A GPU-Assisted Personal Video Organizing System

K Wasif Mohiuddin
Center for Visual Information Technology
IIIT Hyderabad
wasif.m@research.iiit.ac.in

P J Narayanan
Center for Visual Information Technology
IIIT Hyderabad
pjn@.iiit.ac.in

Abstract

Video data is increasing rapidly along with the capacity of storage devices owned by a lay user. Users have moderate to large personal collections of videos and would like to keep them in an organized manner based on its content. Video organizing tools for personal users are way behind even the primitive image organizing tools. We present a mechanism in this paper to help ordinary users organize their personal collection of videos based on categories they choose. We cluster the PHOG features extracted from selected key frames to form a representation for each user-selected category during the learning phase. During the organization phase, labels from a K-NN classifier on these cluster centres for each key frame are aggregated to give a label to the video while categorizing. Video processing is computationally intensive. To perform the computationally intensive steps involved, we exploit the CPU as well as the GPU that is common even on personal systems. Effective use of the parallel hardware on the system is the only way to make the tool scale reasonably to large collections that will be available soon. Our tool is able to organize a set of 100 sport videos of total duration of 1375 minutes in about 9.5 minutes. The process of learning the categories from 12 annotated videos of duration 165 minutes took 75 seconds on a GTX 580 card. These were on a standard desktop with an off-the-shelf GPU. The labeling accuracy is about 96% on all videos.

1. Introduction

Video recording devices are now commonly available to the average consumer. The amount of personal video content generated by users is increasing exponentially as a result. With the popularity of video sharing services and increased access to web, a great number of videos are available today. Taking into account the volume of the data, tasks such as organization or searching through the available content can be very time consuming and tiresome. Therefore, such tasks need to be done automatically, preferably using the content of the videos to accomplish this. Collecting videos of interest has become a hobby lately given the fact that people have large storage devices. Everyone has different category of interest in terms of video. Sports is a genre with large following around the world. People tend to store full recorded matches or selected highlights of sport events of interest on their personal storage devices. An individual may have interest in multiple categories of sports and would like to maintain a personal collection in an organized fashion. There is no tool to do this today for videos; even those available for images are not sufficiently sophisticated. Other genres of interest to everyday users would be family events, outings, graduation ceremonies, etc., which also need to be organized appropriately for easy access in the future.

We present in this paper the design and initial implementation of a scheme that helps categorize large collections of videos on a desktop or laptop. Videos are bulky and so content based organization can be computationally heavy. On the other hand, we can use evidence from multiple parts of the video for its categorization. Individual interests may vary; a personal categorization is thus preferable. In this paper, we focus on the sports genre which has wide range of categories and will be a good test subject for our approach. Our categorization approach is an extension to videos of current scene classification and object detection research carried out by computer vision community. The user annotates a few videos or parts of it by one of the category names which he wishes to use. Our scheme uses the Pyramidal Histogram of Oriented Gradients (PHOG) features and their clustered collection as the representation for each category the user provides, which is computed by the system in the learning phase. A K-NN classifier and aggregation of votes by individual key frames are used to assign categories to the unlabeled collection in the categorization phase. The focus is on a scalable implementation in this work and not on pushing the state of the art on image/video classification.

Video processing is computationally expensive. We exploit all compute power available in a typical desktop or

laptop of the user to achieve this. Graphic Processor Units (GPU) have become popular on even personal systems and have tremendous compute power in them. Our system exploits the parallelism of multicore CPUs as well as the GPUs found on personal desktops and laptops today. Video segmentation is assisted by the GPU and the PHOG computation is performed entirely on the GPU. The GPU also does bulk of the K-Means clustering to select representative vectors in the learning phase. In the categorization phase, the distance evaluation for the K-NN classifier for each frame is performed on the GPU in addition to feature extraction. The final aggregation and decision making takes place on the CPU. Our tool is able to organize a set of 100 sport videos of total duration of 1375 minutes in about 9.5 minutes. The process of learning the categories from 12 annotated videos of duration 165 minutes took 75 seconds. These were on a standard desktop with an off-the-shelf GPU.

2. Related Work

Much has been done in the field of computer vision towards analyzing the image content for scene classification, object detection, image search, etc. Less work has gone on doing the same on videos. A video genre can be discriminated from the other based on the analogous features and attributes that is disparate from other genres. Based on the content of video, the video could be categorized into different genres such as Cartoon, Sports, Commercials, News, Serials etc. The technique described in Vakkalanka *et al.* [19] uses different types of spatial and temporal features. The features are modeled using two different classifier methodologies, namely Hidden Markov Model (HMM) and Support Vector Machines (SVMs).

A number of algorithms have been designed for key frame extraction for videos based on flow. Lui *et al.* proposed a triangular model of perceived motion energy to model motion patterns in videos [7]. The key frames were the turning points of motion acceleration and deceleration. Chen *et al.* extracted optical flows and used them to cluster human crowds into groups in unsupervised manner using adjacency-matrix based clustering (AMC) [2]. Gross image features such as motion and color were used to classify video genre, along with a decision tree classifier [18] concentrated on background or camera motion and the foreground object motion using Gaussian Mixture Model (GMM) as the classifier [16]. Ekenel *et al.* addressed the problem of video genre classification for five classes with a set of visual features, with SVM used for classification [4]. They used temporal and spatial information to build an HMM classifier. Rea *et al.* near-automatically classified tennis videos by modeling the spatio temporal behaviour of the serving player [15]. They then summarized a match using a number of key frames on a synthesized court.

A survey of techniques for automatic indexing and retrieval of video data can be found in Lebart *et al.* [6] and Wang *et al.* [5]. A number of descriptors have been proposed for image representation like GIST [11], PHOG [1], etc. GIST has been found useful for scene classification while PHOG has been used for object identification [3]. We follow the current approach using such features to classify videos into appropriate categories. The learning phase uses a few videos tagged by the user of each sports category. We extract a few keyframes from each video and build a representation using K-Means clustering. Matching is done for similar keyframes of the testing videos using a K-Nearest Neighbour approach. The votes by individual keyframes are aggregated to give one final label to each video.

3. GPU Architecture

Graphic cards have become popular and are being used by a large number of people. The use of graphic cards is no longer restricted to gaming; general purpose computing on them (GPGPU) is used for wide range of applications. GPUs are computationally powerful devices which are able to perform data intensive tasks quickly. NVIDIA's Fermi architecture [12]consists of 16 Streaming Multiprocessors (SM) as shown in Figure 1 with each SM having 32 cores. On the whole, their GTX580 model has 512 CUDA cores. Every core in an SM executes the same instruction. A set of threads forms a block which put together forms a grid. Blocks are assigned to SMs for execution. An SM processes one warp at a time where each warp is of 32 threads from a block. The function calls are made in the form of kernels which unleash multiple threads to perform a task in a Single Instruction Multiple Data (SIMD) fashion. The Compute Unified Device Architecture (CUDA) [10] programming model allows programmers to design a kernel.

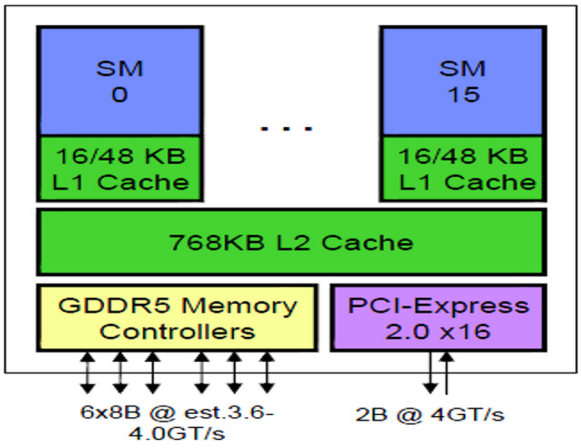

Figure 1. The Fermi Architecture

Each SM has its registers divided equally amongst the

threads that time share it in a kernel. Each thread has a private local memory. The off-chip global device memory per card can be accessed by every thread in the grid but consumes hundred of clock cycles for a single fetch. The GPUs have a moderate amount of shared memory per SM, which is shared between the blocks that map to it at a time. The Fermi GPUs have upto 48KB of shared memory while the older architectures had only 16KB. The Fermi architecture has a single unified memory request path for loads and stores using the L1 cache per SM multiprocessor and unified L2 cache that services all operations. L1 cache is configurable to support both shared memory and caching of local and global memory operations. The 64 KB memory can be configured as either 48 KB of shared memory with 16 KB of L1 cache or vice-versa. By configuring 48 KB of shared memory, programs that make extensive use of shared memory performed up to three times faster. The lifetime of this memory is same as that of a block. Fermi features a 768 KB unified L2 cache that services all load, store, and texture requests. The L2 provides efficient, high speed data sharing across the GPU. The GPUs prior to Fermi had no caching facility.

4. Video Classification

Describing the entire video using a single descriptor – as is often done in images – is not easy. Videos for many purposes can be considered as a collection of images which have a certain association or pattern with each other. In our discussions, we use the term frame to refer to an image from the video stream. A video can be broken into small shots which are a collection of similar frames. Our classification approach has two parts: Category Determination and Category Assignment. There is no off-line training data available; so we need to form a basis for classification in order to categorize the remaining test videos. Also our training process adapts to the user's collections.

4.1. Category Determination

In this phase, we use a few videos tagged with their categories by the user. These are used to train our organization system. Figure 2 shows the flow of the algorithm for this step. The representation formed at the end is the basis for categorizing the videos. We extract the key frames using color histogram differencing. We remove small shots from the obtained result using a threshold on the number of frames to keep only major content. Each remaining shot is represented using a small number of key frames for further processing.

We use PHOG feature descriptor. In our experience, video frames are best distinguished based on objects present in the frames like pitch in the case of cricket, net in the case of tennis, etc. In the image, each region is represented as a Histogram Of Gradients (HOG) as explained in [3]. The

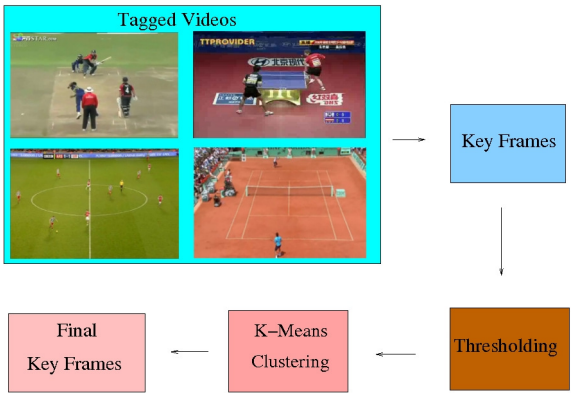

Figure 2. Category Determination: Algorithm Flow

HOG vector is computed for each grid cell at each pyramid resolution level. The final PHOG descriptor for the image is a concatenation of all the HOG vectors. In forming the pyramid, the grid at level l has 2^l cells along each dimension. Level 0 is represented by a V-dimensional vector corresponding to V bins of the histogram, level 1 by a $4V$ vector, etc. Three spatial pyramid levels are used $(1 \times 1, 2 \times 2, 4 \times 4)$. The PHOG vector is normalized to sum to unity. This normalization ensures that images with more edges or texture rich or larger are not weighted more strongly than others. The dimension of our PHOG descriptors is 640. We compute descriptors for all key frames of the tagged videos. The distance between two PHOG image descriptors then reflects the extent to which images contain similar shapes and corresponding in their spatial layout. We use K-Means clustering to group similar frames so that minor variations can be accounted for, similar to how visual words are formed in the object process of Video Google [17]. This results in a set of meaningful centers to represent each category. Our aim is to reduce the large number of descriptors to those that describe a particular category. We typically use 200 clusters to represent each video category. The cluster centers of form the representation for each category.

4.2. Category Assignment

The above representation is used to assign categories to other videos in the user's collection. Shot segmentation and PHOG extraction on user videos are carried out as described earlier. We use a K-nearest neighbor (K-NN) approach using Euclidean distance to assign a label to each key frame of the video. Algorithm 1 shows flow for category assignment phase. Some of the frames may be classified wrongly due to the similarity in individual frames between different categories, such as the ground seen in different sports videos. We give all labels to each frame based on the distances to K nearest neighbors. K may be decided based on the closeness of these sport categories. We use a number that is close

Algorithm 1 Category Assignment

Input: Test Keyframe Descriptors, Centers
Output: Category Label
l: number of key frames, kF: Key frame

for $y = 1$ to l in parallel **do**
  K-NN (kF$_y$) /* Returns K neighbors, distances */
end for
for $y = 1$ to l **do**
  $d1 \leftarrow$ distance(1^{st}-NN(kF$_y$))
  $d2 \leftarrow$ distance(2^{nd}-NN(kF$_y$))
  if $d1/d2 > r$ **then**
    Allocate kF$_y$ a single category, of Centre[1]
  else
    Allocate kF$_y$ to all K categories for the frame
  end if
end for
topper $\leftarrow$ Highest count among categories for video
runner-up $\leftarrow$ Next highest count
if topper is 20% more than runner-up **then**
  Assign video to category of the highest count
else
  Ask user for category assignment
end if

to half the number of categories for K in practice.

Lowe's criterion [8] considers the distances $d1$ and $d2$ ($d2 > d1$) to the first and the second nearest neighbors. A query and its closest neighbor are matched when the ratio $r = d1/d2$ between these two distances is below a threshold. This criterion is more robust than a global threshold on distances and behaves well when the structure to be matched is present exactly once in the candidate database. In our case, if the top match passes the ratio test, only one category is assigned to that frame. Otherwise, multiple labels are assigned to it as per the K nearest neighbors. In this manner, we are able to ensure that multiple close matches against the training frames are considered for final scoring. Each frame has one or more labels at the end of this process. We aggregate them for a video to give it a final label. We compute a category histogram for each video by combining the labels of its key frames. Frames with multiple labels contribute to multiple bins. We assign the most probable label to the video based on the final histogram, provided its score is clearly above all others. To assign a category based on score we require that the top category of a video to have 20% more score than the next best category. We let the user decide the category manually if a clear label cannot be assigned using this criterion. Involving user at the end ensures even ambiguous videos are classified properly also in our view, preferable over mislabeling them. Fewer than 5% of the videos required user intervention in our experiments.

5. Implementation Details

The computationally intensive tasks like segmentation, PHOG feature extraction, K-Means clustering and K nearest neighbor are performed on GPU. Figure 3 indicates the division of work between CPU and GPU for different steps. We adapted the histogram sample code provided by NVIDIA for segmentation and key frame extraction. After the histograms are evaluated, we perform difference between consecutive frames to mark boundaries for shots. PHOG feature descriptors are evaluated on GPU using the approach by Prisacariu and Reid [14]. Their approach uses one thread per pixel and the thread block size is 16×16. Trilinear interpolation is used in cell/block configuration and pixel contributes to up to 4 histograms (one for each cell), and up to 2 bins per histogram. To compute the color gradients, we use two separable convolution kernels similar to the ones from the NVIDIA CUDA SDK. There are two kernels, the first kernel convolutes the row with the centered 1-D mask, while the second kernel computes the column convolution, gradient orientations and magnitudes. HOG is computed for different scales and then merged to get PHOG descriptors. Using the results we get from K-NN we perform the final scoring on CPU.

5.1. K-Means on GPU

We use our own GPU implementation [9] of K-Means. The implementation is divided into two parts: membership evaluation and mean evaluation. We extended parallelism to the computation done on the d components of each input and center vector. Center evaluation involves finding the sum of all vectors with the same label. For a parallel approach this task involves concurrent writes since data objects having the same membership may add the histogram count at a time. Performing the entire process on the GPU is one of our contributions compared to earlier GPU implementations. The row major storage of the vectors make component-wise addition uncoalesced in memory accesses and hence inefficient on the GPUs. The mean evaluation used was able to fix the concurrent write problem which most of the previous approaches failed to solve. The pure coalesced memory access for data rearrangement proved to be vital step in enhancing the mean evaluation on GPU.

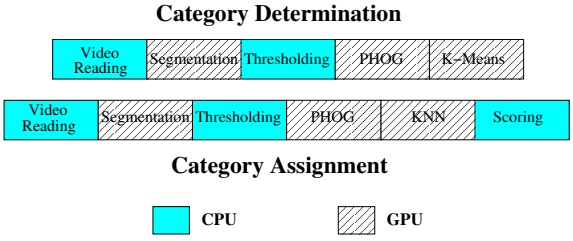

Figure 3. Work division between CPU and GPU.

5.2. K-NN on GPU

The basic process of K-NN algorithm involves finding K nearest neighbors for each keyframe of our test video. To achieve this, we find the distances to the given centers for each test vector and sort them on a distance criteria. The class label of the point is given by the labels of the closest K vectors. For K nearest neighbor evaluations, we use Algorithm 2. Distance of a vector from a centre is evaluated using Algorithm 3. Data elements which are frequently accessed are stored onto the shared memory. We store the following onto the shared memory: s_data holds the input keyframe vector, s_dist stores square of the differences for every dimension, min_y and $membership_y$ store the global minimum distance and label for vector id y. The $syncThreads$ ensures that threads in a block which have completed the task to wait for other threads to finish the task before executing the next instruction. We have all the distances for the testing frames from the centers. The number of testing frames is typically large. We use the SplitSort operation to get K nearest centers. We sort the index of center based on their distance from a single data object, using the *split* primitive [13]. This is done by forming a list of 64-bit records combining the distance and center index. We split this using the distance value as the key as shown in Figure 4, shuffling the original center index order. After sorting the centers index based on distance we send the top K centers from the sorted list to the CPU for the final categorization.

Algorithm 2 KNN in parallel for all keyframes

1: Each block handles l new keyframes at a time loops over all keyframes
2: Find distances for each keyframe against all centers sequentially using Algorithm 3
3: k distances are in the global memory
4: Sort the distances using Splitsort for each keyframe

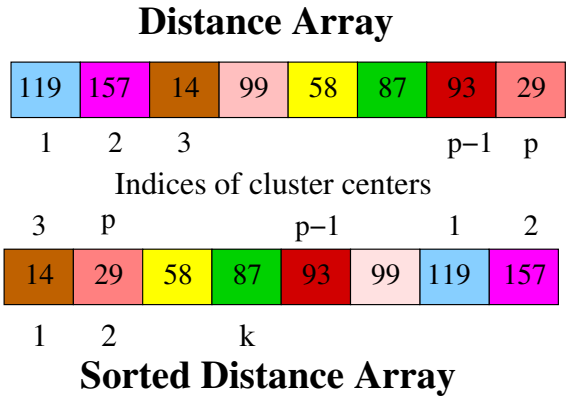

Figure 4. The distance array is sorted as per the distance values. The top k values represent the K Nearest Neighbors

Algorithm 3 Distance Finding for K-NN

Input: inpVector, Centers
Output: distances
tid: Thread Id in a block, dim*: Vector dimensionality*
dist$_z$*: Distance between vector and center* z
s_dist*: Distance components in the shared memory*
/ d threads process a vector, each a component */*

1: $s\_data \leftarrow inpVector[tid]$
2: ***for*** $z = 1$ *to* k *clusters* ***do***
3: $s\_dist[tid] = (s\_data - center_z[tid])^2$
4: *SyncThreads*
5: $i = dim/2$
6: ***while*** $i \geq 0$ ***do***
7: ***if*** $tid \leq i$ ***then***
8: $s\_dist[tid] = s\_dist[tid] + s\_dist[tid + i]$
9: ***end if***
10: *SyncThreads*
11: $i \leftarrow i/2$
12: ***end while***
13: $dist_y[z] \leftarrow s\_dist_y z[0]$
14: ***end for***

6. Results

We worked on a collection with four sport category videos, including cricket, tennis, football, table tennis, for our initial experiments. We downloaded highlights of these sports from popular websites like You Tube, Vimeo, Metacafe, etc., ranging over a period of 5-6 years. The videos typically have 30 frames per second and range from 8-15 minutes of duration. For our experiments, we took 100 videos belonging to 4 categories. Each video roughly had 20K frames. The user tagged 12 videos belonging to four categories. We performed our experiments using computers which have a dual-core Intel CPU and a GPU. A high-end Nvidia GTX 580 and a low end Nvidia 8600 GPUs under the CUDA programming model were tried to gauge the effect of performing the classification on a desktop and on a laptop. In case of cricket videos, we observed that the following shots emerged as key frames: Full screen Score card, Pitch, Wicket celebration, Boundary, Focus on a player, Hawkeye prediction and Crowd (Figure 5). Even in case of football we found such similar frames of player positioning, goal post, goal celebration, football trajectory, crowd etc. Pretty similar was the case for other categories. During the category determining phase, we performed video segmentation, PHOG evaluation, and K-Means clustering on the GPU device. Instead of representing a shot by single frame we take 4 frames to represent a single shot. In this manner, we get an entire summary of the shot consisting of adequate details. Figure 5 shows typical key frames in case of cricket, as discussed above we can see the typical frames

GPU Device	No of Videos	No of Key frames	Segmenting video	PHOG features	K-Means Clustering
8600	4	756	182.7	139.6	3.94
	12	2432	584.3	468.4	14.3
280	4	756	24.8	19.2	0.59
	12	2432	76.9	61.8	1.97
580	4	756	11.8	9.1	0.26
	12	2432	37.91	30.2	0.89

Table 1. Time taken in seconds to process the Category Labeling phase on NVIDIA 8600, GTX 280 and GTX 580 cards

standing out. Figure 6 shows the key frames extracted during the category determining phase using the tagged information. We perform 10 iterations of K-Means on each category to get representative centers for each. The final representation consists of about 200 key frames for each category, distributed evenly.

During the labeling phase we perform the video segmentation, PHOG feature descriptor, K-nearest neighbor on the GPU. Table 1 shows the time taken for each stage of training process for 4 videos of one category and 12 videos of all categories by the user. The times in seconds are shown on a low-end (8600) and a hig-end (GTX 280, GTX 580) GPU showing that the algorithm can be scaled to number of cores. The comparison shows that even the low-end users can benefit from our application.

Table 2 shows the processing time of K-NN algorithm for the category assignment phase on 8600, GTX 280 and GTX 580. The time consuming steps in our application are the video segmentation and PHOG feature extraction. The ratio test was useful for frames which had close ups of an individual or the crowd. Such frames were labeled in multiple categories. Frames consisting of the cricket pitch, the table in tennis table, etc., were classified clearly. In Table 3 we see the percentages of individual frames which are correctly classified for different number of neighbors for a single test video in each category. In case of 3 neighbors, we label the category of a frame based on the majority neighbor criteria. In cases where there was no majority the frame may belong to any of the 3 neighbor categories. Improvement

Figure 5. Key Frames for Cricket

Figure 6. Training Frames

could be seen in categories which have frames consisting of field, players, etc. Not much improvement could be seen in table tennis as majority of the frames consisted of the table views which were easy to match. Using K-NN was beneficial in boosting the classification of frames. We did have some misclassification in certain categories like football due to lack of information from majority of the key frames which have field, players in it. Our tool is able to organize a set of 100 sport videos of total duration of 1375 minutes in about 9.5 minutes. The process of learning the categories from 12 annotated videos of duration 165 minutes took 75 seconds. We achieved an accuracy of nearly 96% on our testing dataset.

GPU Device	No of Videos	No of Key frames	K-NN
8600	88	16946	40.33
280	88	16946	5.39
580	88	16946	2.46

Table 2. Time taken in seconds for K-NN during the category assignment phase on NVIDIA 8600, GTX 280 GTX 580 cards

7. Conclusions

In this paper, we presented the design and initial results from a GPU-assisted system to organize a personal collection of videos. We used a combination of the CPU and the

No of Neighbors	Cricket Videos	Football Videos	Tennis Videos	TT Videos
1	64%	58%	69%	82%
3	73%	66%	77%	84%

Table 3. The percentage of frames correctly classified using K-NN

GPU to get good computational performance. We used simple methods adapted from object recognition literature to classify video frames. We would like to use more sophisticated methods from the visual object detection and scene classification literature in the future. We have to be selective about this as computational requirements have to be kept reasonable for any personal system. We intend to test the system on larger databases with more categories popular in personal videos. In future, with people having thousands of videos, GPU devices on their machines such application will always be handy for maintaining an organized collection of personal videos.

References

[1] A. Bosch and A. Zisserman. Representing shape with spatial pyramid kernel. In *ACM International Conference on Image and Video Retrieval*, CIVR, 2007. 2

[2] D.-Y. Chen and P.-C. Huang. Motion-based unusual event detection in human crowds. *J. Vis. Comun. Image Represent.*, 22:178–186, February 2011. 2

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR, pages 886–893, 2005. 2, 3

[4] H. K. Ekenel, T. Semela, and R. Stiefelhagen. Content-based video genre classification using multiple cues. In *Proceedings of the 3rd international workshop on Automated information extraction in media production*, AIEMPro, pages 21–26, 2010. 2

[5] W. Lao, J. Han, and P. H. N. de With. Automatic sports video analysis using audio clues and context knowledge. In *Proceedings of the 24th IASTED international conference on Internet and multimedia systems and applications*, pages 198–202, 2006. 2

[6] K. Lebart, C. Smith, E. Trucco, and D. Lane. Automatic indexing of underwater survey video: algorithm and benchmarking method. *IEEE Journal of Oceanic Engineering*, 28(4):673 – 686, 2003. 2

[7] T. Liu, H. Zhang, and F. Qi. A novel video key-frame-extraction algorithm based on perceived motion energy model. *IEEE Transactions on Circuits Systems for Video Technology*, 13(10):1006–1013, 2003. 2

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, November 2004. 4

[9] K. W. Mohiuddin and P. J. Narayanan. Scalable Clustering using multiple GPUs. In *IEEE International Conference on High Performance Computing*, HiPC, 2011. 4

[10] NVIDIA. www.developer.nvidia.com/object/cuda.html. 2

[11] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, May 2001. 2

[12] V. Osipov, N. Leischner, and P. Sanders. Nvidia fermi architecture white paper - www.nvidia.com, 2009. 2

[13] S. Patidar and P. J. Narayanan. Scalable split and sort primitives using ordered atomic operations on the gpu. In *IIIT/TR/2009/99. IIIT Hyderabad Technical Report*, 2009. 5

[14] V. Prisacariu and I. Reid. FastHOG - a real-time GPU implementation of HOG. Technical Report 2310/09, Department of Engineering Science, Oxford University. 4

[15] N. Rea, R. Dahyot, and A. C. Kokaram. Classification and representation of semantic content in broadcast tennis videos. In *IEEE International Conference on Image Processing*, ICIP, pages 1204–1207, 2005. 2

[16] M. J. Roach, J. D. Mason, and M. Pawlewski. Video genre classification using dynamics. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Volume 03*, ICASSP '01, pages 1557–1560, 2001. 2

[17] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1470–1478, 2003. 3

[18] B. T. Truong, S. Venkatesh, and C. Dorai. Automatic genre identification for content-based video categorization. In *International Conference on Pattern Recognition*, ICPR, pages 4230–4233, 2000. 2

[19] S. Vakkalanka, C. Krishna Mohan, R. Kumaraswamy, and B. Yegnanarayana. In *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing*, pages 187 – 192, 2005. 2