A Scalable Multimedia Content Processing Framework with Application to TV Shopping

Fausto C. Fleites
Florida International University, fflei001@cs.fiu.edu

DOI: 10.25148/etd.FI14071113
Follow this and additional works at: https://digitalcommons.fiu.edu/etd

Recommended Citation
https://digitalcommons.fiu.edu/etd/1452
FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A SCALABLE MULTIMEDIA CONTENT PROCESSING FRAMEWORK WITH
APPLICATION TO TV SHOPPING

A dissertation submitted in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY
in
COMPUTER SCIENCE
by
Fausto C. Fleites

2014
To: Dean Amir Mirmiran  
College of Engineering and Computing

This dissertation, written by Fausto C. Fleites, and entitled A Scalable Multime-
dia Content Processing Framework with Application to TV Shopping, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Jainendra K. Navlakha

Xudong He

Keqi Zhang

Mei-Ling Shyu

Nagarajan Prabakar

Shu-Ching Chen, Major Professor

Date of Defense: June 12, 2014

The dissertation of Fausto C. Fleites is approved.

Dean Amir Mirmiran  
College of Engineering and Computing

Dean Lakshmi N. Reddi  
University Graduate School

Florida International University, 2014
DEDICATION

To my family
ACKNOWLEDGMENTS

Hereby, I express my utmost gratitude to my advisor Professor Shu-Ching Chen for this invaluable guidance, encouragement, and support over so many years of research. In addition, I thank the suggestions provided by professors Xudong He, Jainendra K Navlakha, and Nagarajan Prabakar of the School of Computing and Information Sciences, Professor Keqi Zhang of Department of Environmental Studies and International Hurricane Research Center, and Professor Mei-Ling Shyu of the Department of Electrical and Computer Engineering at University of Miami.

Secondly, I gratefully acknowledge the financial support I have received from the U.S. Department of Homeland Security under grant Award Number 2010-ST-062-000039 and the U.S. Department of Education’s Graduate Assistance in Areas of National Need (GAANN) Fellowship.

Last but not least, I thank my family for their support and love during so many sleepless nights, for which this dissertation could not have been completed. Additionally, I thank my friends and colleagues from the Distributed Multimedia Information Systems Laboratory, in particular, Min Chen, Na Zhao, Kasturi Chatterjee, Yimin Yang, and Hsin-Yu Ha for their help.
The advent of smart TVs has reshaped the TV-consumer interaction by combining TVs with mobile-like applications and access to the Internet. However, consumers are still unable to seamlessly interact with the contents being streamed. An example of such limitation is TV shopping, in which a consumer makes a purchase of a product or item displayed in the current TV show. Currently, consumers can only stop or pause the current show and attempt to find a similar item in the Web or an actual store by providing a description of the product. It would be more convenient if the consumer could interact with the TV to purchase interesting items.

Towards the realization of TV shopping, this dissertation proposes a scalable multimedia content processing framework. Two main challenges in TV shopping are addressed: the efficient detection of interesting products in the content stream, and the retrieval of similar products given a consumer-selected product. The proposed framework consists of three components. The first component performs computational and temporal aware multimedia abstraction. The goal of this component is to select a reduced number of frames that summarize the important information in the video stream. By both reducing the number of frames and taking into account the computational cost of the subsequent detection phase, the abstraction component allows the efficient detection of products in the stream. The second component realizes
the detection phase. It executes scalable product detection using multi-cue optimization. Additional information cues are formulated into an optimization problem that allows the detection of complex products, i.e., those that do not have a rigid form and can appear in various poses. After the second component identifies products in the video stream, the consumer can select an interesting one for which similar ones must be located in a product database. To this end, the third component of the framework consists of an efficient, multi-dimensional, tree-based indexing method for multimedia databases. The proposed index mechanism serves as the backbone of the search. Moreover, it is able to efficiently bridge the semantic gap and perception subjectivity issues during the retrieval process to provide more relevant results.
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction and Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Proposed Solution</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Computational and Temporal Aware Multimedia Abstraction</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Product Detection with Multi-cue Optimization</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3 Product Retrieval via Semantic Index Method</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Scope and Limitations</td>
<td>9</td>
</tr>
<tr>
<td>1.5 Outline</td>
<td>11</td>
</tr>
<tr>
<td>2 RELATED WORK</td>
<td>12</td>
</tr>
<tr>
<td>2.1 Video Abstraction and ROI Detection</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Context-aided Object Detection</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Tree-based Multimedia Indexing</td>
<td>22</td>
</tr>
<tr>
<td>3 OVERVIEW OF THE FRAMEWORK</td>
<td>29</td>
</tr>
<tr>
<td>3.1 Multimedia Abstraction</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Product Detection</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Distributed Detection Pipeline</td>
<td>35</td>
</tr>
<tr>
<td>3.4 Product Retrieval</td>
<td>39</td>
</tr>
<tr>
<td>4 COMPUTATIONAL AND TEMPORAL AWARE ABSTRACTION</td>
<td>42</td>
</tr>
<tr>
<td>4.1 Multimedia Abstraction Framework</td>
<td>42</td>
</tr>
<tr>
<td>4.1.1 Optimal Solution</td>
<td>47</td>
</tr>
<tr>
<td>4.2 Implementation Details</td>
<td>49</td>
</tr>
<tr>
<td>4.2.1 Preliminary Definitions</td>
<td>52</td>
</tr>
<tr>
<td>4.2.2 Proposed Solution</td>
<td>54</td>
</tr>
<tr>
<td>4.3 Experimental Results</td>
<td>56</td>
</tr>
<tr>
<td>4.3.1 ROI Effectiveness</td>
<td>59</td>
</tr>
<tr>
<td>4.3.2 Computational Scalability</td>
<td>61</td>
</tr>
<tr>
<td>4.3.3 Computational Effect of ROI Detection</td>
<td>65</td>
</tr>
<tr>
<td>4.3.4 Abstract Quality</td>
<td>68</td>
</tr>
<tr>
<td>4.3.5 Weight Parameter</td>
<td>70</td>
</tr>
<tr>
<td>4.4 Conclusions</td>
<td>70</td>
</tr>
<tr>
<td>5 PRODUCT DETECTION WITH MULTI-CUE OPTIMIZATION</td>
<td>73</td>
</tr>
<tr>
<td>5.1 Multi-cue Product Detection</td>
<td>74</td>
</tr>
<tr>
<td>5.1.1 Additional Cues</td>
<td>75</td>
</tr>
<tr>
<td>5.1.2 Optimal Solution</td>
<td>79</td>
</tr>
<tr>
<td>5.2 Experiments and Results</td>
<td>80</td>
</tr>
<tr>
<td>5.2.1 Detection Performance</td>
<td>82</td>
</tr>
<tr>
<td>TABLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>4.1</td>
<td>59</td>
</tr>
<tr>
<td>4.2</td>
<td>60</td>
</tr>
<tr>
<td>5.1</td>
<td>83</td>
</tr>
<tr>
<td>5.2</td>
<td>83</td>
</tr>
<tr>
<td>5.3</td>
<td>89</td>
</tr>
<tr>
<td>6.1</td>
<td>99</td>
</tr>
<tr>
<td>6.2</td>
<td>99</td>
</tr>
<tr>
<td>6.3</td>
<td>118</td>
</tr>
<tr>
<td>7.1</td>
<td>134</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>3.1</td>
<td>Control flow of the proposed framework.</td>
</tr>
<tr>
<td>3.2</td>
<td>High-level design of the proposed framework.</td>
</tr>
<tr>
<td>3.3</td>
<td>Design of the Multimedia Abstraction component.</td>
</tr>
<tr>
<td>3.4</td>
<td>Design of the Product Detection component.</td>
</tr>
<tr>
<td>3.5</td>
<td>High-level design overview of the proposed framework.</td>
</tr>
<tr>
<td>3.6</td>
<td>Design of detection phase in Storm.</td>
</tr>
<tr>
<td>4.1</td>
<td>Example video of four frames depicting two views of $O^*$.</td>
</tr>
<tr>
<td>4.2</td>
<td>Illustrations of a frame’s ROI.</td>
</tr>
<tr>
<td>4.3</td>
<td>Examples of motion scatterness. Actual motion angles are shown rounded to the nearest 22.5 degree angle, each color represents a different motion angle, and the block size is 35 × 35 pixels; (a) corresponds to the frame of Figure 5.3a, and (b) to the frame of Figure 4.1a.</td>
</tr>
<tr>
<td>4.4</td>
<td>Illustration of steps for detecting zoom motion. Each color represents a different motion angle, and the block size is 35 × 35 pixels; (a) shows the motion angles generated by block-based motion estimation for a frame global zoom, (b) shows the motion angles rounded to the nearest 45 degrees, and (c) shows in red the symmetric blocks whose combined motion is 180 degrees.</td>
</tr>
<tr>
<td>4.5</td>
<td>Sample frames from test videos.</td>
</tr>
<tr>
<td>4.6</td>
<td>Running time of the proposed approach with respect to $\gamma = 300\text{ms}$ for clip VC1 as $\gamma$ increases, with $\zeta = 0.019182956$.</td>
</tr>
<tr>
<td>4.7</td>
<td>Running time of the CUFB approach with respect to $\gamma = 300\text{ms}$ for clip VC1 as $\gamma$ increases, with $\zeta = 0.019182956$.</td>
</tr>
<tr>
<td>4.8</td>
<td>Percentage change in running time of proposed approach vs. the CUFB approach for clip VC1 as $\gamma$ increases, with $\zeta = 0.019182956$.</td>
</tr>
<tr>
<td>4.9</td>
<td>Percentage change in running time of proposed approach and the CCFB vs. the CUFB approach for clip VC1 as $\gamma$ increases, with $\zeta = 0.019182956$.</td>
</tr>
<tr>
<td>4.10</td>
<td>Number of output frames as $\gamma$ increases, $\zeta = 0.009591478$.</td>
</tr>
<tr>
<td>4.11</td>
<td>Number of output frames as $\gamma$ increases, $\zeta = 0.019182956$.</td>
</tr>
<tr>
<td>4.12</td>
<td>Quality improvement as the number of output frames increases.</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>4.13</td>
<td>Quality improvement for clip VC1 as the weight parameter $\mu$ varies from 0.0 through 1.0.</td>
</tr>
<tr>
<td>5.1</td>
<td>Example of how the appearance cue helps discern between possible object detections.</td>
</tr>
<tr>
<td>5.2</td>
<td>Example of topological relationship between bags and faces in a fashion show. Yellow bounding boxes depict object detections.</td>
</tr>
<tr>
<td>5.3</td>
<td>Illustration of the optimal path.</td>
</tr>
<tr>
<td>5.4</td>
<td>Sample frames from test videos.</td>
</tr>
<tr>
<td>5.5</td>
<td>Precision-recall curve for VC1.</td>
</tr>
<tr>
<td>5.6</td>
<td>Precision-recall curve for VC2.</td>
</tr>
<tr>
<td>5.7</td>
<td>Precision-recall curve for VC3.</td>
</tr>
<tr>
<td>5.8</td>
<td>Examples of predicted bounding boxes for handbags in clip VC1. The plain product detection approach predicts all bounding boxes (both yellow and red ones), whereas the proposed multi-cue approach only predicts the red boxes and eliminates the yellow boxes as false positives.</td>
</tr>
<tr>
<td>5.9</td>
<td>Precision improvement (%) of the proposed approach vs. the sub-optimal version.</td>
</tr>
<tr>
<td>5.10</td>
<td>Example of incorrect re-scoring by TAS approach.</td>
</tr>
<tr>
<td>5.11</td>
<td>Abstraction effect on VC1.</td>
</tr>
<tr>
<td>5.12</td>
<td>Abstraction effect on VC2.</td>
</tr>
<tr>
<td>5.13</td>
<td>Abstraction effect on VC3.</td>
</tr>
<tr>
<td>6.1</td>
<td>Example of space organization using Euclidean distance.</td>
</tr>
<tr>
<td>6.2</td>
<td>Structure of the AH$^+!!$-tree.</td>
</tr>
<tr>
<td>6.3</td>
<td>Sample images in the product database.</td>
</tr>
<tr>
<td>6.4</td>
<td>Sample foreground masks after background removal.</td>
</tr>
<tr>
<td>6.5</td>
<td>Examples of how detections of product occurrences have challenging backgrounds.</td>
</tr>
<tr>
<td>6.6</td>
<td>Examples of retrieval results.</td>
</tr>
<tr>
<td>6.7</td>
<td>Effectiveness result for range query with radius = 0.4.</td>
</tr>
<tr>
<td>6.8</td>
<td>Effectiveness result for 10NN on horses in a natural environment.</td>
</tr>
</tbody>
</table>
6.9  I/O cost on $\mathcal{D}_1$  ...................................................... 124
6.10  I/O cost on $\mathcal{D}_2$  ...................................................... 126
6.11  Real-time cost on $\mathcal{D}_2$ using range queries with radius 0.3.  .... 127
6.12  AH-Tree’s implementation in PostgreSQL  ...................................... 128
7.1   Sample ROI detections on basketball clip.  ................................. 134
1.1 Introduction and Motivation

Recently, smart TVs have raised the TV experience to a new level by combining the TV, Internet, and PC technologies. Consumers are able to browse the web, interact with a variety of applications, and watch TV channels. Nevertheless, smart TVs still do not allow consumers to seamlessly interact with the contents being streamed. One example of such a limitation is TV shopping. In this use case, the consumer interacts with the TV to purchase an interesting product that is displayed in the current show. For instance, consider the consumer is watching a fashion show. The TV system detects handbags and apparel in the content stream and notifies the consumer via a non-intrusive notification. When the consumer sees the notification, he or she can activate it and is then presented with the list of products detected in the show over a time window. If the consumer is interested in an item, he or she selects the item and proceeds to purchase it. The purchase can be realized, for example, by providing the detected items as search input in an online store. Clearly, such a content-enabled application is of commercial value and would significantly enrich the interaction of consumers with their TV systems.

Two main tasks must be implemented in a system for TV shopping applications. The first task is the detection of interesting products in the content stream. The need for the products to be automatically detected stems from the fact that the interaction mechanisms consumers have with their smart TVs are limited to remote controls. More recently, though, some smart TV sets are able to recognize voice and gesture commands [1]. Nonetheless, these interaction mechanisms are by no means ideal for consumers to specify the products they are interested in, much less to delineate the
boundary of the products in the TV screen. The second task arises once the consumer selects one of the detected products. The selection is specified by a bounding box on some video frame, which represents the product image. This image must then be searched in an online product store to be able to present consumers with possible items for purchase.

The automatic detection of products in the video stream poses a significant detection challenge. Particularly, the system must be able to detect complex object, i.e., those that do not have a rigid form or can appear in a variety of poses. In the fashion show use case, detecting handbags is very difficult as handbags do not have a definitive shape, can present deformations, be occluded by hands or arms, and appear in many poses. The object detection task can be addressed by considering the video stream as an unrelated sequence of frames and perform static object detection [2, 3, 4, 5, 6]. On the other hand, it can be tackled by utilizing the additional information offered by the progression of the video sequence [7, 8, 9, 10, 11, 12, 13]. Nevertheless, most of these approaches fail to detect complex objects and perform well only on ideal conditions. In the case of video based approaches, these mostly concentrate on using motion information to detect moving objects, which may not work well for difficult objects; in the case of handbags, the motion of the bag would be masked by the motion of the person carrying the bag.

Furthermore, the computational costs of the detection phase must be taken into account. Naively executing object detectors for each frame of the video stream is not a practical approach, as it does not consider computational costs, a critical factor to providing effective and responsive TV-shopping applications. The detection cost is influenced by the number of object detectors to be applied and the complexity of the objects to be detected. In the fashion show example, the detection phase may require detecting clothing, human faces, and bags, which demands the execution of multiple
object detectors that significantly increase the computational cost. Complex objects pose an even higher level of difficulty. Hence, there is a trade-off between accuracy and detection speed. Low-complexity object detectors are not sufficiently powerful, and high-complexity object detectors require significant computational resources. Moving object detectors can be used, but detection using only motion estimation is not sufficient.

The second task of searching a product image in a product database is related to the task of content-based multimedia information retrieval (CBMIR), specifically image retrieval. CBMIR focuses on understanding the semantics and retrieval of multimedia data, and its importance is highlighted by the TRECVID conference series sponsored by NIST [14, 15]. Two inherent challenges manifest in CBMIR. One is the subjectivity of the users’ perception, where different users may perceive the meaning of the multimedia data differently and may also change their perceptions over time. The other challenge is the well-known problem of the semantic gap between low-level features and high-level semantic concepts [16]. Visual data are usually modeled using low-level color [17], shape [18], and texture [19] features, but such information may not be discriminative enough at the concept level. In the fashion show example, searching for a product image may easily return unrelated products with similar color distributions.

From an infrastructure perspective, solutions to the aforementioned tasks must consider efficiency and scalability. One aspect of the TV shopping use case is that it must process large amounts of video sequences for the detection phase. Such a large-scale processing can be performed in dedicated servers or in the cloud, and product detection results would then be streamed to the smart TV sets. In the domain of large-scale batch processing, the popular computing framework MapReduce (MR) [20] is the choice for distributed computing as it offers an easy-to-use programming
model and supporting processing framework for data-intensive applications. For real-time processing, Storm [21] is a more recent computing framework that is becoming very popular as it can complement MR for real-time analysis and processing.

1.2 Proposed Solution

This dissertation proposes a scalable multimedia content processing framework for TV shopping applications. The components of the framework address the two main tasks required in the TV shopping use case: the detection of products in the video stream and the retrieval of similar products. The first two components address the first task. Taking into consideration computational costs of the detection phase, the first component performs multimedia abstraction to select a subset of the video frames that best summarizes the information in the video sequence. The second component executes product detection on the abstract frames and utilizes additional cues of information to formulate an optimization problem that allows the detection of complex objects. Lastly, the third component consists of an efficient, multi-dimensional tree-based index method to support product retrieval.

1.2.1 Computational and Temporal Aware Multimedia Abstraction

Given the humongous amount of videos that must be processed to support TV-enabled shopping, it is not feasible to apply the object detection phase to every frame in the input video sequence. That is, the system may not have sufficient computational resources to generate the results under a specified computational constraint. Additionally, even with sufficient computational resources, the system may benefit from
processing only a selection of the video frames that best represent the information in the video sequence, instead of processing all the frames.

To provide a scalable solution, the Multimedia Abstraction component presents a computational and temporal aware multimedia abstraction framework that supports the scalable execution of multiple and complex object detection [22]. Given computational and temporal rate constraints, the proposed framework solves an optimization problem to select the optimal video frames that best represent the video content and allows the execution of the object detection task as a computationally-bound process. The object detection task is not part of the abstraction process; the latter includes an estimate of the cost of the object detection task in the optimization problem. In this sense, the framework is computationally scalable as it can adapt to the given constraints and generate optimal abstraction results accordingly. Additionally, the framework utilizes “object views” as the basis for the frame-selection process, which depict salient information and are represented as regions of interest (ROI). In general, a ROI can be a whole frame or a region that discards background information.

1.2.2 Product Detection with Multi-cue Optimization

The significant challenge posed by complex objects gives rise to using additional information to improve the detection results. For still images, most approaches employ additional information from co-occurrence and/or spatial relationships between object labels [23, 24, 25, 26, 27, 28, 29, 30], but these do not incorporate the temporal information embedded in video sequences. Very few approaches do address the temporal information as an additional cue [31, 32, 33]. However, most of these are meant for surveillance applications, which have different requirements and scene characteristics than those of fashion shows. The latter are characterized by many shots with
varying background motion, making very difficult the differentiation of foreground motion.

Building upon the idea of utilizing additional information to improve detection results, the second component presents a generic, multi-cue product detection framework. The approach is generic as it is not tied to a specific detection approach. It makes use of multi-cue information to enhance the detection of complex objects in unconstrained video sequences, i.e., no assumptions are made about foreground or background motion in the video. Three cues are considered to detect objects of a target product class. The first one is the appearance cue, which dictates that the visual appearance of an object must represent the target product class. The second and third are topological and spatio-temporal cues, which consider the relationship between the target product class and a related, easier-to-detect object class. Within a video frame, the topological cue enforces a spatial relationship between detections of both classes. Across consecutive frames, the spatio-temporal cue assumes there is a correlation between the spatial positions of detections in both classes. The three cues are jointly considered to formulate an optimization problem that selects the best path objects of the target product class can follow in the video.

1.2.3 Product Retrieval via Semantic Index Method

The Product Retrieval component is related to the efficient indexing and retrieval of similar products given a consumer-selected product. For text data, open-source solutions such as Apache Solr and Elasticsearch provide commercial-grade, distributed indexing and search functionality. The indexing component of these solutions is powered by Apache Lucene [34], a library that offers high-performance search-engine functionality. For visual data, however, retrieval solutions are much less mature because of the aforementioned problems of the semantic gap and subjectivity of the
users’ perception. Potential solutions thus need to provide not only efficient access but also meaningful retrieval. Multi-dimensional, tree-based index structures provide an efficient solution to the problem of efficiently retrieving multimedia data [35, 36, 37, 38]. Multimedia data are usually modeled in a multi-dimensional feature space, where tree-based index structures offer effective indexing solutions. Of special interest are height-balanced, tree-based structures because they provide constant access to indexed records and data retrieval through similarity queries.

As the backbone of product retrieval, this component presents the AH+tree [39, 40], an efficient, height-balanced, multi-dimensional, tree-based index structure for multimedia data. The AH+tree utilizes high-level affinity information from the Markov Model Mediator (MMM) [41] during the retrieval process to address the semantic gap and the subjectivity of the users’ perception. Mapping low-level features to high-level concepts, affinity information captures concept-based relationships between images from the users’ perspective. The precursor of the AH+tree is the AH-Tree [36], which is the first tree-based index structure that utilizes semantic information during the retrieval process. However, the AH-Tree incurs in significant I/O overhead, a problem that is eliminated by the AH+tree. Moreover, an implementation of the AH-Tree in PostgreSQL [42], an open-source, commercial-grade DBMS, is presented, which paves the way for a similar implementation of the AH+tree [43].

1.3 Contributions

The proposed framework for TV shopping applications makes several contributions in the areas of multimedia abstraction, context-aided object detection, and multimedia retrieval:
1. A novel temporal and computationally scalable multimedia abstraction approach is proposed to support object detection. The problem of efficiently obtaining an abstract of a video is formulated as an optimization problem that takes into account computational and temporal constraints to ensure the task of object detection is performed under computational bounds. Moreover, the abstraction approach is based on regions of interest instead of whole video frames, which (a) brings computational savings when the object detection task is complex and (b) increases the quality of the abstract by eliminating background information from the optimization problem.

2. A region-of-interest detection algorithm based on motion estimation and relevant texture is proposed to detect important regions in the multimedia abstraction component. The algorithm uses block-based motion estimation and formulates an optimization problem based on the texture and changes in motion direction between adjacent motion blocks.

3. A multi-cue product detection approach is proposed for TV shopping. The framework is generic as it is not tied to specific object detection approaches. Instead, it utilizes appearance, topological, and spatio-temporal cues that make use of a related, easier-to-detect object class to improve the detection results of the target, more-difficult product class. The three cues are jointly considered to select the best path that occurrences of the target product class can follow in the video and thus eliminate false positive occurrences.

4. The AH$^+$-tree, a tree-based indexing mechanism for multimedia data is presented to efficiently retrieve multimedia data while addressing inherent challenges in multimedia information. Since the AH$^+$-tree is a multi-dimensional, height-balanced tree structure, it serves as an efficient indexing mechanism that can be incorporated into full-fledged database management systems. In fact,
an implementation of the AH-Tree, the predecessor of the AH\textsuperscript{+}-tree, is showcased in PostgreSQL. Furthermore, the incorporation of semantic information during the retrieval process allows the AH\textsuperscript{+}-tree to address the problems of semantic gap and user perception subjectivity inherent to multimedia retrieval. Compared to the AH-Tree, the AH\textsuperscript{+}-tree utilizes affinity information in a novel way to eliminate the I/O overhead of the AH-Tree while maintaining the same functionality and quality of results.

1.4 Scope and Limitations

The proposed framework has the following assumptions and limitations:

1. The problem formulation of the proposed scalable video abstraction framework assumes the function that estimates the computational cost of the object detection process is accurate. Such assumption requires the implementation of a mechanism to measure the desired object detection task, which may not be trivial to implement in cases where object detection tasks depends on external factors, such as an approach that requires an external database for face recognition. Furthermore, the external factors that may render inaccurate the estimations of the object detection task, e.g., the computer being shared with other resource-consuming processes, is not considered in the problem formulation. Hence, the object detection task should be isolated in real-world applications.

2. Low resolution videos may hinder the accuracy of the region-of-interest detection method in the proposed video abstraction framework. The proposed approach to obtain regions of interest relies on block-based motion estimation and texture analysis to differentiate areas with foreground objects, and accurately measuring motion and texture requires high-resolution videos. Nevertheless,
requiring reasonably-good quality videos is not too restrictive due to increasing trend of storage and data transmission speeds.

3. Object detection is out of the scope of this dissertation. The Product Detection component proposes a multi-cue product detection approach that acts as a pruning layer and is generic to the specific object detector utilized to detect product occurrences. For prototyping purposes, this dissertation presents detection results based on the object detector of Felzenszwalb et al. [44, 2], but any object detector can be incorporated. Consequently, limitations inherent to the chosen object detector are carried over to the Product Detection component.

4. The multi-cue product detection approach requires the definition of the topological cue. In this dissertation, a definition is provided based on the distance between the target product detections and the related objects. Nevertheless, depending on the application, other definitions can be incorporated, and/or this cue can be extended to prune out detections earlier on the algorithm. For example, based on the relative size of the target product detections and related object detections, a constraint can be used to eliminate target product detections that are too large.

5. The ability of the proposed AH$^+$-tree in addressing the semantic gap and the subjectivity of the users’ perception lies in the choice of low-level features used to represent the multimedia data and the accuracy of the mechanism used to compute affinity information. The research on these two topics is far from maturity. Nonetheless, the AH$^+$-tree is independent of the particular features and affinity mechanism, a design feature that makes the AH$^+$-tree a practical indexing tool on which several combinations of features and affinity information can be experimented with.
6. The proposed AH$^+$-tree requires the multimedia data to be represented as a multi-dimensional numerical vector. Hence, nominal or qualitative representations must be expressed in numerical form.

1.5 Outline

This dissertation is organized as follows. Chapter 2 reviews the advantages and limitations of related work in the areas of video abstraction and ROI detection, context-aided object detection, and tree-based multimedia indexing. Including details of each component, Chapter 3 provides an overall description of the proposed multimedia content processing framework for TV shopping applications. Chapter 4 presents the proposed computational and temporal aware multimedia abstraction framework that supports object detection. Chapter 5 describes the details of the multi-cue product detection approach. Addressing multimedia retrieval, Chapter 6 presents the AH$^+$-tree. Lastly, Chapter 7 concludes this dissertation and explores future research directions.
CHAPTER 2

RELATED WORK

Based on the specific challenges addressed by the proposed multimedia content processing for TV shopping, this chapter thoroughly reviews related work areas of video abstraction and ROI detection, context-aided object detection, and tree-based multimedia indexing.

2.1 Video Abstraction and ROI Detection

The proposed computational and temporal aware video abstraction framework aims at supporting efficient object detection. To achieve this goal, it jointly considers a computational constraint, which bounds the total time to be spend in abstraction and detection, and a temporal constraint, which limits the maximum number of key frames that should appear in the abstract. To the best of our knowledge, this is the first abstraction approach to jointly consider these two constraints. Related work can thus be categorized into (a) key-frame based video abstraction, (b) computationally aware object detection, and (c) ROI detection.

Video abstracts can be categorized into key-frame based approaches and video-skim based approaches. An extensive survey of the work in this area can be found in [45], and a summary is provided as follows. A video skim corresponds to a continuous sequence of frames in the video (i.e., a video segment or shot video clip), and the temporal concatenation of several of these skims represents a summary of the video. Key-frame based approaches, on the other hand, focus on selecting individual key-frames. Hence, since the proposed abstraction approach is a key-frame approach, the subsequent review skips video-skim methods.

Key-frame based abstraction methods are related to the temporal rate constraint used in the proposed abstraction framework. Random sampling offers the best ap-
approach in terms of computational efficiency; however, this approach may skip semantically important segments of the video. More complicated approaches can be formulated as optimization problems that select a subset of the frames based on some cost criteria. A primary categorization of these approaches can be made with respect to whether the desired number of abstract frames is known beforehand or not. A \textit{a priori} approaches use the number of abstract frames $k$ as input parameter and then select the $k$ frames that differ the least from the video based on the cost function \cite{46,47,48}. On the other hand, \textit{a posteriori} approaches use a dissimilarity threshold $\epsilon$ and then select the least number of frames whose dissimilarity to the video sequence based on the cost function is less than $\epsilon$ \cite{49,50,51,52}. The proposed approach falls in the category of \textit{a priori}.

These are indeed good candidates to obtain a good abstract of the input video. However, most of them use the entire area of the video frames as the basis of the abstraction approach and do not consider the computational cost of subsequent tasks such as object detection. This dissertation proposes, instead, an approach that uses regions of interest that discard most of the background information. Although it does not consider computational costs and requires supervised labeling, one recent approach that deviates from using the whole frames is that of Liu et al. \cite{53}. They proposed a supervised video abstraction framework that retains the frames containing an object of interest (OOI), e.g., cars. It can be thought of as an object tracker whose objective is to be consistent with limited ground-truth labeling. Based on salient visual information, their approach builds frame-level and window-level classifiers that are used to determine salient patches that are likely to contain the OOI. For each frame, windows are sampled where each window represents an OOI hypothesis. Subsequently, the frame-level classifiers are used to propagate label information from the
frame to the windows to learn a window-level classifier, which is used to assign scores to the salient patches. Final windows are sampled from a learned spatial distribution.

Computationally aware object detection is related to the proposed framework because the latter enables the execution of object detectors within a computationally-bound time frame. Approaches in this area can be divided in two groups. The first group proposes methods that speed up object detection without directly addressing computational constraints. Butko et al. [54] offered a method to reduce the computational cost of general-purpose object detectors in static images. Their method reduces the image scanning cost by simulating a digital fovea around fixation points that are sequentially placed at various regions to maximize the expected information gain. Greig [55] proposed a method to improve the computational performance of filter-based object detection and tracking in video sequences. The improvement consists of enhancing the standard sampling (selection of spatial and scale positions) strategy by staggering the sampling grid. Zhang et al. [56] presented an object detection and pose estimation method called probabilistic boosting network (PBN) that combines a multi-class boosting classifier for pose estimation and boosted cascade detection for object detection. Their method achieves real-time performance by incorporating a pose invariant pre-filter that quickly rejects negative instances. The second group does directly address computational constraints although they are different from the proposed approach. Very few approaches exist in this group. Vijayanarasimhan et al. [57] addressed the problem in static images and proposed a detection method that can adapt to computational constraints by selecting the image regions and features that promise the most evidence at the least cost. The method can run to completion if there are enough resources or terminate earlier and provide a reasonable result. In addition, it uses a parts-based object detection model, which does not consider pose changes and deformations. This method is not powerful enough for complex objects.
and cannot be utilized in video sequences when the computational constraint does not allow the processing of all the frames in the sequence.

Correlated with the proposed definition of ROIs, the literature on ROI detection defines ROIs as portions of multimedia documents to which audiences are more interested in than others [58]. Approaches in this area can be divided into (a) application-specific, which target applications such as video coding and conferencing; or (b) attention-based, which rely instead on user/visual attention models.

Application-specific approaches typically propose methods that are unsuitable for more general cases and inapplicable to the use case addressed in this work. In the case of video coding, most approaches sacrifice accuracy for speed. Zhang et al. [59][60] targeted ROI-based video coding for multi-view videos. Their method determines ROIs by applying a set of operations on depth, motion, and contour maps, which are obtained via threshold-based rules. Moreover, it builds upon the assumptions that (a) the camera is static, (b) moving objects are more interesting, and (c) the importance of video objects decrease as their depth decrease. Chiang et al. [61] proposed a ROI-based rate control scheme for the H.264/AVC codec. Their proposal extracts ROIs by thresholding a saliency map, which is obtained via the linear combination of motion vectors and simple contrast, texture, and skin color features. In the case of video conferencing, existing approaches heavily rely on domain knowledge. For example, Sun et al. [62] assumed a speaker centered in a classroom setting that is captured by a panoramic video system, and Liu et al. [63] presumed conversational situations to make use of skin tone detection.

Attention-based solutions are more general and built upon on modeling what users pay attention to. In fact, the proposed approach for determining ROIs can be thought as being based on a visual attention model that combines texture and motion cues. Chi et al. [64] presented a method that utilizes visual rhythm analysis as the
basis of the user attention model. Capturing temporal information, visual rhythm is a 2D thumbnail version of a video that is constructed by sampling from diagonal, anti-diagonal, vertical, and horizontal lines in the video frames. Their approach processes sampled information using threshold- and morphological-based methods to obtain smoothed data that can be classified into six attention models. The ROI is then obtained as the region that intersects the attention models across the four visual rhythms. One shortcoming of this approach is that it can only detect one ROI per frame, which could potentially include too much background pixels in frames containing small salient objects that are not spatially close. Rapantzikos et al. [65] and Guo et al. [66] utilized a visual attention model based on spatio-temporal information. Rapantzikos et al. [65] extracted intensity, salient color, and orientation features to form a saliency volume. The latter is then post-processed via thresholding, morphological filtering, and distance and watershed transforms to segment the frame into smaller regions that are sorted based on their average saliency. Instead of motion information, the temporal aspect of the attention model is realized via the post-processing operations applied on the saliency volume. Guo et al. [66] proposed an approach that obtains ROIs by thresholding a saliency map. Video frames are represented by quaternions that encode color, intensity, and motion features, and the map is generated using the phase spectrum of quaternion Fourier Transform (PQFT). One drawback in these two approaches is that they use elementary mechanisms to obtain motion/temporal information, which only works in ideal situations. In addition, global motion is not differentiated from local motion so saliency results can represent camera motion instead of object motion.
2.2 Context-aided Object Detection

The utilization of additional information has been approached in recent years to try to overcome the challenges posed by object detection. Plenty of approaches tackle this task in static images utilizing co-occurrence and/or spatial relationships \[67, 68, 23, 24, 25, 26, 27, 28, 29, 30\]. However, very few approaches address this problem for videos. These additionally include temporal relationships to exploit the inherent spatio-temporal information \[31, 32, 33\].

Co-Occurrence and Spatial Relationships

Even though they utilize additional information to aid object detection, the following approaches are intended for image data and thus do not take into consideration temporal information.

Rabinovich et al. \[69\] present an object detection framework that utilizes co-occurrence relationships between object labels to improve the detection results. In fact, the way the authors utilize additional information as a post-detection step suggests it can be used to enhance any off-the-shelf object categorization method. The detection mechanism the authors employ consists of image segmentation, followed by the bag of features (BoF) classification model. Training images are not segmented but rather just represented via BoF. Segmentation is then applied to test images, and the label of an image segment is that of the training image closest to the segment according to the $L_1$ distance. Segment labels are ranked based on a confidence measure based on the two “best” labels for the segment. As a post-processing stage, semantic context is incorporated via a conditional random field (CRF) to enforce label agreement between the segments in the test image. The semantic relationships between
object labels are represented via context matrices, which are constructed using the co-occurrence frequency of object labels in the training data.

Galleguillos et al. [25] extend the work of Rabinovich et al. [69] to incorporate spatial relationships as an additional source of information. Since objects are initially segmented based on their appearance, their method is called CoLA for co-occurrence, location, and appearance. Similar to Rabinovich et al., the contextual information is also integrated via a conditional random field (CRF) that aims at maximizing the agreement between object labels. However, context matrices are extended to include spatial information. The latter consists of four relationships: above, below, inside, and around, where each relationship is represented via a context matrix that encodes corresponding co-occurrence information.

Heitz et al. [24] propose a probabilistic framework that models contextual information to enhance the detection results of off-the-shelf detectors. It does so by re-scoring the detections scores with the intent of lowering the scores of false-positive detections. The framework captures contextual relationships between “stuff” (i.e., regions with homogeneous or repetitive patterns) and “things” (i.e., monolithic objects) and does not require manual labeling of the “stuff” regions, only limited ground-truth labeling of object detections. In this sense, this approach combines co-occurrence and spatial relationships. Image regions (“stuff”) are provided as input to this approach, and the framework clusters the regions based on their ability to server as context for object detection. A probabilistic model is then learned to link detections with “stuff” clusters.

Wang et al. [29] introduce a multi-class detection framework based on multi-class Hough forests and contextual constraints, where the contextual information stems from spatial relationships between the object labels. Inspired by Galleguillos et al. [25], the authors classify spatial relationships into five groups: ontop, far, next to,
above, and below. Each group is modeled by a relationship matrix, i.e., a context matrix in Galleguillos’s work, that captures the probability distribution of the object classes via co-occurrence information.

Zheng et al. [30] propose a context-modeling framework that extends the idea of Heitz et al. [24]. The authors categorizes types of contexts as “Scene-Thing”, “Stuff-Thing”, “Thing-Thing”, and “Thing-Stuff”, and their framework models “Thing-Thing” and “Thing-Stuff” contexts by learning co-occurrence and spatial contextual relationships. Contextual information is represented via a polar geometric context descriptor. The framework then utilizes a maximum margin context (MMC) model to evaluate the usefulness of contextual information and fuse context information with appearance information. It does so by discriminatively learning a context risk function that measures the rank information between true positives and false positives detections. The empirical results on several PASCAL VOC datasets show that the framework outperforms that of Heitz et al. [24] for some concepts and achieves similar performance for others.

**Additional Temporal Relationship**

The following approaches target video data and include temporal information. However, most of them target surveillance applications, which have different requirements and assumptions (e.g., fixed camera) that are invalid in the TV shopping use case.

Sheikh et al. [31] introduce an object detection framework for surveillance videos that is able to model dynamic backgrounds. Different from previous approaches, the framework does not model pixel intensities as independent random variables; instead, the framework models the background as a single probability density using non-parametric kernel density estimation over joint location-color representation of image pixels. Object detection is approached by also maintaining a foreground model
that is modeled similarly to that of the background and using both models competitively in a decision framework. The foreground model is enhanced with a temporal criterion, under which foreground objects are assumed to maintain small frame-to-frame color transformations and spatial changes. The decision framework is based on a maximum a posteriori Markov Random Field (MAP-MRF), which transforms the problem of object detection as a pixel-level binary classification problem combines the foreground and background probability models in the likelihood function for each pixel. Even though this approach models dynamic backgrounds, it does so specifically for surveillance videos and thus assumes a fixed camera. Hence, it cannot be applied in completely unconstrained videos.

Yan et al. [32] propose to use pairwise constraints to aid video object classification with insufficient labeled data in surveillance videos. Indicating whether two examples are of the same class or not, pairwise constraints exploit the spatio-temporal continuity of video streams. As an example, the authors illustrate their method with the use case of classifying people’s identities. In this task, two overlapping objects from consecutive frames can be considered to have the same identify, but two objects that appear in the same video frame cannot. Moreover, identities can be differentiated using a face comparison mechanism, which represents another source of pairwise constraints, instead of building statistical models for every possible subject. The authors present three discriminative learning methods that minimize the regularized empirical risk and incorporate pairwise constraints by penalizing their violation. This approach focuses on a use case that is different from TV shopping, where the definition of pairwise constraints is not directly applicable. In addition, it is meant for surveillance videos.

Yang et al. [33] introduce an object-tracking framework that utilizes additional information to diminish the possibility of drifting. Their idea is to automatically
mine auxiliary regions that have high co-occurrence and motion correlation, at least for a short period of time, with the target object and use their collaborative tracking to prevent the target tracker from drifting. Such auxiliary regions consist of “significant” color regions that are obtained using the classical split-merge quad-tree color segmentation and are represented via color histograms. Using simple histogram matching, coherent color regions are matched as the frame sequence progresses, a transaction set is constructed, and regions with high co-occurrence with the target object are chosen as candidate auxiliary regions. Such candidate regions are then tracked using a mean-shift tracker. The framework determines the candidate regions that have motion correlation with the target object via a subspace analysis on an assumed affine model between the candidate auxiliary regions and the target object. Once concurrent, motion-correlated auxiliary regions are determined, collaborative tracking is achieved by modeling a random field among the auxiliary regions and the target object. The random field is formulated under a Markov network with a star topology, and a two-step belief propagation algorithm is used to estimate the posterior probabilities of the network. Lastly, the framework includes a mechanism to detect inconsistent tracking estimates, which are regarded as outliers. If the outlier is an auxiliary object, then it is removed from the collaborative tracking; however, if the outlier is the target object, then it is considered to be experiencing occlusion or drift, and it is suspended from the tracking temporarily. This approach uses auxiliary regions to improve object tracking, which can be considered to have the same purpose as the occurrences of the related object class uses in the proposed framework. However, the method is different than the proposed one and assumes that auxiliary regions can be both obtained via color segmentation and tracked by a simple mean-shift tracker, which are not practical assumptions in the TV shopping use case.
2.3 Tree-based Multimedia Indexing

Multimedia data are usually modeled in a multi-dimensional feature space, where multi-dimensional tree-based index structures provide efficient access and multimedia retrieval through similarity queries. Such index methods can be categorized into feature-based and distance-based structures [36]. Moreover, a functional aspect in which tree-based index mechanisms can be analyzed is in their ability to address the semantic gap and the users’ perception subjectivity problems inherent to multimedia data.

Feature-based index structures partition the space based on the values of the feature vectors along each dimension; popular feature-based index structures are multi-dimensional spatial access methods such as the KDB-tree [70] and the R-tree [71][72]. Both of these are height-balanced, dynamic, multi-way tree structures for secondary storage. The KDB-tree combines the properties of K-D-trees [70] and B-trees [73]. Internal nodes store space regions and their corresponding node pointers. Keys are values in a $k$-dimensional space. For each node, its space is partitioned into two sub-spaces based on a value of a single dimension, thus creating disjoint sub-spaces. R-trees generalize $B^+$-trees [74] onto multi-dimensional spaces and can index point and region data. Each internal node represents a bounding rectangle that covers the bounding rectangles of the node’s child nodes, forming a hierarchy.

Distance-based index structures partition the space based on the relative distances between pairs of objects. Some distance-based structures are the BP-tree [75], the M-Index [35], iDistance [76], the M-tree [37], and the vp-tree [77]. Since only pairwise distances are considered, distance-based structures do not assume any specific data representation and offer a more general approach. Nevertheless, they require the distance function to be a metric, that is, to satisfy the following properties:
\[
\begin{align*}
\text{symmetry} & \quad d(O_i, O_j) = d(O_j, O_i) \\
\text{non-negativity} & \quad d(O_i, O_j) \geq 0 \\
\text{identity} & \quad d(O_i, O_j) = 0 \iff O_i = O_j \\
\text{triangular inequality} & \quad d(O_i, O_j) \leq d(O_i, O_z) + d(O_z, O_j)
\end{align*}
\]

These two types of index structures can be further sub-divided into space partitioning or data partitioning. Space partitioning structures recursively partition the space into disjoint sub-spaces forming a hierarchy that is represented in a tree structure. Representative structures in this category are the hB-Tree [78], the KDB-tree [70], and the vp-tree [77]. Data partitioning structures arrange bounding regions in a spatial hierarchy in a containment relation. Common data partitioning structures are the R-tree [71], the M-tree [37], and the X-tree [38].

Feature-based index structures have a severe drawback when it comes to handling multimedia data. To introduce object-level similarity information in the index structure, feature-based index structures require that such similarity be translated into individual feature-level weights [36]. Such an approach is quite difficult due to the well-known problems of semantic gap between low-level features and high-level concepts and user perception subjectivity. Therefore, distance-based index structures can be considered as a more general approach to index multimedia data. The following paragraphs overview distance-based tree index structures.

The vp-tree (vantage point tree) [77] uses vantage points to divide the data based on their distances to the vantage points. At the top level, the vp-tree splits the space according to a chosen vantage point into two sub-spaces: a left/inside sub-space and a right/outside sub-space. The inside sub-space forms a ball region, and the outside sub-space is what is out of this region. This space partitioning process is recursively applied to the inside and outside sub-spaces creating a binary tree where each node is represented by a vantage point. At each node, the space covered by the node is divided
using the median distance of the elements in the node’s space to the vantage point; elements whose distance to the vantage point is less than the median are allocated to the inside sub-space (left sub-tree) and the other elements to the outside space (right sub-tree). The major disadvantage of the vp-tree is that it is a static binary tree that does not guarantee a balanced tree and thus could yield high search costs.

The M-tree \cite{37} is a height-balanced, dynamic tree-based index structure for secondary storage. The M-tree is kept balanced by building the tree in a bottom-up fashion, splitting nodes that overflow, and growing the tree by one level when the root node splits. Every time a node splits, its set of objects is divided into two sets: one set is kept in the node being split (the left node) and the other set is stored in a newly allocated node (the right node). Two objects are promoted to the parent node to represent the new sets of objects in the left and right nodes, where these promoted objects are the routing objects of these two nodes. An object may be promoted more than once in the tree as a routing object. Each internal node is represented by a routing object and a covering radius, forming a ball region; all objects reachable from the internal node lie within a distance equal to the covering radius from the routing object. Moreover, each routing object maintains the distance to its parent object. Entries in leaf nodes are similar to those in internal nodes, but the covering radii are zero and store object identifiers instead of pointers to sub-trees. Several improved tree-based indexed methods have been developed from the M-tree \cite{36,79,80,81,82}.

The Slim-tree is a version of the M-tree that aims at improving the efficiency of the tree by reducing the overlap between the nodes of the tree \cite{80}. The overlap is reduced using the Slim-down, a post-processing algorithm that utilizes measures that take into account the number of objects that lie in the intersection regions of the tree. The split algorithm is based on the Minimal Spanning Tree (MST); when
a node overflows, a MST of the node’s keys is built and one of the longest arcs is dropped.

The M$^+$-tree \cite{82} improves the M-tree by increasing the fanout of the tree and reducing its height. This is achieved by utilizing the concept of a key dimension, i.e., the dimension that affects the most the number of distance computations. The key dimension used in the M$^+$-tree is that with the maximal variance. The split algorithm works as follows. When a node overflows, the same procedure as that of the M-tree is followed: the node’s objects are partitioned and two routing objects are selected. Then, a key dimension is chosen for each partition, and the two partitions are each further divided into two twin spaces using two values from their dimensions. Although it achieves moderate performance improvements, the M$^+$-tree is restricted to $L_p$ metric spaces \cite{83}.

The M$^3$-tree tackles the problem of using more than one metric function to index the data to improve retrieval results \cite{79}. The idea is to use a dynamic distance function that depends on the query object. The M$^3$-tree uses a linear combination of metric functions whose individual weights are chosen at query time by the retrieval process. The weights are in the range $[0, 1]$. For any set of weights, an upper-bounding metric is that where all weights are 1.0. The M$^3$-tree uses such upper-bounding metric to index the objects in the tree; hence, the distances used in the tree (i.e., distances from routing objects to parent objects and covering radii) are upper bounds to the appropriate distances computed using another set of weights, such as those determined for a given query. During query processing, M$^3$-tree uses the stored partial distances to compute tight approximations to the query distance.

iDistance maps the objects to be indexed from a multi-dimensional space onto a one-dimensional space upon which it utilizes a B$^+$-tree as the index method \cite{76}. First, having a sample set of the objects, iDistance partitions the data and selects
a reference point or vantage point for each partition. Subsequently, each object of a partition is represented in a one-dimensional space by the object’s distance to the reference point of the partition. A $B^+$-tree is utilized to index the one-dimensional space. Given $m$ partitions, $C_0, C_1, ..., C_{m-1}$, and their corresponding reference points $O_0, O_1, ..., O_{m-1}$, each object $O$’s one-dimensional value is given by $i \times c + d(O, O_i)$, where $O_i$ is the reference point of the partition that contains $O$, $d(O, O_i)$ is the distance between $O$ and $O_i$, and $c$ is a constant used to keep the partitions distinct.

Inspired by iDistance, the M-index (or Metric Index) partitions the data following a multi-level approach to allow the index to scale to very large data sets and utilizes space pruning techniques that enhance those of iDistance [35]. The M-Index partitions the space using a Voronoi-like partitioning approach creating a multi-level partitioning of the space. Such multi-level partitioning allows assigning close objects to smaller partitions. Having identified $m$ reference objects $O_0, O_1, ..., O_{m-1}$, in the first level, each data object is assigned to its closest reference object, creating $m$ clusters. For the second level, each cluster is further partitioned into $m-1$ clusters using $m-1$ reference points (the set of $m$ reference points originally chosen except the reference point of the cluster being partitioned) following the same procedure. Hence, in the second level each object is assigned to its second closest reference point. This process iterates $l$ times, $1 \leq l \leq m$. For $l$ levels, the one-dimensional value of object $O$ is given by $d(O, O_i) + \sum_{j=0}^{l-1} k_j \times m^{l-1-i}$; where $O_i$ is the reference point of the first-level partition that contains $O$, i.e., the closest reference point to $O$; $m$ is the chosen number of reference points; $d(O, O_i)$ is the distance between $O$ and $O_i$; and $k_j$ corresponds to $O_{k_j}$, i.e., $O$’s $j^{th}$ closest reference object. During the search process, the M-index employs the repetitive Voronoi partitioning and the double-pivot distance constraint [83] to more effectively prune the clusters as compared to that of iDistance.
The BP-tree (Ball-and-Plane tree) is a recent tree access method for secondary storage that aims at combining space and data partitioning strategies [75]. The tree consists of index nodes (similar to internal nodes in the $\text{AH}^+$-tree) and leaf nodes, and it is built in a top-down fashion from a set of objects. Using a partitional clustering algorithm, such as k-medoids, the BP-tree is constructed by partitioning the objects of each index node into $k$ partitions and assigning each as a sub-tree of the index node. The recursive partitioning process terminates in a leaf node if the partitioning algorithm yields less than two partitions or the number of objects to be partitioned is less than the capacity of an index node. The BP-tree has two main drawbacks: it is not height-balanced and thus could degrade into a sequential search, and it is not dynamic and thus does not allow further modifications to the tree once it is constructed.

With respect to the challenges inherent to multimedia data, only the AH-Tree [36] incorporates additional semantic information to cope with these challenges. Using the M-tree as a base, the AH-Tree utilizes the concept of affinity information from the Markov Model Mediator (MMM) model [41] during the retrieval process. Affinity information maps the low-level features of the multimedia data to high-level concepts to represent the users perspective of concept-based relationships between the multimedia data. An important, practical feature is that in absence of semantic information, the AH-Tree behaves as the M-Tree, providing the same efficiency and accuracy.

Notwithstanding, the I/O overhead of the AH-Tree prevents it from being utilized on large multimedia datasets. The limitation arises from the process of affinity promotion, which is executed every time the AH-Tree is queried. Beginning at the leaves, the AH-Tree promotes the affinity values of the indexed records with respect to the given query upward the tree up to the root node. The affinity value of each internal
node is computed as the maximum affinity value between the node’s children. Once
the whole tree has been populated with affinity values, the search algorithms are able
to visit only nodes whose affinity values with respect to the given query object are
higher or equal than a supplied query minimum affinity value, thus computing less
distance computations during the query process by avoiding visiting sub-trees that
do not comply with the affinity requirement. Nevertheless, although the process of
affinity promotion saves distance computations, it forces a complete traversal of the
tree structure for each query, which equates to a significant I/O overhead since each
node in the tree has to be read from the disk onto the main memory.

The proposed AH⁺-tree utilizes the affinity values in a novel way to eliminate the
affinity-related I/O overhead, while still providing same accuracy and functionality
of the AH-Tree. Instead of traversing the whole tree to promote affinity values,
the AH⁺-tree efficiently stores and makes use of the affinity information along with
object-to-node and node-to-node relationships to identify the sub-trees that need to
be trimmed out during the retrieval process.
Smart TVs have enhanced the consumer experience by incorporating Internet access and mobile-like applications in the TV sets. However, consumers are still unable to seamlessly interact with the contents being streamed as it is highlighted by the TV shopping use case. Towards the realization of TV shopping applications, this dissertation proposes a scalable content processing framework. Figures 3.1 and 3.2 depict the control flow and high-level design of the proposed framework, respectively. The framework consists of three components: Multimedia Abstraction, Product Detection, and Product Retrieval, and its goal is to support the detection and retrieval tasks of the TV shopping use case. The input to the framework comes from a TV Content Distributor that streams TV contents to both TV sets and the framework.

The detection task executes first in the TV shopping use case, for which the content stream must be processed as fast as possible to support a pleasant user experience. To accomplish this goal, the components of the framework that are involved in this task execute in a fully distributed/parallel approach. The first component to execute is the Multimedia Abstraction component. It takes into consideration temporal and computational constraints to select a reduced set of frames that best summarize the information in the video sequence. The selection of a subset of frames enables the subsequent component to execute within computational bounds. On the selected frames, the Product Detection component detects products using additional information cues. Such cues consist of appearance, topological, and spatio-temporal constrains that are formalized into an optimization problem that allows the detection of complex objects. The results of the detection task are bounding boxes of product occurrences that are saved in a detection database and then asynchronously streamed to the TV sets. The purpose of the detection database is to remember shows that
 haven already processed and for which the detection results can be readily sent to the TV sets.

During the retrieval task, consumers can select desired product occurrences, which are sent as input to the Product Retrieval component. The latter utilizes a multi-dimensional, tree-based index structure that retrieves similar products from a product database. The retrieval results are streamed to the TV sets, and consumers may provide feedback about the results as well as submit purchase orders. The consumer feedback is of particular importance to the Retrieval component. The feedback information is logged and processed to learn semantic information that is used by the tree-based index structure to efficiently bridge the semantic gap and perception subjectivity problems, and thus provide more relevant results.
3.1 Multimedia Abstraction

Video abstraction is the task of representing a video into a subset of representative frames, which are also called key-frames. This task has been extensively researched [45]; however, most of the existing video abstraction approaches cannot be easily made scalable because they are formulated as optimization problems. To address this problem, the Multimedia Abstraction component incorporates a video abstraction approach that takes into consideration computational constraints.

By jointly considering computational and temporal rate constraints, the proposed video abstraction framework generates an abstract video on which multiple and complex object detectors can be executed such that the complete process is under a specified computational cost. Hence, the framework is scalable as it can generate results to specific temporal and computational constraints. Such an abstraction approach is formulated as an optimization problem, where regions of interest (ROIs)
are the unit of abstraction instead of the whole video frames. In its most general case, a region of interest can cover a whole video frame or encapsulate areas of salient motion and texture information. Since ROIs roughly identify the foreground objects, the abstraction framework is able to discard most of the background information and provide a higher quality video abstract.

For an input video, the abstraction framework consists of two stages, as depicted in figure 3.3. The first stage obtains the video ROIs by executing the ROI detection algorithm described in Section 4.2. This stage makes use of techniques such as shot-boundary detection, block-based motion estimation, and low-level texture feature extraction. The algorithm is then formulated as a block-labeling problem that is solved via graph-cut optimization on a graph representation of the motion blocks. The second stage obtains the optimal set of key frames on which to execute the object detection task and that satisfy the temporal and computational constraints. The
problem of obtaining the optimal set of frames is transformed into a longest-path problem in a dynamic acyclic graph (DAG) that is solved using dynamic programming. The cost function is formulated using the criteria of local representation and content coverage. Chapter 4 completely describes the proposed abstraction approach.

3.2 Product Detection

The task of detecting products in the video stream poses a significant challenge as complex objects can vary their form and appear in many poses. Three areas of related work can be utilized to tackle this challenge. The first two consist of object detectors for still images [2, 3, 4, 5, 6], which can be applied to each video frame, and approaches that mainly rely on motion estimation [7, 8, 9, 10, 11, 12, 13]. These only perform well in ideal conditions and do not specifically address complex objects. The third area consists of approaches that incorporate additional information as context in the detection process, and thus are better suited to handle complex objects. However, most approaches in this area are meant for still images [26, 27, 28, 29, 30] and do not include the temporal information. The few approaches that do [31, 32, 33] are meant for surveillance videos.

To address the detection of complex objects in the TV shopping use case, the Product Detection component formulates an optimization problem that incorporates additional information to improve the detection results of generic object detectors. Figure 3.4 depicts the design of this component. The approach consists of two steps that are executed for each video shot. In the first step, an object detector is executed on the video frames generated by the Multimedia Abstraction component. The purpose of this step is to detect objects using a low detection threshold, causing an increase in false positive detections but increasing the likelihood of detecting complex objects. In the second step, the additional cues are utilized to obtain the optimal path.
Figure 3.4: Design of the Product Detection component.
true positive detections should follow in the video shot. The optimal path identifies the true positive occurrences and serves to weed out false positive detections.

The additional cues consist of appearance, topological, and spatio-temporal relationships. The appearance is related to the likelihood of an object occurrence to belong to the target product class and is modeled by means of the detection score. The other two cues make use of a related object class based on the underlying assumption that such a related class must be easier to detect. The topological relationship constrains the possible locations for occurrences of the target product class with respect to the locations of the related object class. Lastly, the spatio-temporal cue makes use of the correlation between translations of both classes.

3.3 Distributed Detection Pipeline

To provide an effective interface, the detection task of the TV shopping use must be executed as close to real-time as possible. Both the Multimedia Abstraction and Product Detection components are parallelized as depicted in figure 3.5. The parallelization scheme consists of two levels: shots and blocks. The system processes shots in parallel, and in turn, the blocks of a shot are processed in parallel as well. A block is a logical unit in a video shot that consists of a sequence of consecutive frames within the shot, i.e., a block does not span more than one shot. A shot remains its traditional definition, except that it has now maximum length. The reason for capping shots is that a shot can be considerably long, which would degrade the computational performance of the system by diminishing parallelization.

The distributed and parallel execution is implemented in Storm [21], a distributed computational framework for real-time operations that is designed to process unbounded streams of data. Its goal is not to replace MapReduce [20] but to complement it for real-time applications. A job in Storm is represented by a “topology”, which
Figure 3.5: High-level design overview of the proposed framework.

depicts a graph representation of the computation. Vertices in such a graph represent compute nodes, and edges indicate how data should be transmitted between nodes. Also, different from MapReduce jobs, a topology processes data records indefinitely until killed. A data stream is modeled as an unbounded sequence of tuples, and the purpose of the nodes in the topology is to transform streams into new streams. For example, a stream of frames can be transformed into a stream of object occurrences in the frames.

Two types of nodes exist in a topology: spouts and bolts. A spout is a source of streams, and a bolt consumes a set of streams, processes them, and may generate new streams. Complex operations that require multiple steps can be realized using multiple bolts, where each bolt can execute any operation, e.g., aggregations, communicating with external databases, or any arbitrary code. In terms of execution, the nodes in the topology run in parallel with a user-defined task parallelization factor. Each machine in a Storm cluster can run one or more worker processes for the topology, where a worker process may spawn multiple threads of execution that correspond to instantiations of spouts or bolts, and such instantiations are referred to as tasks in
the Storm terminology. The type of communication between nodes in the topology is defined via stream groupings. Examples of groupings are “shuffle grouping”, which sends a message to a random task, and “fields grouping”, which lets the developer group a stream by a subset of the message (i.e., tuple) fields. Furthermore, the Storm framework provides some fault-tolerance by re-assigning failed nodes and guarantees that no data will be lost even when data messages are dropped or nodes go down.

Figure 3.6 depicts the execution of the detection task in Storm. The topology consists of a spout that generates a stream of video frames, which implement both the Multimedia Abstraction and Product Detection components to ultimately generate the product detection results. The components of the topology are described as follows:

- The Frame Stream spout receives the TV stream from the TV Content Provider and emits a stream of video frames that is subsequently processed by the Shot Detection bolt.

- The Shot Detection bolt performs shot boundary detection and emits a stream of tuples, whose keys are the shots ids and values consist of the frames (pixel data as well as frame temporal positioning). The stream is then processed by the Shot Abstraction bolt, which subscribes to the Shot Detection bolt using field grouping on the shot ids. This grouping forces all the frames that belong to the same shot to go to the same instance of the Shot Abstraction bolt.

- The Shot Abstraction bolt reduces the number of frames in each video shot, thus creating a shot summary that is partitioned into blocks. The output stream of this bolt is composed of tuples whose keys consist of both the shot ids and blocks ids, and values of both the number of blocks in the shot and the set of frames that belong to the same block. That is, each tuple value aggregates the frames that belong to the block. The total number of blocks in the corresponding video
Figure 3.6: Design of detection phase in Storm.
shot is included in each tuple so the Multi-cue Optimization bolt can process the object detections across all the blocks that belong to the same shot. The stream is then processed by the Product Detection bolt, which subscribes to the Video Abstraction bolt using shuffle grouping. This grouping randomly assigns the tuples to the Product Detection bolt instances.

- The Product Detection bolt loops through the frames of each block and generates the detection occurrences for the block. The output stream is composed of tuples whose keys consist of the shot ids and the values of the list of detections for a block and the number of blocks in the corresponding shot. The stream is then processed by the Multi-cue Optimization bolt, which subscribes to the Product Detection bolt using field grouping on the shot ids. This grouping forces all the block-level detections that belong to the same shot to be processed by the same instance of the Multi-cue Optimization bolt.

- The Multi-cue Optimization bolt executes the multi-cue optimization step of the Product Detection component at the shot level. It must aggregate all the block-level detections in the same shot to be able to do the optimization. Since it can identify the shot each tuple belongs to as well as the total number blocks in the shot, this bolt can aggregate all the block-level detections by counting them and temporarily storing those block-level detections that belong to a different shot. The pruned detections can then be stored to disk or passed to another subsystem.

3.4 Product Retrieval

The Product Retrieval component fulfills the task of retrieving, from a product database, similar products to a user selected product occurrence. This task has been
extensively studied in the CBMIR community, but it is still a challenge to address the inherent problems in multimedia retrieval of the semantic gap between low-level features and high-level concepts and the subjectivity of the users’ perception.

To efficiently address these challenges in the retrieval task, the proposed framework presents the $\text{AH}^+$-tree \cite{40,39}, a height-balanced, multi-dimensional tree structure for multimedia data that utilizes semantic information to address the challenges inherent to multimedia retrieval. The semantic information consists of high-level affinity relationships that numerically represent how semantically related multimedia objects are from the consumers perspective, based on the idea that the more frequent two images are accessed together, the more related they are. Currently, the $\text{AH}^+$-tree obtains the affinity information from the Markov Model Mediator (MMM) \cite{41}. Nevertheless, the $\text{AH}^+$-tree is not tied to the MMM mechanism, and other data mining approaches can be utilized to derive affinity information. In addition, the $\text{AH}^+$-tree improves upon the $\text{AH}$-Tree \cite{36}, which is the first tree-based indexing mechanism to utilize semantic information to tackle the aforementioned multimedia data problems. However, the $\text{AH}^+$-tree extends the tree structure to utilize the semantic information in a novel way that eliminates the significant I/O overhead incurred by the $\text{AH}$-Tree during the retrieval process. The modification of the tree structure in the $\text{AH}^+$-tree consists of two global arrays that are used to track where indexed objects are located in the tree (object-to-node relationships) and how the tree must be traversed to reach certain leaf nodes (node-to-node relationships).

Two major processes drive the functionality of the $\text{AH}^+$-tree: tree construction and similarity searching. The $\text{AH}^+$-tree is constructed in a similar fashion as that of the M-tree \cite{37} and the $\text{AH}$-Tree. The tree structure is kept balanced by inserting new keys at the leaves, splitting nodes that overflow, and propagating changes upward in the three. When a new object is inserted in a leaf node, the object-to-node
relationships are updated, and when a tree node overflows and splits, the node-to-
node relationships are updated to keep track of the nodes being modified. Similarity
searching allows the AH$^+$-tree to provide content-based retrieval. Traditional exam-
pies of similarity queries are range and k nearest neighbor (kNN) queries. Range
queries are fulfilled by traversing down the tree and at each level selecting relevant
sub-trees to traverse by making use of the triangular inequality of the metric space as
well as the object-to-node and node-to-node relationships. Nearest neighbor queries
are implemented using the same branch-and-bound technique utilized by the AH-
Tree and the M-tree. Like for range queries, the object-to-node and node-to-node
relationships are utilized to select qualifying sub-trees. Chapter 6 fully describes the
AH$^+$-tree.
In the TV Shopping use case, the computational cost of detection task is dictated by two factors. One factor is the complexity of the object detectors that must be executed to detect complex objects, considering that low-complexity object detectors are not a viable solution as they may not be powerful enough. The second factor is the number of video frames that must be processed. The latter is significant, and the system may be constrained by computational resources. Additionally, even with enough available resources, the system may benefit from not processing all the video frames but a selection of them that best represent the video contents.

To facilitate the execution of the detection task in a timely fashion, this chapter proposes a multimedia abstraction framework [22]. Jointly considering computational and temporal rate constraints, the proposed framework generates an abstract video on which multiple and complex object detectors can be executed such that the complete process is under a specified computational cost. In this sense, the framework is computationally scalable as it can adapt to given constraints and generate optimal abstraction results accordingly. The abstraction problem is formulated as an optimization problem based on the specified constraints and obtains an optimal solution using dynamic programming. Moreover, the abstraction process uses salient information as the basis for the selection process. Such information is represented via regions of interest, which can cover the whole extent of the video frames or encapsulate areas of salient motion that are also visually relevant.

4.1 Multimedia Abstraction Framework

As previously introduced, effectively addressing the TV-shopping use case requires the application of numerous and complex object detection methods. Given a video
\[ V = \{ F_i \}_{i=1}^N \], where \( F_i \) is the \( i \)th frame in \( V \), the task of detecting moving objects of class \( C \) in \( V \) can be approached by relying on motion information coupled with the application of an object detector \( D_o(\cdot) \) of class \( C \). However, since executing \( D_o(\cdot) \) on each \( F_i \) is not feasible, it is assumed the applicability of \( D_o(\cdot) \) is constrained by computational resources. Given a computational rate constraint \( \gamma \) and a temporal rate constraint \( \kappa \), the problem is then to find the moving object occurrences of class \( C \) in \( V \) such that the computational cost of the detection process is less than or equal to \( \gamma N \) and the generated object occurrences come from at most \( \kappa N \) frames in \( V \). The computational rate constraint enforces an upper bound on the maximum time that can be consumed by the detection process across \( V \), and the temporal rate constraint specifies the maximum number of video frames that can be processed using \( D_o(\cdot) \).

The stated problem is solved using a multimedia abstraction framework that selects the best subset of frame indices \( S = \{ a_u \}_{u=1}^{N_S} \), such that \( D_o(S) \) satisfies the constraints. The selected subset of frames summarize the "views" that a moving object \( O^* \) follows in \( V \).

A view of a moving object \( O^* \) can be thought of an occurrence of \( O^* \) that depicts a visual perspective of \( O^* \). For example, Figure 5.2 depicts a video of four frames where the object \( O^* \) consists of the handbag carried by the model. In this sample video, the first two frames depict one view of \( O^* \), whereas the last two frames depict another view. If the task is detecting the bag carried by the model, then \( D_o(\cdot) \) should be executed on only two frames that depict the first and second views, respectively.

The relevant question is then how to select the frames that best represent the views of \( O^* \) in \( V \), which equates to selecting the frames that best represent the view path. The proposed framework uses regions of interest (ROIs) as the basis for selecting \( S \). In general, a ROI can be a whole frame, or a smaller region of it that both exhibits salient motion and is visually relevant. Section 4.2 presents our proposal for
determining ROIs. Nevertheless, the proposed multimedia abstraction framework is
generic with respect to the algorithm used to detect ROIs.

The proposed framework selects the set $\mathcal{S} = \{a_u\}_{u=1}^{N_S}$ that best represents the
view path. The formulated selection problem considers the temporal rate constraint
as well as the fact that the computational allowance left for applying $D_o(\mathcal{S})$ has
been reduced by the computational costs incurred in both selecting the ROIs and the
abstraction process itself, i.e., selecting $\mathcal{S}$. Let $Q(\mathcal{S})$ represent the quality of $\mathcal{S}$ in
representing the view path, $R_{DC}(\mathcal{S})$ be the computational allowance to be consumed
by executing $D_o(\mathcal{S})$, $R_{TC}(\mathcal{S})$ be the temporal cost of $\mathcal{S}$, $R_{RC}^V(\mathcal{V})$ be the computational
allowance already consumed by the ROIs, and $R_{SC}^V(\mathcal{V})$ be the computational allowance
to be consumed by the abstraction process. The framework selects the optimal $\mathcal{S}^*$ by
solving the following optimization problem:

$$\begin{equation}
\begin{aligned}
\text{maximize } & Q(\mathcal{S}) \\
\text{subject to } & R_{DC}(\mathcal{S}) \leq \gamma N - R_{RC}^V(\mathcal{V}) - R_{SC}^V(\mathcal{V}) \\
& \text{and } R_{TC}(\mathcal{S}) \leq \kappa N
\end{aligned}
\end{equation}$$
Since the framework should select as many frames as possible from $\mathcal{V}$, the quality $Q(.)$ must increase as the cardinality of $S$ increases. Among several options for $S$ with the same cardinality and that satisfy the constraints, it is desirable to choose the one which maximizes $Q(S)$. However, if the options have different cardinality values, the desirable option is to choose the set with the largest cardinality.

To determine $Q(.)$, the proposed framework utilizes the criteria of local representation and content coverage [45] at the ROI level. A frame with a good local representation would ensure the frame’s ROI is highly similar to the ROIs of its neighboring frames, thus representing a specific view in the view path. A high content coverage among the selected frames would choose frames whose ROIs are highly dissimilar. Selecting dissimilar ROIs ensures a good representation of the different views in the view path.

The local representation of frame $F_i$ is defined as:

$$B_{LR}(i) = \begin{cases} 
C(\mathcal{R}_{i-1}, \mathcal{R}_i) & \text{if } i = N \\
C(\mathcal{R}_i, \mathcal{R}_{i+1}) & \text{if } i = 1 \\
\frac{C(\mathcal{R}_{i-1}, \mathcal{R}_i) + C(\mathcal{R}_i, \mathcal{R}_{i+1})}{2} & \text{otherwise}
\end{cases}$$

where $C(.)$ is a measure of the visual correspondence (i.e., similarity) between two ROIs. It is important that computing $C(.)$ is not computationally expensive. The visual content of the ROIs can be represented, for example, using the YCbCr color histograms with the cosine function as the similarity measure or the CEDD feature [84], which encodes both color and texture information. The choices for the visual features and similarity measure are not restricted, as long as the latter yields a value in $[0, 1]$.

The content coverage of frames $F_i$ and $F_j$ is defined as given in equation (4.3). As such, $F_i$ and $F_j$ have high content coverage if $B_{CC}(i,j)$ is low.
\[
B_{CC}(i, j) = \begin{cases} 
0 & \text{if } i = 0 \\ 
C(R_i, R_j) & \text{otherwise}
\end{cases}
\quad (4.3)
\]

With these definitions, a high-quality \( S \) should have high \( \sum_{u=1}^{N_S} B_{LR}(a_u) \) and low \( \sum_{u=1}^{N_S} B_{CC}(a_{u-1}, a_u) \). The quality function can then be defined as follows.

\[
Q(S) = \sum_{u=1}^{N_S} \left\{ \eta B_{LR}(a_u) + (1 - \eta) \left[ 1 - B_{CC}(a_{u-1}, a_u) \right] \right\}
\quad (4.4)
\]

where \( \eta \) is a weighting parameter in \([0, 1]\). Clearly, \( Q(S) \) is non-decreasing as the cardinality of \( S \) increases, which is a feature desired for \( Q(\cdot) \). Having defined \( Q(\cdot) \), the constraints are implemented by means of the following:

\[
R_T(S) = N_S \quad (4.5)
\]

\[
R_C^D(S) = \sum_{a_u \in S} g_D(a_u) \quad (4.6)
\]

\[
R_C^R(V) = \sum_{i=1}^{N} g_R(i) \quad (4.7)
\]

\[
R_S^C(V) = g_S(V) \quad (4.8)
\]

where the function \( g_D(a_u) \) estimates the computational cost of executing \( D_o(\cdot) \) on \( R_{a_u} \), and \( g_R(i) \) and \( g_S(V) \) provide the computational costs incurred in selecting \( R_i \) and obtaining the abstract \( S \) from \( V \), respectively. If the whole video frames are used as the ROIs, then \( g_R(\cdot) = 0 \) for all frames. It is worth pointing out that the functions \( g_R(i) \) and \( g_S(V) \) can be measured during the abstraction process. On the contrary, \( g_D(a_u) \) cannot be measured before finding the optimal solution to equation \((4.1)\) as
$D_o(.)$ can only be applied once the optimal solution is found. The object detectors to be applied are not part of the proposed approach and should be chosen based on the specific application. The function $g_D(a_u)$ must then be appropriately defined in each case. As an example, $g_D(.)$ can be defined as $g_D(a_u) = \zeta p(\mathcal{R}_{a_u})$, where $\zeta$ is a constant and $p(\mathcal{R}_{a_u})$ measures the size in pixels of $\mathcal{R}_{a_u}$. In this case, the computational cost of applying $D_o(.)$ on $\mathcal{R}_{a_u}$ is proportional to the size in pixels of $\mathcal{R}_{a_u}$. Other functions can also be used to measure such a variable cost. It is important to highlight that applying $D_o(.)$ at the ROI level allows us to identify more frames in $\mathcal{S}$, given that the pixel area covered by the ROIs is less than or equal to the area covered by the whole frames. In addition, it may eliminate false positive detections as the detection area is concentrated on the ROI, although this claim is dependent on the quality of the ROI.

Based on the given definitions, equation (4.1) can be written as follows.

$$\text{maximize } Q(\mathcal{S}) = $$
$$\sum_{u=1}^{N_S} \left\{ \eta B_{LR}(a_u) + (1 - \eta) \left[ 1 - B_{CC}(a_{u-1}, a_u) \right] \right\}$$

subject to $R_D^C(\mathcal{S}) \leq \gamma N - R^C_V - R^S_V$ (4.9)

and $R_T(\mathcal{S}) \leq \kappa N$

4.1.1 Optimal Solution

This section shows how to use dynamic programming (DP) to obtain the optimal solution to the optimization problem stated in equation (4.9). Firstly, the cost function $G_k(a_k)$ is created to represent the maximum cost solution for the first $k$ elements of $\mathcal{S}$ subject to the $k^{th}$ element is $a_k$. It is clear that maximizing $G_{N_S}(a_{N_S})$ implies maximizing $Q(\mathcal{S})$. In addition, $G_{k+1}(a_{k+1})$ can be written as:
\[ G_{k+1}(a_{k+1}) = G_k(a_k) + \left\{ \eta B_{LR}(a_{k+1}) + (1 - \eta) \left[ 1 - B_{CC}(a_k, a_{k+1}) \right] \right\} \] (4.10)

which shows that the selection of the \( k^{th} \) frame index does not depend on the previously selected frame indices. This recursive representation makes the next step of the optimization process independent of the previous step, which is the foundation of DP.

This problem can be interpreted as finding the longest path in a weighted, directed acyclic graph (DAG) \( G = (V, E, w_Q, w_C) \), where \( V \) is the set of vertices, \( E \) is the set of edges, \( w_C : V \to (R) \) is a vertex-weight function which assigns a cost to each vertex in \( G \), and \( w_Q : E \to (R) \) is the edge-weight function. Let \( I \) be the set of frame indices in \( V \). Then \( V = I \) as all frame indices in \( V \) can be in \( S \), and \( E = \{(a_u, a_v) \in I^2 | a_u < a_v \} \) as frame indices in \( S \) must respect their temporal order. The edge-weight function \( w_Q \) and the vertex-weight \( w_C \) correspond to the stepwise values of \( Q(,) \) and \( R^D(,) \), respectively. They are defined as follows.

\[
w_Q(a_u, a_v) = \eta B_{LR}(a_v) + (1 - \eta) \left[ 1 - B_{CC}(a_u, a_v) \right] \tag{4.11}
\]

\[
w_C(a_u) = g_D(a_u) \tag{4.12}
\]

Let \( q_t \) be a path in \( G \) with \( t \) vertices, \( q_t = \{a_{j_1} \}_{j=1}^t \), let the quality length of \( q_t \) be the function \( L_Q(q_t) = \eta B_{LR}(a_1) + \sum_{(a_{j-1}, a_j) \in q_t} w_Q(a_{j-1}, a_j) \), and the computational length of \( q_t \) be the function \( L_C(q_t) = \sum_{a_j \in q_t} w_C(a_j) \), for which it is said \( q_t \) has a quality cost and a computational detection cost, respectively.

To find the optimal path \( q^* \) that satisfies computational and temporal rate constraints, the DP algorithm uses the fact the path with the largest computational cost
must have the same number of vertices that the path with the largest quality cost. This is true since $L_Q(q_t)$ and $L_C(q_t)$ are non-decreasing as $t$ increases. Moreover, given the temporal rate constraint, the path with the optimal cost must have $\lfloor \kappa N \rfloor$ or fewer vertices. Therefore, $q^*$ can be found by backtracking from the $\lfloor \kappa N \rfloor$ state of the DP algorithm. Beginning with state $t = \lfloor \kappa N \rfloor$, the DP algorithm iteratively decrements $t$ and makes $q^* = q_t$ the first path in the iteration with $\max_t (L_Q(p_t))$ and that also satisfies $L_C(p_t) \leq \gamma N - R_C^R(V) - g_S(V)$. The longest path in $G$ can then be found in $O(I^2)$ by fixing the first frame index in $S$ or in $O(I^3)$ otherwise.

4.2 Implementation Details

This section presents our proposal for determining the ROI that represents the views of moving objects in a frame $F$. Let $R$ denote the ROI of frame $F$. The simplest case of $R$ is the whole extent of $F$. This case has the benefit that no computational resources are spent on detecting ROIs. Notwithstanding, this approach is only effective when the objects $O^*$ occupy most of $F$, which is not a practical assumption.

To better differentiate between object views across different frames, it is necessary to select areas of $F$ that better enclose the moving objects. Specifically, $R$ should comprise the areas that coarsely enclose zones of salient motion and relevant texture in $F$. The need for coarse areas stems from a computational performance requirement, as the computational cost of determining $R$ should be far lower than the cost of executing object detectors on $F$, as well as the fact that it is not practical to assume that objects can be determined using only motion information.

Two types of motion can appear in a frame: global and local motion. Representing the background motion, global motion refers to the motion induced by the motion of the camera. Local motion is the motion of moving objects $O^*$ in the frame. This work focuses on translational and zoom global motions as they are the most common ones in
TV shows. In the ideal case, it would be possible to use any of the existing methods \cite{85,86,87,88,89} to estimate the global motion in the frame, use global motion compensation to remove the global motion, and then obtain the region that exhibits salient motion. However, estimating global motion parameters is computationally expensive and susceptible to local motion noise.

Using block-based motion estimation \cite{90,91}, this chapter proposes a novel approach to determine areas of the local motion. It consists of analyzing changes in motion direction between adjacent blocks in $\mathcal{F}$ that exhibit relevant texture. Let’s analyze the ideal case where all background blocks in frame $\mathcal{F}$ move in the same direction and the estimated motion for all the blocks is correct. Let $B_l$ be a block of $\mathcal{F}$ that exhibits local motion, and $B_g$ be a block of $\mathcal{F}$ that belongs to the background. If $B_l$ and $B_g$ are adjacent in the frame, then their motion directions will be different. Hence, the ROI would be the area whose locations differ in motion direction. However, in practical situations, if two adjacent blocks differ in motion direction, the reason could be (a) an erroneous motion estimation, (b) the effect of the camera zoom, or (c) one of the blocks belongs to $\mathcal{O}^*$ and the other block to the background. Usually, the cause of erroneous motion estimation is low texture in the block.
Figure 4.2 depicts the idea of a frame’s ROI on a fashion video, in which $O^*$ consist(s) of model(s). Figure 5.3a depicts the first frame of a video sequence showing a model walking across the stage while the camera moves to follow the model, injecting a translational global motion. Figure 5.3b depicts a possible ROI for the frame shown in Figure 5.3a. Similarly, Figure 4.2c depicts two models as they walk across the stage while the camera does a slight zoom on the models. Figure 4.2d depicts a possible ROI for the two models, where the ROI consists of two areas, one for each model. As depicted in these figures, the ROI coarsely encloses the moving object(s) but effectively removed most of the background area, thus allowing for a more concrete representation of the view in the frame.
4.2.1 Preliminary Definitions

Let $\tau(\mathcal{B})$ provide a measure of texture for block $\mathcal{B}$, and $\Pi(\mathcal{B})$ be a neighborhood of blocks for $\mathcal{B}$. In this work, $\tau(\mathcal{B})$ is defined as the pixel-level intensity variance in $\mathcal{B}$ and $\Pi(\mathcal{B})$ as the 4-connect blocks of $\mathcal{B}$, although other texture measures can be used for $\tau(\mathcal{B})$.

In addition, let $\mathcal{F}_i$ be the current frame for which $\mathcal{R}_i$ needs to be estimated. Let $\theta_\omega(\mathcal{B})$ and $\omega_\omega(\mathcal{B})$, $z > 0$, be the direction and magnitude of motion, respectively, for block $\mathcal{B}$ between $\mathcal{F}_i$ and $\mathcal{F}_{i+z}$. The dissimilarity between the motion directions of two blocks $\mathcal{B}_n$ and $\mathcal{B}_m$ of $\mathcal{F}_i$ is computed by the predicate $H_z(\cdot)$ as follows:

$$H_z(\mathcal{B}_n, \mathcal{B}_m) = \begin{cases} 1 & \text{if } \left| \frac{\theta_\omega(\mathcal{B}_m) - \theta_\omega(\mathcal{B}_n)}{\theta_\omega(\mathcal{B}_n)} \right| \geq \rho \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

That is, the motion of two blocks is dissimilar if their percentage change difference is greater than some threshold $\rho$, which should be determined based on the amount of motion scatterness between $\mathcal{F}_i$ and $\mathcal{F}_{i+z}$. For example, if the motion angles between $\mathcal{F}_i$ and $\mathcal{F}_{i+z}$ depict low scatterness, as depicted in Figure 4.3a, the threshold $\rho$ should have a low value. In this case, it is easier to differentiate local motion from global motion. However, if the motion angles depict high scatterness, as shown in Figure 4.3b, $\rho$ should be high so that it does not detect motion differences in the background.
Figure 4.3: Examples of motion scaterness. Actual motion angles are shown rounded to the nearest 22.5 degree angle, each color represents a different motion angle, and the block size is $35 \times 35$ pixels; (a) corresponds to the frame of Figure 5.3a, and (b) to the frame of Figure 4.1a.

There are several possible ways of measuring the scaterness among motion angles. One possible way is to compute the frequency of motion angles, and assume low scaterness if the highest frequency is below some threshold. The approach used in this work, very effective based on empirical results, is to assign the threshold $\rho$ based on the type of global motion between $F_i$ and $F_{i+z}$. Different from the traditional approach of estimating global motion parameters through expensive computational approaches, it is possible to estimate whether there is no global motion, a translational motion, or a zoom motion between $F_i$ and $F_{i+z}$ using the motion angles rounded to the nearest 45 degrees. Rounding the motion angles eliminates spurious motion values and tends to reveal global motion patterns. Using thresholding, the type of global motion can be determined as follows:

- **No motion**: if most blocks have the same motion direction and the average motion magnitude is small.
- **Translational motion**: if most blocks have the same motion direction but the average motion magnitude is not small.
- **Zoom motion**: if a significant number of symmetrical blocks exhibit a combined motion of 180 degrees. Symmetrical blocks are those located on a horizontal
Figure 4.4: Illustration of steps for detecting zoom motion. Each color represents a different motion angle, and the block size is $35 \times 35$ pixels; (a) shows the motion angles generated by block-based motion estimation for a frame global zoom, (b) shows the motion angles rounded to the nearest 45 degrees, and (c) shows in red the symmetric blocks whose combined motion is 180 degrees.

Figure 4.4 depicts the steps of determining whether there is global zoom motion. Figure 4.4a shows the results of block-based motion estimation for a frame with a model walking towards the camera while the camera zooms on the model. Figure 4.4b shows the motion angles after having been rounded to the nearest 45 degrees. Figure 4.4c depicts in red the symmetric blocks whose combined motion is 180 degrees.

4.2.2 Proposed Solution

Our proposed solution for determining $\mathcal{R}_i$ formulates a binary block labeling problem, building on the idea of existing pixel-labeling approaches on image data \[92\][93][94]. It combines motion angles, texture, and differences in motion at the neighborhood level to define how to label blocks. Let $T$ be the number of blocks in frame $\mathcal{F}_i$, where $\mathcal{F}_i = \{B_n\}_{n=1}^T$. Let $l_n \in \{0, 1\}$ be a label for $B_n$. The search space is $\{0, 1\}^T$. Label 0 means background motion and 1 means local motion.
\[
E_n^d(l_n) = \begin{cases} 
\xi_d & \text{if } l_n \left\{ 1 - \text{sign}(\omega_z(B_n)) \right\} > 0 \\
\xi_d & \text{if } l_n \left\{ 1 - \text{sign}[\tau(B_n) \sum_{B_m \in \Pi(B_n)} H_z(B_n, B_m)] \right\} \text{sign}(\omega_z(B_n)) > 0 \\
\xi_d & \text{if } (1 - l_n) \left\{ \text{sign}[\tau(B_n) \sum_{B_m \in \Pi(B_n)} H_z(B_n, B_m)] \right\} \text{sign}(\omega_z(B_n)) > 0 \\
\xi_d^{\tau_{\text{min}}}(\tau(B_n)) & \text{if } l_n \left\{ \text{sign}[\tau(B_n) \sum_{B_m \in \Pi(B_n)} H_z(B_n, B_m)] \right\} \text{sign}(\omega_z(B_n)) > 0 \\
0 & \text{otherwise}
\end{cases}
\] (4.14)

Defined in equation (4.14), let \( E_n^d(l_n) \) be a cost function for labeling block \( B_n \) using label \( l_n \), where \( \xi_d > 0 \) is a texture-independent cost value for assigning label \( l_n \). The function \( \text{sign}(.) \) is zero if its argument is zero, one otherwise. Additionally, \( \xi_d^{\tau_{\text{min}}}(\tau(B_n)) > 0 \) is a texture-dependent cost whose value depends on the texture of \( B_n \) compared to a minimum texture \( \tau_{\text{min}} \). Basically, \( \xi_d \) and \( \xi_d^{\tau_{\text{min}}}(\tau(B_n)) \) assign a cost based on the motion of the block and that of its neighbors.

For label \( l_n \), according to equation (4.14), the costs are assigned based on the following cases:

1. If \( B_n \) is to be labeled with local motion and it has no motion, the cost is \( \xi_d \).
   That is, blocks with zero motion should not have local motion. The following cases are triggered when the magnitude of motion is \( > 0 \).

2. If \( B_n \) is to be labeled with local motion and either (a) \( B_n \)'s texture is zero or (b) \( B_n \)'s motion is the same as that of its neighbors, the cost is \( \xi_d \). That is, blocks with no texture or no salient motion in their neighborhoods should not have local motion. This case is triggered when \( \tau(B_n) \sum_{B_m \in \Pi(B_n)} H_z(B_n, B_m) = 0 \) and \( l_n = 1 \).

3. If \( B_n \)'s motion is different from that of its neighbors and its texture is not zero, then there are two possible costs, depending on the label. This case is triggered when \( \sum_{B_m \in \Pi(B_n)} H_z(B_n, B_m) > 0 \). If \( l_n = 0 \) (i.e., if \( B_n \) is to be labeled...
with background motion), then the cost is $\xi_d$. However, if $B_n$ is to be labeled as foreground motion ($l_n = 1$), then the cost is $\xi_d^{\tau_{\text{min}}}(\tau(B_n))$. The supporting argument for $\xi_d^{\tau_{\text{min}}}(\cdot)$ is that $B_n$ should not be labeled as foreground motion if its texture is too low, as there is a high chance that in this case $B_n$ belongs to the background or its motion has been incorrectly estimated. If $\tau(B_n)$ is lower than $\tau_{\text{min}}$, then $\xi_d^{\tau_{\text{min}}}(\tau(B_n))$ will increase more when $\tau(B_n)$ gets closer to zero.

Let $E^p(l_n, l_m)$ be another cost function to inject a smoothness prior in the labeling problem:

$$E^p(l_n, l_m) = \begin{cases} 
  \xi_p & \text{if } l_n \neq l_m \\
  0 & \text{otherwise}
\end{cases} \quad (4.15)$$

Basically, $E^p(l_n, l_m)$ penalizes with cost $\xi_p$ adjacent blocks with different labeling. Having defined $E^d(\cdot)$ and $E^p(\cdot)$, the selection of the ROI is stated as the following block labeling problem:

$$\text{minimize } M(l_1, \ldots, l_T) = \sum_{n=1}^{T} E^d_n(l_n) + \sum_{n<m} E^p(l_n, l_m) \quad (4.16)$$

The cost function $M(l_1, \ldots, l_T)$ is referred to as the Ising model, and Greig et al. [95] show how to solve this type of model using the min-cut algorithm on a properly-constructed graph. Please refer to [95][96] for a description of how to construct the graph. Using the Edmonds–Karp algorithm [97], the minimum cut can be found in polynomial time.

### 4.3 Experimental Results

This section presents and analyzes experiments that demonstrate the advantages of using the proposed multimedia abstraction framework. Five experiments are pre-
The first experiment demonstrates the effectiveness of the proposed method in determining ROIs. In its most general case, a ROI is a whole frame of the input video, and the proposed framework can disable ROI-detection to utilize whole video frames. However, when ROIs need to be detected, the accuracy of the ROI-detection algorithm does affect the advantage in terms of quality and number of generated abstract frames of the proposed abstraction framework, which is the reason for commencing the experiments with the analysis of the proposed ROI-detection approach.

Subsequent experiments compare the proposed framework to computationally unconstrained frame-based (CUFB) and computationally constrained frame-based (CCFB) abstraction approaches. Frame-based approaches use whole-frames as the basis of the abstraction approach. Since the novelty of this work lies in the usage of ROIs and a computational constraint, both frame-based approaches were simulated by turning off the detection of ROIs in the proposed framework. In addition, the computational constraint was set to a sufficiently large value when simulating the CUFB approach. The CUFB approach represents existing abstraction approaches in the literature, whereas the CCFB approach represents the proposed approach when ROIs are just the whole video frames. The second and third experiments study the proposed framework from a computational perspective. The second experiment focuses on the computational scalability of the proposed abstraction method. The comparison is made against the CUFB approach to showcase how the proposed method can “accommodate” abstraction results based on computational and temporal requirements. The other computational aspect of the proposed approach is ROI detection. Hence, the third experiment compares the proposed framework with the CCFB approach to analyze the computational effect of utilizing ROIs as the basis of the abstraction process. After addressing the computational aspects of the proposed framework, the last two experiments address its abstraction performance in terms of quality. The fourth
experiment compares the abstraction quality of the proposed method with that of the CUFB approach, which serves to evaluate whether the inclusion of ROIs affects the resulting quality. Finally, the fifth experiment explores how the weight parameter of equation (4.9) affects the abstract quality of the proposed framework.

The videos used in the experiments are three fashion show videos, which were obtained from YouTube [98] with high-resolution. The goal of the proposed framework is to support TV-enabled shopping, and fashion shows are good candidates for a prototype system. These three videos correspond to different fashion shows, and a significant difference between them is that VC2, contrary to VC1 and VC3, contains the public closer to the fashion models, and motion in the public makes more difficult the detection of ROIs in the video. Experiments across specific types of videos, e.g., slow vs. high motion is left for future work. Sample frames of such videos are shown in figure 5.4. Given the required manual labeling and evaluation efforts, a 2,075-frame clip was extracted from each video (i.e., a total of 6,225 frames with a resolution of 576×324). The experiments were conducted in a PC running Ubuntu 12.04 with 3GB of RAM, 65GB of HDD, a dual core Intel(R) Core(TM) i7 CPU, and 128MB of integrated video memory. Moreover, the framework was implemented using Java SDK 6, C++, JavaCV 0.5 [99], and OpenCV 2.4.5 [100].
Table 4.1: ROI-detection parameter values

<table>
<thead>
<tr>
<th>Param.</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z$</td>
<td>2</td>
<td>step between frames for estimating motion, i.e., $\theta_{z=2}$</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.1</td>
<td>the minimum percentage change upon which the motions of two blocks are dissimilar (static and translational camera)</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>the minimum percentage change upon which the motions of two blocks are dissimilar (zoom camera)</td>
</tr>
<tr>
<td>$\xi_d$</td>
<td>5</td>
<td>the texture-independent cost assigned to block $B_n$ in block-labeling function $E^d_n(l_n)$</td>
</tr>
<tr>
<td>$\tau_{\min}$</td>
<td>0.08</td>
<td>the minimum texture threshold used to assign texture-dependent block-labeling cost $\xi_{\tau_{\min}}$</td>
</tr>
<tr>
<td>$\xi_p$</td>
<td>2</td>
<td>cost assigned in smoothness prior function $E^p$</td>
</tr>
</tbody>
</table>

4.3.1 ROI Effectiveness

This experiment addresses the effectiveness of the proposed ROI detection method based on the accuracy results. Our implementation uses an exhaustive block-based motion estimation [90] using a block size of $35 \times 35$ pixels. The block dimensions were chosen based on empirical studies. Blocks with too small sizes would likely represent only the border areas of the ROI, and blocks with too large sizes would represent too many background areas.

Additional parameters were set as shown in Table 4.1. The texture-dependent cost assigned to block $B_n$ in the block-labeling function $E^d_n(l_n)$ (as shown in equation (4.14)) is assigned based on the function $\text{sign}(\tau_{\min} - \tau(B_n)) \times \left\{ \frac{\lambda_2}{2} + \frac{\lambda_1 - \tau(B_n)}{\tau_{\min}} \right\}$. That is, if the texture of $B_n$ is lower than the threshold $\tau_{\min}$, the cost $\frac{\lambda_2}{2} + \frac{\lambda_1 - \tau(B_n)}{\tau_{\min}}$ is assigned for label $l_n = 1$.

ROIs were detected for each frame of the three clips, and the results were manually annotated as follows.

- "Missed": the frame ROI was not detected at all.
- "Partial": the frame ROI was partially detected.
Table 4.2: Accuracy of ROI Detection Method.

<table>
<thead>
<tr>
<th>Clip</th>
<th>Missed</th>
<th>Partial</th>
<th>Good</th>
<th>Pixel Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>0.05%</td>
<td>1.69%</td>
<td>98.26%</td>
<td>55.22%</td>
</tr>
<tr>
<td>VC2</td>
<td>3.47%</td>
<td>1.59%</td>
<td>94.94%</td>
<td>53.18%</td>
</tr>
<tr>
<td>VC3</td>
<td>0.13%</td>
<td>0.48%</td>
<td>99.37%</td>
<td>29.96%</td>
</tr>
</tbody>
</table>

- “Good”: the frame ROI was detected as a whole.

Table 4.2 shows statistics for “Missed”, “Partial”, and “Good” results, along with the total percentage of pixels that were labeled as background by the ROI detection algorithm. The latter is important as it demonstrates the benefit of using ROIs to identify the important area of a frame and remove from further processing the remaining area. As shown, the proposed ROI detection approach is effective in obtaining the area with local motion. The percentage of “Good” ROIs is above 90% for all three clips. Nevertheless, for VC2, the percentage of missed ROIs is larger than that of the other clips because in VC2 the background public was closer to the fashion models, making more difficult the detection of ROIs. In addition, for clips VC1 and VC2, more than half of the total pixels do not have to be processed by further processing. For clip VC3, about 30% of the pixels were removed, which is still significant.

The ability of reducing a significant percentage of pixels has a direct impact in reducing the computational cost of further processing. For example, for the task of detecting bags in fashion shows, on the aforementioned computing platform, the parts-based object detector by Felzenszwalb et al. [2] takes 0.98 seconds, and detecting the ROI for a frame takes 0.16 seconds. Hence, for clip VC1, the total computational cost of ROI detection and bag detection would be $2075 \times 0.16 + 2075 \times 0.4478 \times 0.98 = 1242.60$ seconds, or 20.71 minutes. However, if the bag detection algorithm is applied on the whole frames, the computational cost would be $2075 \times 0.98 = 2033.50$ seconds, or 33.89 minutes.
4.3.2 Computational Scalability

This experiment compares the running times of the proposed framework and the CUFB approach vs. the allowed computational time, which is given by $\gamma \ast N$. Besides the ROI-detection configuration described in the previous experiment, carrying out this experiment requires the definition of $C(.)$, i.e., the measure of visual correspondence between two ROIs (or two whole frames). The CEDD descriptor [84] was chosen for $C(.)$, as it jointly considers color and texture information. It is important to highlight that using different visual features does not affect the conclusions of the experiment since $C(.)$ is applied equally to ROIs and the whole frames. Thus, changing $C(.)$ has the same effect in both cases.

For clip VC1, the proposed abstraction approach was executed multiple times for $\kappa$ equal to 0.1, 0.3, and 0.5. Only VC1 was used as running times do not depend on the input video. For each $\kappa$, the running times were measured as $\gamma$ increased from 300ms to 1700ms per frame. The weighting factor $\eta$ in equation (4.9) was kept constant at 0.1. Finally, the parts-based object detector of Felzenszwalb et al. [2] was chosen to estimate $D_o(.)$. Such a detector was run multiple times, and a detection time of 0.004795739 ms per pixel was obtained after averaging the detection results for 100 frames sampled from the video clips. All the runs in this experiment utilized $\zeta = 0.004795739 \times 4 = 0.019182956$ so that $D_o(.)$ simulates four object detectors.

The results are shown in figures 4.6, 4.7, and 4.8. Figure 4.6 plots the running times normalized by the allowed time for $\gamma = 300\text{ms}$ to visually depict how the proposed approach computationally scales as $\gamma$ increases. It can be seen that the proposed approach, for all three values of $\kappa$, scales according to the allowed running time, and upon reaching the number of abstract frames specified by $\kappa$, the running times become constant. Consequently, the time spent by the proposed approach
Figure 4.6: Running time of the proposed approach with respect to $\gamma = 300\text{ms}$ for clip VC1 as $\gamma$ increases, with $\zeta = 0.019182956$.

Figure 4.7: Running time of the CUFB approach with respect to $\gamma = 300\text{ms}$ for clip VC1 as $\gamma$ increases, with $\zeta = 0.019182956$. 
on the abstraction process plus the estimated time to execute $D_o(\cdot)$ on such results did not go over the allowed computational time. This is not the case for the CUFB approach, which produces constant running times regardless of the value of $\gamma$ as shown in Figure 4.7. The larger the value of $\kappa$, for more values of $\gamma$ the CUFB approach went over the allowed computational time.

Figure 4.8 plots the percentage change in running time of the proposed approach vs. the CUFB approach. For $\kappa = 0.3$ and 0.5, the proposed approach significantly outperforms the CUFB approach across all values of $\gamma$. For smaller values of $\gamma$, the percentage change is greater as the CUFB approach produces constant running times. After a certain value of $\gamma$, the percentage change becomes constant given that the proposed approach is able to generate the number of abstract frames specified by each $\kappa$. Another observation that can be made from these two curves is that the larger the value of $\kappa$, the more significant is the reduction in running time of the proposed approach. In addition, after $\gamma = 1300\text{ms}$ for $\kappa = 0.3$ and $\gamma = 1500\text{ms}$ for $\kappa = 0.5$, even though there is enough available computational time for both approaches, the
The proposed approach improves the running time by more than 20%. The reason is that, even though the proposed approach spent an average extra 6.8 min obtaining the ROIs, the aggregate number of pixels of the ROIs in the abstract frames is far less than that of the whole frames, so applying $D_o(.)$ to the ROIs reduces the running times, an effect that is more noticeable the larger the value of $\kappa$.

For $\kappa = 0.1$, the curve exhibits the same trend, except that after $\gamma = 700$ms, the proposed approach produces slightly slower running times than the CUFB approach. The reason is that the number of abstract frames to generate is too small, and the application of $D_o(.)$ on the resulting ROIs does not offset the time spent on detecting the ROIs. It can then be concluded that for very small values of $\kappa$, it is best to use the CCFB approach, which is able to scale with $\gamma$ but does not have the extra cost of obtaining ROIs. This conclusion is exemplified in Figure 4.9, which plots the percentage change of the proposed approach and the CCFB approach vs. the CUFB approach for $\kappa = 0.1$. As shown in the figure, the CCFB approach produces the same percentage change as the proposed approach for $\gamma \leq 700$ms, but afterwards produces the same running times as the CUFB given that it does not have to obtain ROIs. It is worth highlighting that the CCFB should not be used for larger values of $\kappa$ as the reduction of the number of pixels when using ROIs outweighs the cost of obtaining the ROIs. Devising a mechanism to automatically choose between the proposed approach and the CCFB approach is left as future work.
4.3.3 Computational Effect of ROI Detection

The second experiment addresses the computational effects of including the ROI-detection algorithm in the abstraction process. Even though the computational effects of determining ROIs was briefly addressed in the previous experiment, this experiment provides more insight in this respect with regards to the number of generated abstract frames. The conclusions of this experiment are important towards understanding the practicality of utilizing ROIs as the basis of the abstraction process because the computational costs of ROIs are obviously higher than those of the whole frames.

For fixed constraints and detection costs, the goal of the experiment is to compare the number of generated abstract frames by the CCFB approach and the proposed approach to evaluate whether the computational costs of ROIs are too significant. If they are, then using ROIs is not practical. The CCFB approach would select much more frames under the same constraints, given the extra cost of computing ROIs. However, ROIs do reduce the pixel area to be processed by the object detection step.
The reduced pixel area produces computational savings that, for certain values of the computational constraint, should allow the proposed approach to generate more frames than the CCFB method.

As in the previous experiment, the parts-based object detector of Felzenszwalb et al. [2] was chosen to estimate $D_o(.)$. For two values of $\zeta$, i.e., the per-pixel computational cost of executing $D_o(.)$, the experiment consists of fixing the temporal constraint $\kappa = 0.1$, 0.3, and 0.5, and varying the computational constraint. Only clip VC1 is utilized as the same conclusions are obtained for the other two clips.

Figure 4.10 shows percentage change in the number of abstract frames of the proposed approach vs. the CCFB approach as the computational constraint $\gamma$ increases from 300ms up to 1700ms. The constant $\zeta$ is set to $0.004795739 \times 2 = 0.009591478$, a value that simulates two object detectors. The plots are representative of expected results. For $\kappa = 0.1$, the percentage change never goes above 0%, which means that the CCFB approach generated the same number of more frames than the proposed approach. After $\gamma = 600$ms, both approaches produce the same number of frames as both meet $\kappa = 0.1$ constraint. The reason is the same as in the previous experiment. For small values of $\kappa$, the cost of obtaining ROIs outweighs the benefits of having fewer pixels on which to execute $D_o(.)$.

Nevertheless, for $\kappa = 0.3$ and 0.5, after a certain value of $\gamma$, the proposed approach produces more abstract frames than the CCFB approach. For $\kappa = 0.3$, after $\gamma = 800$ms, the proposed approach generates more frames, and for $\kappa = 0.5$, it generates more frames after $\gamma = 600$ms. For $\gamma = 1100$ms, both approaches meet the constraint $\kappa = 0.3$, and for $\gamma = 1400$ms, both meet $\kappa = 0.5$. The reason for the proposed approach generating more frames than the CCFB approach after some $\gamma$ is that the application of $D_o(.)$ on the reduced number of ROI pixels does produce computational benefits that allow the proposed approach to generate more frames, even though it
spends extra time in obtaining the ROIs. Again, the larger the value of $\kappa$, the more noticeable is the benefit of using ROIs.

Considering the case of more complicated processing for $D_o(.)$, Figure 4.11 shows the results for the same experiment, with $\zeta = 0.004795739 \times 4 = 0.019182956$ instead, i.e., a simulation of four object detectors. The same conclusions are drawn. For $\kappa = 0.1$, the proposed approach does not generate more frames, but for $\kappa = 0.3$ and 0.5, the proposed approach does produce more frames after some $\gamma$. However, as shown by the area above 0% for the $\kappa = 0.3$ and 0.5 plots, the proposed approach is able to increase the number of output abstract frames compared to the CCFB approach. Since $D_o(.)$ is more costly in this case, the benefits of only applying $D_o(.)$ on the ROIs have a greater impact.

It can be further concluded that the CCFB approach is to be chosen only when the computational constraint is very stringent, and that the proposed approach will outperform the CCFB approach for larger values of $\gamma$ and $D_o(.)$. 

Figure 4.10: Number of output frames as $\gamma$ increases, $\zeta = 0.009591478$. 

![Graph showing percentage change in number of abstract frames vs allowed computational cost per frame]
4.3.4 Abstract Quality

To test our claim that selecting frames based on ROIs allows us to better abstract the view path, this experiment compares the abstract “quality” generated by the proposed approach vs. that by the CUFB approach. For all the runs, the parameters $\gamma$ and $\eta$ were kept constant to eliminate the influence of the computational constraint and the weighting scheme of equation (4.9), respectively. Specifically, $\gamma$ was set to 100000 ms, a significantly large value to avoid the proposed approach being limited by computational costs, and $\eta$ to 0.3. To generate a fixed amount of abstract frame each time, $\kappa$ was changed from 0.01 through 0.5. For each run, the value of $Q(.)$ from the proposed approach was compared to that of the CUFB approach in terms of percentage change.

Figure 4.12 depicts the quality improvement of the proposed approach as the number of abstract frames increases, for clips VC1, VC2, and VC3, respectively. The abstract quality based on ROIs is consistently better than that of the CUFB approach. As expected, in all three cases, the quality improvement is larger towards
Figure 4.12: Quality improvement as the number of output frames increases.

small values of $\kappa$ and decreases as the number of abstract frames increases. However, for $\kappa = 0.01$, the quality improvement is small, reaching its peak for all three plots around $\kappa = 0.05$. The explanation is that both approaches encounter frames that are very dissimilar for $\kappa = 0.01$, but, as $\kappa$ gets close to 0.5, the discriminative power of ROIs over whole frames allows the proposed approach to produce frames that differ between them much more than those selected by the CUFB approach. After $\kappa = 0.5$, since the number of abstract frames gradually increases, more similar frames are included in the abstract, which causes the proposed approach to reduce its quality improvement over the CUFB approach.

Another aspect worth highlighting is the relation between quality improvement and the accuracy of the ROI detection algorithm. As shown in section 4.3.1, the ROI detection algorithm produces more accurate results for VC1 and VC3 when compared to VC2. With similar results, the proposed approach has a higher quality improvement in VC1 and VC3. Such a direct proportionality can be explained considering that the higher the quality of the ROIs, the less influence the background in the
video has in selecting the abstract frames, and thus a higher quality is achieved in the abstract video.

4.3.5 Weight Parameter

This experiment explores how the weight parameter $\eta$ in equation (4.9) affects the advantage of the proposed abstraction approach over the CUFB approach. Following the setup of the previous experiment, the proposed abstraction framework was run multiple times, each time varying $\eta$ while keeping all the other parameters constant. In particular, $\kappa$ was set to 0.3. For each run, the percentage change in $Q(.)$ of the proposed abstraction approach with respect to the CUFB approach was plotted.

Figure 4.13 depicts the quality improvement as $\eta$ goes from 0.0 to 1.0. A value of 0.0 forces the abstraction approach to use only content coverage (equation (4.3)) in determining $Q(.)$, while a value of 1.0 forces it to use only the local representation (equation (4.2)). As shown in the figure, the closer the value of $\eta$ to 1.0, the lower the improvement in quality of the proposed ROI-based abstraction. The largest values of percentage change occur the closer the value of $\eta$ to 0.0. The reason for such a trend is that the CUFB approach generates higher values of local representation due to the inclusion of more background information, and for larger values of $\eta$, the local representation becomes more significant in driving the value of $Q(.)$.

4.4 Conclusions

Smart TVs fail to provide consumers with a seamless TV shopping experience. Instead of directly interacting with the streaming content, consumers must resort to the stores or Web to purchase the products displayed in TV sets. The implementation of such a use case lies in the ability of TV systems in understanding the streaming
content, and object detection is one of the tasks that are necessary to realize such an understanding. However, the question is how to apply the existing wealth of content detection methods in a timely fashion so that consumers are provided with a responsive and effective user interface. To this end, this chapter introduces a multimedia abstraction framework that facilitates the execution of content detection methods in a computationally bound time frame. The framework jointly considers computational and temporal rate constraints to select a subset of video frames that best represent the video contents and ensure the efficient execution of object detection methods. In this sense, the framework is scalable as it is able to restrict the computational cost of the detection phase based on the given constraint values. The abstraction process is formulated as an optimization problem over regions of interest (ROIs), where an optimal solution is obtained via dynamic programming. In its most general case, a region of interest can cover a whole video frame or encapsulate areas of salient motion and texture information. In the latter case, ROIs allow the framework to discard most of the background information, which is beneficial to better represent the

Figure 4.13: Quality improvement for clip VC1 as the weight parameter $\mu$ varies from 0.0 through 1.0.
moving objects. Experimental results demonstrate the benefits of performing content abstraction using ROIs as well as the practicality of the proposed framework.
CHAPTER 5

PRODUCT DETECTION WITH MULTI-CUE OPTIMIZATION

One requirement in the TV shopping use case is the ability to detect complex objects that are characterized by pose variations and irregular forms. Tackling this challenge requires the utilization of additional contextual information as most approaches for static object detection [2, 3, 4, 5, 6] as well as those that mainly rely on motion estimation [7, 8, 9, 10, 11, 12, 13] are not designed for complex objects and perform well mostly in ideal conditions.

The utilization of additional information in the object detection task can be categorized into approaches for still images and those for videos. In the former group, co-occurrence and/or spatial relationships between object labels characterize the additional information [23, 24, 25, 26, 27, 28, 29, 30]. However, these fail to incorporate the temporal information provided by video sequences. Approaches designed for videos are very few [31, 32, 33] and are meant for surveillance applications, which assume specific scene characteristics that are not applicable in the TV shopping use case.

To address the detection of complex objects in the TV shopping use case, this chapter proposes a multi-cue product detection framework for unconstrained video sequences. The framework is generic as it does not depend on specific object detectors and relies on multiple cues to enhance the detection results of complex objects. Two classes of objects are required in the framework. One is the target product class and the other one consists of a related object class. The latter must be easier to detect, in the sense that there is a mature technology for detecting objects of this class, and its purpose is to be the source of some of the additional cues. The multiple cues are formulated into an optimization problem that obtains the optimal path that detections of the target product class should follow in the video considering the path
of the related object occurrences. The first cue relates to the appearance of the object occurrence, i.e., how likely is that the occurrence belongs to the target class. In a video frame, the second cue specifies topological relationships between the target class and the related object class. Lastly, the spatio-temporal cue correlates the paths of the target product and related classes across a video shot.

5.1 Multi-cue Product Detection

As previously introduced, effectively addressing the TV-shopping use case requires the detection of possibly complex objects of a target product class $C$ in an unconstrained video sequence $V = \{F_i\}$, where $F_i$ is the $i^{th}$ frame in $V$. Hence, the problem at hand consists of obtaining the product occurrences of class $C$ in $V$. This chapter proposes to solve the stated problem by dividing $D$ into shots $\{S_k\}$, followed by detecting all product occurrences of class $C$ in each shot $S_i$ using additional cues. More specifically, the approach consists of two steps that are applied on $S_i$. Firstly, an object detector is executed on each frame. The detection threshold of the particular object detector is lowered to increase the chances of detecting complex objects, at the expense of increasing the number of false positive detections but increasing the chances of detecting complex product occurrences. Secondly, additional cues are utilized to obtain the optimal path product occurrences should follow across $S_i$. The optimal path identifies the true positive occurrences and serves to weed out false positive detections. Since the video is divided into shots, it is assumed the shot-boundary method employed for this task will not fragment a continuous scene into many separate shots.
5.1.1 Additional Cues

The additional cues consist of appearance, topological, and spatio-temporal relationships. The appearance cue refers to the visual appearance of the target product class, i.e., how much influence has the visual appearance of the object in determining its class. An example of this cue is depicted in figure 5.1. It represents the task of detecting bags in fashion shows, where the target product class is “bags”. An object detector for bags could generate the three bounding boxes shown in the figure. However, the visual features of the red bounding box should indicate that indeed this is the correct detection. The detection score of each bounding box can be used to quantify the visual information. Additionally, besides analyzing a frame in vacuum, the appearance cue applies to nearby frames. Product occurrences in a neighborhood of frames must have similar visual appearance as the same object should have small changes in appearance from frame to frame.

Figure 5.1: Example of how the appearance cue helps discern between possible object detections.

In contrast, the topological and spatio-temporal cues refer to relations the target product class has with a related object class $C_R$. An implicit requirement is the related object class must be easier to detect in the sense there is a mature technology that robustly detects objects of $C_R$. For example, for the task of detecting bags in fashion shows, the related object class is “faces”. The technology for face detection is quite
Figure 5.2: Example of topological relationship between bags and faces in a fashion show. Yellow bounding boxes depict object detections.

robust, and thus it is possible to use face detection results to enhance the detection of objects of the class “bags”. Nonetheless, false positive detections of the related object class can still occur.

The topological relationship constricts the possible locations for occurrences of the target product class with respect to locations of occurrences of the related object class. Resuming the fashion show example, in the case the model is carrying a bag as depicted in Figure 5.2, there is a clear positional relationship between the model’s face and the bag. Based on this topological relationship, it is clearly possible to use the position of the model’s face to restrict the possible locations for the bag. Lastly, the progression of video frames creates a spatio-temporal correlation between consecutive positions of the target product class and consecutive positions of the related object class, as depicted in Figure 5.3a. Hence,

Based the multiple cues, the proposed approach analyzes the best path occurrences of the target product class can follow in a video shot. Figure 5.3b depicts how the
optimal path weeds out false positive detections. In a succession of video frames \(\{F_i\}_{i=1}^M\), let a possible path be \(P = \{O_i\}_{i=1}^M\), where \(O_i\) is an occurrence of the target product class in \(F_i\); \(Q(P)\) be the quality of \(P\); and \(R_i\) is an occurrence of the related object class in \(F_i\). The definition of \(Q(P)\) is based on the following criteria imposed by the aforementioned cues.

**Appearance.** The appearance cue is modeled as the probability \(P(O_i|C)\), which can be obtained by training an object detector to detect occurrences of target concept class. Any object detection can be utilized, but it is recommended a robust one that can detect occlusions and pose variations. The trained object detector must provide \(P(O_i|C)\) as well as the corresponding bounding box, i.e., the location and size of \(O_i\). Moreover, consecutive product occurrences in the path must have a high appearance similarity, which is defined by
\[ \Omega(\mathcal{O}_i, \mathcal{O}_j) = \begin{cases} 0 & \text{if } i \leq 0 \\ s(\tau(\mathcal{O}_i), \tau(\mathcal{O}_j)) & \text{otherwise} \end{cases} \tag{5.1} \]

where \( i < j \), \( \tau(.) \) is the feature vector representation of a product occurrence’s bounding box, and \( s(.) \) is a function that measures the similarity between the feature vectors of two occurrences, where the image of \( s(.) \) is \([0, 1]\).

**Topological.** Based on the location of the related object class, the topological cue specifies the set of locations from which a product occurrence should not deviate in a frame. With respect to the example of detecting bags in a fashion show, occurrences of “bags” should not be located too far from the location of the model’s face. This requirement is modeled via the following function:

\[ \Psi(\mathcal{O}_i) = \frac{d_l(l(\mathcal{O}_i), l(\mathcal{R}_i))}{b_m} \tag{5.2} \]

where \( l(.) \) provides the location of a detection in the video frame, \( d_l(.) \) is the Euclidean distance between two positions in the frame, and \( b_m \) is a constant that measures the diagonal length of the video frames. Hence, \( \Psi(\mathcal{O}_i) \) assigns larger values the farther the product occurrence is from the related object, and its image is in \([0, 1]\).

It is important to highlight that \( \Psi(.) \) can be defined differently depending on the use case.

**Spatio-Temporal.** With respect to the path of the related object class, the spatio-temporal cue imposes a similar within-path deviation in the trajectory of the target product class. This constraint is modeled via the function:

\[ \Gamma(\mathcal{O}_i, \mathcal{O}_j) = \begin{cases} 0 & \text{if } i \leq 0 \\ \frac{\min(d_l(\mathcal{O}_i, \mathcal{O}_j), d_l(\mathcal{R}_i, \mathcal{R}_j))}{\max(d_l(\mathcal{O}_i, \mathcal{O}_j), d_l(\mathcal{R}_i, \mathcal{R}_j)) + \epsilon} & \text{otherwise} \end{cases} \tag{5.3} \]
where $i < j$ and $\epsilon$ is a small constant $\geq 0$ to avoid dividing by zero. The function $\Gamma(.)$ is proportional to the translational difference between the target product class and the related object class, and its image is in $[0, 1]$.

The best path $\mathcal{P}^*$ that occurrences of the target product class can follow in $\{ \mathcal{F}_i \}$ must have the highest $\sum_{i=1}^{M} P(O_i|C)$, the highest $\sum_{i=1}^{M} \Omega(O_{i-1}, O_i)$, the lowest $\sum_{i=1}^{M} \Psi(O_i)$, and the highest $\sum_{i=1}^{M} \Gamma(O_{i-1}, O_i)$. The optimal path can then be obtained by solving the following optimization problem:

$$
\maximize Q(\mathcal{P}) = \sum_{i=1}^{M} \left\{ \alpha P(O_i|C) + \beta \Omega(O_{i-1}, O_i) \right. \\
\left. \gamma [1 - \Psi(O_i)] + (1 - \alpha - \beta - \gamma) \Gamma(O_{i-1}, O_i) \right\}
$$

where $\alpha$, $\beta$, and $\gamma$ are weight parameters in $[0, 1]$.

5.1.2 Optimal Solution

The optimal solution to equation (5.4) can be efficiently obtained using dynamic programming (DP). Let $\mathcal{P}_k$ denote the path formed by the first $k$ elements in $\mathcal{P}$. Firstly, the cost function $G_k(\mathcal{P}_k)$ is created to represent the maximum cost solution for the first $k$ elements of $\mathcal{P}$ subject to the $k^{th}$ element is $O_k$:

$$
G_k(\mathcal{P}_k) = \maximize_{\mathcal{P}_{k-1}} Q(\mathcal{P}_k)
$$

It is clear that maximizing $G_M(\mathcal{P}_M)$ implies maximizing $Q(\mathcal{P})$. In addition, $G_{k+1}(\mathcal{P}_{k+1})$ can be written as:

$$
G_{k+1}(\mathcal{P}_{k+1}) = G_k(\mathcal{P}_k) + \left\{ \alpha P(O_{k+1}|C) + \beta \Omega(O_k, O_{k+1}) \right. \\
\left. \gamma [1 - \Psi(O_{k+1})] + (1 - \alpha - \beta - \gamma) \Gamma(O_k, O_{k+1}) \right\}
$$
which shows that the selection of the $k + 1$ occurrence in the path does not depend on the previously selected occurrences. This recursive representation makes the next step of the optimization process independent of the previous step, which is the foundation of DP.

Therefore, the problem can be interpreted as finding the longest path in a weighted, directed acyclic graph (DAG) $G = (V, E, w)$, where $V$ is the set of vertices consisting of all the product occurrences found in $\{F_i\}_{i=1}^{M}$, $E$ is the set of edges $\{(O_i, O_{i+1})\}$ where $O_i$ is a product occurrence in $F_i$ and $O_{i+1}$ in $F_{i+1}$, and $w : E \rightarrow (R)$ is an edge-weight function that assigns a weight to each edge as follows:

$$w(O_i, O_{i+1}) = \begin{cases} 
\alpha P(O_{i+1}|C) + \beta \Omega(O_k, O_{i+1}) \\
\gamma [1 - \Psi(O_{k+1})] + (1 - \alpha - \beta - \gamma) \Gamma(O_i, O_{i+1}) 
\end{cases} \quad (5.7)$$

Obtaining the longest path in the DAG via DP takes $O(M t_{\text{max}}^2)$, where $t_{\text{max}}$ is the maximum number of object appearances in a frame of $\{F_i\}$.

### 5.2 Experiments and Results

This section presents and analyzes experiments that demonstrate the advantages of the proposed multi-cue product detection approach. The experiments were conducted in a MacBook Pro with 4GB of RAM, 200GB of HDD, a dual core Intel(R) Core(TM) i7 CPU, and 512MB of video memory. The proposed approach was implemented using Matlab 2012a [101] and OpenCV 2.4.5 [100].

The evaluation was performed on the same fashion show clips utilized in chapter 4 that were obtained from YouTube [98] with high resolution. Figure 5.4 shows more sample frames. Each clip has 2,075 frames, for a total of 6,222 frames with a resolution of $576 \times 324$, and they are referred to as VC1, VC2, and VC3. The target
product class consists of handbags, which represent a significant detection challenge as described in the introduction, and the models’ faces make up the related object class. Figure 5.2 shows bounding boxes for these two classes.

Color histograms were utilized as the feature representation $\tau(.)$ required in equation (5.1), with histogram intersection used as the similarity function $s(.)$. In addition, to detect occurrences of the target and related classes, the following object detectors were utilized:

- The widely utilized Viola-Jones object detector [102] was used to detect the models’ faces. The implementation and trained models that are provided by OpenCV 2.4.5 were incorporated into the Matlab implementation of the proposed approach via the mexopencv [103] library, which provides Matlab mex functions for the OpenCV library.
- The discriminatively trained object detector based on deformable part models of Felzenszwalb et al. [44, 2] was utilized to detect handbags. It represents
variable object classes by using mixtures of deformable part models at different scales. This detector has achieved state-of-the-art results in the PASCAL object detection challenges [104], and its ability to detect non-rigid deformations and partial occlusions in the objects make it a suitable approach for detecting handbags. Moreover, a Matlab implementation of this approach is available online [44], which was directly incorporated into the proposed approach. To train this detector, 500 frames were extracted from the three fashion show videos (excluding the extracted clips). Out of the 500 frames, 250 were positive frames (i.e., contained handbags), while the other 250 were negative frames (i.e., no handbags). The bounding boxes for the handbags in the positive set were manually labeled, thus creating the ground-truth set of bounding boxes.

The experiments consisted in (a) comparing the detection performance of the proposed approach with four other approaches and (b) assessing the effect of video abstraction on the performance results.

5.2.1 Detection Performance

The first comparison is with the approach of Sheikh et al. [31], referred to as the surveillance-application approach. It is representative of the majority methods that use temporal information, which are meant for surveillance videos. Since they assume a fixed camera, these approaches are not suitable for fashion shows that are characterized by many shots and varying background motion. Nevertheless, the comparison with the proposed approach is made to validate this claim. To make a proper comparison, the clips VC1, VC2, and VC3 were divided into shots, and the approach of Sheikh et al. was executed for each shot. The code was obtained from the project’s web site\(^1\). It generates a detection mask for each frame, which was post-processed

\(^1\)http://crcv.ucf.edu/projects/backgroundsub/
Table 5.1: Detection results for the surveillance-application approach.

<table>
<thead>
<tr>
<th></th>
<th>VC1</th>
<th>VC2</th>
<th>VC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.14</td>
<td>0.01</td>
<td>0.09</td>
</tr>
<tr>
<td>Recall</td>
<td>0.21</td>
<td>0.06</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Table 5.2: Detection results for the proposed approach.

<table>
<thead>
<tr>
<th></th>
<th>VC1</th>
<th>VC2</th>
<th>VC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.94</td>
<td>0.92</td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td>0.38</td>
<td>0.54</td>
<td>0.31</td>
</tr>
</tbody>
</table>

using morphological operators. The resulting regions were then represented by bounding boxes. For this and following comparisons, the proposed approach was executed with the weight parameters $\alpha$, $\beta$, and $\gamma$ of equation (5.1) set to $\frac{1}{4}$; i.e., it equally utilized all the equations derived from the three cues.

Tables 5.1 and 5.2 show the detection results in terms of precision and recall values. Precision is defined as the fraction of predicted bounding boxes that are true positives, and recall is defined as the fraction of ground-truth bounding boxes that are predicted. A bounding box is considered true positive if it overlaps more than 50% of the ground-truth bounding box; otherwise, it is considered a false positive. Moreover, if multiple bounding boxes are predicted that overlap with the same ground-truth bounding box, only one is considered correct, and the other ones are labeled as false positives. Precision-recall curves were not generated as the approach of Sheikh et al. does not generate detection scores.

As shown in the tables, the proposed approach significantly out-performs that of the surveillance-application approach. For all three clips, the proposed approach achieved higher recall values as well as precision values over 0.90. On the other hand, the maximum values achieved by the surveillance-application approach were 0.14 precision and 0.21 recall. Such a difference in performance highlights the claim that approaches meant for surveillance videos are not appropriate for unconstrained
video sequences. Additionally, some complex object classes such as handbags are very difficult to detect using mainly motion estimation as these are likely to be immersed in the motion of the person carrying them.

The other comparisons are with the following three approaches, which assume nothing about background motion in the video sequences. The first one is a “plain” product detection approach, which does not use any additional cues and consists of performing object detection on each frame of the video sequence. The second approach is that of Heitz et al. [24] that uses a things-and-stuff (TAS) context model as described in section 2.3. This approach is representative of methods that do not use temporal cues. They can be applied to the TV shopping use case by processing each frame of the video sequence individually. The TAS approach was chosen as its goal is in line with the proposed approach. That is, it is not tied to a specific detection approach but rather utilizes contextual information to enhance the detection results of object detectors. The code was obtained online\(^2\), the CEDD features [84] were used to represent the image regions this approach uses for context, and the code was trained using the same ground-truth bag data used to train object detector of Felzenszwalb et al. Finally, the third approach consists of one sub-optimal version of the proposed approach. This version sets the weight parameters \(\alpha\) and \(\beta\) to 0.5, thus effectively disabling both the topological and spatio-temporal cues to be able to analyze their combined effect. For each frame, each approach predicts bounding boxes of the handbags along with corresponding detection scores. These scores were thresholded to then obtain precision-recall curves for each clip.

Figures 5.5, 5.6, and 5.7 depict the precision-recall curves for VC1, VC2, and VC3, respectively. As shown, the proposed approach significantly outperforms both the plain product detection and TAS detection approaches in terms of precision. The

\(^2\)http://ai.stanford.edu/ gaheitz/Research/TAS/
Figure 5.5: Precision-recall curve for VC1.

Figure 5.6: Precision-recall curve for VC2.
advantage is achieved by removing a significant number of false positive detections and thus improving the precision for the same recall value. For VC1, the maximum recall achieved is 0.38, at which the proposed approach obtains a precision of 0.94 vs. 0.56 for the plain object detector (67.86% improvement) and 0.57 for the TAS approach (64.91% improvement). For VC2, the maximum recall achieved is 0.54, at which the proposed approach obtains a precision of 0.92 vs. 0.43 for the plain object detection (113.56% improvement) and 0.36 for the TAS approach (155.56% improvement). For VC3, the maximum recall achieved is 0.31, at which the full version obtains a precision of 0.97 vs. 0.66, (47.07% improvement) and 0.63 for the TAS approach (53.97% improvement). Sample bounding boxes generated by the proposed and sub-optimal approaches in VC1 are depicted in figure 5.8. The sub-optimal detector generates both true positive (red bounding boxes) and false positive (yellow bounding boxes) detections due to the difficulty of detecting handbags. On the other hand, the proposed approach is able to eliminate the false positive boxes. This result empirically proves the claim that the proposed approach can be used with
Figure 5.8: Examples of predicted bounding boxes for handbags in clip VC1. The plain product detection approach predicts all bounding boxes (both yellow and red ones), whereas the proposed multi-cue approach only predicts the red boxes and eliminates the yellow boxes as false positives.

a low detection threshold to increase the recall of complex objects but still achieve high precision values.

Figure 5.9 highlights the improvement in precision of the proposed approach compared to that of the sub-optimal version. As shown, the improvement is zero for low recall values but rises as the recall increases. A maximum improvement of 8.15% is achieved for VC1, 14.80% for VC2, and 8.92% for VC3. The reason for a larger improvement in VC2 is that more false positive detections were corrected by using the topological and spatio-temporal cues. Moreover, this result underlines the importance of utilizing the related object class in achieving high precision at low recall values.

Another aspect worth discussing is the improved performance of the plain detection approach over that of the TAS approach, in all three video clips. The TAS approach failed to correctly re-score a significant portion of the detections. An ex-
Figure 5.9: Precision improvement (%) of the proposed approach vs. the sub-optimal version.

Figure 5.10: Example of incorrect re-scoring by TAS approach.

ample of which is depicted in figure 5.10, where the red bounding box is the one that was scored the highest by the TAS approach in that frame. The explanation is that the image regions used by the TAS approach as context around the handbags are very similar to those around other parts in the models, which makes it difficult for the TAS approach to correctly re-score the detections.

Moreover, it is important to highlight that the relatively low recall values reported in the precision-recall curves do not detriment the applicability of the proposed approach on fashion shows. These values can be considered to be on a “per occurrence” basis, i.e., computed for each true positive bounding box. However, for the purpose
Table 5.3: Maximum recall achieved.

<table>
<thead>
<tr>
<th></th>
<th>VC1</th>
<th>VC2</th>
<th>VC3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Per Occurrence</strong></td>
<td>0.38</td>
<td>0.58</td>
<td>0.34</td>
</tr>
<tr>
<td><strong>Per Product</strong></td>
<td>1.00</td>
<td>1.00</td>
<td>0.90</td>
</tr>
</tbody>
</table>

of TV shopping, the important criterion is to detect each particular bag at least once, and not necessarily on all the frames the bag appears on. For example, if a bag appears consecutively from frames 1 through 100 as the model walks through the stage, it can be considered a successful product detection if the bag is detected on a subset of the 100 frames, even if it is detected only once or twice. The rationale is that the consumer will still be noticed of the existence of a possibly interesting bag. Hence, it can be stated that recall on a “per product” basis is more important. Table 5.3 compares the per-occurrence and per-product recall values for the three clips. It shows that on a per-product basis, the recall is very high, achieving 1.0 for both VC1 and VC2, and 0.9 for VC3. Precision, on the contrary, still should be considered on a per-occurrence basis as having too many false-positive occurrences has a negative impact on the usability of the system.

5.2.2 Effect of Abstraction

The goal of this experiment is to evaluate the effect the Multimedia Abstraction component has on the performance of the multi-cue product detection framework. Since the former reduces the number of frames in the video sequence, it brings about computational gains but diminishes the information provided by the temporal progression of the frames. That is, subsequent frames in the abstract set may be separated by more than one frame in the original video sequence. As described in section 2.2, most approaches in the literature that utilize temporal relationships are meant for surveillance applications. These rely on motion estimation and thus are affected by
an abstraction step. However, the proposed multi-cue product detection approach utilizes temporal relationships differently; it defines them with respect to the occurrences of the related object class. Therefore, the temporal information in the proposed approach should be effective in the abstract set of frames as well.

To evaluate this claim, figures 5.11, 5.12, 5.13 compare the precision-recall curves of several abstraction levels for VC1, VC2, and VC3, respectively. Equivalent to the original video clips, the curves “100Pct” mean 100% of the original frames. On the other hand, the curves “50Pct”, “30Pct”, and “10Pct” correspond to 50%, 30%, and 10% of the original frames, respectively. These abstracts are the same ones utilized in chapter 4 to compare the effect on abstract quality at various levels of abstraction. As shown in the figures, the effect of abstraction on the detection performance of the multi-cue product detection approach is not significant. For all three clips, the small differences in the curves are the result of different abstraction levels eliminating different number of frames with false positives.
Figure 5.12: Abstraction effect on VC2.

Figure 5.13: Abstraction effect on VC3.
5.3 Conclusion

The lack of content understanding does not allow smart TVs to provide consumers with a seamless TV shopping experience. To purchase interesting items displayed in the current TV show, consumers must inconveniently resort to a store or the Web. Object detection is one of the tasks that are required for realizing the TV shopping use case, but the detection of complex objects poses a significant challenge. To this end, this chapter proposes a multi-cue product detection framework for TV shopping. Three main characteristics define the proposed approach. Firstly, it is generic in the sense that it is not tied to a specific object detection approach. Secondly, it does not make any assumption about motion in the video. Thirdly, it utilizes three cues as additional information to improve the detection results of a target product class. The appearance cue is related to the probability of a product occurrence of corresponding to the target class. The other two consist of topological and spatio-temporal relationships between the target product class and a related, easier-to-detect object class. These enforce spatial relationships within a video frame and across consecutive frames, respectively. The proposed approach jointly considers the three cues as a path-optimization problem that aims at selecting the correct product occurrences and weed out false positive detections. The empirical results demonstrate the advantages of the proposed framework in improving the detection results.
CHAPTER 6

PRODUCT RETRIEVAL

Once products are detected in the video stream and consumers select interesting ones, the TV shopping framework must provide consumers with a list of similar or related products consumers can purchase. This task is related to multimedia information retrieval. However, due to the nature of multimedia data, traditional multimedia retrieval mechanisms are unable to provide efficient and meaningful multimedia data retrieval [16] due to the well-known problems of the semantic gap between low-level features and high-level concepts and the subjectivity of the users’ perception [105]. This chapter presents a novel index structure called the AH\(^{+}\)-tree for image data that addresses such problems during the retrieval process by efficiently incorporating semantic information [40][39]. By indexing a product database, the AH\(^{+}\)-tree provides product retrieval via similarity queries, e.g., nearest neighbor search. In addition, an implementation of the AH-Tree in PostgreSQL is provided, which serves as preliminary work for a similar implementation of the AH\(^{+}\)-tree [43].

6.1 Tree-based Indexing

Efficiently accessing and meaningfully retrieving multimedia data are challenging tasks due to the semantic gap between low-level features and high-level concepts and the users’ perception subjectivity. Tree-based index structures offer an efficient solution to such challenges since they provide both constant access to the indexed records and data retrieval through similarity queries. Consisting of range and nearest neighbor queries, similarity queries allow the retrieval of objects that are “similar” to a given query object. Given the fact that multimedia data are usually transformed into feature vectors on a multidimensional feature space on which a suitable (dis)similarity function can be defined, similarity queries provide content-based retrieval by allowing
the retrieval of objects that are close to the query object. Multidimensional, tree-
based indexing mechanisms, therefore, ought to provide similarity queries to access
multimedia data.

For multidimensional data, several index structures exist in the literature, which
can be categorized into feature-based and distance-based index structures. Please
refer to Chapter 2 for a discussion on this taxonomy. Feature-based index structures
make difficult the introduction of object-level information that is not represented as
a vector because they do not allow any correlation between the feature values in the
distance function used to represent the (dis)similarity of data objects [36]. Providing
a solution to this issue, distance-based structures index the pairwise distances or
similarities of the objects. Representative distance-based index structures are the
vp-tree [77], the M-tree [37], and the AH-Tree [36]. To the best of our knowledge, the
AH-Tree is the only distance-based, tree-structured index method that tackles the
problem of CBIR by incorporating high-level relationships in the index structure.

The AH-Tree combines both vector and metric spaces in a novel way to organize
large image databases and supports CBIR [36]. An important characteristic of the
AH-Tree is that it utilizes information from a learning mechanism to address the issues
of semantic gap and user subjectivity. Specifically, the AH-Tree utilizes the concept
of affinity relationship from the Markov Model Mediator (MMM) model [41]. Affinity
relationships map low-level features to high-level concepts and aim to represent the
users’ perspective of concept-based relationships between the images. The AH-Tree
provides k-nearest-neighbor (kNN) queries that directly support CBIR. Moreover,
the AH-Tree was built upon the M-tree, so in the case when no affinity values are
available, the AH-Tree has the same efficiency and accuracy of the M-tree.

Notwithstanding, the I/O overhead of the AH-Tree prevents it from being uti-
lized on large multimedia datasets. The limitation arises from the process of affinity
promotion, which is executed every time the AH-Tree is queried. The promotion of affinity values [36] begins at the leaves of the tree and promotes the affinity values of the indexed records with respect to the given query upward the tree up to the root node. The affinity value of each internal node is computed as the maximum affinity value between the node’s children. Once the whole tree has been populated with affinity values, the search algorithms are able to visit only nodes whose affinity values with respect to the given query object are higher or equal than a supplied query minimum affinity value, thus computing less distance computations during the query process by avoiding visiting sub-trees that do not comply with the query’s minimum affinity requirement. The process of affinity promotion saves distance computations but forces a complete tree traversal for each query, which causes a significant increase in I/O overhead since each node in the tree has to be read from the disk onto the main memory.

To address the I/O overhead problems of the AH-Tree, this chapter presents the AH$^+$-tree [40][39], which utilizes the affinity values in a novel way to avoid having to perform the process of affinity promotion while still providing the same accuracy of the AH-Tree. Instead of traversing the whole tree to promote affinity values, the AH$^+$-tree efficiently stores and makes use of the affinity information along with object-to-node and node-to-node relationships to identify the sub-trees that need to be trimmed out during the retrieval process. Each query specifies a minimum affinity value, and the AH$^+$-tree only visits nodes whose sub-trees contain indexed objects that comply with the query’s affinity, thus providing the same functionality as that of the AH-Tree but without having to traverse the whole tree. Furthermore, the size of the internal data structures used by the AH$^+$-tree to keep track of object-to-node and node-to-node relationships is insignificant when compared to the size of the multimedia data being indexed. The contribution of this chapter is in presenting an index structure that
utilizes high-level multimedia information in a novel way to significantly improve the
I/O efficiency of the AH-Tree.

6.2 The AH\textsuperscript{+}-tree

The AH\textsuperscript{+}-tree is a height-balanced, tree-based metric index structure for secondary
storage. To handle affinity information in an efficient way, the tree structure is coupled
with two global arrays that track object-to-node and node-to-node relationships in the
tree. Such information is utilized during similarity queries to identify sub-trees that
lead to indexed objects that share semantic relationships with the query objects. The
following sub-sections describe the representation of multimedia objects, the storage
for the affinity information, and the structure of the tree.

6.2.1 Representation of Multimedia Objects

The AH\textsuperscript{+}-tree indexes the keys of the multimedia objects stored in the database.
For example, in an image database, images can be represented by multi-dimensional
feature vectors, and these vectors would be the indexed objects that are stored in the
entries of the tree. Moreover, the AH\textsuperscript{+}-tree requires multimedia objects to be identified via id’s, which are integer values that begin at one and increase incrementally by
one. These id values act as primary keys for the multimedia objects in the database
and do not limit the type of multimedia objects that can be indexed at all. Since the
AH\textsuperscript{+}-tree is a metric tree, meaning that it only indexes pairwise distances between
objects, multimedia objects can be represented using any suitable approach.

Formally, the AH\textsuperscript{+}-tree indexes objects in a database $\mathcal{D}$ with cardinality $T$. $\mathcal{D}$
stores tuples $(O_i, I)$, where $O$ is the key (e.g., multi-dimensional feature representa-
tion) of multimedia object $I$, and $i$, $1 \leq i \leq T$, is the unique identifier of $O$. Moreover,
there exists a total distance function $d$ for the keys in $\mathcal{D}$ that satisfies the metric postulates (see section 2.3). Since the $\text{AH}^+$-tree only handles key values, in subsequent sections the terms key, object, and multimedia object are used interchangeably to refer to the keys indexed by the $\text{AH}^+$-tree.

6.2.2 Affinity Information

The $\text{AH}^+$-tree utilizes affinity information to address the issues of semantic gap and user perception subjectivity. Currently obtained from the Markov Model Mediator mechanism [41], affinity relationships numerically represent how semantically related objects are from the user’s perspective and are based on the idea that the more frequent two images are accessed together, the more related they are. The MMM mechanism is defined as a 5-tuple $\lambda = (\mathcal{S}, \mathcal{F}, \mathcal{A}, \mathcal{B}, \Pi)$, where $\mathcal{S}$ is the set of objects (e.g., images or documents), $\mathcal{F}$ is the set of distinct features (e.g., color features in the HSV color space), $\mathcal{A}$ is the state transition probability distribution, $\mathcal{B}$ is the feature matrix, and $\Pi$ is the initial state probability distribution. The matrix $\mathcal{A}$ holds the affinity information, where $a_{i,j}$ indicates the relationship between object $i$ and object $j$. These affinity relationships are captured through an offline training process that utilizes access patterns and access frequencies from the objects in the database.

Given at set of $q$ queries issued over a period of time, the relative affinity measurement between two images $m$ and $n$ in an image database $\mathcal{D}$ is defined as follows:

$$\text{aff}_{m,n} = \sum_{k=1}^{q} \text{use}_{m,k} \times \text{use}_{n,k} \times \text{access}_k$$

The terms $\text{use}_{m,k}$ and $\text{use}_{n,k}$ denote the usage pattern of image $m$ with respect to query $k$ per time period and of image $n$ with respect to query $k$ respectively; $\text{access}_k$ denotes the access frequency of query $k$ per time period. Through usage patterns and frequencies, the affinity measurements capture high-level image relationships and
thus provide an approach to tackle the problems of semantic gap and user perception subjectivity. Once relative affinity measures are computed, the values of the $A$ matrix are obtained by the following normalization process:

$$a_{m,n} = \frac{aff_{m,n}}{\sum_{i \in D} aff_{m,i}}$$

The AH$^+$-tree is not tied to the MMM mechanism; how the affinity information is obtained is irrelevant to the AH$^+$-tree. Besides the MMM mechanism, other methodologies that make use of mining approaches to identify relationships between multimedia data, such as [106][107][108], can be utilized in the AH$^+$-tree without any loss of generality.

The data structure used to store the affinity relationships is critical to the overhead in memory consumption as well as the performance of the tree. The affinity relationships provide an affinity value between two multimedia objects; therefore, the natural method of storing this relationship is via a matrix, which will be called affinity matrix. However, in a large data set of multimedia objects, the affinity matrix will most likely be a sparse matrix, since it is not feasible for users to provide enough information to generate affinity values for every possible pair of multimedia objects. Consequently, using the traditional data structure used to store sparse matrices, the AH$^+$-tree makes use of linked lists to store the affinity relationships in a way that enables the tree’s retrieval algorithms to efficiently use this information. The affinity relationships are stored in an array of pointers, where index $i$ identifies object $O_i$, and the pointer at index $i$ references a sorted linked list $alist(O_i)$ that is ordered decreasingly by affinity. The affinity value between $O_i$ and $O_j$ is denoted as $aff(O_i, O_j)$. The $k^{th}$ value of $alist(O_i)$ consists of the tuple $(j, aff(O_i, O_j))_k$, where $j$ is the identifier of object $O_j$ and $aff(O_i, O_j) > 0$. 

98
The AH$^+$-tree partitions the metric space into different regions and directs the search process to appropriate regions for efficient retrieval. Its structure consists of internal and leaf nodes, both of which store sets of entries, correspond to regions of the metric space, and are stored in fixed-size pages, similar to nodes in the M-tree [37]. Entries in internal nodes represent ball regions by a routing object $O^r$ and a covering radius $r$, so that all objects reachable from the entry lie within a distance less or equal than $r$. Similarly, entries in leaf nodes have the same elements, but their covering radii is zero, as their reference objects in the database instead of regions in the metric space. Furthermore, entries in the tree keep the distances to their parent entries to make use of the triangular inequality of the metric space to avoid making unnecessary distance computations. Tables 6.1 and 6.2 describe the elements of entries in internal and leaf nodes.

### Table 6.1: Elements of entries in internal nodes

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O^r$</td>
<td>routing object</td>
</tr>
<tr>
<td>$id(O^r)$</td>
<td>$O^r$’s id</td>
</tr>
<tr>
<td>$r$</td>
<td>covering radius</td>
</tr>
<tr>
<td>$d$</td>
<td>distance between $O$ and its parent routing object $O^p$</td>
</tr>
<tr>
<td>$ptr(T(O^r))$</td>
<td>pointer to the corresponding sub-tree</td>
</tr>
<tr>
<td>$id_{node}$</td>
<td>the id of the node referenced by the entry</td>
</tr>
</tbody>
</table>

### Table 6.2: Elements of entries in internal nodes

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O$</td>
<td>indexed object</td>
</tr>
<tr>
<td>$id(O)$</td>
<td>$O$’s id</td>
</tr>
<tr>
<td>$r$</td>
<td>covering radius with value 0</td>
</tr>
<tr>
<td>$d$</td>
<td>distance between $O$ and its parent routing object $O^p$</td>
</tr>
<tr>
<td>$oid(O)$</td>
<td>identifier of the indexed object in the database</td>
</tr>
</tbody>
</table>

6.2.3 Structure
Figure 6.1: Example of space organization using Euclidean distance.

Figure 6.1 depicts two routing objects in some AH$^+$-tree, where the metric distance used is the Euclidean distance. In this figure, all objects in $ptr(T(O^r))$ are within a distance $r_2$ from $O^r$; $O^p$ is the parent object of $O^r$; and $d$ is the distance between $O^r$ and its parent $O^p$. Figure 6.2 depicts the structure of the AH$^+$-tree. $L_1$, $L_2$, and $L_3$ represent leaf nodes; $O_1$, $O_{20}$, and $O_3$ are indexed objects; and $N$ is an internal node that has two entries: $e_1$ and $e_2$.

Figure 6.2: Structure of the AH$^+$-tree.
The AH$^+$-tree enhances the structure of the AH-Tree with the inclusion of two globally accessible arrays: $ObjectToNode$ and $NodeTrack$. Their purpose is to track object-to-node and node-to-node relationships respectively. More specifically, for each indexed object, $ObjectToNode$ allows the identification of the leaf node where the object is located (object-to-node relationship), and for every node in the tree, $NodeTrack$ allows the identification of its parent node in the tree structure (node-to-node relationship). Index $i$ of $ObjectToNode$ corresponds to the indexed object $O_i$, and $ObjectToNode[i]$ is the id of the leaf node that holds $O_i$; for example, in Figure 6.2, $ObjectToNode[3] = 2$, that is, $O_3$ is stored in leaf $L_2$. Following the same idea, index $k$ of $NodeTrack$ corresponds to the node in the tree whose id is $k$, denoted as $N_k$, and $NodeTrack[k]$ stores the id of the parent node of $N_k$ in the tree; in Figure 6.2, $NodeTrack[id(B)] = id(A)$.

The space occupied by these two arrays is “negligible” when compared to the size occupied by the database. For example, consider a database of 10,000,000 images, where images are represented by 500 dimensional feature vectors and corresponding id values. In this database, each image consumes $4 + 8 \times 500 = 512$ bytes, assuming each feature is an 8-byte floating-point number and the id is a 4-byte integer value; and the whole database would consume approximately 4.77 GB. Nevertheless, $ObjectToNode$ would only consume $10,000,000 \times 4 \sim 38.1$ MB, and $NodeTrack$, with a fanout of 100, consume $\sum_{i=1}^{\log_{100} 10,000,000} 10^{i-1} \sim 4$ MB. The small size footprint of $ObjectToNode$ and $NodeTrack$ allows the AH$^+$-tree to manipulate these arrays in main memory without a detrimental impact on real-time costs.

### 6.3 Tree Construction

The AH$^+$-tree is constructed in a similar way as that of the M-tree and R-Tree. The tree structure is kept balanced by inserting new keys at the leaves, splitting nodes that
overflow, and propagating changes upward in the three. *ObjectToNode* is updated appropriately when an object is inserted in a leaf node. When there is a split in the tree due to a node overflow, *NodeTrack* is updated to keep track of the nodes being modified. The algorithm for building the AH+-tree is depicted in algorithms 1, 2, and 3.

### Algorithm 1 Insert($O$)

1. Create entry $e(O)$
2. if the tree is empty then
3. create new leave node $N$ with $id(N) = 1$
4. insert $e(O)$ in $N$
5. Set $MAXINDEX = 1$
6. return
7. end if
8. Set $N =$ root of the tree
9. return $\text{InternalInsert}(N,e(O))$

The Insert method begins in Algorithm 1. For a new object to be inserted $O$, the algorithm either initializes a new tree if the tree is empty (lines 2-7) or calls the helper method $\text{InternalInsert}$. Starting at the root node, $\text{InternalInsert}$ traverses the tree to the leaf node where $O$ should be inserted. At each internal node (lines 1-8), the algorithm picks the sub-tree whose parent entry has the minimum insertion penalty. The penalty policy is guided by two heuristics: either the closest entry to $O$ whose covering radius overlaps $O$ (line 2) or the entry for which the enlargement of its covering radius is minimum among all the entries in $N$ (lines 3-6); in the latter case, the covering radius of the chosen entry is updated to $d(O^i,O)$ (line 5) to enforce the property of the tree that all objects reachable from an entry are within the entry’s covering radius. Respectively, these two heuristics try to obtain well-clustered sub-trees and minimize the overall volume of the current node [37], both of which are beneficial to the performance of the tree. Once the algorithm reaches the bottom level (lines 8-15), it stores the object in the chosen leaf node (line 10) if the node is
not full; upon inserting the new entry, the object-to-node relationship is maintained in line 11 by updating $ObjectToNode$. If the leaf node was full, then $Split$ method is called (line 13).

**Algorithm 2** InternalInsert($N, e(O)$)

1: if $N$ is an internal node then
2: Let $e(O^i) \in N$ be the entry closest to $O$ such that $O$ is within the covering radius, that is, such that $d(O^i, O) \leq e(O^i).r$ and $\forall e(O^j) \in N \ d(O^j, O) \leq d(O^i, O)$
3: if $O$ is not within the covering radius of any entry in $N$ then
4: Let $e(O^i) \in N$ be the entry such that $d(O^i, O) - e(O^i).r$ is minimum
5: Set $e(O^i).r = d(O^i, O)$ // update the covering radius of the chosen entry
6: end if
7: return InternalInsert(ptr($T(O^i)$), $e(O)$)
8: else // $N$ is a leaf node
9: if $N$ is not full then
10: Store $e(O)$ in $N$
11: Set $ObjectToNode[id(O)] = id(N)$ // update object-to-node relationship
12: end if
13: return $Split(N, e(O))$
14: end if
15: end if

Algorithm 3 splits overflowed nodes and propagates changes upward in the tree. The idea of the $Split$ algorithm is to create a new node, choose two entries to be the routing objects of the node being split and the node, divide the entries between the two nodes, and try to insert the chosen two entries in the parent node, then propagate changes upward if necessary.

Given a node to split $N$ and an entry $e(O)$, the algorithm first creates a new node $N^r$ (line 2) and initializes its id to $MAXINDEX$ (line 3), a value that is incremented by one every time a new node is inserted in the tree. From the set of entries in $N$ plus $e(O)$, line 4 promotes two entries that will take the role of routing objects for nodes $N$ and $N^r$. Several criteria can be chosen for promotion algorithms [37]; examples of which are choosing two entries for which (a) the sum of their covering radius is
Algorithm 3 Split($N, e(O)$)

1: $U_N$ be the union of the entries of $N$ and $e(O)$
2: Create new node $N^r$
3: Set $\text{id}(N^r) = \text{MAXINDEX}$ and increment $\text{MAXINDEX}$ // update the id of the newly created node
4: Promote $(e(O^l), e(O^r))$ from $U_N$ // promote two entries to be the routing objects of $N$ and $N^r$ respectively
5: Partition $U_N$ into two sets of entries, one for $N$ (where $O^l$ is the parent object) and the other for $N^r$ (where $O^r$ is the parent object)
6: Set $e(O^l).\text{id}_{node} = \text{id}(N)$
7: Set $e(O^r).\text{id}_{node} = \text{id}(N^r)$
8: Set $\forall$ entries $e(O') \in N^r$ $\text{NodeTrack}[\text{id}(O').\text{id}_{node}] = \text{id}(N^r)$ // NodeTrack needs to reflect that $N^r$ is now the parent of its assigned children nodes
9: \textbf{if} $N$ is the root of the tree \textbf{then}
10: Create a new node $N^p$
11: Set $\text{id}(N^p) = \text{MAXINDEX}$ and increment $\text{MAXINDEX}$
12: Set $\text{NodeTrack}[\text{id}(N)] = \text{id}(N^p)$ // update $\text{NodeTrack}$
13: Set $\text{NodeTrack}[\text{id}(N^r)] = \text{id}(N^p)$ // update $\text{NodeTrack}$
14: Store $e(O^l)$ and $e(O^r)$ in $N^p$
15: \textbf{return}
16: \textbf{else}
17: Let $e(O^p)$ be the entry that points to $N$, stored in $N^p$
18: Replace $e(O^p)$ with $e(O^l)$ in $N^p$
19: Set $\text{NodeTrack}[\text{id}(N^r)] = \text{id}(N^p)$ // update $\text{NodeTrack}$
20: \textbf{if} $N^p$ is not full \textbf{then}
21: Insert $e(O^r)$ in $N^p$
22: \textbf{return}
23: \textbf{else}
24: $\text{Split}(N^p, e(O^r))$
25: \textbf{return}
26: \textbf{end if}
27: \textbf{end if}
minimum or (b) the maximum of their two radii is minimum. Once two entries are chosen, line 5 partitions the remaining entries into two sets, one for node $N$ and the other for $N^r$. Two possible partition criteria are discussed in Ciaccia et al. [37]; the generalized hyperplane partition assigns an entry to its nearest routing object ($O^l$ or $O^r$) which could yield an imbalanced split, and the balanced partition assigns to $O^l$ and $O^r$ their closest entry from the set in a round-robin fashion. In addition to assigning entries to their respective parent entries, the partition process also updates the entries’ distances to their new parents and also updates the covering radii of the two chosen routing objects based on the set of entries assigned to each, keeping updated the $d$ and $r$ values of all entries being modified. Lines 6 and 7 update the chosen entries’ $id_{node}$ values.

At this point, two routing objects have been chosen and the entries have been distributed between $N$ and $N^r$. Before the split process, $NodeTrack$ contained the information that $N$ was the parent node of all of $N$’s children node, but now, some of such entries have been moved to the new node $N^r$, and $NodeTrack$ is outdated. Line 8 addresses this issue and updates $NodeTrack$ by updating the value of all the entries in $N^r$ to their new parent node, thus maintaining node-to-node relationships.

The rest of the $Split$ algorithm (lines 9-27) inserts the chosen routing objects into $N$’s parent node, if there is one, or grows the tree by one level. If $N$ was the root node (line 9-15), lines 10 and 11 create a new node and initialize its id. Since both $N$ and $N^r$ have a new parent node, lines 12 and 13 update $NodeTrack$ accordingly, and line 14 inserts the parent entries for $N$ and $N^r$ in the new root node $N^p$. If $N$ was not the root of the tree (lines 16-27), line 17 obtains $e(O^p)$, the parent entry of $N$ that is stored in node $N^p$. $N$’s parent entry can be easily obtained because the parent node of the current node in $InternalSearch$ can be stored and passed to the $Split$ method. Line 18 replaces $e(O^p)$ with $e(O^l)$ in $N^p$, and line 19 updates the
parent of $N^r$ in $NodeTrack$ to maintain node-to-node relationships. Subsequently, the algorithm tries to insert $e(O^r)$ in $N^p$. If $N^p$ has space for the new entry (line 20), lines 21 inserts $e(O^r)$ in $N^p$, and the algorithm finishes. However, if $N^p$ is full (line 23), line 24 calls the algorithm recursively to split $N^p$ and propagate changes upward in the tree.

6.4 Similarity Queries

Similarity queries are defined by a query object $Q$, explicitly or implicitly specified, and a constraint on the form and extent of proximity required [83], where proximity is typically expressed as a distance. The basic similarity queries are range and $k$ nearest neighbor ($k$NN) queries. A range query receives as parameter a query object $Q$ and a radius $r(Q)$ and returns all objects $O^*$ in the database such that $d(O^*, Q) \leq r(Q)$. Given a query object $Q$ and an integer $k \geq 1$, a $k$NN query returns the $k$ objects in the database that are closest to $Q$. In both cases, $Q$ has to be expressed in the same form as that of the indexed objects in the AH\(^+-\)tree, e.g., as a multi-dimensional feature vector. Through range and $k$NN queries, the AH\(^+-\)tree can support content-based retrieval.

In addition to the aforementioned parameters, range and $k$NN queries in the AH\(^+-\)tree receive the parameter $aff$, which denotes a minimum affinity value the result objects must have with respect to $Q$. For example, $RangeSearch(Q, 0.3, 0.1)$ returns all objects in the database that lie within a distance 0.3 from $Q$ and have an affinity with $Q$ higher or equal than 0.1; similarly, $kNNSearch(Q, 10, 0.1)$ retrieves from the database the 10 closest objects that have an affinity with $Q$ higher or equal than 0.1. The minimum affinity requirement $aff$ allows users to specify a minimum level of semantic relationship between the retrieved objects and the query object. In subsequent sections, if an indexed object $O$ has an affinity with $Q$, i.e., $aff(O, Q)$, higher
or equal than $aff$, it is said that $O$ complies with the minimum affinity requirement; furthermore, if an entry of an internal node leads to indexed objects that satisfy the minimum affinity requirement, it is said that the entry (or the node the entry points to) also satisfy the minimum affinity requirement. Similarly, if an indexed object $O$ lies within a distance $r(Q)$ of $Q$, i.e., $d(O, Q) \leq r(Q)$, it is said that $O$ satisfy the query’s radius; if an entry $e(O^*)$ of an internal node leads to indexed objects that satisfy the query’s radius, i.e., $d(O^*, Q) \leq r(Q) + e(O^*).r$, it is said that the entry satisfies the query’s radius. The following two paragraphs describe the algorithms for range and $k$NN queries. Please refer to Fleites et al. [40] for an in-depth description of these algorithms.

**Range Queries**

The basic idea of the algorithm for range queries is to traverse down the tree, at each level selecting relevant internal nodes and retrieving from the selected leaf nodes the oid’s of the key values that match the query parameters. An internal node is selected if it satisfies both the query’s radius and the minimum affinity requirement. The global arrays $ObjectToNode$ and $NodeTrack$ and $alist(Q)$, the affinity linked list structure that corresponds to $Q$, are utilized to implement the verification of the minimum affinity requirement. The algorithm for range queries is shown in algorithms 4, 5, and 6.

The retrieval process of range queries begins in Algorithm 4. It begins by selecting the root node of the tree (step 1) and locating $alist(Q)$ (line 2). If $alist(Q)$ is empty, then the algorithm performs a range search exactly the same as in the M-tree [37] (lines 3-6). Otherwise, the algorithm constructs an array called $NP$, which stands for node path, and invokes the method $InternalSearch$ (lines 8 and 9). The array $NP$ serves to identify the id’s of the nodes that satisfy the minimum affinity requirement.
Algorithm 4 RangeSearch($Q, r(Q), aff$)

1: Set $N = \text{ROOT}$ // set the root node as the starting point
2: Locate $alist(Q)$
3: \textbf{if} $|alist(Q)| = 0$ \textbf{then} // the list is empty
4: \hspace{1em} Perform search as in the M-tree
5: \hspace{1em} \textbf{return} Result
6: \textbf{end if}
7: \textbf{if} $|alist(Q)| > 0$ \textbf{then} // the list is non-empty
8: \hspace{1em} $NP = \text{ObtainNodePath}(alist(Q), aff)$ // mark in $NP$ the nodes that will be visited
9: \hspace{1em} \textbf{return} InternalSearch($N, Q, r(Q), aff, NP$) // begin range search on the root node
10: \textbf{end if}

Algorithm 5 ObtainNodePath($alist(Q), aff$)

1: Create $NP$ with $|NP| = M$ and all values = $\text{false}$
2: Let $O_q$ be the object at the top of $alist(Q)$
3: Set $id = \text{ObjectToNode}[\text{id}(O_q)]$ // $id$ is the id of the leaf node that holds $O_q$
4: Set $NP[id] = \text{true}$ // mark $id$ in $NP$
5: Set $id = \text{NodeTrack}[id]$ // $id$ now is the id of the parent node of the leaf node that holds $O_q$
6: \textbf{while} $id \neq 0$ \textbf{do} // loop through NodeTrack to mark all the nodes that lead to the leaf node holding $O_q$
7: \hspace{1em} $NP[id] = \text{true}$
8: \hspace{1em} $id = \text{NodeTrack}[id]$
9: \textbf{end while}
10: \textbf{for all} $(id_i, aff_i) \in alist(Q)$ \textbf{do} // iterate through the remaining elements
11: \hspace{1em} Break the loop if $aff_i < aff$
12: \hspace{1em} $id = \text{ObjectToNode}[id]$
13: \hspace{1em} \textbf{if} $NP[id] == \text{false}$ \textbf{then}
14: \hspace{1em} \hspace{1em} $NP[id] = \text{true}$
15: \hspace{1em} \hspace{1em} \textbf{else} // if node was already marked, then continue to next element in the list
16: \hspace{1em} \hspace{1em} \hspace{1em} \textbf{continue}$id$
17: \hspace{1em} \hspace{1em} \textbf{end if}
18: \hspace{1em} \textbf{end for}
19: \hspace{1em} \textbf{while} $id \neq 0$ \textbf{do} // mark the nodes that lead to the current element in $alist(Q)$, stop if a node has already been marked
20: \hspace{1em} \hspace{1em} Break the loop if $NP[id] == \text{true}$
21: \hspace{1em} \hspace{1em} $NP[id] = \text{true}$
22: \hspace{1em} \hspace{1em} $id = \text{NodeTrack}[id]$
23: \hspace{1em} \hspace{1em} \textbf{end while}
24: \hspace{1em} \textbf{end for}
25: \hspace{1em} \textbf{return} $NP$
Algorithm 5 returns an array $NP$ of size $M$ (refer to Section 6.2.3 for the value of $M$). The idea of Algorithm 5 is to select from $alist(Q)$, in decreasing order by affinity, the id’s of objects that have an affinity with $Q$ higher or equal than $aff$, utilize $ObjectToNode$ to mark the leaf nodes that lead to the selected object id’s, and use $NodeTrack$ to mark the internal nodes in the tree that lead to the marked leaf nodes. Lines 1, 2, and 3 respectively create the array $NP$, obtain the first object $O^q$ in $alist(Q)$, and obtain the id of the leaf node that holds $O^q$ using $ObjectToNode$. $O^q$ is the object that has the maximum affinity with $Q$. Line 4 marks in $NP$ the leaf node that holds $O^q$, and line 5 obtains the id of the leaf node’s parent node. Lines 6-9 loop through $NodeTrack$ and mark in $NP$ all the internal nodes in the path from the root of the tree to the parent of the leaf node that leads to $O^q$. Subsequently, the algorithm iterates through all the entries in $alist(Q)$ (lines 10-24) that have an affinity with $Q$ higher or equal than $aff$ (line 11). For each iteration, the algorithm obtains the id of the leaf node that holds the current entry of $alist(Q)$ (line 12), but only marks $NP$ if the leaf node has not previously marked (lines 13-17). Lines 18-23 do the same as lines 6-9; they loop through $NodeTrack$ and mark in $NP$ all the internal nodes that lead to the current entry of the iteration, but abort marking $NP$ (line 20) if the current node had already been marked in a previous iteration. $NP$ is returned in line 25.

With regards to number of iterations, the cost of Algorithm 5 is negligible as it only makes use of small arrays that are easily loaded into memory. It marks $NP$ as many times as nodes need to be searched in the tree in order to find the objects listed in $alist(Q)$ that satisfy the minimum affinity requirement; only needs to work with $ObjectToNode$ and $NodeTrack$, which only need to be created when the tree is constructed; and returns an array of size $M$, which is 0.5 MB for one million images.
Algorithm 6 InternalSearch\((N, Q, r(Q), \text{aff}, NP)\)

1: \textbf{if} \quad N \text{ is an internal node \hspace{0.5cm}} \textbf{then} \\
2: \hspace{1cm} \textbf{for all} \quad e(O^r) \in N \hspace{0.5cm} \textbf{do} \quad \text{// loop through } N\text{'s entries} \\
3: \hspace{2cm} \textbf{if} \quad |d(O^p, Q) - e(O^r).d| \leq r(Q) + e(O^r).r \text{ and } NP[e(O^r).id_{\text{node}}] = \text{true} \hspace{0.5cm} \text{then} \quad \text{// the value } d(O^r, Q) \text{ needs to be computed and the node satisfies the minimum} \\
4: \hspace{3.5cm} \text{affinity requirement} \\
5: \hspace{2cm} \textbf{Compute} \quad d(O^r, Q) \\
6: \hspace{2cm} \textbf{end if} \\
7: \hspace{1cm} \textbf{end if} \\
8: \hspace{1cm} \textbf{end for} \\
9: \hspace{1cm} \textbf{else} \quad \text{// the node is a leaf node} \\
10: \hspace{2cm} \textbf{for all} \quad e(O) \in N \hspace{0.5cm} \textbf{do} \quad \text{// loop through the objects in } N \\
11: \hspace{3cm} \textbf{if} \quad |d(O^p, Q) - e(O).d| \leq r(Q) \text{ and } \text{aff}(O, Q) \geq \text{aff} \hspace{0.5cm} \text{then} \quad \text{// the value} \\
12: \hspace{4.5cm} d(O^r, Q) \text{ needs to be computed and the object satisfies the minimum affinity} \\
13: \hspace{4.5cm} \text{requirement} \\
14: \hspace{3cm} \textbf{Compute} \quad d(O, Q) \\
15: \hspace{3cm} \textbf{end if} \\
16: \hspace{2cm} \textbf{end if} \\
17: \hspace{1cm} \textbf{end for} \\
18: \hspace{1cm} \textbf{end if} 

110
After obtaining the array \( NP \), Algorithm 4 invokes the method \( InternalSearch \), which, beginning at the root node, traverses down the tree selecting internal nodes that satisfy the query’s radius and the minimum affinity requirement. If the current node \( N \) is an internal node (line 1-9), the algorithm iterates through all the entries of \( N \) (line 2). For each entry \( e(O^r) \), it checks (a) if the actual distance \( d(O^r, Q) \) needs to be computed and (b) if \( e(O^r) \)’s node complies with the minimum affinity requirement (line 3). The entry’s child node satisfies the minimum affinity requirement if the node’s id is marked as \textit{true} in the array \( NP \). Because of the triangular inequality property of the metric space, \( d(O^r).d \) (the distance of the current entry to its parent entry) and \( d(O^p, Q) \) (the distance of the parent entry to the query object which was computed in the previous call to the method) can be used to determine if \( d(O^r, Q) \) need be computed. More details about the usage of the triangular inequality property to prune internal nodes and avoid making unnecessary distance computations can be found in Zezula et al. [83]. If the entry cannot be pruned, line 4 computes \( d(O^r, Q) \). Line 5 actually checks if the entry complies with the query’s radius; if so, the method is invoked recursively on the selected entry (line 6). Otherwise, if the current node is a leaf node (line 10-19), the algorithm iterates through all the entries of \( N \) (line 11). For each entry, the algorithm checks if \( d(O, Q) \) needs to be computed and the indexed object complies with the minimum affinity requirement (line 12). If so, line 13 computes \( d(O, Q) \). If the indexed object actually complies with the query’s radius (line 14), the oid of the object is added to the result set (line 15).

Compared to the AH-Tree’s range and \( k \)NN queries without taking into consideration the process of affinity promotion, the extra cost added by the \( AH^+ \)-tree’s range search algorithms consist of Algorithm 5. Nevertheless, Algorithm 5 does not incur in I/O overhead because it makes as many iterations as the number of nodes that must be visited in the tree in order to retrieve all objects that comply with the minimum
affinity requirement, and these iterations are carried out very efficiently in memory with negligible I/O cost.

**kNN Queries**

The algorithm for kNN queries is based on a branch-and-bound technique [36] that makes use of a priority queue $PR$ and a $k$-element array $NN$. $PR$ holds pointers to sub-trees where qualifying objects can be found. $NN$ stores nearest neighbor candidates and at the end of the algorithm contains the $k$ nearest neighbors of $Q$. As with range queries, qualifying objects must comply with the query’s radius and the minimum affinity requirement. Nonetheless, the query’s radius has a different meaning since no radius is specified as parameter. In the kNN query algorithm, the query’s radius $r(Q)$ is set to the largest distance value in $NN$, so $r(Q)$ is dynamic and changes during the execution of the algorithm. Algorithms 7 and 8 implement kNN queries.

Algorithm 7 kNNSearch($Q,k,aff$)

1: Initialize $PR$ with the root node
2: Set $r(Q) = \infty$ // the search radius
3: Locate $alist(Q)$
4: $NP = ObtainNodePath(alist(Q),aff)$ // mark in $NP$ the nodes that will be visited
5: while $PR$ is not empty do // there are nodes where nearest neighbors can be found
6: $NextNode \leftarrow$ choose node $N$ such that $N$ is the node from $PR$ that has the minimum $d_{min}$ (i.e. the first node in $PR$)
7: Remove $NextNode$ from $PR$
8: $NodeSearch(NextNode,Q,k,aff,NP)$ // delegate the search to $NodeSearch$
9: end while

Algorithm 7 initializes $PR$ with the root node of the three (line 1) and sets the query’s radius to $\infty$ (line 2). Line 3 loads $alist(Q)$, and line 4 constructs $NP$ by invoking the method $ObtainNodePath$. Subsequently, the algorithm iterates over
PR while the latter is not empty (line 5). At each iteration, the algorithm retrieves and removes from PR the node N with the smallest \(d_{\text{min}}\) (lines 6-7), which is defined as the minimum distance to Q from any object in the sub-tree of which N is the root node. Utilized in the M-tree [37], this heuristic tries to visit nodes in the tree as efficiently as possible. The chosen N is passed as a parameter to Algorithm 8 (line 8).

Algorithm 8 performs the search on the given node N and updates the queue PR and the array \(NN\). If N is an internal node (lines 2-10), the algorithm loops through the entries of N (line 3). For each entry, the algorithm checks if the entry \(d(O^r, Q)\) need to be computed (utilizing the triangular inequality property exactly the same as in range queries) and also if the entry satisfies the minimum affinity requirement (line 4). If so, \(d(O^r, Q)\) is computed (line 5). Line 6 checks if \(d_{\text{min}}(e(O^r).ptr) \leq r(Q)\), that is if the minimum distance of the sub-tree referenced by \(O^r\) is lower than the current search radius. If so, the node pointed by the entry is added to PR (line 7). If N is a leaf node (lines 11-19), the algorithm loops through all the entries (line 12) and checks each entry exactly the same as in the algorithm for range queries, i.e., if the entry satisfies the current search radius \(r(Q)\) and minimum affinity requirement, making use of the triangular inequality property. If the entry complies with the requirements, line 15 orderly inserts the entry in \(NN\), and line 16 updates the current search radius \(r(Q)\).

6.5 Experiments

This section provides experiments and detailed analyses for the AH\(^+\)-tree index structure. Four experiments are presented. The first experiment provides retrieval results of the AH\(^+\)-tree on a sample product database. The other three compare the AH\(^+\)-tree with the AH-Tree and the M-tree on two large image datasets. The sample product
Algorithm 8 NodeSearch($N, Q, k, \text{aff}, NP$)

1: Let $O^p$ be the parent key of $N$ // needed to avoid computing unnecessary distance computations

2: if $N$ is an internal node then

3: for all $e(O^r)$ in $N$ do // loop through $N$’s entries

4: if $|d(O^p, Q) − e(O^r).r| ≤ r(Q) + e(O^r).r$ and $NP[e(O^p).id_{node}] == \text{true}$ then // the entry is within the current search radius and has been marked in $NP$

5: Compute $d(O^r, Q)$

6: if $d_{\text{min}}(e(O^r).ptr) ≤ r(Q)$ then

7: Add the node $e(O^r).ptr$ to $PR$ // add node referenced by $e(O^r)$ to $PR$ so that it is later used in the search

8: end if

9: end if

10: end for

11: else

12: for all $e(O)$ in $N$ do // loop through the objects of $N$

13: if $|d(O^p, Q) − d(O, O^p)| ≤ r(Q)$ and $\text{aff}(O, Q) ≥ \text{aff}$ then // the object complies with both the current search radius and the query’s affinity requirement

14: Compute $d(O, Q)$

15: Orderly insert ($O, d(O, Q), oid(O)$) into $NN$

16: Set $r(Q) = k^{th}$ distance in $NN$ // update current search radius

17: end if

18: end for

19: end if
database, referred to as $P$, contains 231 product images downloaded from the Web. The number of images in $P$ is small and no affinity relationships are included, but it serves to showcase that the $\text{AH}^+\text{-tree}$ can be utilized for product retrieval. Since the task of product retrieval equates to image retrieval, a more in-depth analysis of the $\text{AH}^+\text{-tree}$ is performed on the other two datasets, which are those used in Fleites et al. [40]. Set $D_1$ consists of 10,000 images from the Corel data set, and set $D_2$ consists of 1,000,000 random data elements. The experimental setup consisted of the same computer hardware and implementation framework.

The first experiment models the product images on the HSV color space via a three-dimensional color histogram. In addition, it discusses the challenges of retrieving products where the query images are cluttered by background information and pose variations. The experiments on $D_1$ and $D_2$ model the images in a 12-dimensional feature space. Extracted from the HSV color space, the features for $D_1$ are “black”, “white”, “red”, “red-yellow”, “yellow”, “yellow-green”, “green”, “green-blue”, “blue”, “blue-purple”, “purple”, and “purple-red”, and the affinity relationships were obtained from the MMM model. For the $D_2$, the feature and affinity values were randomly generated.

The second experiment targets effectiveness, i.e., what is the benefit with regards to relevance when using affinity information. The third one focuses on I/O efficiency, aiming to reflect the benefits of the $\text{AH}^+\text{-tree}$ with regards to I/O performance. Finally, the forth one is directed towards real-time cost, measured in seconds. Real-time cost is not very popular in analyzing metric index structures because this measure is hardware-dependent [109]; however, comparing the real-time cost of the aforementioned three index structures on the same hardware and data set will single out the effect of the $\text{AH}^+\text{-tree}$’s in-memory utilization of the affinity information.
The validity of comparing the AH$^+$-tree with the AH-Tree and the M-tree stems from the fact that the AH-Tree is an extension of the M-tree, and the AH$^+$-tree is an improved version of the AH-Tree. With regards to the base tree structure that indexes the metric space, these three trees are the same. Moreover, it is worth highlighting that the chosen metric space is irrelevant in the subsequent experiments since a change in the metric space (or feature space) has the same impact on these three index structures since their difference lies solely in the incorporation and handling of affinity information.
6.5.1 Product Retrieval

This experiment addresses the effectiveness of the AH$^{+}$-tree as a product retrieval mechanism on $\mathcal{P}$. One challenge in this task is that only the products are important and not the background pixels, i.e., the background should be segmented out from the product images. Background removal is relatively easy to perform on $\mathcal{P}$ as it contains “ideal” images. Figure 6.3 shows some examples. These ideal images contain the product in the center against a white or a clear, consistent background. For this experiment, GrabCut [110, 111] was utilized to remove the background pixels. This is a very effective segmentation algorithm, but it is designed to be interactive, i.e., users must identify a background region in the image so the algorithm correctly segments the foreground. For the images in $\mathcal{P}$, since the images have “ideal” characteristics, user-intervention was avoided by automatically selecting pixels near the edges of the image as background region. Figure 6.4 depicts the accurate foreground masks obtained after applying GrabCut to the images in figure 6.3.

Unfortunately, it is a challenge to obtain the same foreground segmentation results on the query images. The reason is that the product occurrences detected in the video sequences have “difficult” backgrounds as depicted in figure 6.5. The level to which such difficult backgrounds affect the retrieval results depends on how tight the bounding boxes enclose the product occurrences, and usually, it is not practical to assume the bounding boxes will include only foreground information. Addressing this challenge is left as future work in this dissertation, and background removal was not attempted on the queries.
The experiment was carried out by extracting the query images from VC1, VC2, and VC3, i.e., the same fashion show video clips utilized in both chapters 4 and 5. For each product, ten occurrences were extracted as query images. However, only those products for which $P$ contained relevant images were chosen. In total, 450 product occurrences were queried, and for each result set the relevant images were manually annotated. Table 6.3 contains the retrieval results in terms of mean average precision (MAP) for the top five and 10 retrieval results. As shown, reasonable good precision values are achieved for all three video clips. Compared to the other two clips, the precision for VC2 is slightly slower because of a characteristic purple background color that had a greater influence in the results.

### 6.5.2 Effectiveness with Affinity Information

To test the effectiveness of the $AH^+$-tree when affinity information is used, this experiment compares the precision of the results returned by the $AH^+$-tree and the
Figure 6.6: Examples of retrieval results.
Figure 6.7: Effectiveness result for range query with radius = 0.4.
M-tree on $D_1$. Semantically related images were identified through a manual annotation process, and *precision* is defined as the percentage of retrieved images that are semantically related to the query image. Only these two index methods were compared because, on the same database and affinity information, the AH-Tree and the AH$^+$-tree manifest the same precision. Furthermore, $D_2$ is not used in this experiment as it is a randomly generated data set.

The experimental results echo the results obtained by Chatterjee et al. [36] for both range and $k$NN queries. The results obtained using the AH$^+$-tree exhibit a much higher relevance toward the query image than those obtained using the M-tree. For example, figures 6.7 and 6.8 depict the results for two sample queries, one range and one 10NN respectively. The query images are those highlighted with a red border, located on the top left section of the windows. As it is shown, the M-tree retrieves images that are much less relevant, with most images being not relevant at all but having very similar color features. The query image in Figure 6.7 depicts a mountainous landscape, and the images returned by the M-Tree contain cars and buildings, and dolphins (see Figure 6.7a); only three out of 15 images are relevant to the query image. On the other hand, in the result set returned by the AH$^+$-tree (see Figure 6.7b), 13 out of 15 images are semantically related to the query image. Likewise, in the sample 10NN query (Figure 6.8), the AH$^+$-tree is more effective. In this case, the query image depicts animals in natural surroundings. Out of 10 returned images, the M-tree retrieved only three relevant images (see Figure 6.8a), whereas the AH$^+$-tree retrieved nine relevant images (see Figure 6.8b).

On average, the AH$^+$ exhibited a precision of 80%, compared to less than 40% for the M-tree. It is important to reiterate that any possible improvement in the feature representation of the image set to achieve a higher accuracy has the same effect on both the M-tree and the AH$^+$-tree; without affinity information, the AH$^+$-
Figure 6.8: Effectiveness result for 10NN on horses in a natural environment.
tree behaves exactly the same as the M-tree. Furthermore, the fact that only 80% precision was achieved in this experiment for the AH+-tree is not a statement about the maximum precision achievable by the AH+-tree. With a better-trained affinity set, the AH+-tree can achieve higher accuracy. This is one of the benefits of the AH+-tree; it can be supplied with affinity sets from different models, each suitable for particular applications, to achieve results with desirable precision.

6.5.3 I/O Cost

This experiment measures I/O read operations for range and kNN queries on both data sets $\mathcal{D}_1$ and $\mathcal{D}_2$. The query parameters were chosen to be different than those used in Fleites et al. [40].

Figure 6.9 shows the number of I/O read operations performed on data set $\mathcal{D}_1$. For range queries using 0.5 as query radius and 0.001 minimum affinity (see Figure 6.9a), the AH+-tree outperforms both the M-tree and the AH-Tree, and the M-tree performs better than the AH-Tree as the former does not incur in a complete traversal of the tree due to the process of affinity promotion. These results echo Fleites et al. [40]. Since they visit the same nodes after the AH-Tree carries out affinity promotion, the AH+-tree performs a constant amount of I/O reads less than the AH-Tree.

For kNN queries with 0.001 minimum affinity and $k = 10$ (see Figure 6.9b), overall, the AH+-tree performs better than the M-tree. For some queries, e.g., 18 and 64, both perform the same number of I/O reads; and for queries 16, 37, 56, 81, and 86, the AH+-tree performed slightly more I/O reads than the M-tree. As in Fleites et al. [40], these two cases are outliers. Their cause is these query objects mostly have zeroes as their feature values except for two or three features with nonzero values, causing these query objects to have their closest neighbors in the same node or nearby. For these outliers, the M-tree, since it does not consider their affinity relationships, finds their
(a) Range queries with radius = 0.5 and affinity = 0.001

(b) $k$NN queries with affinity = 0.001 and $k = 10$

Figure 6.9: I/O cost on $\mathcal{D}_i$. 

124
nearest neighbors quickly; however, the AH$^+$-tree does consider affinity relationships and finds other nearest neighbors that are more semantically related to the query objects but are not located in the same or nearby nodes.

Figure 6.10 shows the number of I/O read operations performed on data set $D_2$ on a logarithmic scale. For range queries using 0.9 as query radius and 0.001 minimum affinity (see Figure 6.10a), the AH$^+$-tree significantly outperforms both the AH-Tree and the M-tree. By using affinity information, the AH$^+$-tree quickly finds semantically related objects that lie within the specified radius of the query object. As shown in the figure, the M-tree performs a large number of I/O read operations, close to the numbers of the AH-Tree. The reason is that for $D_2$ the query radius 0.9 is large and forces the M-tree to traverse a significant number of nodes in the tree. Figure 6.10b depicts the results for $k$NN queries with 0.001 minimum affinity and $k = 10$. Similar to range queries, the AH$^+$-tree performed best. Again, these results match those in Fleites et al. [40].

### 6.5.4 Real-time Cost

The AH$^+$-tree is able to eliminate the I/O overhead incurred by the AH-Tree by using the global arrays $ObjectToNode$ and $NodePath$ as well as the $NP$ array during query processing (see Section 6.4). The previous experiment shows that such approach is efficient with regards to I/O cost but does not reveal any insight into the overhead caused by the handling of such arrays in main memory. To analyze such overhead, this experiment measures the real-time cost (response time) between the M-tree and the AH$^+$-tree for range queries with radius 0.3 and minimum affinity 0.0 (see Figure 6.11). The minimum affinity value was set to 0.0 to force the AH$^+$-tree visits the same nodes as the M-tree and still utilize the aforementioned global arrays; consequently,
(a) Range queries with radius = 0.9 and affinity = 0.001

(b) $k$NN queries with affinity = 0.001 and $k = 10$

Figure 6.10: I/O cost on $\mathcal{D}_2$. 

126
any difference in real-time cost shown in the plot is due to the handling of the global arrays during query processing.

Figure 6.11 depicts real-time cost vs. number of objects in data set $\mathcal{D}_2$ for both the AH$^+$-tree and the M-tree. The plot shows the overhead incurred by the AH$^+$-tree in utilizing the arrays $ObjectToNode$, $NodeTrack$, and $NP$ during query processing is negligible, thus supporting the claim the AH$^+$-tree significantly reduces the I/O cost of the AH-Tree without a tangible penalization in real-time cost.

6.6 The AH-Tree in PostgreSQL

This section presents a system based on PostgreSQL [112] that utilizes the AH-Tree [36] as index method to efficiently support CBIR [43]. The AH-Tree is incorporated inside PostgreSQL’s kernel by internally modifying PostgreSQL’s GiST access mechanism. Furthermore, to eliminate the I/O overhead incurred by the AH-Tree when populating the affinity information through the tree structure for each query, the presented system only utilizes the affinity information at the leafs of the tree, mak-
ing possible its implementation in PostgreSQL while still providing the same level of functionality.

6.6.1 System Architecture

The proposed system consists of a PostgreSQL DBMS whose kernel-level indexing mechanism has been extended to support the AH-Tree. The incorporation of the AH-Tree was achieved via modifying PostgreSQL’s internal GiST mechanism [113], which serves as a template framework for implementing balanced, tree-based index structures such as the B-tree. Figure 6.12 depicts the implementation of the AH-Tree in PostgreSQL. Given a user query, the query processor in the DBMS kernel parses the query, finds the optimal execution plan, and executes the plan by interacting with the access method, i.e., the AH-Tree.

![Figure 6.12: AH-Tree’s implementation in PostgreSQL.](image)

The GiST-based implementation of an index method requires several user-implemented index support functions, which are utilized by GiST in its insert, search, and delete core functions. Figure 6.12 shows the support functions in a blue font. For example, for an insert query, the insert algorithm descends the tree by selecting at each level the node with minimum insertion penalty (\( \text{Penalty} \)) function, and when it reaches a
leaf node, the insertion of the new key is attempted. If there is space in the selected leaf, the key is inserted; otherwise, the node is split (PickSplit function), and changes are propagated upward in the tree using the Union support function. For a search query, the Consistent function is utilized to determine the tree nodes that have to be visited. GiST’s delete algorithm searches the key to be deleted, removes the key from its leaf node, and if the deletion causes an underflow, changes are propagated upward in the tree. Furthermore, during the insert, search, and delete processes, the access method needs to interact with the storage manager to access persistent data and guarantee concurrent executions via locking mechanisms.

One key feature of the AH-Tree is that it uses the triangular inequality property to avoid making unnecessary distance computations. This feature is partially implemented by having tree entries store the distances to their parents, which are updated during node splits. However, the Union function, which is used to create parent keys when a node is split, does not support such updates as it does not distinguish between parent and child keys. For this reason, the proposed system modifies PostgreSQL’s core GiST insert, search, and delete functions to use an extended set of support functions that replaces the Union function by the functions UpdateAncestor and UpdateDescendant and adds the KnnConsistent function for nearest neighbor queries. The updated set of support functions is shown in a read font in Figure 6.12. The functions UpdateAncestor and UpdateDescendant allow the update of the parent entry based on its child entries and a child entry based on its parent entry, respectively. These two functions are invoked during a node split.

6.7 Conclusions

To implement the TV shopping use case, consumers must be able to make purchases of products that are similar to the product occurrences they selected. To achieve this
goal, a multimedia retrieval mechanism must be in place to retrieve similar items from a product database. However, such a task suffers from the semantic gap between low-level features and high-level concepts and the subjectivity in users’ perception. Index structures should not only provide efficient access but also meaningful retrieval. This chapter has presented the AH$^+$-tree, a height-balanced, distance-based, tree index structure that efficiently addresses the problems of semantic gap and user perception subjectivity by incorporating during the retrieval process high-level affinity information. The AH$^+$-tree eliminates the I/O overhead incurred by the AH-Tree during query processing while still providing the same effectiveness. The structure of the AH$^+$-tree efficiently stores the affinity information and keeps track of object-to-node and node-to-node relationships in the tree to efficiently prune irrelevant tree nodes during query processing and retrieve indexed objects that have a semantic relationship with the query object. Experimental results demonstrate the improved relevance in the results when high-level affinity information is utilized, confirm the I/O efficiency of the AH$^+$-tree over the AH-Tree and the M-tree, and corroborate the AH$^+$-tree reduces I/O cost with a negligible impact on real-time cost. The ability of the AH$^+$-tree to efficiently incorporate high-level affinity information in the retrieval process to address problems inherent to multimedia retrieval make the AH$^+$-tree a promising index access method for large multimedia databases. Additionally, an implementation of the AH-Tree in PostgreSQL is described, which lays down the preliminary work for implementing the AH$^+$-tree in PostgreSQL.
CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This dissertation proposes a scalable multimedia content processing framework for TV shopping applications. TV shopping is a novel use case that allows consumers interact with the contents being streamed to make purchases. From the consumer’s perspective, while watching a TV show, he or she is notified by a non-intrusive notification on the TV set that products have been detected in the show. The consumer can then see the detections and browse through them, and select one for purchase. Subsequently, the TV set provides the consumer with a list of related products that he or she can buy. Two core tasks must be implemented to allow such a use case. The first one is the detection task, which detects products in the content stream. The second one is the retrieval task that, given the product occurrence selected by the consumer, provides a retrieval mechanism for similar products in a product database.

The components of proposed framework address the two TV shopping tasks. For the detection task, both the Multimedia Abstraction and Product Detection components are executed. The former abstracts the video sequence into a set of representative frames that allows the efficient execution of the second component. This step considers the computational cost of the detection phase and provides a mechanism to detect products in the content stream efficiently irrespective of the particular object detectors that are employed. On the abstract frames, the Product Detection component utilizes multiple cues of information to detect complex products. This component is generic to the particular object detectors that are utilized. Instead, the Product Detection component acts as a post-detection layer that eliminates false positives and thus enhances the detection results. For the retrieval task, the Retrieval component
incorporates an efficient indexing mechanism that is able to retrieve similar products from a product database. Additionally, this index method utilizes semantic information during the retrieval process to address the inherent problems of multimedia retrieval.

The Multimedia Abstraction component offers a computationally scalable solution that supports the efficient execution of complex detection algorithms. The computational costs are modeled via user-specified computational and temporal constraints under which the object detection task must be completed. The framework then abstracts the video into a subset of frames that best represent the contents and for which object detection executes under the specified constraints. Given computational and temporal rate constraints, the abstraction problem is formulated as an optimization problem based on local representation and content coverage criteria. It is worth pointing out that the object detection task is not part of the proposed approach, but rather an estimation of its computational cost is used when solving the optimization problem. Furthermore, the abstraction problem is based on regions of interest, not on the whole video frames, which allows obtaining solutions that better represent the important moving information in the video. Such regions of interest are computed via a novel and efficient approach that incorporates block-based motion estimation and texture information. The advantages of the proposed abstraction approach, as well as the benefits of using regions of interest, are demonstrated by the experimental results.

The Product Detection component addresses the detection of products on the abstract frames. Specifically, it is capable of detecting products that are challenging, i.e., those that do not have a definite shape and can present pose deformations. This component first executes an external object detector and lowers the detection threshold to increase the chances of detecting complex objects, at the expense of increase the
number of false positive detections. Subsequently, the component combines multiple cues of information into an optimization problem that selects the optimal path product occurrences should follow across the frames. The detection results are improved by eliminating the false positive detections that are pruned out by the optimal path. Three additional cues are implemented. The first one is the appearance cue, which concerns the visual appearance of the product occurrences. This cue is modeled as the probability that the occurrences belong to the target product class, and it also favors occurrences that are similar to those in nearby frames. The other two cues depend on a related object class that is easier to detect. In the case of TV shopping in a fashion show, one example of the related object class is the models’ faces as there is a mature technology for detecting faces, and it is possible then to use the detection results of the faces to select the true positive results of a target product class such as handbags or clothing. The second cue is the topological cue, which constrains the possible locations for the target product class based on the locations of the related object class. Lastly, the spatio-temporal cue correlates the paths across the frames of both the target and related classes.

The Product Retrieval component supports product retrieval via the AH$^+$-tree, a height-balanced, multi-dimensional tree-based index mechanism that utilizes semantic information to address the aforementioned problems of multimedia retrieval. Obtained from the Markov Model Mediator (MMM) mechanism, the semantic information consists of affinity relationships between the multimedia elements that capture concept-based relationships based on user access patterns. The AH$^+$-tree improves upon the AH-Tree by eliminating the I/O overhead incurred by the latter when promoting affinity information throughout the tree for each retrieval query. The I/O overhead is eliminated by modifying the structure of the tree with global arrays that
keep track of object-to-node and node-to-node relationships. Experimental results confirm the I/O and query efficiency of the $\text{AH}^+$-tree.

### 7.2 Future Work

Future research directions for the proposed framework include:

1. **Assessing the applicability of the proposed framework in other application domains.** In this dissertation, TV Shopping is the use case on which the proposed scalable content processing framework is evaluated. However, more experiments should be performed on other domains, such as sports videos. One possible use case in this domain consists of detecting “events” in a sport video and retrieving similar events from a collection of videos. In this case, target products would consist of game events. The three components of the framework would fulfill such a use case as follows:

<table>
<thead>
<tr>
<th>Missed</th>
<th>Partial</th>
<th>Good</th>
<th>Pixel Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.16%</td>
<td>1.77%</td>
<td>87.07%</td>
<td>39.12%</td>
</tr>
</tbody>
</table>
The Multimedia Abstraction component would provide the benefit of obtaining an optimal set of frames that summarizes the information in the video, while considering computational and temporal constraints. The scalability advantages would also apply, but accurately detecting ROIs across several domains is a challenging task. The proposed ROI detection method detects frame areas of local motion. It does so by analyzing texture and changes in motion direction between adjacent blocks. To be able to detect ROIs on different video domains, the various thresholds used in the ROI detection algorithm must be experimented with and possibly employ data mining techniques to devise adaptive approaches that detect and accommodate to the video pattern. Nevertheless, preliminary results do confirm that the proposed ROI detection algorithm can produce promising results on sports videos, as depicted in figure 7.1 and table 7.1 for a basketball clip. As shown in the figure, the ROIs are coarsely detected, even though there is camera motion as well as motion caused by the public waving hands and standing up. Specifically, the clip consisted of 1,075 frames, on which the ROI detection algorithm was executed, and the results were annotated following the same scheme described in section 4.3.1. Table 7.1 shows the statistics for “Missed”, “Partial”, and “Good”, along with the total percentage of pixels that were categorized as background by the ROI detection algorithm. The percentage of “Missed” and “Good” are 11.16% and 87.07%, respectively, and 39.12% of the total pixels were labeled as background pixels. As a comparison, for the fashion show clips utilized throughout this dissertation, the average percentage of “Missed” and “Good” are 1.22% and 97.52%, respectively, and 46.12% of the pixels
were labeled as background. The percentage values of “Partial” are similar in both domains.

(b) The Product Detection component would detect events in the sport video. The existing literature on event detection is extensive, but the effect of video abstraction on the detection accuracy is unknown and would need to be researched. Moreover, to support a variety of use cases, the Product Detection component would have to incorporate an array of techniques (e.g., object detection, concept detection, event detection).

(c) The Product Retrieval component would search for similar events in a collection of sport videos, after the consumer selects an interesting event. The proposed $AH^+\text{-tree}$ index structure is able to search for similar events based on individual frames and incorporates semantic information to enhance the retrieval performance. Additionally, several frames in the same video event can be searched simultaneously to enhance the retrieval results at the event level.

2. **Enhancing product retrieval results.** The performance of the Product Retrieval component can be enhanced by (a) reducing the influence of the background pixels in the product detections and (b) combining the retrieval results of nearby frames. As described in chapter 5, images in a product database usually have “ideal” characteristics and their background pixels can be removed effectively. However, the product detections are cluttered by background pixels whose removal is not trivial. This challenge can be addressed by analyzing the contextual areas surrounding the product detections to thus model the probabilities of the background pixels. Furthermore, product detections in nearby frames are likely to represent different poses of the same product. Hence, the retrieval results
of a neighborhood of frames can be combined to enhance the relevancy of the results.

3. **Incorporating a recommendation component.** The current framework is able to retrieve similar products allowing the consumer to make a purchase. However, a recommendation component would allow the framework to not only retrieve similar products but also recommend related ones to the consumers. Recommendation is an active research area [114, 115, 116], and based on the consumer purchase histories, collaborative filtering approaches can be implemented. Additionally, novel solutions can be provided by making use of purchase histories to try to identify the members of the household and thus be able to make more accurate recommendations.

4. **Implementing the AH$^+$-tree in PostgreSQL.** Incorporating the AH$^+$-tree as an index access method in PostgreSQL would combine the benefits provided by traditional DBMSs with those of a meaningful and efficient content-based retrieval mechanism. Section 6.6 already describes how to implement the AH-Tree, the precursor of the AH$^+$-tree, in PostgreSQL. However, incorporating the modifications of the AH$^+$-tree into PostgreSQL’s access method subsystem is not a trivial task. The modifications of the tree structure consist of two global arrays that track object-to-node and node-to-node relationships in the tree. One possible approach would be to store these arrays in the header section of the root node of the tree and rely on PostgreSQL’s cache subsystem to process them in memory.
BIBLIOGRAPHY


video segmentation from superpixel flows,” in Proceedings of the 11th European

era,” in Image Processing (ICIP), 2010 17th IEEE International Conference

[12] G. van Essen, S. Marsland, and J. Lewis, “Hierarchical block-based image regis-
tration for computing multiple image motions,” in Image and Vision Computing
430.

video object segmentation,” Image Processing, IEEE Transactions on, vol. 17,
no. 6, pp. 958–967, 2008.

trecvid: A 5-year retrospective of achievements,” in Multimedia Content Anal-
ysis, ser. Signals and Communication Technology, A. Divakaran, Ed. Springer

Smeaton, and G. Quénot, “Trecvid 2012 - an overview of the goals, tasks, data,
evaluation mechanisms and metrics,” in TRECVID, 2012.

[16] A. Yoshitaka and T. Ichikawa, “A survey on content-based retrieval for mul-
timedia databases,” IEEE Transactions on Knowledge and Data Engineering,

color feature based color image compression,” in Smart Structures and Systems
(ICSSS), 2013 IEEE International Conference on, March 2013, pp. 82–86.

on shape-feature and mpeg-7 standard,” in Consumer Electronics, Communications
and Networks (CECNet), 2011 International Conference on, April 2011,
pp. 3095–3097.


VITA

FAUSTO C. FLEITES

Born, Havana, Cuba

2001–2004
B.S. in Computer Science (105 credits completed)
Havana University
Havana, Cuba

2005–2009
B.S. in Computer Science
School of Computing and Information Sciences
Florida International University
Miami, Florida

2009–2012
M.S. in Computer Science
School of Computing and Information Sciences
Florida International University
Miami, Florida

2009–2014
Ph.D. Candidate in Computer Science
School of Computing and Information Sciences
Florida International University
Miami, Florida

PUBLICATIONS


