A Comparative Analysis of the Efficiency of Video Reader Object for Frame Extraction in MATLAB

Nachamai M, Joy Paulose, Samuel Marandi
Christ University
Bangalore, India
nachamai.m@christuniversity.in
joy.paulose@christuniversity.in
samuel.marandi@cs.christuniversity.in

ABSTRACT: *Frame extraction in video files is a challenging and arduous task. This process involves the extraction first from a video file and selecting the desired frame as the output. In this study we have provided two new approaches for the process and generated algorithm to ease the process. In the experimentation we found that both the approaches to read a video file using the video reader object in MATLAB which lead to the conclusion that the second approach performed much better in terms of time and space*.

Keywords: Frame Extraction, Video File Analysis, MATLAB

Received: 13 May 2017, Revised 17 June 2017, Accepted 23 June 2017

1. Introduction

Reading of video files is crucial to most applications in the field of computer vision. Extraction of a frame from a video generally involves reading the video and extracting all the frames in the video and then picking the desired frame as the output. For example, in MATLAB the reading of video files can be done using the computer vision toolbox which provides a video reader object to read a video. The *VideoReader* function is used with the *read* method to read the data from an input file. There are varying restrictions of the video formats available in MATLAB. The most common of these formats are *wmv*, *mp4*, *m4v*, *avi* and *mpg*[1]. The general syntax for creating a video reader object is given below:-

<*ObjectName*>=VideoReader(<*filename*>)

The above syntax will create a *VideoReader* object from the file specified in the filename. If it cannot construct the object for any reason, *VideoReader* will generate an error. Some of the properties returned by this object are defined in the table below.

BitsPerPixel	**Bits per pixel of the video data.**
Duration	Total length of the file in seconds.
FrameRate	Frame rate of the video in frames pers second.
Height	Height of the video frame in pixels.
Name	Name of the file associated with the object.
NumberOfFrames	Total number of frames in the video stream.
Path	String containing the full path to the file associated with the reader.
Tag	String that identifies the object.
Type	Class name of the object '*VideoReader*'
UserData	Generic field for data of any class that needs to be added.
VideoFormat	String indication MATLAB representation of the video format
Width	Width of the video frame in pixels.

Table 1. VideoReader Object Properties

The methods used with video reader object are defined in the table below.

Get	**Query property values for video reader object**
getFileFormats	File formats that *VideoReader* supports
Read	Read video frame from data
Set	Set property values for the video

Table 2. VideoReader Object Methods

The properties and methods are extensively used to make the complete analogy between the two methods in the next section.

2. Literature Review

Frame extraction is monotonous and arduous process when implemented manually. It can prove to be quite challenging and exhausting very quickly if there are multiple constraints from which a frame is defined uniquely from the other. As such various techniques have been developed and implemented to identify a frame and extract the same. In paper[2] Huang and Mehrotra discuss the importance of key frame extraction and evaluate the exiting methods for doing the same. To overcome the shortcomings of the existing approaches, a new algorithm for key frame extraction based on unsupervised clustering is introduced. The key frame extraction technique for the removal of duplicates from a video stream has been optimized by Thepade and Tonge in their work[3]. They have extracted key frames by computing the consecutive frame differences and eliminating the relative neighboring frames with near match as duplicate copies. Tabassum and Islam[4] talk about watermarking a video file and how a single frame from a group of identical frames can be extracted for digital watermarking to tackle copyright challenges. Using the concept of identical frames watermark embedding is applied in a video file to implement digital watermarking. Key frame extraction is implemented using an improved version of hierarchical clustering algorithm by Liu and Hao. The improved hierarchical clustering algorithm is used to obtain an initial clustering result. And K-means is conducted to optimize the initial clustering result and obtain the final clustering result[5]. In another work we see the implementation based on nonparametric clustering based on density estimation for key frame extraction[6]. Here key frame extraction is accomplished by density-estimation-based nonparametric clustering, the mean shift method, which can efficiently analyze complex multimodal feature space and delineate arbitrarily shaped clusters in it. An experiment conducted by Li et al., uses nonparametric motion feature for capturing a key frame[7]. A temporal K-means clustering algorithm for the key frame extraction has been used, which incorporates the sequential constraint into extracted key frames. Liu and Fan explore a fresh approach to key frame extraction and object segmentation from a video[8]. The authors have proposed a new approach to combine two video segmentation techniques together. Sun and Zhou on the other hand proposed an alternative method for key frame extraction based on mutual information and image entropy[9]. Experimental results show that the proposed approach is able to extract key frames adaptability and represent video content more effectively as presented in the paper. Chen et al., proposed a novel scheme to extract key frames based on kernel locality preserving learning, for the purpose of video shot summarizing. They have represented the key frame by the linear combination of several neighboring frames, corresponding to the center of the feature vectors in the higher dimensional space[10]. Yong et al., presents framework that produces semantic context features for image frame understanding and further employs a one-class classifier for key-frame extraction. The framework starts with image segmentation, followed by low-level feature extraction and classification of the image blocks extracted from image segments[11]. Yet in another approach we can see the frame extraction takes place based on the entropy of the images present in the video sequence. Using the color information in the video frames, the algorithm looks every frame of a shot as a special sample and selects appropriate feature[12]. The work carried out by Calic and Izuierdo introduces a real-time algorithm for scene change detection and key-frame extraction. This method generates frame difference metrics by analyzing the statistics of the macro-block features extracted from an MPEG compressed stream. The method is implemented using difference metrics in curve simplification by means of a discrete contour evolution algorithm[13]. Key-frame extraction based on adaptive threshold detection of multi-features was suggested by Huang et al.,[14]. A color histogram, edge histogram and a wavelet transform are used to describe visual content, and combine to form a frame difference measure of multi-feature integration. An innovative method for extracting key frame, based on optimized frame difference is mentioned in the work concluded by Meng and Liu. It measures the similarity of two adjacent frames' in terms of the information of frame difference, and extracts key frames after optimizing the frame difference[15]. Honghua et al., experimented with a method, which is based on multi-scale-phase-based local features in their work. Their work features retrieval of key frames from a surveillance video and the characteristics of user attention focus[16]. Wu and Xu carried out a research work by extracting the most representative frame from each group as a key frame. They have grouped the video frames in accordance with the correlation of the visual content by clustering and have extracted the most representative frame from each group as a key frame[17].

3. Analysis and Results

This analogy for frame extraction compares two methods over the same video file and benchmarks the time and space required for the execution of both approaches. The specifications of the video used in this analysis are as mentioned in the table below.

Total Length	**11:38 seconds**
Width in Pixels	1920 pixels
Height in Pixels	1080 pixels
Frame Rate	25 fps
Data Rate	7996 kbps
Size	676 MB

Table 3. Video properties used in the analysis

The first approach implements the *VideoReader* object in the same generic format as specified and extracts all the frames that are present in the video into the buffer memory. The frames are then read from the buffer memory one at a time using the *read* method and are written to a directory. This results in all the frames being written to a secondary memory. If the number of frames in the video is n, irrespective of the length of the video, a total of n frames will be written into the memory. This can sometimes be a bottleneck in performance as often all the frames may not essentially be required since the differential coefficient in frames that are adjacent to each other will be close to a negligible value. The adjacent frames do not provide any prominent difference and any information retrieved from such a frame will simply be a replication of already obtained values. The extensive Input/Output (I/O) operations that are hence performed before the relevant frames are picked appear to vitiate the throughput of the system. A generic algorithm for the above mentioned approach is provided below for the purpose of illustration.

Algorithm: FrameExtraction

Input: A video file of supported format.

Output: Frames relevant for further processing.

1. Read the video using video reader object.

2. Retrieve the number of frames using the '*NumberOfFrames'* property of the *VideoReader* object.

3. For each frame incremented by 1 until the total number of frames is reached

   a. Read the current frame using the *read* method of *VideoReader* object

   b. Write the frame to secondary memory as image.

4. End for

5. Pick the relevant frames required for processing.

6. End FrameExtraction

The second approach on the other hand also uses the *VideoReader* object of the computer vision toolbox. Contrary to the first approach where all the frames are extracted on the system buffer, in this approach all the frames are stored into a *struct* array of size 'n', where 'n' is the total no of frames in the video. The interval for picking the frames is then taken from the user and is converted into an index. This index acts as a frame number for the frame to be picked from the *struct* array and writes the frame to the disk. The I/O activity of this approach is significantly reduced as opposed to the first approach where all the frames are written to the secondary storage whereas here, only the frames that are required are picked from the *struct* array. Since the number of frames to be written is significantly less, this approach also removes the overhead of using extensive disk space. A generic algorithm for the above mentioned approach is provided below for the purpose of illustration.

Algorithm: Frame_Extraction(*Interval*)

Input: A video file of supported format.

Output: Frames relevant for further processing.

1. Read the video using video reader object.

2. Convert the internal to frame number '*f*'

3. Initialize f to f_i.

4. Retrieve the number of frames using the '*NumberOfFrames'* property of the *VideoReader* object.

5. While frame f is less than total number of frames

   a. Read the frame f using the *read* method of *VideoReader* object

   b. Write the frame f to secondary memory as image.

   c. f = f + f_i

6. End while

7. End FrameExtraction

The analysis of both the approaches was made using the stopwatch timer available in MATLAB. The stopwatch timer uses two commands tic, toc which measures the total execution time in seconds for any given script. The tic command is called at the very beginning of the script. This starts the stopwatch timer and any command executed hereafter is timed by the stopwatch. The toc command is called at the end of the script which prints the time elapsed since the tic was used. The general syntax of these commands is given below:-

```
tic
        <any statements>
toc
t = toc
```

The statistical performance analyses of both the approaches are provided in the table below.

	Approach 1	Approach 2
Total Execution Time	28 *minutes*(approx.)	132 *seconds*
Total Disk Space	2.74 *GB*	1.77 *MB*

Table 4. Benchmark of both approaches

4. Conclusion

The detailed analysis of both the approaches to read a video file using the video reader object in MATLAB concludes that the second approach performs much better in terms of time and space. The latter approach performs 12 times better when it comes down to total execution time. The space and hardware resources required are almost negligible in comparison to the first approach.

The second approach also provides a new way of extracting a frame from a video file with respect to time series dilation. The analysis proves very insightful when a video stream of very large size has to be processed and when each and every frame in the video stream does not play a pivotal role in the outcome.

Acknowledgement

A Comparative Analysis of the Efficiency of Video Reader Object for Frame Extraction in MATLAB is part of major research project undertaken by Department of Computer Science, Christ University. We would like to thank the Center For Research, Christ University and the management for their incessant support and opportunity to work on the project. This work was carried out as part of the major research project titled 'Facilitating Feedback Reckoning of Teaching Session using Artificial Intelligence' fully supported and funded by Center For Research, Christ University under the Grant No: MRPDHS-1210.

References

[1] Read video files - MATLAB - MathWorks India. [Online]. Available: http://in.mathworks.com/help/matlab/ref/videoreader-class.html. [Accessed: 16-Mar-2015].

[2] Huang, T. S., Mehrotra, S. (1998). Adaptive key frame extraction using unsupervised clustering, *In*: Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269), 1, p. 866–870.

[3] Thepade, S. D., Tonge, A. A. (2014). An optimized key frame extraction for detection of near duplicates in content based video retrieval, *In*: 2014 International Conference on Communication and Signal Processing, p. 1087–1091.

[4] Tabassum, T., Islam, S. M. M. (2012). A digital video watermarking technique based on identical frame extraction in 3-Level DWT, *In*: 2012 15th International Conference on Computer and Information Technology (ICCIT), p. 101–106.

[5] Liu, H., Hao, H. (2014). Key frame extraction based on improved hierarchical clustering algorithm, *In:* 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), p. 793–797.

[6] Congyan, Lang., De, Xu., Wengang, Cheng., Yiwei, Jiang. (2004). Efficient key-frame extraction using density-estimation-based nonparametric clustering, *In*: 2004 IEEE Region 10 Conference TENCON 2004., A, p. 391–394.

[7] Li, L., Zhang, X., Wang, Y.-G., Hu, W., Zhu, P. (2008). Nonparametric Motion Feature for Key Frame Extraction in Sports Video, *In*: *2008 Chinese Conference on Pattern Recognition*, p. 1–5.

[8] Liu, Lijie., Fan, Guoliang. Combined key-frame extraction and object-based video segmentation, *IEEE Trans. Circuits Syst. Video Technol.*, 15 (7), p. 869–884, (July).

[9] Sun, Lina., Zhou, Yihua. (2011). A key frame extraction method based on mutual information and image entropy, *In*: 2011 International Conference on Multimedia Technology, p. 35–38.

[10] Chen, Z., Bao, F., Fang, Z., Li, Z. (2010). Key-Frame Extraction Using Kernel-Based Locality Preserving Learning, *In*: 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, p. 655–658.

[11] Yong, S.-P., Deng, J. D., Purvis, M. K. (2010). Modeling semantic context for key-frame extraction in wildlife video, *In*: 2010

25th International Conference of Image and Vision Computing New Zealand, p. 1–8.

[12] Pan, R., Tian, Y., Wang, Z. (2010). Key-Frame Extraction Algorithm Based on Entropy, *In*: 2010 International Conference on E-Product E-Service and E-Entertainment, p. 1–4.

[13] Calic, J., Izuierdo, E. (2002). "Efficient key-frame extraction and video analysis, *In*: Proceedings. International Conference on Information Technology: Coding and Computing, p. 28–33.

[14] Huang, M., Shu, H., Jiang, J. (2009). An algorithm of key-frame extraction based on adaptive threshold detection of multi-features, *In*: 2009 International Conference on Test and Measurement, 1, p. 149–152.

[15] Liu, H., Meng, W., Liu, Z. (2012). Key frame extraction of online video based on optimized frame difference, *In*: 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, p. 1238–1242.

[16] Honghua, Lin., Xuan, Yang., Jihong, Pei. (2008). Key frame extraction based on multi-scale phase-based local features, *In*: 2008 9th International Conference on Signal Processing, p. 1031–1034.

[17] Wu, Z., Xu, P. (2012). Research on the Technology of Video Key Frame Extraction Based on Clustering, *In*: 2012 Fourth International Conference on Multimedia Information Networking and Security, p. 290–293.