Faculty of

Electronics and Communication Engineering

Gesture Recognition From video

B.sc project of Electronic and Telecommunication Engineering

Submitted by

Arifa Nisha

ID : 2010-2-55-008 and

Jihan Sultana

ID : 2010-1-55-007

Supervised by

Hafiz Imtiaz

Assistant professor

Department of EEE

Bangladesh University of Engineering and Technology

RESEARCH PAPER ON

Gesture Recognition From Video

By

Arifa Nisha

ID: 2010-2-55-008

and

Jihan Sultana

ID: 2010-1-55-007

A thesis submitted in partial fulfillment of the

requirements for the degree of

Bachelor of science in the field of

Electronics and Telecommunication Engineering

Supervised by:

Mr.Hafiz Imtiaz

at the

East West University

2014

Dedicated to my All teachers, Family and Friends

Letter of Approval

Title of thesis : Gesture Recognition From Video

Submitted by : Arifa Nisha

ID : 2010-2-55-008

and

Jihan Sultana

ID : 2010-1-55-007

The above thesis,submitted in partial fulfillment of the requirements for the degree of Bachelor of Electronics and Telecommunication Engineering ,East West University has been accepted.

Superviser : Hafiz Imtiaz

Signature :

Date : 3-5-2014

Declaration

I hereby declare that this thesis, submitted to East West University as a partial fulfillment of the requirements for the degree of Bachelors of Electronics and Telecommunication Engineering, has not been submitted as an exercise for a degree at any other university. I also certify that the work described here is entirely my own, except for quotations and summaries whose sources have been appropriately cited in the references.

This thesis may be made available within the university library and may be photocopied or loaned to other libraries for the purpose of consultation.

Arifa Nisha:

.....ArifaNisha.....

and

Jihan Sultana:

Jehan

.....

Date:

Acknowledgement

At first I would like to express my sincere gratitude to Almighty GOD for successfully completing my project paper.

I am very much fortunate to work with my honorable supervisor Mr.Hafiz Imtiaz for his valuable assistance and insights leading to the writing of this paper and also very thankful to him for his patience and understanding during this eight months of effort that went into the production of this paper.

I am also grateful to my parents and family for their support and blessings.Another special thanks to my faculty members of ECE department,my friends and batch-mates for their endless support and inspiration to work on this thesis topic.

Thank you all for supporting me in every way.

Signature

Arifa Nisha Jehan

Abstract

Recognizing Human Actions from video is a challenging problem which has received much attention during the recent years due to its many applications in different fields. A reliable system capable of recognizing various human actions has many important applications such as human-computer interaction,content-based video retrieval,visual surveillance,analysis of sports events and more. The problem of human action recognition from video sequence was addressed in this project. The aim is to develop an algorithm which can recognize low-level-actions from the input video sequences. For the purpose of evaluation we introduce a new video database of such low- level five human actions (walking,running,jogging,boxing and hand clapping)performed by 25 people in four different dimentions.The presented results of action recognition justify the proposed method and demonstrate its advantage compared to other relative approaches for action recognition.

TABLE OF CONTENTS

Chapter 1

1 Introduction

A huge number of videos (e.g., BBC[1] and Youtube[2]) are available online today and the number is rapidly growing. Human actions constitute one of the most important parts in movies, TV shows, and consumer-generated videos. Analysis of human actions in videos is considered a very important problem in computer vision. The term "action" refers to a simple motion pattern as performed by a single subject, and in general lasts only for a short period of time, namely, just a few seconds. Action is often distinguished from activity in the sense that action is an individual atomic unit of activity. In particular, human action refers to physical body motion. Recognizing human actions from video is a very challenging problem due to the fact that physical body motion can look very different depending on the context. For instance, similar actions with different clothes or in different illumination and background can result in a large appearance variation, or the same action performed by two different people may look quite dissimilar in many ways.

In this paper,each action is represented by one or several video sequences,where the actors who performs one or more times a characteristic motion.

1.1 Problem Specification

Recognition is generally divided into two parts: category classification and detection/localization. The goal of action classification is to classify a given action query into one of several prespecified categories (for instance, five categories from the KTH action data set [1]: boxing, hand clapping, jogging, running, and walking). Meanwhile, action detection is meant to separate an action of interest from the background in a target video (for instance, spatiotemporal localization of a walking person). This paper tackles both action detection and category classification problems simultaneously by searching for an action of interest within other "target" videos with only a single "query" video. We focus on a sophisticated feature representation with an efficient and reliable similarity measure, which also allows us to avoid the difficult problem of explicit motion estimation.

In general, the target video may contain actions similar to the query, but these will

typically appear in completely different context.Examples of such differences can range from rather simple optical or geometric differences (such as different clothes, lighting, action speed, scale, and view changes) to more complex inherent structural differences rather than a real human action.Also this action database contains 2391 sequences. All sequences were taken over homogeneous backgrounds with a static camera with *25*fps frame rate. The sequences were downsampled to the spatial resolution of *160x120* pixels and have a length of four seconds in average.

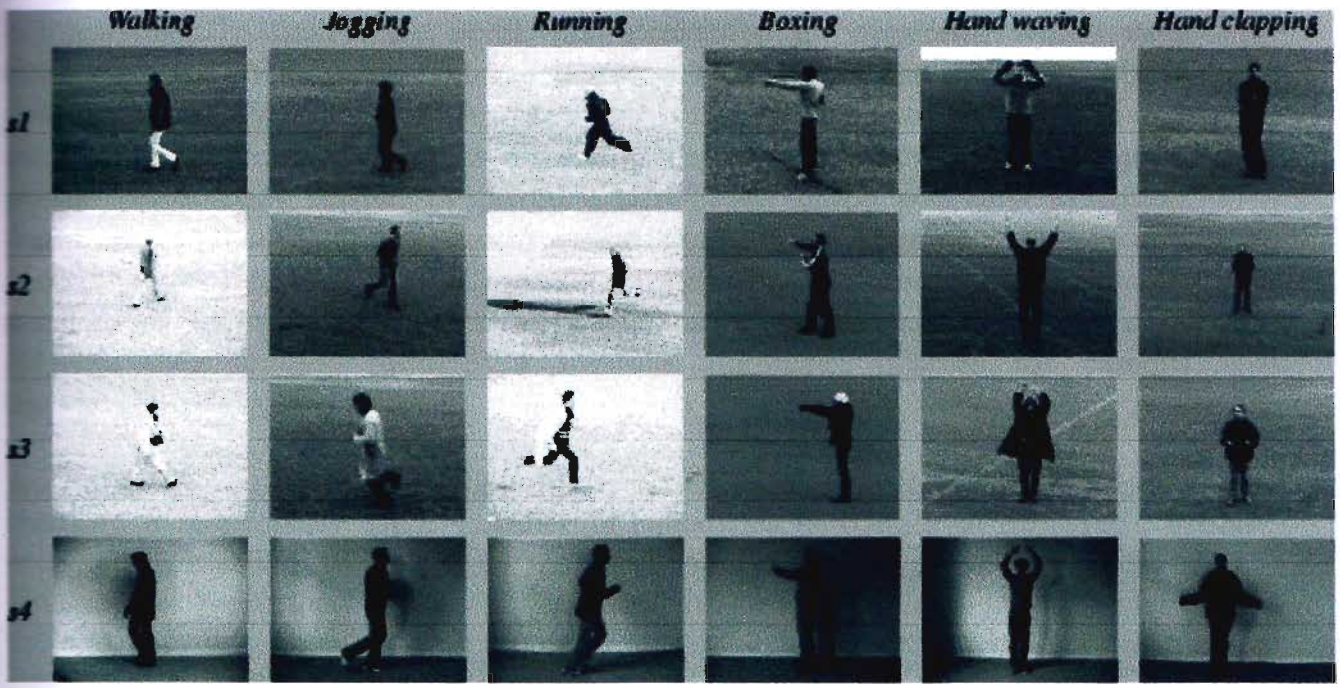

Fig 1 : Action database (available on request): examples of sequences corresponding to different types of actions and scenarios

All sequences are stored using AVI file format and are available on-line (DIVX-compressed version). Uncompressed version is available on demand. There are *25x5x4=500* video files for each combination of 25 subjects, 5 actions and 4 scenarios. Each file contains about four subsequences used as a *sequence* in our experiments. The subdivision of each file into sequences in terms *start_frame* and *end_frame* as well as the list of all sequences is given in.

1.2 Related Work

.Over the past two decades, many studies have attempted to tackle this problem and made impressive progress.Approaches can be catagorized on the basis of various action representations.This project is related to 2D features KTH database system. Shechtman and Irani [22] employed a 3D correlation scheme for action detection. They focused on subvolume matching in order to find similar motion between the two space-time volumes, which can be computationally heavy. Ke et al. [23] presented an approach which uses boosting on 3D Haar-type features inspired by similar features in 2D object detection [32]. While these features are very efficient to compute, many examples are required to train an action detector in order to achieve good performance. They further proposed a part-based shape and flow matching framework [33] and showed good action detection performance in crowded videos. Recently, Kim and Cipolla [24] generalized canonical correlation analysis to tensors and showed very good accuracy on the KTH action data set, but their method requires a manual alignment process for camera motion compensation.

As opposed to 2D object recognition, which has recently proven capable of learning a respectably large number of categories (a couple of hundred), action recognition is still only limited to about a dozen categories at best (6 for the KTH, 10 for the Weizmann, and 12 for the Hollywood2 action data sets). Even though learning-based action recognition methods appear to be practical in a small number of categories, they have not yet proven to be scalable with a larger number of categories.[4]

Due to the advent of large database-driven nonparametric approaches [34], [35], [36], instead of training sophisticated parametric models, we can reduce the inference problem to matching a query to an existing set of annotated databases, posing a video-to-video matching problem. As a successful example, Boiman et al. [30] showed that a rather simple NN-based image classifier in the space of the local image descriptors is efficient and even outperforms the leading learning-based image classifiers, such as SVM-KNN [37] and pyramid match kernel [38].

Methods such as those in [33], [22], [25], [39], and [40], which aim at recognizing actions based solely on one query,are very useful for applications such as video

retrieval from the Web (e.g., viewdle[5] and videosurf[6]). In these methods, a single query video is provided by users and every gallery video in the database is compared with the given query.

2. Proposed Method

In this paper,our contributions to the action recognition task are mainly based on several methods.Here we represent a new video database of six human actions performed by 25 people in four different dimensions.First,we use Motion JPEG software for coverting these videos in to images.Secondly.we use MATLAB for representing images using its functions.There are many functions of Motion JPEG and MATLAB which are used in this project to get desire results.

2.1 Motion JPEG

Motion JPEG(M-JPEG or MJPWG) is a video format in which each video frame or interlaced field of a digital video sequence is compressed separately as a JPEG image.Originally developed for multimedia PC applications,M-JPEG is now used by video-capture devices such as digital cameras,IP cameras and webcams and by non-linear video editing systems.It continues to enjoy native support by the Quick Time Player,the PLAY Station console and browsers such as Safari,Google Chrome and Mozilla Firefox.

2.1.1 History

MJPEG was first used by the QuickTime Player in the mid 1990s.

2.1.2 Applications

Software and devices using the M-JPEG standard include web browsers, media players, game consoles, digital cameras, IP cameras, webcams, streaming servers, video cameras, and non-linear video editors.

2.1.3 Video capture and Editing

M-JPEG is frequently used in non-linear video editing systems.Modern desktop CPUs are powerful enough to work with high definition video so no special hardware is required and they in turn offer native random -access to a frame,M-JPEG support is also wide spread in video-capture and editing equipment.

2.1.4 Digital video

Digital video (DV)adopts a similar method by compressing video frames individually.

2.1.5 Encoding

M-JPEG is an intraframe-only compression scheme (compared with the more computationally intensive technique of interframe prediction. Whereas modern interframe video formats, such as MPEG1,MPEG2 and H.264/MPEG-4 AVC, achieve real-world compression-ratios of 1:50 or better, M-JPEG's lack of interframe prediction limits its efficiency to 1:20 or lower, depending on the tolerance to spatial artifacting in the compressed output. Because frames are compressed independently of one another, M-JPEG imposes lower processing and memory requirements on hardware devices.

As a purely intraframe compression scheme, the image-quality of M-JPEG is directly a function of each video frame's static (spatial) complexity. Frames with large smooth-transitions or monotone surfaces compress well, and are more likely to hold their original detail with few visible compression artifacts. Frames exhibiting complex textures, fine curves and lines (such as writing on a newspaper) are prone to exhibit DCT-artifacts such as ringing,smudging and macroblocking. M-JPEG compressed-video is also insensitive to motion-complexity, i.e. variation over time. It is neither hindered by highly random motion (such as the surface-water turbulence in a large waterfall), nor helped by the absence of motion (such as static landscape shot by tripod), which are two opposite extremes commonly used to test interframe video-formats.

For Quick Time formats, Apple has defined two types of coding: MJPEG-A and MJPEG-B. MJPEG-B no longer retains valid JPEG Interchange Files within it, hence it is not possible to take a frame into a JPEG file without slightly modifying the headers.

2.2 MATLAB

2.2.1 Introduction

MATLAB is a high-performance language for technical computing created by *The*

MathWorks in 1984. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. [22]

2.2.2 History of MATLAB

MATLAB was invented in the late 1970s by Cleve Moler, then chairman of the computer science department at the University of New Mexico. He designed it to give his student's access to LINPACK and EISPACK without having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community.

Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded The MathWorks in 1984 to continue its development. These rewritten libraries were known as JACKPAC.

MATLAB was first adopted by control design engineers, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved with image processing.

2.2.3 What is MATLAB

MATLAB is a numerical computing environment and programming language. It allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. Although it specializes in numerical computing, an optional toolbox interfaces with the Maple symbolic engine, allows it to be part of a full computer algebra system. Besides dealing with explicit matrices in linear algebra, it can handle differential equations, polynomials, signal processing, and other applications. Results can be made available both numerically and as excellent graphics. [22]

MATLAB solves many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

The name MATLAB stands for *Matrix Laboratory*. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB features a family of add-on application-specific solutions called *Toolboxes*. Toolboxes allow learning and applying specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

As of 2004, MATLAB was reported to be used by more than one million people in industry and academia.

2.2.4 The MATLAB system

The MATLAB system consists of the following five parts:

1) Desktop Tools and Development Environment 2) MATLAB Mathematical Function Library
3) MATLAB Language 4) MATLAB Graphics
5) MATLAB External Interfaces/API

2.2.5 Desktop tools and Development environment

This is the set of tools and facilities that help to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, a code analyzer and other reports, and browsers for viewing help, the workspace, files, and the search path.

2.2.6 MATLAB Mathematical Function Library

The MATLAB mathematical function library is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

2.2.7 MATLAB Language

The MATLAB language is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both programming in the small to rapidly create quick and dirty throw-away programs, and programming in the large to create large and complex application programs.

2.2.8 MATLAB Graphics

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

2.2.9 MATLAB External Interfaces/API

The MATLAB External Interface is a library that allows writing C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from

MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

2.2.10 Conclusion

The reason for using MATLAB for the design and development of this project is its toolboxes and superior processing capabilities. The MATLAB Neural Network toolbox and Image Processing toolbox played an important role for the development of this project.

3. Applied Functions of MATLAB

3.1 " imread " Function

Read image from graphics files

Syntax

```
A = imread(filename,fmt)
[X,map] = imread(filename,fmt)
[...] = imread(filename)
[...] = imread(URL,...)
[...] = imread(...,idx)    (CUR, ICO, and TIFF only)
[...] = imread(...,'frames',idx) (GIF only)
[...] = imread(...,ref)    (HDF only)
[...] = imread(...,'BackgroundColor',BG) (PNG only)

[A,map,alpha] = imread(...) (ICO, CUR, and PNG only)
```

Description

The `imread` function supports four general syntaxes, described below. The `imread` function also supports several other format-specific syntaxes. See Special Case Syntax for information about these syntaxes.

`A = imread(filename,fmt)` reads a grayscale or truecolor image named `filename` into `A`. If the file contains a grayscale intensity image, `A` is a two-dimensional array. If the file contains a truecolor (RGB) image, `A` is a three-dimensional (`m`-by-`n`-by-3) array.

`filename` is a string that specifies the name of the graphics file, and *`fmt`* is a string that specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname of the location on your system. If `imread` cannot find a file named `filename`, it looks for a file named `filename.fmt`. See Formats for a list of all the possible values for *`fmt`*.

`[X,map] = imread(filename,fmt)` reads the indexed image in `filename` into `X` and its associated colormap into `map`. The colormap values are rescaled to the range [0,1].

`[...] = imread(filename)` attempts to infer the format of the file from its content.

`[...] = imread(URL,...)` reads the image from an Internet URL. The URL must include the protocol type (e.g., `http://`).

3.2 "imshow" Function

Display image

Syntax

`imshow(I)` example

- `imshow(I,RI)` example
- `imshow(X,map)` example
- `imshow(X,RX,map)`
- `imshow(filename)` example
- `imshow(___,Name,Value...)`
- `imshow(gpuarrayIM,___)` example
- `imshow(I,[low high])` example
- `himage = imshow(___)`

Description

example

`imshow(I)` displays the image `I` in a Handle Graphics® figure, where `I` is a grayscale, RGB (truecolor), or binary image. For binary images, `imshow` displays pixels with the value `0` (zero) as black and `1` as white.

example

`imshow(I,RI)` displays the image `I` with associated 2-D spatial referencing object `RI`.

example

`imshow(X,map)` displays the indexed image `X` with the colormap `map`. A color map matrix may have any number of rows, but it must have exactly 3 columns. Each row is interpreted as a color, with the first element specifying the intensity of red light, the second green, and the third blue. Color intensity can be specified on the interval 0.0 to 1.0.

`imshow(X,RX,map)` displays the indexed image `X` with associated 2-D spatial referencing object `RX` and colormap `MAP`.

example

`imshow(filename)` displays the image stored in the graphics file `filename`. The file must be in the current directory or on the MATLAB® path and must contain an image that can be read by `imread` or `dicomread`. `imshow` calls `imread` or `dicomread` to read the image from the file, but does not store the image data in the MATLAB workspace. If the file contains multiple images, `imshow` displays only the first one.

`imshow(___,Name,Value...)` displays the image, specifying additional options with one or more `Name,Value` pair arguments, using any of the previous syntaxes.

example

`imshow(gpuarrayIM,___)` displays the image contained in a `gpuArray`. This syntax requires the Parallel Computing Toolbox™.

example

`imshow(I,[low high])` displays the grayscale image `I`, specifying the display range as a two-element vector, `[low high]`. For more information, see the`DisplayRange` parameter.

`himage = imshow(___)` returns the handle to the image object created by `imshow`.

3.3 "fft2" Function

2-D fast Fourier transform

Syntax

```
Y = fft2(X)
Y = fft2(X,m,n)
```

Description

`Y = fft2(X)` returns the two-dimensional discrete Fourier transform (DFT) of `X`. The DFT is computed with a fast Fourier transform (FFT) algorithm. The result, `Y`, is the same size as `X`.

If the dimensionality of `X` is greater than `2`, the `fft2` function returns the 2-D DFT for each higher dimensional slice of `X`. For example, if `size(X) = [100 100 3]`, then `fft2` computes the DFT of `X(:,:,1)`, `X(:,:,2)` and `X(:,:,3)`.

`Y = fft2(X,m,n)` truncates `X`, or pads `X` with zeros to create an `m`-by-`n` array before doing the transform. The result is `m`-by-`n`.

3.4 "abs" Function

Absolute value and complex magnitude

Syntax

```
Y = abs(X)
```

Description

`abs(X)` returns the absolute value, $|X|$, for each element of `X`.
If `X` is complex, `abs(X)` returns the complex modulus (magnitude):

```
abs(X) = sqrt(real(X).^2 + imag(X).^2)
```

3.5 "sort" Function

Sort array elements

Syntax

- `B = sort(A)` example
- `B = sort(A,dim)` example
- `B = sort(___,mode)` example
- `[B,I] = sort(___)` example

Description

example

`B = sort(A)` sorts the elements of `A` in ascending order along the first array dimension whose size does not equal 1. For strings, this is a sort in ASCII dictionary order. The sort is case-sensitive; uppercase letters appear in the output before lowercase.

- If `A` is a vector, then `sort(A)` sorts the vector elements.
- If `A` is a nonempty, nonvector matrix, then `sort(A)` treats the columns of `A` as vectors and sorts each column.
- If `A` is an empty 0-by-0 matrix, then `sort(A)` returns an empty 0-by-0 matrix.
- If `A` is a multidimensional array, then `sort(A)` treats as vectors all values along the first array dimension whose size does not equal 1, and then sorts each vector.

example

`B = sort(A,dim)` sorts the elements of `A` along dimension `dim`. For example, if `A` is a matrix, then `sort(A,2)` sorts the elements in each row.

example

`B = sort(___,mode)` sorts in the order specified by `mode` using any of the above syntaxes. The single string, `'ascend'`, indicates ascending order (default) and `'descend'` indicates descending order.

example

`[B,I] = sort(___)` also returns a collection of index vectors in an array, `I`, using any of the above syntaxes. `I` is the same size as `A` and describes the rearrangement of the elements along the sorted dimension. In general, if `B` is an array with *N* dimensions whose sizes do not equal 1, then *N-1* loops are required to use the indexing array, `I`. This requirement is because `I` contains only location information for the single dimension being sorted. For example,

- If you sort a numeric vector or cell array of strings, then `B = A(I)`.
- If you sort the columns of a matrix , then the above relation holds for each column independently. That is, each column of `I` is a permutation vector of the corresponding column of `A` such that

```
for j = 1:size(A,2)
    B(:,j) = A(I(:,j),j);
end
```

3.6 "find" Function

Find indices and values of nonzero elements

Syntax

- `indices = find(X)`
- `indices = find(X, k)`
- `indices = find(X, k, 'first')`
- `indices = find(X, k, 'last')`
- `[i,j] = find(...)`

`[i,j,v] = find(...)`

Description

`indices = find(X)` returns the linear indices corresponding to the nonzero entries of the array `X`. If none are found, `find` returns an empty, 0-by-1 matrix. In general, `find(X)` regards `X` as `X(:)`, which is the long column vector formed by concatenating the columns of `X`.

`indices = find(X, k)` or `indices = find(X, k, 'first')` returns at most the first `k` indices corresponding to the nonzero entries of `X`. `k` must be a positive integer, but it can be of any numeric data type.

`indices = find(X, k, 'last')` returns at most the last `k` indices corresponding to the nonzero entries of `X`.

`[i,j] = find(...)` returns the row and column indices of the nonzero entries in the matrix `X`. This syntax is especially useful when working with sparse matrices. If `X` is an N-dimensional array with N > 2, `j` contains linear indices for the dimensions of `X` other than the first.

`[i,j,v] = find(...)` returns a column vector `v` of the nonzero entries in `X`, as well as row and column indices.

4. Results

To quantify the performance of our algorithm, we also conducted experiments on the KTH data set. The KTH action data set contains five types of human actions (boxing, hand clapping, walking, jogging, and running), performed repeatedly by 25 peoples in four different scenarios.This data set seems more challenging because there are large variations in human body shape, view angles, scales, and appearance. We also evaluate our method on the KTH data set under various split setup.We were able to achieve a recognition average rate of 81.4 percent on these five actions(walking,running,boxing,jogging and handclapping) of average confusion matrix across all scenarios for this setup. The recognition rate is provided in Table-1 as well.

Table-1

Action	Walking	Running	Boxing	Jogging	Handclapping
KTH Database	83%	79%	85%	80%	80%

Fig: Recognition Accuracy of human actions

In the KTH data sets, target videos contain only one type of action. However, a target video may contain multiple actions in practice. In this case, simple nearest neighbor classifiers can possibly fail. Therefore, we might benefit from contextual information to increase the accuracy of action recognition systems, as similarly done in [89]. In fact, there is broad agreement in the computer vision community about the valuable role that context plays in any image understanding task.

4.1 Performance Analysis

In this human actions recognition method,we use mainly a MATLAB and Motion JPEG Converter.By all this applications we finally find the recognition accuracy performance.For recognition accuracy we use KTH database system and find the desire resuts.

5. Conclusion

In this paper, we have proposed a novel action recognition algorithm by employing K T H D a t a b a s e s y s t e m . The proposed method can automatically detect in the target video the presence, the number, as well as the location of actions similar to the given query video. Multiscale implementation dealt with large variations in scale of actions and outperformed the single-scale version. In order to increase the detection accuracy and further deal with action classification, we employed an action cropping method based on MATLAB coding. Challenging sets of real-world human action experiments demonstrated that the proposed approach achieves high-recognition accuracy .

5.1 Future Detections

. Since the proposed method is designed with detection accuracy as a high priority, extension of the method to a large-scale data set requires a significant improvement of the computational complexity of the proposed method. Toward this end, we could benefit from an efficient searching method . These aspects of the work are the subject of ongoing research.

References

[1] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," Proc. IEEE Conf. Pattern Recognition, June 2004.

[2] T. Darrell and A. Pentland, "Classifying Hand Gestures with a View-Based Distributed Representation," Proc. Advances in Neural Information Processing Systems, vol. 6, pp. 945-952, 1993.

[3] J. Yamato, J. Ohya, and K. Ishii, "Recognizing Human Action in Time Sequential Image Using Hidden Markov Model," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1992.

[4] H. Jiang, M. Crew, and Z. Li, "Successive Convex Matching for Action Detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2006.

[5] T. Starner and A. Pentland, "Visual Recognition of American Sign Language Using Hidden Markov Model," Proc. Int'l Workshop Automatic Face and Gesture Recognition, 1995.

[6] C. Carlsson and J. Sullivan, "Action Recognition by Shape Matching to Key Frame," Proc. Workshop Models versus Examplars in Computer Vision, 2001.

[7] A. Yilmaz and M. Shah, "Actions Sketch: A Novel Action Representation," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2005.

[8] K. Cheung, S. Baker, and T. Kanade, "Shape-from-Silhouette of Articulated Objects and Its Use for Human Body Kinematics Estimation and Motion Capture," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2003.

[9] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 29, no. 12, pp. 2247-2253, Dec. 2007.

[10] A.F. Bobick and J.W. Davis, "The Recognition of Human Move-ment Using Temporal Templates," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 1257-1265, Mar. 2001.

[11] J. Little and J. Boyd, "Recognizing People by Their Gait: The Shape of Motion," J. Computer Vision Research, vol. 1, pp. 2-32, 1998.

[12] S. Ali and M. Shah, "Human Action Recognition in Videos Using Kinematic Features and Multiple Instance Learning," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, no. 2, pp. 288-303, Feb. 2010.

[13] Y. Yacoob and M. Black, "Parameterized Modeling and Recogni-tion of Activities," Computer Vision and Image Understanding, vol. 73, pp. 232-247, 1999.

[14] J. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words," Int'l J. Computer Vision, vol. 79, no. 3, pp. 299-318, Mar. 2008.

[15] J. Niebles and L. Fei-Fei, "A Hierarchical Models of Shape and Appearance for Human Action Classification," Proc. IEEE Conf. Computer Vision and Pattern Recognition, June 2007.

[16] Z. Laptev and T. Lindeberg, "Space-Time Interest Points," Proc. IEEE Int'l Conf. Computer Vision, Oct. 2003.

[17] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning Realistic Human Actions from Movies," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008.

[18] O. Boiman, E. Shechtman, and M. Irani, "In Defense of Nearest-Neighbor Based Image Classification," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008.

[19] C.H. Lampert, M.B. Blaschko, and T. Hofmann, "Beyond Sliding Windows: Object Localization by Efficient Subwindow Search," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008.

[20] P. Viola and M. Jones, "Robust Real-Time Object Detection," Int'l J. Computer Vision, vol. 57, no. 2, pp. 137-154, 2004.

[21] Y. Ke, R. Sukthankar, and M. Hebert, "Event Detection in Crowded Videos," Proc. IEEE Int'l Conf. Computer Vision, 2007.

[22] A. Torralba, R. Fergus, and W. Freeman, "80 Million Tiny Images: A Large Data Set for Non-Parametric Object and Scene Recogni-tion," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, no. 11, pp. 1958-1970, Nov. 2008.

[23] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman, "LabelMe: A Database and Web-Based Tool for Image Annota-tion," Int'l J. Computer Vision, vol. 77, nos. 1-3, pp. 157-173, 2008.

[24] J. Hays and A. Efros, "Scene Completion Using Millions of Photographs," Proc. ACM SIGGRAPH, 2007.

[25] H. Zhang, A. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Cate-gory Recognition," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2006.

[26] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Efficient Learning with Sets of Features," J. Machine Learning Research, vol. 8, pp. 725-760, 2007.

[27] C. Yeo, P. Ahammad, K. Ramchandran, and S.S. Satry, "High-Speed Action Recognition and Localization in Compressed Domain Videos," IEEE Trans. Circuits and Systems for Video Technology, vol. 18, no. 8, pp. 1006-1015, Aug. 2008.

[28] W. Yang, Y. Wang, and G. Mori, "Human Action Recognition from a Single Clip Per Action," Proc. Second Int'l Workshop Machine Learning for Vision-Based Motion Analysis, 2009.

[29] H.J. Seo and P. Milanfar, "Training-Free, Generic Object Detection Using Locally Adaptive Regression Kernels," IEEE Trans. Pattern Analysis and MachinIntelligence, vol. 32, no. 9, pp. 1688-1704, Sept. 2010.

[30] H.J. Seo and P. Milanfar, "Static and Space-Time Visual Saliency Detection by Self-Resemblance," J. Vision, vol. 9, no. 12, no. 15, pp. 1-27, 2009, http://journalofvision.org/9/12/15/ (doi:10.1167/9.12.15).

[31] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel Regression for Image Processing and Reconstruction," IEEE Trans. Image Proces-sing, vol. 16, no. 2, pp. 349-366, Feb. 2007.

[32] H. Takeda, S. Farsiu, and P. Milanfar, "Deblurring Using Regularized Locally-Adaptive Kernel Regression," IEEE Trans. Image Processing, vol. 17, no. 4, pp. 550-563, Apr. 2008.

[33] H. Takeda, P. Milanfar, M. Protter, and M. Elad, "Super-Resolution without Explicit Subpixel Motion Estimation," IEEE Trans. Image Processing, vol. 18, no. 9, pp. 1958-1975, Sept. 2009.

[34] Y. Fu, S. Yan, and T.S. Huang, "Correlation Metric for Generalized Feature Extraction," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, no. 12, pp. 2229-2235, Dec. 2008.

[35] Y. Fu and T.S. Huang, "Image Classification Using Correlation Tensor Analysis," IEEE Trans. Image Processing, vol. 17, no. 2, pp. 226-234, Feb. 2008.

[36] C. Liu, "The Bayes Decision Rule Induced Similarity Measures," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1086-1090, June 2007.

[37] D. Lin, S. Yan, and X. Tang, "Comparative Study: Face Recognition on Unspecific Persons Using Linear Subspace Methods," Proc. IEEE Int'l Conf. Image Processing, 2005.

[38] Y. Ma, S. Lao, E. Takikawa, and M. Kawade, "Discriminant Analysis in Correlation Similarity Measure Space," Proc. IEEE Int'l Conf. Machine Learning, 2007.

[39] J.W. Schneider and P. Borlund, "Matrix Comparison, Part 1: Motivation and Important Issues for Measuring the Resemblance between Proximity Measures or Ordination Results," J. Am. Soc. for Information Science and Technology, vol. 58, no. 11, pp. 1586-1595, 2007.

[40] P. Ahlgren, B. Jarneving, and R. Rousseau, "Requirements for a Cocitation Similarity Measure, with Special Reference to Pearson's Correlation Coefficient," J. Am. Soc. for Information Science and Technology, vol. 54, no. 6, pp. 550-560, 2003.

[41] J. Rodgers and W. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," Am. Statistician, vol. 42, no. 1, pp. 59-66, 1988.

[42] https://www.google.com.bd/#q=human+action+recognition+in+videos+using+kinematic+features+and+multiple+instance+learning

[43] https://www.google.com.bd/#q=action+recognition+for+one+sample