Geetanjali Khanvilkar / International Journal of Engineering Research and Applications (IJERA)
ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 2,Mar-Apr 2012, pp.909-911

# Domain Specific Video Data Mining Using Multicore Fp-Growth Tree

**Geetanjali Khanvilkar**
Department Of Computer Engineering,RMCET, Ratnagiri.

**Asmeena Mhate**
Department of Information Technology,RMCET,Ratnagiri

**Abstract:**
Content based methods are necessary when text annotations are non-existent or incomplete. Furthermore, content based methods can potentially improve retrieval accuracy even when text annotations are present by giving additional insight into the media collections. For domain specific videos, associations among different events are used for indexing purpose, which contributed a lot in bringing vast knowledge from video libraries and databases. In this paper, we are introducing multicore Fp (frequent pattern)-growth tree building algorithm applicable to streamed data. This algorithm will improve efficiency to retrieve videos through parallelizing frequent pattern constructing steps.

**Index Terms:** Multicore, multithreading, indexing.

## I. Introduction

Associations among different events exhibiting temporal semantics are used for indexing videos, which are domain specific. Shot boundary detection, text recognition, camera motion, specific audio events; which are again domain specific; are used for indexing purpose. When a piece of video is provided as input these indices are used for comparison and wherever match is found, the corresponding video is retrieved. Fp (frequent pattern)- growth tree is used to mine associations among events, which provides semantic insight to users. This paper addresses the multicore capability of FP-growth tree to fulfill the requirement of fast video retrieval.

Multi-core processors are extensively used across many application areas including general-purpose, embedded, network, digital signal processing (DSP), and graphics. However, existing softwares are not able to exploit the multiple cores available in the machine. This is partly because of sequential algorithms implemented by existing software.[2] Scanning the whole video for searching associations among events is time demanding work. This process can be fastened by taking the advantage of multicore inability to build frequent-pattern tree.

The paper is structured in following manner: Knowledge based video indexing and FP- growth to mine the associations between the different events is presented in section 2, which is followed by the multicore inability of FP-tree in section 3.

## II. Related Work

With general content based video retrieval, certain features are selected in order to index them. When a part of video is given as input to the system, the same features are extracted from the video and are compared with features of stored videos. The threshold decides the extent with which the videos should match and accordingly retrieves the videos from databases. But, this technique does not exhibit any temporal semantics, it cause to retrieve irrelevant videos also. To achieve great relevance, association based video indexing architecture is followed for indexing purpose.

The architecture shown in fig.1 provides the insight of steps involved in association based indexing specific to basketball videos.

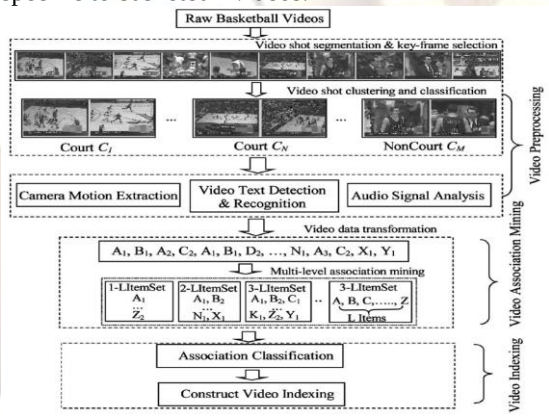


**Fig. 1** The architecture of association-based video indexing

Firstly, shot boundary detection technique is applied for separating shots, in further these are divided into court and non-court shots.

Caption text detection technique is used for extracting team names for building next level of indexing. Still, pan(left & right), zoom (in & out), like camera motions are recognized from a qualitative camera motion extraction method. Special audio events, e.g., audience applause & a referee's whistle help to get semantic cues. To detect audience cheering, the pitch of audio signal is used whereas to detect a referee's whistle,

spectrum domain features are used. The corresponding four streams are combined into single hybrid stream.To have relevant associations, temporal support and temporal confidence which are basics of association mining are chosen as thresholds. Fp-growth tree algorithm is applied to have the associations among different events. It always start with node 'null'. Depending upon the temporal support whole stream is divided into equal sized windows from

```
1. count [ ] = { countA, countB,……..}
      countA: counts event A
      countB: counts event B and so on
2. scan the stream
      If (A) countA++
      If (B) countB++
      And so on
3. Arrange count[ ] in descending order.
      Sort the events and their frequencies in descending order.
4. Select TDT
5. ShotNo=1
6. root=null                                          //fp tree construction
7. scan shots(ShotNo+TDT)
8. arrange them in order found in step2
9. add first event as child node to root, second node as child node to first, and so on
10 ShotNo++
11. while(end of stream)
12.     repeat steps 7 and 8
13.     if prefix is common follow the already available path else add it as new branch.
14.     ShotNo++
15. end of while
```

**Fig. 2** Algorithm to Construct FP

which the frequency of each event is decided and then all events are arranged in descending order. Same order of events is considered while building frequent pattern tree.[1][8][9]

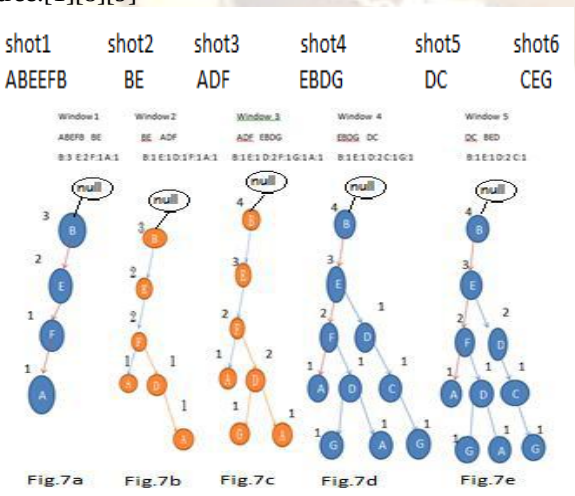


**Fig. 3** FP-tree construction.

| Video Data | Conditional pattern base | Conditional fp-tree |
|---|---|---|
| A | <B:5 E:4 F:2 A:1> <B:5 E:4 F:2 D:1 A:1> | null |
| C | <B:5 E:4 D:2 C:1> | null |
| D | <B:5 E:4 F:2 D:1> <B:5 E:4 D:2> | <B, E, D:2> |
| E | <B:5 E:4> | <B, E:4> |
| F | <B:5 E:4 F:2> | <B, E, F:2> |
| G | <B:5 E:4 F:2 D:1 G:1><B:5 E:4 D:2 C:1 G:1> | null |

**Fig. 4** Frequent Event Determination

By searching patterns from hybrid stream with constraints, appropriate association can be mined as well as manually these associations are evaluated to put in appropriate class of event.[9] Once event is detected for each of the sequence of associated video data, indexing is done.

## Iii Proposed Work

Essential factor for measuring performance of any system is its response time, users want the immediate response to their request. To address this issue, multithreading concept of Java programming language can be implemented, which will give results in short time, even in presence of single processor. When multiple processors are available, due to simultaneous execution of certain instructions, we will automatically achieve efficiency. As we know, sorting and searching algorithms are involving great extent of space and time complexity, in order to minimize it, this paper is presenting algorithm which multithreads certain steps involved in sorting method.

From pseudo code mentioned in fig.5, it is clear that dependent instructions are arranged and executed in sequential order. Step1 and 2 count the frequency of each event contributing to the final hybrid stream. Proceeding with step 3, all events are arranged in descending order of their frequencies, this sorting is to be done parallelizing the execution of instructions.[2] To have temporal semantic, TDT is selected and accordingly equal number of shots are grouped in one window by step no.4. Now these windows can be processed independently as mentioned in[2]

1. Count[]={countA,countB,.....}
   countA: counts event A
   countB: counts event B &
   so on.
2. Scan the video stream
   If (A) countA++
   If (B) countB++ &
   so on.

   //step 3 is to be multithreaded.
3. Arrange count[] in descending order.
   (Sort the events and their frequencies in descending order.)
4. Select TDT and accordingly create equal sized windows.
   Where each window is consisting of TDT+1 number of shots.
5. Root=null
6. Integer i
   //part to be multithreaded (from step 7 to 8).
7. Scan Shots(window [i]) for all i=1,2,3....
   Feed each window consisting of shots of number TDT+1 to thread
8. In each window arrange events in order found in step2 and return.
9. Scan window(i)
10. Add first event as child node to root, second node as child node to first and so on.
11. While (end of stream)
12. Repeat steps 7 and 8 for all events belonging to same window.
13. If prefix is common follow already available path else add it as a new branch.
14. Window(i++)
15. End of while.

**Fig.5** Multithreaded Fp-tree construction.

independently as mentioned in [2] to get descending order of events from same window. These ordered events from each window are processed sequentially to build Fp-tree by following steps 9 to 15.

## Iv. Conclusion

This paper addresses the expectation of users of getting response within short amount of time. Fp-growth tree constructing algorithm involves sorting of events, multithreading this part of algorithm will definitely affect the response time in positive manner.

## References

[1] Geetanjali Khanvilkar, Dr.B.B.Meshram " Video Data Mining: Event Detection from the Association Perspective using FP-growth Tree" December 2011

[2] D.Abhyankar, M.Ingle "An Efficient Parallel Sorting Algorithm for Multicore Machines" 2011

[3] Jian-Kang Wu: "Content-Based Indexing of Multimedia Databases" November/December 1997

[4] Eung Kwan Kang, Sung Joo Kim, and Joon So0 Choi :"Video Retrieval Based On Scene Change Detection In Compressed Streams" August 1999

[5] Mohand-SaõÈd Hacid, Member and Jacques Kouloumdjian :"A Database Approach For Modeling And Querying Video Data" Septeber/October 2000

[6] Jianping Fan, Ahmed K. Elmagarmid, Xingquan Zhu, Walid G. Aref and Lide Wu: "ClassView:Hierarchical Video Shot Classification, Indexing, and Accessing" February 2004

[7] Domenico Cantone, Alfredo Ferro, Alfredo Pulvirenti, Diego Reforgiato Recupero, and Dennis Shasha : "Antipole Tree Indexing to

[8] Support RangeSearch and K-Nearest Neighbor Search in Metric Spaces" April 2005

[9] Xingquan Zhu, Ahmed K. Elmagarmid, Zhe Feng, and Lide Wu: "Video Data Mining: Semantic Indexing and Event Detection from the Association Perspective" May 2005

[10] M. M. Yeung and B.-L. Yeo : "Video visualization for compact presentation and fast browsing of pictorial content" Oct 1997

[11] X. Zhu, A.K. Elmagarmid, X. Xue, L. Wu, and A. Catlin: "InsightVideo: Towards Hierarchical Video Content Organization for Efficient Browsing, Summarization and Retrieval" 2004

[12] S. Nepal, U. Srinivasan, and G. Reynolds: "Automatic Detection of „Goal" Segments in Basketball Videos"

[13] J. Fan, W.G. Aref, A.K. Elmagarmid, M. Hacid, M. Marzouk, and X. Zhu, "MultiView: Multi-Level Video Content Representation and Retrieval," J. Electronic Imaging, vol. 10, no. 4, pp. 895-908, 2001.