Samera:A Scalable and Memory-Efficient Feature Extraction Algorithm for Short 3D Video Segments

Rahul Malik
Department of Computer Science
University of Illinois at Urbana-Champaign
rmalik4@illinois.edu

Chandrasekar Ramachandran
Department of Computer Science
University of Illinois at Urbana-Champaign
cramach2@illinois.edu

Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
indy@cs.uiuc.edu

Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
klara@cs.uiuc.edu

ABSTRACT

Tele-immersive systems, are growing in popularity and sophistication. They generate 3D video content in large scale, yielding challenges for executing data-mining tasks. Some of the tasks include classification of actions, recognizing and learning actor movements and so on. Fundamentally, these tasks require tagging and identifying of the features present in the tele-immersive 3D videos. We target the problem of 3D feature extraction, a relatively unexplored direction. In this paper we propose *Samera*, a scalable and memory-efficient feature extraction algorithm which works on short 3D video segments. The focus is on relevant portions of each frame, then uses a flow based technique across frames (in a short video segment) to extract features. Finally it is scalable, by representing the constructed feature vector as a binary vector using Bloom Filters. The results obtained from experiments performed on 3D video segments obtained from Laban Movement Analysis (LMA) show that the compression ratio achieved in Samera is 147.5 as compared to the original 3D videos.

1. INTRODUCTION

Tele-immersion systems provide a way for users in distributed geographical locations to collaborate with each other in a shared virtual space that combine audio and video. Tele-immersive systems, by their very nature, generate large amount of 3D video content hence represent testbeds where data-mining techniques are desired for performing several tasks, notably in classification and categorization of 3D videos. As an example, our current Tele-immersive Environments for EVErybody (TEEVE) system [16] allows for collaborative dance classes. Let us assume that the user wishes to search through the 3D video recordings, what kind of movements are being performed by each of the dancers she would then simply interact with an interface and she would easily get a categorized list of movements performed, and their corresponding dancers from the system. Another task could be to mine activity patterns from the recorded 3D video datasets. For example, to find out which dancers in the stored 3D videos are performing a particular dance move.

As a precursor to performing these tasks, we need an efficient way to tag and recognize features in the videos obtained from the recordings in the tele-immersive system. An important aspect of these videos is their 3D nature. The existing algorithms for feature extraction in the computer vision community are currently being focused on 2D videos, and the topic of feature extraction in 3D is relatively unexplored.

3D videos present several difficulties while processing and this factor is compounded in the videos obtained from TEEVE. First, any feature vector for 3D videos would need to account for depth information present in these videos. This gives an added complexity to the feature vector description. Second, even if current feature extraction techniques were to be used for our requirements, they would not be scalable simply because of their in-memory requirements while processing. Thirdly, is that TEEVE generates videos which are noisy in nature and it becomes a challenging task to identify the actors or movements from these videos. As an addition, there could be practical difficulties of using the feature vectors obtained for several of the data-mining tasks mentioned. The feature vectors themselves may need to be pre-processed in a particular format (according to the application requirements).

Thus, an important question becomes: Can we investigate solutions to the feature extraction problem that works well for noisy 3D videos, is scalable, has low in-memory requirements, is reasonably accurate, and requires a low amount of (post feature-extraction) pre-processing? We answer these

Immerscom 2009, May 27-29, 2009, Berkley, USA.
Copyright C 2009 ICST ISBN # 978-963-9799-39-4

questions in the paper.

We present Samera, a scalable and memory-efficient feature extraction algorithm for short video segments obtained from a TEEVE system. The main contributions of Samera are that it:

1. Provides a novel frame descriptor representation that is compact, and handles depth information present in 3D videos.

2. Is scalable to large workloads.

3. Has low in-memory requirements, even for large workloads.

4. Performs well for a diverse set of workloads, that have a high degree of noise.

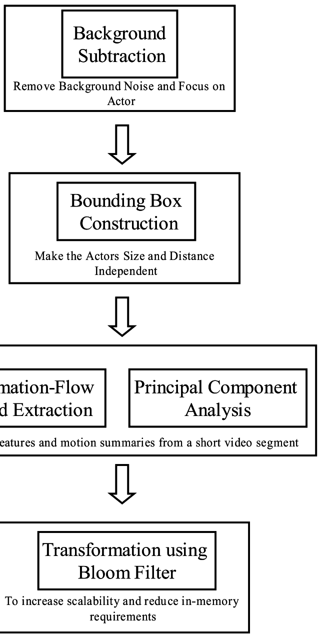
Background Subtraction
Remove Background Noise and Focus on Actor
Bounding Box Construction
Make the Actors Size and Distance Independent
Information-Flow based Extraction
Principal Component Analysis
Extract features and motion summaries from a short video segment
Transformation using Bloom Filter
To increase scalability and reduce in-memory requirements

Figure 1: A high-level flow diagram of Samera.

A flow diagram of Samera is shown in Figure 1. Samera is run off-line after collecting the short video segments for training and testing purposes from TEEVE in a single machine. While the terms, and exact technical description are explained in Section 3, we give a brief overview of the architectural steps followed in its feature extraction process. Earlier, we had mentioned that the video segments obtained are very noisy, and in many video segments the actors are barely recognizable even to the naked human eye. So, to focus our feature-extraction only on the actor and ignore the background, we perform some form of *background subtraction* here. In addition, many actors are of different sizes, and perform similar actions. A good feature extraction algorithm must be size and distance-independent and should recognize the actor and actor movements regardless of these two. We construct a *bounding box* around each actor for this purpose. Next, we need to account for actor movements in all directions along with temporal aspects of these video segments. We use an information-flow based technique along with *Principal Component Analysis (PCA)* to obtain the motion summaries and a frame descriptor representing this. Finally, to make the obtained feature vectors more scalable, we do a transformation using *Bloom Filters* to obtain a binary vector representing the features of the short video segment.

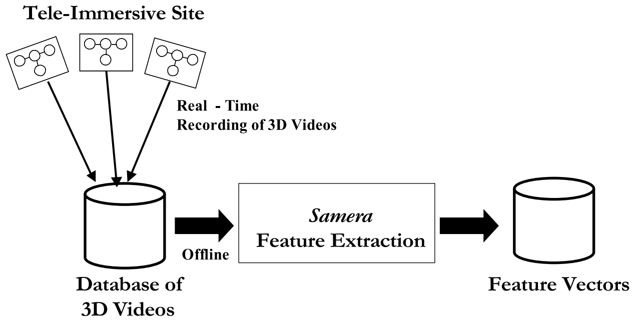

Figure 2: System Overview of Samera.

The rest of this paper proceeds as follows: Section 2 briefly describes related work in feature extraction techniques, Section 3 and 4 describe Samera while Section 5 shows performance results with workloads obtained from TEEVE and finally Section 6 concludes the work and outlines scope for further research, especially in data-mining.

2. RELATED WORK

Related work in feature extraction and representation are abundant, especially in approaches which use local interest points in video images. Some techniques include SIFT, Harris Detector, p-LSA model and so on [11]. Another set of techniques includes using behavioral similarity features such as intensity. Some examples of this technique include the integral video method by [10] using motion energy receptive fields, and a tensor canonical correlation analysis method. These are collectively called patch-based methods. Other methods such as the bag-of-words representation model provide a concise way of describing multiple features and which can be used for matching similarity of different videos. Statistics based approaches like histograms have also been effective. The distance measures used for all these statistical approaches include KL-Divergence and χ^2 techniques are also in use.

Location invariant feature matching and extraction is an important task which has gained popularity in many feature extraction algorithms developed today. One of the first substantial research gains in this were obtained by Schmid et al. [4]. Some more work in this area was done by Lowe [6] who achieved scale invariance. Some work using wavelet coefficients for feature extraction was done by Shokoufandeh et al [13]. There are also many valuable papers in shape description techniques. Two prominent ones are [14] and [9]. A survey of existing research in shape description is provided

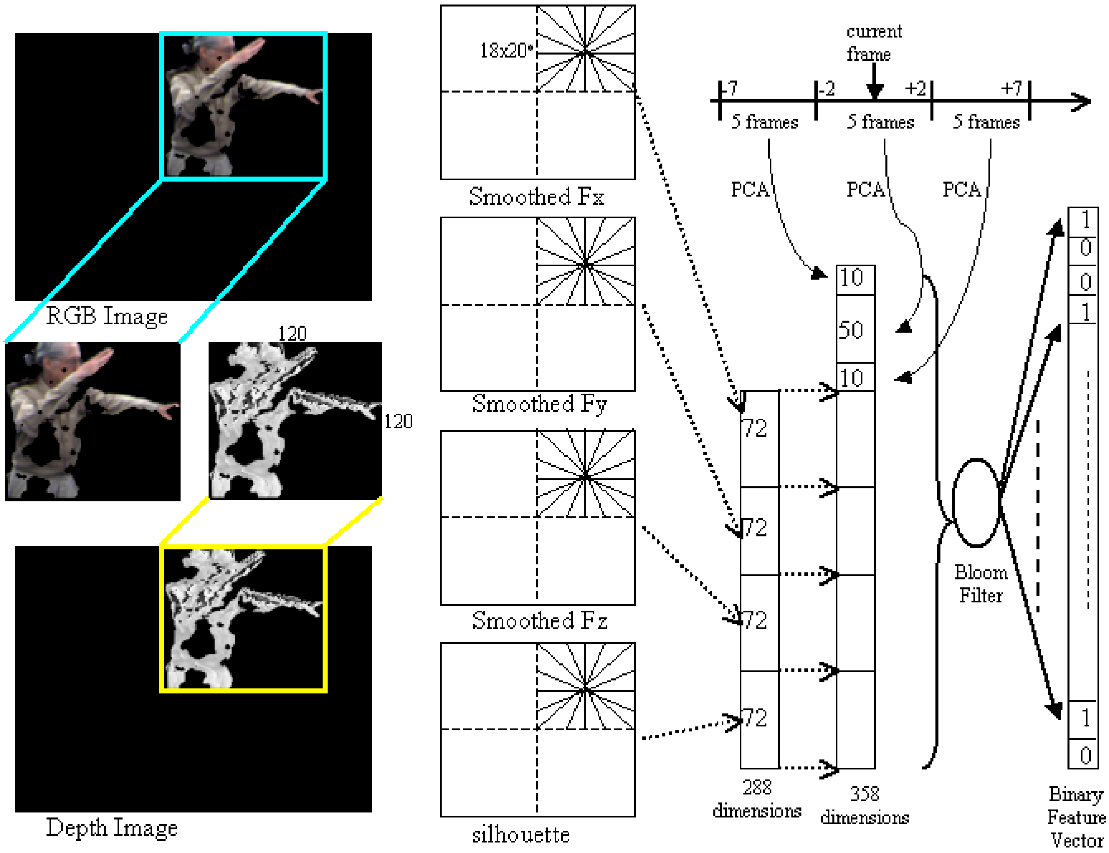

Figure 3: For our feature extraction we use four information channels namely: Horizontal flow, Vertical flow, Depth flow and silhouette. In each channel, similar to our previous step, the measurements are resampled to fit into normalized (120×120) box while maintaining aspect ratio. The normalized bounding box is divided into 2×2 grid. Each grid cell is divided into 18-bin radial histogram $(20^\circ$ per bin). Each of the 4 channels is separately integrated over the domain of each bin. The histograms are concatenated into 288 dimensional frame descriptor. 5-frame blocks are projected via PCA to form medium scale motion summaries. For the immediate neighborhood, the first 50 dimensions are kept and for the two adjacent neighborhoods the first 10 dimensions are kept. The total 70-dimensional motion summary is added to the frame descriptor to form the motion context. Finally, they are represented as a binary vector output of Bloom filter.

in [14] and a review of three-dimensional shape-searching techniques is provided in [9].

From a survey of existing work for 3D videos we found the feature extraction techniques to be quite insufficient. One of the early techniques of feature extraction from 3D objects was [8]. However this technique focused only on graphical models obtained from sources such as scanners, multiple cameras and so on. Another technique [1] computed facial features from frontal and profile view images of a person, and used this further to reconstruct 3D facial models. Most recently, [15] developed a similarity search and retrieval scheme for 3D videos. However their technique worked only on 3D models which are free from noise.

3. SAMERA: SYSTEM DESCRIPTION

In this section we give a high-level overview and system description of Samera. Figure 2 shows our system. Samera consists of 4 major components. The first one is the tele-immersive system called TEEVE where real-time 3D videos are recorded. The next one is a database where these videos are stored. Next and the most important component is the Samera processor. The final component represents the extracted feature vectors obtained from Samera.

When a user interacts with the tele-immersive environment the 3D videos are generated, recorded and stored in real-time in the database of 3D videos. These videos can then be processed offline by Samera to generate the feature vectors. Once the feature vectors are obtained we need not worry about the original 3D videos. In this way the database continuously collects the 3D videos in real-time while the processing is done by Samera off-line.

To understand the characteristics of the video frames that we use for our evaluation purposes, we provide an overview of the relevant components of the tele-immersive environment [16]. Tele-immersion is aimed at enabling users in geographically distributed sites to collaborate in real time in a shared simulated environment as if they were in the same physical room. The capturing of 3D videos in a tele-immersive system consists of a series of C 3D cameras (where C is the number of cameras), each 3D camera consisting of four 2D cameras, organized around the subject (or the actor) and synchronized. At a given point of time, each camera captures one frame of depth image from its viewpoint and provides a 3D video frame. At that time instant, the remote renderer uses C 3D reconstructed frames to show the same scene from different viewing angles. The cameras have a 360^o coverage around an actor, and this enables a viewer to view the actor from any viewing direction. The 3D reconstructed frames are recorded as videos.

4. FEATURE EXTRACTION AND REPRESENTATION

In this section, we present our algorithm for extracting features. A feature, in our context, is an entity that is inferred to abstract a group of frames (obtained from a short video segment). Thus, a feature represents the temporal aspects of a frame along with motion in three directions along with shape of the person as a sequence of bits. The three dimensions refer to the horizontal and vertical component of a video, and the depth value associated with them. These are referred to as x, y, and z directions respectively. The video segments consists of multiple frames that are sequenced temporally. We use those frames for extracting the features. The feature extraction is a two-step process where we first focus on the relevant portions of each frame, and then proceed to a flow-based technique across frames. The process of feature extraction is explained in Figure 3.

4.1 3D Feature Vectors

The background in the captured videos contains a lot of unnecessary information that is not relevant to the task of actor identification and merely acts as noise for feature extraction. In order to focus on the actor without worrying about the background, we perform background subtraction. A model of the background is made from initial few frames for every camera and the background subtraction is done based on the difference of the current frame with respect to the background model. Once the portion of the frame with the actor is extracted out, the portion remaining is the silhouette. Based on the way we capture images, if a rotation is performed by an actor the user would view the actor with the same viewing direction with respect to some other camera as the previous camera. Because our feature vector is a combination of feature vectors of several different cameras, this leads to rotation invariance. Further, actions performed by actors located at different distances from the camera should be treated as the same actions. These actors appear to be of different sizes depending on their distances from the camera. Therefore, to make them size independent, a bounding box is created around the actor and it is normalized to make the person seem distance-independent. This is shown in Figure 3 by boxes around the actor in the frame.

The captured 3D videos contain movements of actors made in the three directions, namely x, y, and z. We need a way to capture the amount of movement made by them. Flow based techniques [7], which we adopt for use in our paper, help us in achieving this goal. A flow represents different amount of movements made by different body parts in those directions. In addition, different body shapes for different actions need to be considered to distinguish body shapes from one another. For this, we use a silhouette mask.

Our frame descriptor is a histogram of the silhouette and of the optic flow inside the normalized bounding box. We scale the larger side of the bounding box to a fixed size R preserving the aspect ratio. The scaled box is then placed at the center bottom of an $R \times R$ square box padded with zeros. The rescaling of this bounding box makes the object size invariant. We use this transformation to resample the values of the flow vectors and of the silhouette.

In order to extract the features, we use flow based techniques [7] in x, y, and z direction. From Figure 3, we can see that four different kinds of flows are extracted, labeled as F_x, F_y, F_z and silhouette. We design our features to capture appearance and motions of small portions of the person. To capture this, we use the concept of radial bins [2]. The concept of radial bins lets us take into account different features that are located at various distances from the point where the measurement is made. The radial bins are shown for each of the quarter portions of flows in Figure 3.

The optic flow measurements are split into horizontal (x), vertical (y) and depth (z) channels. To reduce the effect of noise each channel is smoothed using a median filter [12]. This gives us three real-valued channels F_x, F_y and F_z. The silhouette gives us the fourth (binary) channel S. Each of the four channels is histogrammed using the same technique. The normalized bounding box is divided into 2×2 sub-windows. Each sub-window is then divided into 18 pie slices covering 20 degrees each. Belongie et al [2] have used the concept of pie slices and found these values to be suitable. Intuitively, if the number of pie slices is less then information is lost and finer details are not captured. On the other hand, if the number of pie slices is high then it becomes computationally challenging to extract the features.

The center of the pie is in the center of the sub-window and the slices do not overlap. The values of each channel are integrated over the domain of every slice. The result is a 72 ($2 \times 2 \times 18$)-dimensional histogram. By concatenating the histograms of all 4 channels we get a 288-dimensional frame descriptor. By doing this we are able to take small-local body motions of the actor into account.

The captured movements in these videos represents views from different cameras. Feature vectors are obtained for the images captured from each camera's view. We need a way to make the feature vectors seem independent of each camera's location (since they are capturing the images of the same actor regardless of their location). In order to combine multiple camera views, the 3D feature vectors from all the cameras are concatenated in order to make a single feature vector. The resulting histograms are normalized in order to represent them in the same range.

4.2 Taking Motion (Temporal Aspects) Into Account

In addition to the extracted features from a current frame, we need to consider the motion made by the actors in the 3D videos. Motion takes place over time, and we need a way of taking this into account while constructing our feature vectors. We use a number of frames around the current frame (including past and future frames) to achieve this goal.

We use 15 frames around the current frame and group them into 3 groups of 5 frames: past, current and future. The choice of 15 frames is ideal for taking local temporal aspects into account and without involving much computational overhead. In essence, it is like a short video segment. Previous work such as Shape Context [2] and 30-pixel men [7] have found this choice suitable. Since each block consists of 5 frames and each frame is represented using 288-dimensional frame descriptor, stacking them together results into a block descriptor, which is a 1440 dimensional vector. This block descriptor is then projected onto the first N principal components using principal component analysis (PCA). We keep the first 50, 10 and 10 dimensions for the current, past and future blocks respectively. These parameters are chosen based on a Gaussian distribution around the current frame so that the current block gets maximum weight and past and future frames receive lesser weight. The resulting 70-dimensional context descriptor is appended to the current frame descriptor to form the final 358-dimensional motion context descriptor.

4.3 Representation Framework Using Bloom Filters

Now that we have obtained the final feature vectors, we need to refine them in a way which will be more scalable as well as be quick enough for further processing. The large number of feature vectors deters us from using them directly. Each frame feature vector consists of a 288-dimensional frame descriptor. If we were to consider 15 frames for comparison we would get $288 \times 15 = 4320$ features which would be computationally difficult to process. To solve this, we use Bloom filters [3] and represent the resulting feature vector as a binary vector. Bloom filter is a unique data structure which provides for space-compaction by testing for set-memberships. This consists of a number of hash functions which map key-values to individual positions of the data structure. Initially all elements in this structure are set to zero. Whenever a value is hashed to a particular location, that position is set to 1. Another advantage of using Bloom filters is that the probability of false positives is very low [3].

So, in order to represent a frame as a Bloom filter, we add the 358 elements of the frame in an empty Bloom filter and it serves as a representation of the frame.

Table 1: Value of various parameters used in experimental evaluation

Symbol	Parameter	Value
C	Number of 3D cameras	12
h	Number of hash functions	10
m	Number of bits in Bloom filter	1000000

5. EXPERIMENTAL EVALUATION

5.1 Workloads

The workloads for our experiments are from the tele immersion setup. The workloads that we use are the dancer's data for Laban Movement Analysis [5]. This provides a diverse set of videos for our purposes. Since LMA provides a systematic method of categorizing different basic movements, different movements can be composed using these basic categories. Therefore it is a general representative of different movements actors can make. LMA styles are divided broadly into three categories: dream state, mobile state and rhythm state. Each of these states is further divided into two subcategories. Dream state is divided into free & light and bound & strong. Mobile state is divided into free & quick and bound & sustained, whereas the rhythm state is divided into strong & sustained and light & quick.

5.2 Results

Here we describe the results of experiments done on the workloads using Samera. We evaluate the following metrics here: 1) Accuracy of feature extraction on the workloads with and without Bloom Filters, and 2) Memory requirements and scalability aspects of using Samera.

From each of the six sub-categories Laban Movement Analysis, one video was selected. We removed the last portions of videos from LMA and used the remaining starting portion to train a classifier. We use individual video frames and the idea is that if the classification is accurate, then it will identify those frames to the correct class. The removed portions were used for testing the feature extraction algorithm and we tested each for the category to which it belongs. Each of the trained videos were taken as a category in itself. We used original feature vectors from each frame without using the Bloom filter. This is used to test the effectiveness of our feature extraction algorithm which in turn affects the classification accuracy (of the images).

For the purposes of classification, we train an SVM classifier for the training data. This classifier was then used to allocate labels to the remaining portion of the video. A label was assigned to each of the frames of the testing video.The frame label on the complete testing video segment was assigned as the majority of the labels.

We used libSVM[1] to train an SVM classifier. We used the rbf kernel with $\gamma = 0.00012367$ and $\rho = -0.121748$. These values are obtained using cross-validation on training data and the classifier was performing best for these values. The obtained results are shown in Table 2. The results show the total number of frames that were used for testing purposes. From the results, we can see that for different categories, the percentage of frames that were correctly identified by SVM of that category is around 70%. A match is considered as correct if a frame is identified to be of same category as it was trained. There is some variation in the results for different categories. The lowest results are around 65% and the highest accuracy is around 80%.

Next, we looked at how the performance of the SVM classifier is affected by using the binary vector obtained from the Bloom filter as a feature vector. The results obtained from this are shown in Table 2. In most cases the accuracy results obtained using Bloom filters outperform those without using them.

We now show that the memory requirements of Samera using Bloom Filters are much less than the original videos. The original videos obtained from TEEVE are of size 640 × 480 pixels. Each pixel needs 5 bytes for representation-3 for color and 2 for depth. The total bits required then becomes $640 \times 480 \times 5 \times 8 = 11.71$ Mbits. This is for one camera cluster. We have 12 cameras in our experimental setup and thus the total size required then becomes 140.6 Mbits. However the Bloom filter representation of Samera requires around 0.95 Mbits. The compression ratio which is defined as the ratio of size of original data to compressed data is now $140.6/0.95 = 147.5$. Since the reduction in memory usage by

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvm

Table 2: Results of analysis of feature extraction and Bloom filter over LMA

		Without Bloom Filter		**With Bloom Filter**	
LMA Style	**Total frames**	Number of correct	% correct	Number of correct	% correct
Dream State: Free & Light	245	173	70.6%	188	76.7%
Dream State: Bound & Strong	208	148	71.1%	134	64.4%
Mobile State: Free & Quick	234	185	79.1%	196	83.7%
Mobile State: Bound & Sustained	228	154	67.5%	174	76.3%
Rhythm State: Strong & Sustained	298	193	64.7%	198	66.4%
Rhythm State: Light & Quick	273	188	68.8%	193	70.7%

Samera is significant, it is scalable to larger workloads. From these results we can see that using Bloom Filters not only improves accuracy, but also reduces the memory overhead significantly.

6. CONCLUSIONS

In this paper we have addressed the important problem of extracting features from the 3D videos present in tele-immersive systems. Feature extraction in 3D videos presents several challenges including that of noise, handling depth information, scalability of processing to name a few. We have presented Samera, a scalable and memory-efficient feature-extraction algorithm to address these issues and challenges. Our algorithm, though currently capable of only working over short video segments, can be easily scaled to larger workloads. We showed the quality, memory-efficiency, the ability to handle noise as well as the scalability of Samera in our performance analysis with some diverse workloads. From our results we show that using Bloom Filters not only improves accuracy, but also reduces the memory overhead significantly. The next logical step here, as we mentioned earlier in our introduction, would be to perform several data-mining tasks such as classification, categorization and activity-mining.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF CMS 0427089, NSF CNS 0448246 and NSF CNS 0520182).

7. REFERENCES

[1] A.-N. Ansari and M. Abdel-Mottaleb. Automatic facial feature extraction and 3d face modeling using two orthogonal views with application to 3d face recognition. *Pattern Recognition*, pages 2549–2563, 2005.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:509–522, 2002.

[3] B. Bloom. Space-time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[4] C.Schmid and R.Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(5):530–534, 1997.

[5] E. Davies. *Beyond Dance: Laban's Legacy of Movement Analysis.* Routledge Taylor and Francis Group, 2001.

[6] D.G.Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, pages 1150–1157, 1999.

[7] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. *ICCV*, pages 726–733, 2003.

[8] H. H. S. Ip. Content-based retrieval of 3d models: Feature extraction and representation. *11th International Multimedia Modelling Conference*, pages 16–17, 2005.

[9] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509–530, 2005.

[10] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. *ICCV*, pages 166–173, 2005.

[11] J. Niebles, H. Wang, and L. F. Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[12] W. Pratt. *Digital Image Processing.* Wiley, New York, 1978.

[13] A. Shokoufandeh, S. J. Dickinson, K. Siddiqi, and S. W. Zucker. Indexing using a spectral encoding of topological structure. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'99)*, pages 2491–2497, Ft. Collins, CO, June 1999.

[14] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *SMI*, pages 145–156, 2004.

[15] Y. Toshihiko and A. Kiyoharu. Motion segmentation and retrieval for 3d video based on modified shape distribution. *EURASIP Journal on Advances in Signal Processing*, 2007.

[16] Z. Yang, Y. Cui, B. Yu, J. Liang, K. Nahrstedt, S.-H. Jung, and R. Bajscy. Teeve: The next generation architecture for tele-immersive environments. *ISM*, pages 112–119, 2005.