Feature extraction in Videos

ABSTRACT

The vast amount of data delivered from a video stream provides the need to identify the contents in a video. A higher level of content abstraction and investigation in the distinct region is required to identify the video content. A video can contain several point of interest in every frame. We detect and extract the features such as color, key points and motion vectors in every frame of a video. To do so, we use the libraries such as SIFT and FFmpeg to obtain the dataset. These Algorithms specifically address how to find the most significant features and describe them with a mathematical feature vector that can be used to recognize or find matches of the same feature from multiple objects. Further we divide every frame into individual cells and classify the obtained features. These "feature description" extracted from the training object can then be used to identify the object when attempting to locate the object in a test video containing many other objects.

Group Members:

Anoop Jatavallabha Vijayakumar

Deepak Soundararajan

Narendra Kumar Sampath Kumar

Ravikiran Tangirala

Santosh Mandya Jayaram

Keywords:

Video features, Feature Extraction, Videos, Motion vectors, SIFT vectors, Histogram.

Introduction:

Data redundancy is one of the biggest challenge in Multimedia data management. When the input data to an algorithm is too large to be processed and it is suspected to be redundant, it directly has an impact on the reliability of the end product. To overcome the data redundancy, the input data can be transformed into reduced set of feature vectors (process called as Feature Selection) [1]. Feature Selection largely relies on the purpose of the application. For example, sports video summarization usually focuses more on the motion of foreground athletes and balls, while generic-purpose applications may concern more about the background scenes [2]. In most cases, Motion and color are two primary cues to measure the scene change and summarize the video sequence. This project focuses on extracting the color histogram, SIFT vectors and motion vectors for the given set of video files.

Terminology:

Feature:

A Feature is a piece of information which is relevant for solving the computational task related to a certain application.

Feature vectors:

When two or more different features are extracted, resulting in two or more feature descriptors at each image point. The organized collection of the information provided by two or more feature descriptors as the elements of one single vector is called Feature vector

Feature Detection:

Feature detection refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not

Feature Extraction:

Feature Extraction is a process of extracting features from every frame in a video, independently of past and future frames.

Histogram:

A histogram is a graphical representation of the distribution of numerical data. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are usually equal size.

Motion Vectors:

A motion vector is the key element in the motion estimation process. It is used to represent a macroblock in a picture based on the position of this macroblock (or a similar one) in another picture, called the reference picture.

SIFT (Scale Invariant-feature transform):

Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images

Frames:

A frame is an electronically coded still image in video.

Goal Description:

- Create an *n*-bin color histogram for every frame in the set of videos. The frame may be further divided into cells
- Identify the Interesting points on the object using SIFT algorithm. The points are categorized into cells for every frame in a set of videos.
- Extract the motion vectors for a given set of videos. Both forward and backward motion estimation are required.

Assumptions:

- Valid directory path is passed as an command argument.
- The cell values are taken in the matrix format

  (0,0) - first cell; (0,1) - second cell and so on

- Output file name should be provided with appropriate .txt extension at run time
- nBin for Histogram should not exceed 255.
- The input ‘r’ values are taken to be considerably small r=3,4,7. For ‘r’ values greater than 7 would require more computation time.
- All videos are in same directory
- In motion vectors, the macro blocks that are out of the frames’ resolution are considered as objects that are coming in or moving out (for both forward or backward estimation). These vectors are placed in their respective cells.
- For Histogram, after dividing the cells if there are residue pixels it is taken as cells and printed along row and columns, even if the residue lies in one cell.

Description of proposed Solution:

Task1:

Algorithm:

Get input videos’ directory, r value, histogram size ‘n’, output file

For every I, j (i denotes video , j denotes frame)

If frame can be divided into cells without remainder

    do not consider the remainder

Else

    Consider the remainder as new cell

For every cell l in frame j

    Get the hist values and append it to output file

END

Implementation:

The algorithm considers two cases. One, if the frame's row/column is divisible by given 'r' value then the frame is partitioned perfectly. Second, if the frame's row/column is not divisible by the given 'r' it considers the remainder rows/columns as separate cells and computes the Histogram values

The algorithm is implemented in Matlab IDE. The output is given of the format <VideoFile; Frame; Cell; Histogram values>. The following libraries are used for the required purpose.

v = VideoReader(filename) creates object v to read video data from the file named filename.

X = ones(n) returns an n-by-n matrix of ones.

C = mat2cell(A,dim1,Dist,dimNDist) divides array A into smaller arrays within cell array C

r = rem(a,b) returns the remainder after division of a by b

N = histcounts(X) partitions the X values into bins, and returns the count in each bin

dlmwrite(filename,M,'-append') appends the data to the end of the existing file, filename.

Task 2:

Prerequisite:

Download SIFT library and compile the Sift_compile.m. Install mex compiler for windows to compile supported cpp files.

Algorithm:

Get input for videos,r,output_file

For every i,j (i denotes video , j denotes frame)

    Extract the SIFT vectors for Fj

    For every cell in the matrix

        For Every SIFT Vector

            If the SIFT X,Y lie inside the cell

                PRINT the SIFT vectors

            Else

                Do Nothing

Implementation:

The algorithm computes the SIFT vectors for every frame in a video (having all the video files in loop) and extracts each and every SIFT vector frame values into a variable which in the form of <X,Y,Scale,Orientation> and descriptor values into a variable which in the form of <0..128>. Then based on the X,Y values in the frame variable for each SIFT vectors it decides which cell to put into.

The Algorithm is implemented in Matlab IDE. The output is given in the format <VideoFile;Frame,Cell,SIFT frame, SIFT Descriptor>. The following libraries are used for the required purpose

v = VideoReader(filename) creates object v to read video data from the file named filename.

[f,d] = sift(grayimage) calls the external SIFT library to compute SIFT vectors, f - frame, d- descriptor

B = repmat(A,n) returns an array containing n copies of A in the row and column dimensions

Correctness:

Since SIFT vectors for all frames in all videos should be constant every time when the program is run. The Output always contains same set of SIFT vectors but aligned to different cells based on the input value 'r'.

Task 3:

Prerequisite:

Set up a project in visual studio including all the FFmpeg libraries and the Dlls attached.

Algorithm:

Get the inputs for directory, resolution, Output_filename

Demux/Decode the Video stream using Open_codec_context method

For Every Video

    For every frame in the video

        Read frame from the file

        Call Decode_packet

    Flush the cached Frame

End

Decode_packet Method

IF frame exists

    Get the AVFrameSideData

    Get the AVMotionVector from the AVFrameSideData

    For Every Motion Vector

For Every cells in the matrix

IF the Motion vector DESTx,DESTy is within the cell range

Print the Motion Vector in their cell

Else IF the vectors are out of scope

Print the Motion Vector in the Last row/column based on its axis

Exit;

Implementation:

The motion vector extraction is an open source code [9]. On top of the existing code, we obtain the motion vectors for all the frames in all the videos. For every Motion vector, the destinationX and destinationY values are compared to the cell range and if it falls within the cell range, the values are printed.

One important corner case, for both Forward prediction and backward prediction (Bi-directional prediction). Some vectors are placed out of the frame's resolution. Those vectors are taken into consideration as it might be an incoming/outgoing object. These vectors are also placed in their respective cells.

Correctness:

Since the motion vectors should be constant every time when the given set of videos is run. The Output always contains the same set of motion vectors with different cells aligned based on input r value.

User Interface specification:

Open Issues:

For different videos having the motion vectors outside the frame's column is not resolved. Note: The motion vectors lying outside the frame's row has been resolved.

Display description:

The output file can be found in the directory which the user gave as input while running the program.

System requirements/installation:

Matlab 2016

Visual Studio 2015 or any equivalent IDE

For Task 3, the instructions to set up the project is attached in this report.

Related Works:

The development of image matching by using a set of local interest points can be traced back to the work of Moravec (1981) on stereo matching using a corner detector. The ground-breaking work of Schmid and Mohr (1997) showed that invariant local feature matching could be extended to general image recognition problems in which a feature was matched against a large database of images. Earlier work by the author (Lowe, 1999) extended the local feature approach to achieve scale invariance. This work also described a new local descriptor that provided more distinctive features while being less sensitive to local image distortions such as 3D viewpoint change [3] The author provides a more in-depth development and analysis of this earlier work, while also presenting a number of improvements in stability and feature invariance. He proposes a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. The author also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm. This approach has been named the Scale Invariant Feature Transform (SIFT), as it transforms image data into scale-invariant coordinates relative to local features [4]. But recently, the focus of research is less on the measurement of image or camera motion and more on the labeling of the action taking place in the scene. This shift has been triggered not only by the availability of the computational resources, but also the interest in applications, such as wireless interfaces and interactive environments. The most primitive level, however, is movementa motion whose execution is consistent and easily characterized by a definite space-time trajectory in some feature space. [5].

Applications such as surveillance, video retrieval and human-computer interaction require methods for recognizing human actions in various scenarios. Typical scenarios include scenes with cluttered, moving backgrounds, nonstationary camera, scale variations, individual variations in appearance and cloth of people, changes in light and view point and so forth. All of these conditions introduce challenging problems that have been addressed in computer vision in the past (see [6], [7] for a review). The Author here demonstrates that the action recognition can be achieved using local measurements in terms of spatiotemporal interest points (local features). Such features capture local motion events in video and can be adapted to the size, the frequency and the velocity of moving patterns, hence, resulting in video representations that are stable with respect to corresponding transformations. Support Vector Machines (SVMs) are state-of-the-art large margin classifiers which have recently gained popularity within visual pattern recognition. By combining local features with SVM, the author derived a novel method for motion recognition that gives high recognition performance compared to other relative approaches [8].

Conclusions:

We have extracted the video features such as color, key points and motion vectors and classified them based on the input cell size. Considered all the corner test case scenarios. For instance, the input cell size not being able to exactly divide the frame resolution. Finally, the output data is stored as a text file with suitable formats that can be used for further research.

Bibliography:

[1] - https://en.wikipedia.org/wiki/Feature_extraction

[2] - Recent Advances in Multimedia Signal Processing and Communications By Mislav Grgic, Kresimir Delac, Mohammed Ghanbari , ISBN 978-3-642-02900-4

[3] – “Distinctive Image Features from Scale-Invariant Keypoints” Lowe, D.G. International Journal of Computer Vision (2004) 60: 91. doi:10.1023/B:VISI.0000029664.99615.94

[4] - Towards Intelligent Engineering and Information Technology Imre J.Rudas, Janos Fodor, Janusz Kacprzyk ISBN 978-3-642-03737-5.

[5] - A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 257-267, Mar 2001. doi: 10.1109/34.910878

[6] - J. Aggarwal and Q. Cai, "Human Motion Analysis: A Review", Computer Vision and Image Understanding, vol. 73, no. 3, pp. 428-440, 1999.

[7] - T. Moeslund and E. Granum, "A survey of computer vision-based human motion capture", CVIU, vol. 81, no. 3, pp. 231-268, 2001.

[8] - C. Schuldt, I. Laptev and B. Caputo, "Recognizing human actions: a local SVM approach," Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, 2004, pp. 32-36 Vol.3.doi: 10.1109/ICPR.2004.1334462

[9] - https://www.ffmpeg.org/doxygen/1.1/doc_2examples_2demuxing_8c-example.html