

AUTOMATIC DISCOVERY OF CATEGORIES IN VIDEO

by

CHRIS REYNIA

Advisor

MICHAEL BRANTON

A senior research paper submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mathematics and Computer Science
in the College of Arts and Science
at Stetson University
DeLand, Florida

Spring Term
2011

TABLE OF CONTENTS

Table of Contents	ii
List of Figures	iii
List of Tables	iv
Abstract	1
Introduction	2
Previous Work	3
Automatic Video Categorization	3
Media Categorization	3
Images from Video	4
Image Comparison	6
Solution	10
Multi-Media Graph	10
Frame Picking	14
Image Segmentation	14
Low Level Analysis	15
Categorization	15
Limitations	17
Results	19
Analysis	21
Conclusion	22
References	23

LIST OF FIGURES

Figure 1 – A “video tapestry” from [4].....	4
Figure 2 – A “stained glass” video summary from [5].....	5
Figure 3 – Normalized Cut Image Segmentation [3].....	7
Figure 4 – MMG with Category Nodes	11
Figure 5 – MMG with Category and Video Nodes.....	11
Figure 6 – MMG with Category, Video, and Keyframe Nodes.....	11
Figure 7 – MMG with Category, Video, Keyframe, and Segment Nodes.....	12
Figure 8 – MMG with Category, Video, Keyframe, Segment, and Attribute Nodes	12
Figure 9 – Populated MMG with an Uncategorized Video	13

LIST OF TABLES

Table 1 - Uniform Frame Picking Correctness Rate (%).....	20
Table 2 - Barne's Frame Picking Correctness Rate (%).....	20

ABSTRACT

As we move further in to the information age, the problem is no longer the lack of availability of information, but rather the ability to find the right piece of information in the overabundance of everything available. This is made even more difficult when the piece of information being sought is not text, but a more complicated form of media such as an image or video. We propose a method to help find video based off similarity to other video, as well as automatically categorize this video based off an archetypal categorization, such as genre. We do this by leveraging existing algorithms, which find interesting frames in video, and then doing low level analysis on each of these frames. This allows us to utilize well researched areas of computer vision in order to expand on a comparably sparse field.

INTRODUCTION

We have reached a point in time where information is both extremely valuable, but also overabundant. It is becoming exceedingly difficult to sift through the large amounts of information to find what you are looking for, especially when dealing with rich media such as images, video, or sound. The complexity of the problem only increases when you have no natural language context or annotation to associate with the media. Because of this, we find particularly interesting the subset of this problem that deals with the comparison and categorization of media using the innate features of that media.

While it is not a trivial problem, the task of comparing images is well identified in the area of Computer Vision and many solutions have been proposed. However, there has been little work in the area of comparing and categorizing video, especially using methods unaided by humans. We think it is especially important to focus on solutions that do not require human input after training, or else the applications of the solution will be greatly limited.

With this in mind, we will propose a solution to computationally categorize a library of video. We will first give an overview of related work in the field. Next, we will detail our solution and provide some results from testing that solution. Lastly, we will analyze those results and conclude with some final thoughts.

PREVIOUS WORK

Automatic Video Categorization

As mentioned previously, we found very little work already completed in the area of automatic video categorization, however there has been some. In [1], Truong and Dorai extend a study they did on editing effects, motion, and colors used in video to automatically categorize videos. The analysis of these features was inspired by the common use of the features within a genre as a technique to cause certain emotions within viewers. They then analyzed the trends of each of the features in various genres to help understand similarities, dissimilarities, and confusions between genres.

In [2], Pan and Faloutsos create a tool which analyzes video, still frames, and audio to create a vocabulary describing a video that is similar to a natural language vocabulary. A genre, or class, of video can then be described by a vocabulary, or set of basis functions, found when doing Independent Component Analysis on the features of videos. This is done through a compression method, finding the bases of a class that can be best used to reconstruct a video clip with the smallest error. The videos can then be mined through the vocabulary of each video or through the classes.

Media Categorization

In a different paper, [3], Pan, et al. describe a generalized method for comparing any type of media. This method creates a “Multi-Media Graph” (MMG) to provide a framework for media categorization abstracted from the actual method of comparison. We will go over in detail how this graph is created and structured later in the paper. The

graph can then be traversed using any number of graph traversal algorithms to find coefficients for how related two pieces of media are to each other.

We were immediately attracted to the generalized nature of the solution presented by Pan, et al., as well as its direct applicability to video. One of the attributes of this method that we find to be the most powerful is the independence of the actual attribute comparison from the framework; any metric we implement for measuring a difference between videos can be implemented in to the MMG. With the MMG in mind, it becomes very easy to separate our problem in to two distinct parts: finding descriptive images from video, and then comparing the images found. While neither of these two problems is trivial, they both have a notable research record.

Images from Video

Finding useful images from a video is a very interesting problem in its own right and has many applications completely separate from the problem we are trying to solve. Because of this, we found two primary motivations for work in this area: finding useful frames for intelligently traversing through video, and creating a descriptive summary of video using still frames.

A method motivated by the previous is described in [4], in which a continuous still “tapestry” is produced that describes a variable timeline of the video based off a temporal scale (Figure 1). This solution is very interesting as a means of video traversal, but we are primarily interested in their means for selecting frames to include in the tapestry. This is achieved by minimizing a function outlined by Simakov, et al:

Figure 1 – A “video tapestry” from [4]



$$d_{BDS}(S, T) = \frac{1}{N_S} \sum_{s \subset S} \min_{t \subset T} D(s, t) + \frac{1}{N_T} \sum_{t \subset T} \min_{s \subset S} D(s, t)$$

where S is the source, or original image, T is the target image, small rectangular image patches s and t are sampled from the source and target images, and the number of source and target patches are N_S and N_T , respectively. The patches are of fixed size: we use 7×7 patches in our implementation. The distance $D(s; t)$ is the distance in color space between these square patches: we use L^2 distance [Euclidean distance] in RGB space. When retargeting, the source image S will have different dimensions than the retargeted image T .

As a first level optimization, Barnes, et al. use entire frames for the image patches, creating a set of interesting frames. The function is then minimized again, using the set of frames as the source and the patches as the target to create a set of interesting content from the frames.

A different method, [5], which was motivated by creating descriptive summaries that look similar to stained glass windows (Figure 2), takes a slightly different approach to finding

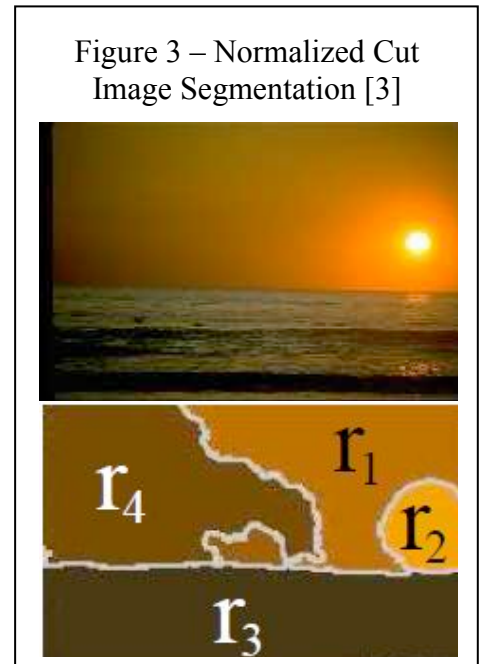


interesting portions of video. First, the video is segmented in to clips of similar frames using techniques such as color histograms or variations of pixel-wise differences. Next, regions of high importance are found. This is done by arranging the frames in a 3D space (x-y-time) and then creating bounding 3D rectangular boxes around pixels of high velocity in the 3D space. The velocity of a pixel is its change in luminance between video frames. These bounding boxes can then be used to describe interesting portions of a series of frames, which can then be used to create a descriptive still image summary of a video.

Image Comparison

Now that we have described some methods for finding useful still frames in a video, we will explore some methods for comparing and categorizing images, and which can be used to compare our useful frames. There is a wide variety of image comparison algorithms, but in our search for different methods we noticed a strong prevalence of semantically based algorithms that tie some sort of vocabulary to images, portions of images, or sets of images. Obviously many low level measures exist for quantifying pixel data between images, but they seem to be trivial enough that there is not very much in depth research on those methods. Below is a series of interesting methods for image comparison to help set the context for the state of the field as well as present possible method that can be implemented as measures in our solution.

Chen and Wang present a method of image categorization, [6], in which images are represented as bags containing instances. A bag is considered positive if it contains at least one instance, otherwise it is negative. Each image is segmented (Figure 3) and each segment, represented by an instance, is described by a feature vector. The algorithm learns a series of instance prototypes and creates a mapping from bags to points in the bag feature space. Vector machines are then trained in the bag feature space to detect the instance of regions in images. When vector machines are trained, they construct a series of binary separations in a high dimensional space. These separations essentially segment the space, allowing for each segment to be considered an instance in our application. Lastly, these vector machines are used to detect the instance of new regions, by using the feature vectors to see in which instance space the region falls.



Shotton, et al. introduce a new low level feature analysis called texton forests[7]. Texton forests are decision trees that act on the pixels of an image to represent semantic textons. Semantic textons are square patches of an image, centered at each pixel with one of four functions applied to it. The semantic textons can then be used to generate both local and global information about the image. The global information is used for categorization, and the local information is used for segmentation.

Li, et al. demonstrate a method of categorizing images through statistical modeling [8]. In their approach, a set of training images for each category is used to

generate a two dimensional multiresolution hidden Markov model. Image being used for training or categorization are used to generate a set of feature vectors extracted at different resolutions and arranged on a pyramid grid. In categorization, an image's pyramid is run through each of the categorical models, which output a likelihood of a match. The likelihood values are used to find statistical significance of each category for that image, and thus indentifying the image as a member of any category that is significant.

Barnard, et al. present a number of different categorization models in [9]. They introduce Shi and Malik's normalized cut segmentation algorithm, [10], and explain a set of 40 features generated for each image used in the models. The first model they present is for annotating images and is the "Multi-Modal Hierarchical Aspect Model". In this model, a tree is created from a training set and then traversed based off feature values. The tree is built such that the greater the depth traversed in to the tree, the more specific the annotation. They also present a "Discrete Data Translation" model, which is essentially a one to one translation of features to annotations. A feature space is created and trained using an annotated training set, and regions on the feature space are mapped to particular annotations. An image is then matched to an annotation if its feature vector falls within the region. They go on to explain a few other methods of annotation, as well as discuss the details of properly evaluating automatic annotations.

Wenyin, et al., [11], describe a system for semi-automatically annotating images. In their system a user searches for an image through keywords. The system then returns an image using a search algorithm. The user can give feedback as to the correctness of the keywords' correlation to the returned image. If the user gives positive feedback,

annotations that were not already assigned to the image are added. The system is designed so that any feature based search algorithm can be used, but the initial implementation just used a simple low level feature vector matching algorithm.

SOLUTION

Multi-Media Graph

Our solution to the automatic categorization problem is primarily based around the MMG, presented in [3]. We decided to utilize this approach for a variety of reasons, but we were particularly attracted to the intuitiveness, abstraction, and modularity that MMGs provide. The intuitiveness of MMGs make the solution easier to conceptualize, which in turn makes it easier to find and solve issues. We will exemplify this later in the paper when we address some of the issues we faced.

We found the abstraction and modularity of MMGs to go hand in hand to provide many benefits. These include the innate separation of the low level parts of the solution, such as the methods of segmentation and low level image analysis, from the high level parts, such as how the relation values between videos are being calculated. This allows us to experiment with the different parts of the solution individually, for example we can easily substitute different methods of frame picking or choose differently low level features to analyze. Following from this, we can now combine different methods of comparison to generate a relation value, rather than having to evaluate an individual method.

We will now detail our MMG implementation. We generate our graph as follows:

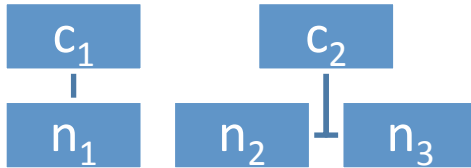
1. Start with empty directed graph G , this will be our MMG
2. Place c disconnected nodes on G , each representing a different category. We will refer to these nodes as “category nodes”.

Figure 4 - MMG with Category Nodes



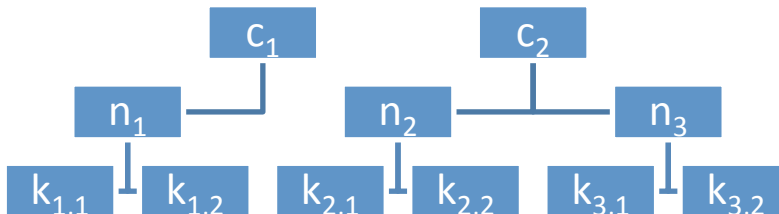
3. Place n nodes on G , each representing a different video with a known category, and for each of the n nodes create a bi-directional edge between that node and one of the c nodes representing its respective category. We will refer to these nodes as “video nodes”.

Figure 5 - MMG with Category and Video Nodes



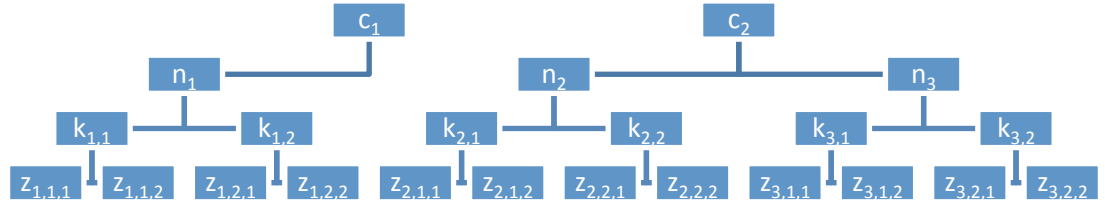
4. Run an algorithm to pick k keyframes from each video, and add k nodes per video to G , each connected by a bi-directional edge to its respective video. We will refer to these nodes as “keyframe nodes”.

Figure 6 - MMG with Category, Video, and Keyframe Nodes



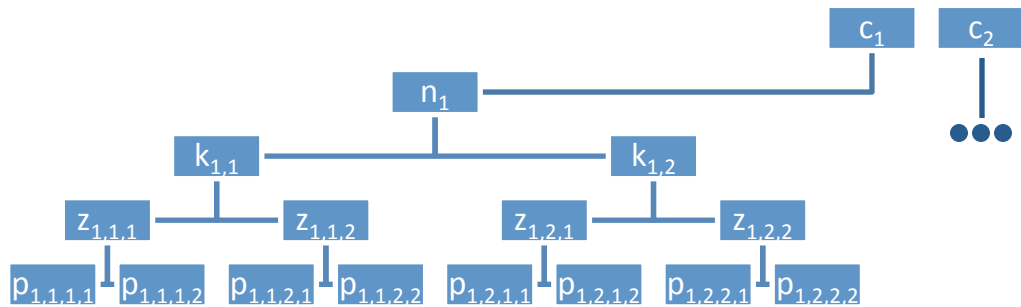
- Run an algorithm to separate each keyframe in to Z keyframe segments, and add Z nodes per keyframe to G , each connected by a bi-directional edge to its respective keyframe. We will refer to these as “keyframe segment nodes”.

Figure 7 - MMG with Category, Video, Keyframe, and Segment Nodes



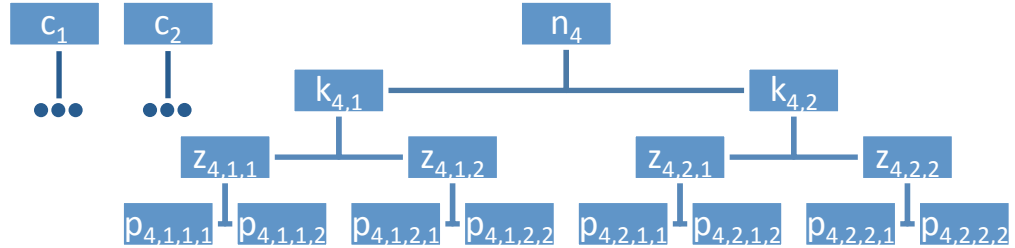
- Run p algorithms on each of the keyframe segments to generate values for p low-level features, and add p nodes per keyframe segment to G , each connected by a bi-directional edge to its respective keyframe segment. We will refer to these nodes as “attribute nodes”.

Figure 8- MMG with Category, Video, Keyframe, Segment, and Attribute Nodes



- For every video to be categorized, complete steps 3-6, but do not create an edge between the video node and a category node

Figure 9 - Populated MMG with an Uncategorized Video



- For each attribute node q , find the j number of attribute nodes, of the same low-level feature type, that are closest to q by comparing the Euclidean distance of the feature values of each node. Add j directed edges to G , from q to each of the j closest attribute nodes. We will refer to these edges as “parameter edges”.

Once our graph is generated, we traverse our graph using the random walks with restarts method, which allows us to find the probability of traversing from one node to every other node. This traversal works by randomly choosing to travel to a node adjacent to our current node or randomly choosing to restart traversal back at the initial node. We chose the random walks with restarts graph traversal algorithm, because it was recommended in [3] to efficiently provide correct results. The algorithm generally works by generating a normalized adjacency matrix of G , which is used to calculate a steady-state vector that describes the probability of traveling from one node to every other node. We take this probability as the relevance value between two nodes. While this means we must regenerate the adjacency matrix every time G changes, this also means we only need to calculate the steady-state vector for the node we are currently classifying. In

detail, the algorithm works as follows [3]:

Given an MMG graph, G , an object, O_q , and a restart probability, c :

1. Let $V_q=0$, for all its N entries, except a '1' for the q th entry.
2. Normalize the adjacency matrix of G , A , by column. That is, make each column sum to 1.
3. Initialize $U_q=V_q$.
4. While(U_q has not converged)
 - a. $U_q = (1-c)A - U_q + cV_q$

Frame Picking

We have used two methods of keyframe picking for comparison. The first method is just uniform sampling, to provide a benchmark for comparison. In our implementation, the keyframes are separated by s frames, and start from the first frame. The value for s is found by the following function:

$s(v,k) = \text{floor}(l(v)/k)$; l is the number of frames in video v , and k is the number of keyframes we are choosing.

Our second method of keyframe picking is a keyframe clustering algorithm described by [4]. This method evaluates context change from frame to frame and then globally searches for keyframes that describe the most difference between frames. This algorithm was previously described in detail in the Images from Video section.

Image Segmentation

To segment our keyframes, we decided to utilize the Normalized Cuts algorithm developed by Shi and Malik. This algorithm has become a standard method of image

segmentation in the computer vision field. Their solution approaches the segmentation as a graph partitioning problem. They attempt to partition the graph globally, rather than locally, such that they maximize the dissimilarities between segments and maximize the similarities within segments. This is done efficiently by solving for eigenvalues. The details of their algorithm can be found in [10].

Low Level Analysis

At the most basic level of our solution, we are doing low level analysis on each of the keyframe segments. We are evaluating average RGB values, area of segment, and x-y location of segment. This could easily be expanded to include other values such as mean luminosity, bounding length of segment, etc. For each of these metrics, a parameter node is created for each segment node on our MMG. After these nodes are added to the graph, for each node, x , a k number of edges are created between x and a k number of nodes. These nodes are the k number of nodes closest to x in the parameter space for that type of low level metric.

While we only generate parameter nodes for each of our segments, this could be expanded to evaluate features of parts of the video higher in the graph hierarchy. For example, parameter nodes could be created that are attached to each video node describing the length of the video, or parameter nodes could be created that are attached to each keyframe node that describe a quality of the sound in the video segment from that keyframe to the following keyframe. So while we are choosing to only use low level image analysis as our parameters, we hope it is obvious that the solution is not limited to this.

Categorization

With our graph generated, to identify the category for a video we simply inspect the steady-state vector for that video's video node. The relative likelihood of belonging to each category is the steady state value for each respective category node. This means once our graph is generated, we only have to calculate our normalized adjacency matrix and then the steady state vectors for the videos we wish to categorize.

LIMITATIONS

We ran in to a series of problems while developing this solution. We would like to describe what these problems were, and address how we overcame the problem or what compromises were made to work around the issue.

First, working in a 32 bit Windows environment, we didn't have enough memory space to store the full bitmap images for long videos, meaning longer than about 10 minutes, depending on frame rate and resolution. This is because a matrix fully populated with the bitmap values was required to generate the affinity matrix (which was also quite large) in the keyframe clustering algorithm outlined in [4]. We worked around this issue by choosing videos from our database that were short enough to not meet our space limitations. Another possible solution would be to arbitrarily reduce resolution or frame rate using a standard algorithm. As most of the videos we used were already stored at a compressed bitrate, we speculate that as long as a certain level of quality is preserved this won't have a great affect on the algorithm.

Another major issue we have had to deal with was a large compute time for videos of any notable length. Most of this time was spent generating the affinity matrix in the keyframe clustering algorithm outlined in [4]. Since the affinity matrix is symmetric we greatly reduced the compute time by only computing one side of the diagonal and then copying the proper values for the rest of the matrix. In addition, we chose to estimate the values in the affinity matrix as only the time difference term when the color difference term became trivial in comparison.

We have also had issues with general stability while computing. These have

mostly been caused by our space limitations. To help reduce the impact of any crashes or failed computations we created a fairly robust system that saves the values of computations at each step of the process to minimize any time lost, and that will repeat each step until every part of that step finishes successfully. If our solution were to be expanded as a commercial product, these issues would need to be addressed more directly, but for our proof of concept and basic experiments these workarounds have served our purposes.

RESULTS

In order to evaluate the effectiveness of our method, we are using the archive.org video database. We first developed a training set of nine different categories (Animation and Cartoons, Arts and Music, Computers and Technology, Cultural and Academic Films, Ephemeral Films, News and Public Affairs, Spirituality and Religion, Sports Videos, and Videogame Videos). The training set includes ten videos from each category, resulting in ninety total videos. We then use a testing set of 18 different videos, two from each category, but which are added to the graph as uncategorized video nodes to test the correctness of our solution. We tested the solution multiple times, while adjusting a series of variables: the frame picking method, the number of edges computed per parameter node, and the traversal restart probability. These are the correctness percentages for each of these tests:

Table 1 - Uniform Frame Picking Correctness Rate (%):

Restart Probability	Parameter Edges	5		10		50	
		Top 1	Top 3	Top 1	Top 3	Top 1	Top 3
0.0		27.8	77.8	22.2	72.2	11.1	50.0
0.01		22.2	77.8	27.8	72.2	11.1	50.0
0.1		27.8	72.2	22.2	77.8	11.1	61.1

Table 2 - Barne's Frame Picking Correctness Rate (%):

Restart Probability	Parameter Edges	5		10		50	
		Top 1	Top 3	Top 1	Top 3	Top 1	Top 3
0.0		16.7	44.4	16.7	61.1	22.2	44.4
0.01		27.8	50.0	27.8	61.1	22.2	50.0
0.1		22.2	55.6	27.8	66.7	22.2	50.0

ANALYSIS

As displayed by the results, the uniform sampling method of picking keyframes seems to be more effective for our solution than Barnes, et al.'s algorithm[4]. We find this surprising because Barnes, et al.'s algorithm is designed to choose keyframes which provide semantic insight in to the meaning of the video being keyframed. The most effective set of parameters was using uniform sampling with a 0.0 probability of restart during traversal and 5 parameter edges created. This configuration chose the correct category as the first choice 27.7% of the time, and chose the correct category as one of the top three choices 77.7% of the time. These are notable findings since both of these values are well above chance (11.1% and 37.9%, respectively). We believe the correctness of the algorithm would continue to increase if more low level parameters were added, as well as some global parameters that are descriptive of whole frames or entire videos.

CONCLUSION

In this paper we outlined the problem of automatic video categorization. We discussed the current state of the field, and presented our solution to the problem. After implementing our solution we tested our design by using an open video database. Our solution proved to be viable, providing result notably greater than chance, but needs to be perfected before it can be tested to perform at a production level. We think this could most effectively be done by adding more parameters for analysis, and especially so by expanding in to parameters that measure video dynamics and audio qualities. In future work we would be interested in exploring the viability of maximizing algorithm efficiency, even through estimation, to develop an algorithm that can run at close to real time.

REFERENCES

- [1] B. T. Trong and C. Dorai, "Automatic Genre Identification for Content-Based Video Categorization," in , Barcelona, Spain, 2000.
- [2] J.-Y. Pan and C. Faloutsos, "VideoCube: a novel tool for video mining and classification," in , Singapore, 2002.
- [3] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic Multimedia Cross-modal Correlation Discovery," in , Seattle, WA, 2004.
- [4] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein, "Video Tapestries with Continuous Temporal Zoom," 2010.
- [5] P. Chiu, A. Girgensohn, and Q. Liu, "Stained-Glass Visualization for Highly Condensed Video Summaries," in , Palo Alto, CA.
- [6] Y. Chen, J. Z. Wang, and D. Geman, "Image Categorization by Learning and Reasoning with Regions," 2004.
- [7] J. Shotton, M. Johnson, and R. Cipolla, "Semantic Texton Forests for Image Categorization and Segmentation," in , Anchorage, AK, 2008.
- [8] J. Li and J. Z. Wang, "Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach," in , 2003.
- [9] K. Barnard, et al., "Matching Words and Pictures," 2003.
- [10] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," 2000.
- [11] L. Wenyin, et al., "Semi-Automatic Image Annotation," in , 2001.
- [12] P. G. Doyle and J. L. Snell, "Random walks and electric networks," 2006.