positives are eliminated based on scarcity of peak points. Then, the count of sub-actions are found using autocorrelation for the segment of the repetitive action. The overall algorithm for detecting periodic action patterns in a video and calculating the number of repetitions in each periodic segment can be summarized in the following steps:

1. Represent the video of $N$ frames as a sequence of $N$ feature vectors. This step involves feature extraction from the video.

2. Calculate the distance between each pair of $N$ vectors obtained from step 1 to construct its distance matrix. The repetitive pattern in the distance matrix indicates repetitive actions. The distance matrix could be processed further to obtain better results in the following steps.

3. Treat each row of the distance matrix as the feature vector of the frame. Calculate the distance between the consecutive rows (or frames) in the distance matrix and obtain a time series data of dimension $(N - 1)$. The repeating actions are expected to appear in this time series data as a repeating pattern.
4. Find candidate segments of repetitive actions by detecting the repeating patterns in the time series data, indicated by consecutive peaks of similar heights in the time-series graph.

5. Filter out the non-periodic action segments from the list of candidate segments obtained in step 4.

6. For each of the periodic segments, find the period of repetition and repetition count by employing autocorrelation.
CHAPTER 4

EXPERIMENTS

In this chapter, we explain the experimental results to evaluate our method. Firstly, we briefly go over the MHAD202-v dataset [1] used in the experiments. After presenting a brief description of the ground-truth data, we briefly provide the quantitative measures to compare with the P-MUCOS method [18]. Finally, the results of experiments are provided. All experiments were performed on a 1.8GHz Intel Core i7-8550U (quad-core, 8MB cache, up to 4.0GHz) CPU.

4.1 MHAD202-v dataset

MHAD202-v dataset [18] consists of the 202 video sequences from the 101 pairs of the MHAD101-v dataset used in [23]. It is derived from the Berkeley Multimodal Human Action Database (MHAD). The Berkeley MHAD contains 11 actions performed by 7 male and 5 female subjects in the range 23 to 30 years of age except for one elderly subject. All subjects performed 5 repetitions of each action, yielding about 660 action sequences which correspond to about 82 minutes of total recording time. [1]
The MHAD101-v dataset video frames are of resolution 640x480 and each pixel of a frame is represented with three channels to represent color. The dataset can be divided into four groups based on the number of common actions. 1) **1-common-action.** In 50 paired sequences, each sequence consists of 3 concatenated action clips and the paired sequences have exactly 1 action in common. 2) **2-common-actions.** In 17 pairs, each sequence consists of 3-7 actions and they have 2 actions in common. 3) **3-common-actions.** There are 17 other pairs containing 3-7 actions with 3 actions in common. 4) **4-common-actions.** Finally, there are 17 pairs more having 4-7 actions with 4 actions in common. The MHAD101-v dataset was created to detect common actions in two video sequences [23]. However, the same dataset is also evaluated for detecting periodic actions in [18] by treating each sequence independently. Hence, we use this same dataset in our experiments, since this allows a direct comparison with the method provided in [18]. Figure 4.1 shows sample snapshots of actions appearing in the Berkeley MHAD data set.

**Figure 4.1:** Sample frames of 11 actions from the MHAD datasets. [1]
4.2 Ground-truth Data

The ground truth of the MHAD202-v dataset was provided by the authors of [18]. The ground-truth data represents a video as a binary vector where each value indicates if the frame belongs to a repetitive action or not. If a frame is part of a repetitive action, its corresponding value is 1; otherwise, it is 0. The length of a binary vector is equal to the number of frames in the video and may vary across videos.

We provide a sample representation of the ground-truth data as follows. Suppose in a video with 12 frames, the first 4 frames and the final 3 frames contain repetitions. Then its ground-truth is represented as

\[ [1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1] \]

4.3 Performance Metrics

For the assessment of the performance evaluation of our methods, the standard metrics of precision (P), recall (R), F1-score and overlap O (intersection-over-union) are used as in [18]. In the context of our experiments, these measures are defined as follows.

**Precision** is the ratio of the number of detected frames appearing in a repetitive action to the number of frames that our algorithm detected as a part of a repetitive action.

\[
Precision = \frac{TruePositives}{TruePositives + FalsePositives}
\]
True Positive (TP) indicates that our algorithm detected a frame belonging to a repetitive action correctly. On the other hand, False Positive (FP) means that our algorithm found the frame as a part of a repetitive action, but it is not.

**Recall** measure returns the ratio of the number of detected frames belonging to a repetitive action to the number of all frames appearing in repetitive actions.

\[
Recall = \frac{TruePositives}{TruePositives + FalseNegatives}
\]

False Negative (FN) indicates that a frame belongs to a repetitive action, but our algorithm missed it. This is especially likely to occur for frames appearing at the beginning or end of a repetitive action.

**Overlap** considers possible gaps in detecting frames belonging to a repetitive action and is defined as the intersection divided by union. In terms of TPs, FNs, and FPs, it is computed as follows.

\[
Overlap = \frac{TruePositives}{TruePositives + FalsePositives + FalseNegatives}
\]

**F1-score** is the geometric mean of the precision and recall as a single measure.

\[
F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}
\]

These measures are computed for each video. When providing the final result, the average of each measure from all videos is reported.
4.4 Results

We compare our results with the method called P-MUCOS [18]. P-MUCOS also detects periodic segments in an unsupervised manner. Compared with P-MUCOS, our method also uses the concept of distance matrix to obtain the results. In P-MUCOS, the distance matrix is calculated using the IDT (improved dense trajectories) of frames. In our approach, the distance matrix is calculated using a) grayscale pixel values of the frames, and b) frames features extracted by VGG16 conv-net. However, the main distinction lies in the fact that P-MUCOS uses the distance matrix directly to find the repetitive patterns by identifying the diagonals parallel to the principal diagonal by framing it as a graph search problem. On the other hand, in our approach, the distance matrix is used rather as a preliminary process - the data from the distance matrix is converted into a time-series and that time-series is used as the main input data for the periodic action detection algorithm.

The published results from their experiments with the 202 video sequences from the MHAD202-v datasets have been compared against the results from our approach in the same dataset. Table 4.1 summarizes the results obtained on detecting the periodic segments by our approach on the MHAD202-v video dataset.

<table>
<thead>
<tr>
<th></th>
<th>Recall (R)</th>
<th>Precision (P)</th>
<th>F1 Score (F)</th>
<th>Overlap (O)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-MUCOS</td>
<td>88.2</td>
<td>95.2</td>
<td>90.0</td>
<td>84.2</td>
</tr>
<tr>
<td>Our Approach (pixel intensities)</td>
<td>89</td>
<td>94.9</td>
<td>91.4</td>
<td>84.9</td>
</tr>
<tr>
<td>Our Approach (deep features)</td>
<td>89.2</td>
<td>94.8</td>
<td>91.5</td>
<td>86.9</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of detecting frames appearing in repetitive actions
The scores are presented as average percentage scores computed over all individual scores per video of a dataset. Our recall is 1% better than the recall of P-MUCOS while the precision values are close to each other. Especially, the overlap value for deep neural features is significantly better than the overlap of P-MUCOS. Our method has achieved overall measure of 0.869 whereas P-MUCOS has only obtained 0.842. These results show that our method performs slightly better than the method presented in [18].

Furthermore, our method can also predict the number of times a periodic action is repeated. We should note that no ground-truth data is provided for the counts of repeating actions in the MHAD202-v dataset. To generate the ground-truth data, we picked 20 random videos from the MHAD202-v dataset and counted how many times a periodic action is repeated in each periodic segment. The error and accuracy of our method are provided in Table 4.2. For each video, there are multiple periodic segments detected by our algorithm and the repetition count for each periodic segment is estimated. In these selected videos, the repetition count is 5. Error is computed as

\[
\frac{|predicted\_count - actual\_count|}{actual\_count}
\]

Table 4.3 provides the statistics per video and the number of repetitions that is found by our method. Note that, the accuracy provided in the table is also affected by the accuracy of the algorithm that detects periodic segment. If the repetitive action is not detected properly, then evidently, its number of repetitions will not be
Table 4.2: Performance for detecting the number of periodic actions in a repetitive action.

<table>
<thead>
<tr>
<th></th>
<th>Mean Absolute Error (%)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Approach</td>
<td>15.8</td>
<td>84.2</td>
</tr>
</tbody>
</table>

computed properly. For fairness, we also considered only the repetitive actions that are accurately detected by the algorithm and calculated the error of repetition count on only those segments. In that case, the error rate of our algorithm was found to be 12%. This error is a better indicator of the autocorrelation method used for counting the repetition of subactions.
<table>
<thead>
<tr>
<th>Test Video</th>
<th>Detected Repetitive Segment</th>
<th>Period</th>
<th>Repetitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video 1</td>
<td>Segment 1</td>
<td>t</td>
<td>5</td>
</tr>
<tr>
<td>Video 1</td>
<td>Segment 2</td>
<td>55</td>
<td>5</td>
</tr>
<tr>
<td>Video 1</td>
<td>Segment 3</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>Video 2</td>
<td>Segment 1</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>Video 3</td>
<td>Segment 1</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>Video 3</td>
<td>Segment 2</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>Video 4</td>
<td>Segment 1</td>
<td>93</td>
<td>3</td>
</tr>
<tr>
<td>Video 4</td>
<td>Segment 2</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>Video 4</td>
<td>Segment 3</td>
<td>61</td>
<td>3</td>
</tr>
<tr>
<td>Video 4</td>
<td>Segment 4</td>
<td>58</td>
<td>5</td>
</tr>
<tr>
<td>Video 5</td>
<td>Segment 1</td>
<td>48</td>
<td>5</td>
</tr>
<tr>
<td>Video 5</td>
<td>Segment 2</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>Video 6</td>
<td>Segment 1</td>
<td>183</td>
<td>3</td>
</tr>
<tr>
<td>Video 6</td>
<td>Segment 2</td>
<td>61</td>
<td>3</td>
</tr>
<tr>
<td>Video 7</td>
<td>Segment 1</td>
<td>38</td>
<td>4</td>
</tr>
<tr>
<td>Video 7</td>
<td>Segment 2</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>Video 8</td>
<td>Segment 1</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>Video 8</td>
<td>Segment 2</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>Video 9</td>
<td>Segment 1</td>
<td>69</td>
<td>5</td>
</tr>
<tr>
<td>Video 9</td>
<td>Segment 2</td>
<td>79</td>
<td>3</td>
</tr>
<tr>
<td>Video 10</td>
<td>Segment 1</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>Video 10</td>
<td>Segment 2</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>Video 10</td>
<td>Segment 3</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>Video 11</td>
<td>Segment 1</td>
<td>64</td>
<td>5</td>
</tr>
<tr>
<td>Video 11</td>
<td>Segment 2</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>Video 12</td>
<td>Segment 1</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>Video 12</td>
<td>Segment 2</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>Video 13</td>
<td>Segment 1</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>Video 13</td>
<td>Segment 2</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>Video 13</td>
<td>Segment 3</td>
<td>91</td>
<td>3</td>
</tr>
<tr>
<td>Video 14</td>
<td>Segment 1</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Video 14</td>
<td>Segment 2</td>
<td>74</td>
<td>4</td>
</tr>
<tr>
<td>Video 14</td>
<td>Segment 3</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Video 14</td>
<td>Segment 4</td>
<td>48</td>
<td>5</td>
</tr>
<tr>
<td>Video 15</td>
<td>Segment 1</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>Video 15</td>
<td>Segment 2</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>Video 16</td>
<td>Segment 1</td>
<td>28</td>
<td>5</td>
</tr>
<tr>
<td>Video 16</td>
<td>Segment 2</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>Video 16</td>
<td>Segment 3</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>Video 17</td>
<td>Segment 1</td>
<td>48</td>
<td>5</td>
</tr>
<tr>
<td>Video 17</td>
<td>Segment 2</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>Video 18</td>
<td>Segment 1</td>
<td>39</td>
<td>5</td>
</tr>
<tr>
<td>Video 18</td>
<td>Segment 2</td>
<td>20</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 4.3: Statistics for videos having periodic actions and the performance of our method for each segment.

<table>
<thead>
<tr>
<th>Video 18</th>
<th>Segment 3</th>
<th>33</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video 19</td>
<td>Segment 1</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Video 19</td>
<td>Segment 2</td>
<td>74</td>
<td>4</td>
</tr>
<tr>
<td>Video 19</td>
<td>Segment 3</td>
<td>41</td>
<td>5</td>
</tr>
<tr>
<td>Video 19</td>
<td>Segment 4</td>
<td>38</td>
<td>4</td>
</tr>
</tbody>
</table>

4.5 Summary

These results show that our algorithm can effectively detect frames of repetitive actions. Our method achieved F1-score greater than 0.91 and precision around 0.95. Our method slightly performed better than P-MUCOS [18] when compared against the MHAD202-v dataset. Our method achieved around 84% accuracy while counting the actions that are repeated in a repetitive action.
CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we proposed a method that temporally localizes the periodic segments in a video by analyzing the time-series data obtained from the distance matrix of frames in a video. The distance matrix is generated based on i) the grayscale pixel values and ii) features learned from a deep learning architecture. Based on these two representations, a symmetric matrix that contains the pairwise distances between each pair of frames in the video is generated. Then, we generate the time-series data by taking the distances between each pair of consecutive rows of the distance matrix. Repetition patterns in the time series appear in the form of similar transition patterns. So clusters of frames that exhibit similarity in the transitions are segregated as the candidate periodic segments. However, in some cases, because of noise, non-periodic segments can also exhibit such similarity in transitions. To handle this artifact, at the final stage, a pruning algorithm discards the non-periodic segments from the set of the candidate periodic segments. Note that being stationary can still be considered as a periodic action although there is no motion in the segment. Therefore, it is necessary to determine whether there is any action in the segment. To achieve this, a gaussian filter with a high variance value is applied to smooth the time
series of the candidate segment. If the resulting graph does not exhibit repetitive
sharp changes after smoothing, it is discarded as being a non-repetitive segment (or a
stationary segment). For a periodic pattern, it is expected to have at least a certain
number of sharp peaks in the time series which cannot be completely flattened by
the smoothing filter. After eliminating these segments, the remaining segments are
considered to be actual repetitive segments of the video. Finally, for each segment
that has been recognized to contain repetitions, the periodicity of the repetition is
determined by using auto-correlation. Correlation values of the time series data for
the segment and a lagged version of the same data are calculated. The lag that
gives the highest auto-correlation value for the time-series is chosen as the period of
the repetitive action. Once the period is determined, the repetition count is simply
computed as the ratio of the total frames in the segment to the length of the period.

To validate our hypothesis, experimental evaluations were performed on the
MHAD202-v dataset and the results were compared with an existing method pro-
posed by Panagiotakis et al. [18]. Our method yields slightly better results on the
MHAD202-v dataset when compared against [18]. We have used two ways to repre-
sent the distance matrix based on features obtained frames: grayscale pixel intensities
and deep neural features. Although the periodic patterns can be observed in distance
matrices, there is a noticeable difference in the quality of these matrices to detect
periodic actions. Rather than generating a method sensitive to the distance matrix,
we have extracted time-series data from the distance matrix. Our experiments show
that time-series data extracted from both matrices performed well. Although the
quality of distance matrix for deep neural features is slightly lower than the quality
of distance matrix for gray-scale intensities, the results for deep neural features were slightly better than the results for gray-scale features. However, we should mention that we had to do some preprocessing to smooth out the distance matrix for deep neural features. The main trade off is that deep neural features take longer to calculate than just generating a flattened array of pixel intensities. The size of deep neural features is significantly smaller than the size of the gray-scale representation.

The results from the experiments confirmed that our method can be used to detect periodic action patterns in videos with F1-score above 0.91 and precision around 0.95. Additionally, the period and repetition count for each periodic segment were computed. In our experiments, the error for the count of repetitive actions was around 15.8% for videos having 5 sub-actions. This error also included the error due to the misdetection of repetitive actions.

Thus, to answer the research questions, time-series data obtained from the distance matrix created from frame features of a video can be used to detect periodic repetitive patterns as well as to count the number of repetitions in each of those periodic action patterns.

In this thesis, we have focused our experiments on human actions in this work. In human actions, the speed of the action is neither very fast nor is very slow which results in the perfectly spaced peaks in the time-series data that we obtained. However, in non-human repetitive actions, for example, motion of a fan or a wheel might be so speedy that the algorithm might be unable to determine the repetition at all. Thus, one future work could be testing and if needed, extending the algorithm further for non-human actions. Moreover, the proposed algorithm is focused solely
on the patterns exhibited by the time-series data obtained from the distance matrix. Although the patterns work great for repetitive actions, there can be cases where even some non-repetitive actions might yield such patterns. Another limitation of the research is that if a repetitive action consists of more than one subactions, but the subactions are very similar, the resultant peaks in the time-series data will be very similar too. In such a case, the algorithm will consider it as a repetitive action with a single subaction which is not accurate.

Moreover, when detecting the periodic action patterns, we consider only the difference in heights of the peaks in the time-series data. In the future, the algorithm can be extended to also consider the number of frames between the two peaks to get a better sense of the periodicity of the action.

Additionally, in this work, we have not analyzed the hierarchy of a repetitive action that could be composed of shorter repetitive actions. Identifying the hierarchy would be helpful to analyze patterns and understand what a repetitive action is composed of. In this thesis, we have developed methods to detect repetitive actions and to count how many times its periodic actions occurred in the video. Another future work is to classify repetitive actions and subactions. That would help to understand the video content better.

As future work, the accuracy of detecting repetitive action, its periodic subactions, and their count could be improved. We worked on the time-series data. Alternatively, the analysis could be performed on the frequency domain. If the Fourier transform is applied to the time series, the power spectrum may provide significant values at the fundamental frequencies of the signal, thus giving some insights on the
periodicity. In addition to the Fourier transform, wavelet transform could also be used for analyzing the time series.

Additionally, there are some parameters that were determined empirically. Moreover, there are some components of our method where supervised methods could be helpful. For example, the best threshold value can be predicted to cluster the periodic frames in the time series. Supervised learning might help to filter out false positives segments which are reported to be periodic but are not. Hence, machine learning techniques can be employed at various stages of the methodology to yield better results.
REFERENCES


[9] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining periodic behaviors for moving objects. In 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1099–1108, 2010.


