Utility-based Network Adaptation for MPEG-4 Systems

Paul Bocheck, Andrew T. Campbell, Shih-Fu Chang and Raymond R.-F. Liao[*]

Dept. of Electrical Engineering,
Columbia University
{bocheck, campbell, sfchang, liao }@ee.columbia.edu

Abstract— In this paper, we present the design, implementation and evaluation of a utility-based network adaptation system for the delivery of MPEG-4 video over time-varying networks such as wireless networks. The system comprises a number of algorithms that can be programmed to support application-specific adaptation in the network or at its edges. By exploiting video content and user preferences, an online utility generator can dynamically formulate *bandwidth utility functions* and *scaling profiles* that capture application-specific adaptation. The utility generator employs a video content classification algorithm to speedup processing and an adaptive prediction algorithm to keep dynamically generated utility functions meaningful over network adaptation time-scales. Network adaptation is based on a *utility-fair* bandwidth renegotiation algorithm that realizes profile-based media scaling. The signaling system efficiently delivers utility functions and scaling profiles to network adaptation agents.

Keywords— utility functions, content-aware adaptation, media scaling, MEPG-4

I. INTRODUCTION

The emerging MPEG-4 video-coding standard [5] is suitable for a wide range of applications (e.g., storage and communication applications). The most distinctive feature of MPEG-4 is its ability to independently encode video objects appearing in a scene. Object-based treatment of video sequences enables exploration of a new type of media scaling called content-based scalability. It is widely agreed that the performance of distributed multimedia content (e.g., video streams) can be improved by exploiting the intrinsic scalability of content through rate control techniques coupled with effective media scaling and periodic bandwidth renegotiation. This can result in a significant increase in the utilization of network capacity [1]. These techniques are well suited toward transporting and scaling video content in response to time-varying bandwidth availability typically found in the Internet and more characteristically in wireless and mobile networks.

Bandwidth utility functions [9] can be used to characterize an application's capability to adapt over a range of available bandwidth. For video applications utility takes the form of a video quality metric. Studies of video quality measurement have placed emphasis on the design of accurate perceptual metrics [15] [7]. However, little attention has been paid to the generation of these perceptual quality metrics. In network economics research, utility functions are used as a theoretical abstraction of application demands for network pricing [14] and the optimization of resource allocation in wireline [13] and wireless [8] [9] networks. Recently, we have developed an approach for the dynamic generation of utility functions [10] [2]. In this paper, we build on this work and design a utility-based network adaptation system enabling content-aware adaptation for MEPG-4 video delivery.

The utility-based adaptation framework comprises three modules as illustrated in Figure 1: a content scaler, a bandwidth allocator and a utility generator. The content scaler performs content-based rate control interacting with the bandwidth allocator to realize utility-based bandwidth allocation [10] [13] in the network. The utility generator dynamically creates bandwidth utility functions on-demand for the content scaler and bandwidth allocator modules. The utility generator can be located at the video server to gain access to MPEG-4 video object information without additional transcoding overhead or at any server in the network or at its edges. Typically, the content scaler would be placed close to or at the network bottleneck. For example, in the case of time-varying wireless networks the content scaler could be installed at base station to dynamically scale downlink media. Being able to program and locate the components of the architecture has a number of benefits. Base-to-mobile scaling can react to time-varying bandwidth over faster time-scales and in a more scalable in comparison to end-to-end adaptation approaches [16].

The structure of the paper is as follows. In Section II we present a system architecture that addresses a number of technical barriers facing the design of utility-based adaptation systems. In Section III we introduce the concept of *scaling profiles* that characterize the scaling actions that can be applied to the transmission of multiple MPEG-4 video objects in the same session. Following this, in Section IV we discuss an approach to increase the speed of utility generation using content classification techniques. In Section V, we present an adaptive prediction technique that keeps generated utility functions meaningful over network adaptation time-scales. Next, in Section VI, we discuss our experiment results and finally, in Section VII, we conclude with some remarks.

[*] Corresponding author.

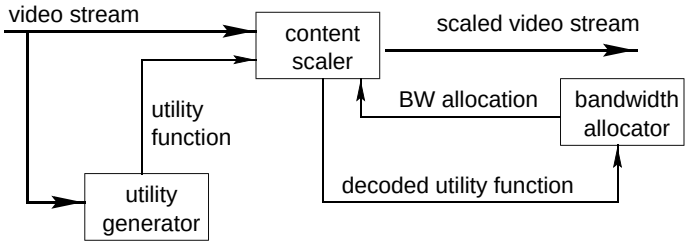


Figure 1: Utility-based Adaptation Framework

II. System Architecture

The design of utility-based network adaptation systems is dependent on a number of systems issues including encoding techniques, video content and user preferences. Therefore, a single generic design solution to these issues is unlikely. Rather, a solution capable of accommodating a range of system considerations through programmability is more likely to succeed.

A. Challenges

A number of technical challenges motivate the design of our utility-based MPEG-4 system. First, with the emergence of MPEG-4 technology, one single packet stream can contain a number of elementary streams corresponding to different media objects with distinct adaptation needs. Scaling techniques applied to each individual media object can vary with video content changes. Therefore any model characterizing MPEG-4 media scalability should take into account at least two "levels of multiplexing"; that is, within media objects and across multiple media objects. Second, utility function generation requires the measurement of video quality distortion for some given video sample rate. This procedure can be computationally intensive when on considers fine-grain sampling. Care has to be taken to reduce the impact of dynamic utility generation on the transport system. This can be achieved in part by making sure that the utility generator is architecturally separated from packet forwarding path and possibly coupled with the video server avoiding any additional transcoding where possible. Any excessive delay introduced in the utility generation process can render the generated utility functions obsolete. The architecture needs to minimize any computationally intensive procedures so that utility functions can be dynamically generated in real-time. Third, utility functions generated for video may dynamically change over fast time-scales (e.g., in the order of tens of milliseconds) as content changes (e.g., due to scene changes). We observe that network adaptation operates over a longer time-scale, potentially in the order of hundreds of milliseconds to tens of seconds. This is a product of the signaling system efficiency, resulting network load of dynamic resource management and the round trip delay between a source encoder and receiving decoder. Since the "content time-scale" may be several orders of magnitude smaller than the "network adaptation time-scale", the utility generator needs to reconcile this mismatch without sacrificing the accuracy of generated utility functions or burdening the network with excessive signaling.

B. System Architecture

The utility-based network adaptation system architecture illustrated in Figure 2 addresses these technical challenges. The system architecture represents a further decomposition of the utility generator, content scaler and bandwidth allocator components. The utility generator comprises a scaling profile selector, content-based utility function estimator and a long-range utility function predictor. The scaling profile selector generates "scaling profiles" that captures MPEG-4 stream scalability. This takes into account user preferences and includes aggregation of multiple media objects and the selection of potentially multiple scaling techniques for a single media object. A content analyzer associated with the scaling profile selector extracts content features from compressed video streams. The scaling profile selector uses content features to select an appropriate scaling profile. Signaling utilizes the MPEG-4 object descriptor structure [5] to signal utility functions and scaling profiles to the content scaler as illustrated in

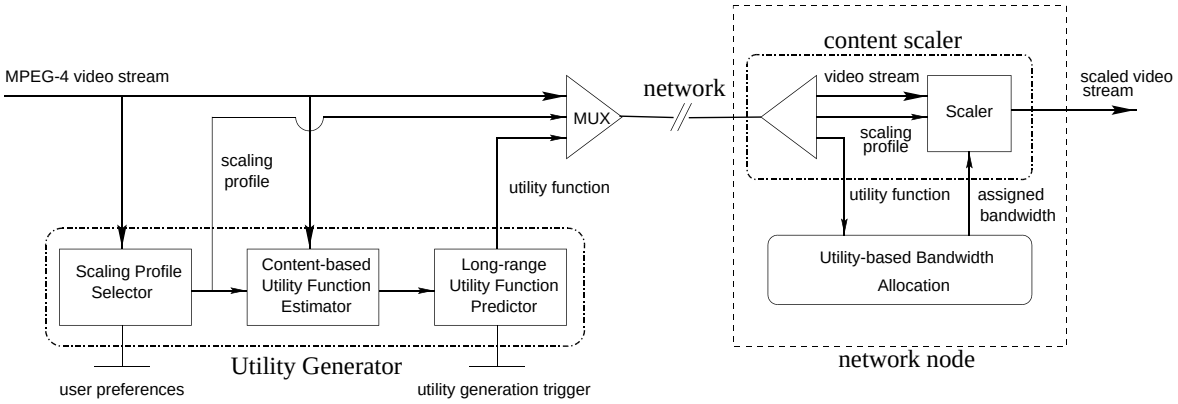


Figure 2: System Architecture

Figure 2.

By applying machine learning techniques to video content classification [2] the utility estimator uses content features to classify incoming video objects. Instead of generating a new utility function for each video object, the estimator reuses utility functions generated for a class containing a particular video object thereby significantly increasing the performance of the overall utility generator.

The long-range utility function predictor resolves any potential mismatch between "content" and "network adaptation" time-scales discussed above. An adaptive filtering algorithm [10] keeps the generated utility functions meaningful over longer network adaptation time-scales. The algorithm adjusts to track the long-term variation in utility functions to balance the tradeoff between increasing the utility generation interval and maintaining the accuracy of generated utility functions.

In the following, we discuss the formation of scaling profiles and present the detail design of the content-based utility function estimator and long-range utility function predictor. Following this, we present our system evaluation.

III. Formulating Scaling Profiles

The role of scaling profiles is to ensure that the content scaler, which can be located inside or at the edges of the network, scales media based on the selects scaling methods used during the generation of utility functions. This results in the effective scaling of video stream data rates keeping the content scaler's operating point for a particular stream on its associated utility curve. Media scaling techniques can be broadly categorized as follows:

- *spatial resolution scaling*, which changes picture size (e.g., from CIF to QCIF require transcoding);
- *temporal domain scaling*, which drops frames;
- *quality scaling*, which changes quantization levels, chrominance dropping, Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) coefficients dropping; and
- *content-based scaling*, which uses MPEG-4 video object prioritization and dropping.

Typically, the majority of scaling actions generate coarse-grained rate changes that can be estimated by the amount of data dropped in terms of frames, layers or objects. The resulting distortion function registers a discrete drop in quality when the rate is scaled-down. In contrast, dropping transform coefficients supports fine-granularity rate changes (e.g., the dynamic rate shaping (DRS) [3] method optimally drops luminance data DCT coefficients to minimize distortion). The combination of scaling techniques that can be applied to a video object is not static. Rather, it can dynamically change with user preference and content type. For example, in the case of fast-motion scenes, spatial resolution or quality scaling techniques are more suitable than temporal-domain scaling techniques (e.g., dropping frames) because the details within a picture may not be as important under fast-motion conditions. In contrast, slow-motion scenes favor the opposite approach. In the system architecture, the scaling profile selector is co-located with a content analyzer algorithm that can gain access to video content features. In addition, the scaling profile selector provides a user preference API that allows end users to specify high-level rules that can be used to generate scaling profiles (e.g., a mobile PDA user may prefer high resolution to rich color). User preferences may be specified, for example, as dropping chrominance, dropping background objects and reducing the frame rate of foreground objects. Scaling profiles accurately capture the scalability of video streams at two distinct levels: (i) the scaling pattern of single video objects (i.e., combination of scaling techniques applied to a single object); and (ii) the aggregation of multiple prioritized video objects associated with the same video stream.

A. Single Video Objects

In our current implementation, the scaling pattern of a single video object is specified as a sequence of scaling actions. As illustrated in Figure 3, the derived utility function results in a concatenation of curves with discrete steps and piecewise-linear shapes. Each discontinuity point (u_i , R_i) on the utility function is associated with one *set* of scaling techniques S_i, that is used to scale down video to rate R_i with the corresponding utility value u_i. The scaling pattern is the set of S_i , denoted by $\{S_i\}$, structured as an array containing pointers to the corresponding scaling actions S_i. The scaling action pointers represent uniform resource locators (URL) pointing to the method implementations in media scaling toolkits located at servers. The content scaler uses the URL information to download scaling toolkits [18] at service creation time if they are not already resident at the content scaler.

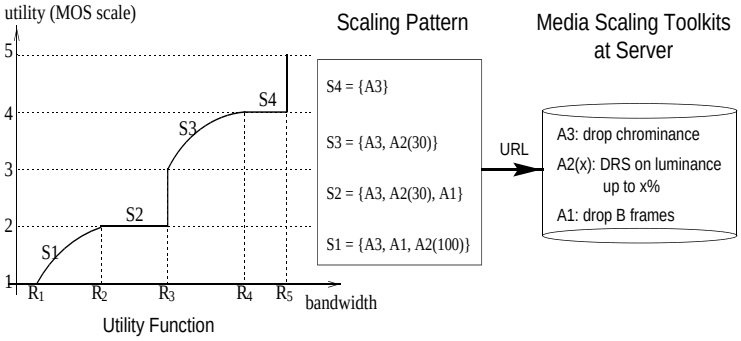


Figure 3: Scaling Pattern of Single Video Object

Figure 3 illustrates an example of a scaling pattern in relation to a utility function. The scaling actions are A_3 (drop chrominance); $A_2(30)$ (drop up to 30% of the DCT coefficients using DRS); A_1 (dropping B frames); and $A_2(100)$ (drop up to 100% of DCT coefficients using DRS). The utility function comprises a concatenation of four parts each corresponding to a different scaling patterns. For example, between utility values 5 and 4 the utility function is a step function as a result of the scaling pattern S_4, which consists of one scaling action A_3, that is dropping chrominance. Between utility values 4 and 3, the utility function has a continuous concave shape as a result of the

scaling pattern S_3, which comprises two scaling actions: drop chrominance (A_3) and drop up to 30% of DCT coefficients using DRS $(A_2(30))$.

B. Aggregated Objects

In MPEG-4, a number of elementary streams corresponding to different video objects can be multiplexed into the same network session at the FlexMux and TransMux layers [5]. Utility functions constructed for single video objects therefore need to be aggregated together into a form suitable for network-wide utility-based bandwidth allocation. By associating a priority with a video object we can define a scaling order for different video objects in the case of aggregation. A lower priority video object could be dropped before a higher priority object is scaled down. In this respect an aggregated utility curve will have a shape representing a concatenation of two utility curves for video objects with high and low priority objects. The resulting curve is normalized over a 5-level mean-opinion-score (MOS) scale [6]. Objects that have the same priority are scaled proportionally according to a *utility-fair* algorithm [9]. In this case, the aggregated utility curve is calculated based on the utility-fair algorithm.

Utility functions and scaling profiles are dynamically created by the utility generator and dispatched to the content scaler as illustrated in Figure 2. The content scaler forwards utility functions to the bandwidth allocator to make resource reservations. Since the generation of utility curves and scaling profiles can occur frequently, an efficient signaling scheme is required to ensure the timely delivery of scaling information to distributed algorithms (e.g., the content scaler). We use the MPEG-4 Object Extension Descriptor component [5] for signaling this information across the network. We define a Scaling Extension Descriptor that contains (i) a utility function for each elementary stream; (ii) an aggregated utility function; and (iii) a scaling profile comprising the scaling pattern and the aggregation priority associated with all active video objects. The utility function is represented by a vector of discontinuity points (u_i , R_i). The scaling pattern takes the form of an array of URLs that point to the location of a specific media scaling implementation. The structure of the Scaling Extension Descriptor is similar to the Elementary Stream Descriptor that carries URLs and stream priority [5].

IV. The Content-based Utility Function Estimator

The dynamic generation of bandwidth utility functions requires large amount of processing power. Given the current state-of-the-art, generation of utility functions on a frame-by-frame basis is difficult to achieve in real-time [10]. In what follows, we describe a technique for content-based utility function estimation that accelerates the generation of utility functions.

A. Model

The proposed technique does not rely on the explicit generation of utility functions for each video object. Rather, it uses video content and machine learning techniques to determine the utility class of an object. Because video content can be dynamically extracted from compressed video streams, this technique is suitable for real-time applications. Note that we use the term "video object" (VO) to refer to either a video frame in the case of frame-based encoding (e.g., MPEG-1, MPEG-2, H.263) or a video object plane (VOP) as in the case of MPEG-4 [5]. Figure 4 illustrates the architecture of a content-based utility function estimator [2] that can support a variety of compression schemes. The sub-system architecture comprises two main components: *an adaptive content classification loop* and a *real-time estimation path*. The adaptive content classification loop comprises the per-frame utility generator, training pool, utility clustering module and decision-tree generator. The real-time estimation path comprises content analyzer and utility selector. The adaptive content classification loop (also referred to as an adaptation loop) and the real-time estimation path operate asynchronously. Based on content features extracted online by the content analyzer, the utility selector dynamically determines the *utility class* and the corresponding *characteristic utility function* for each video object. An adaptation loop is activated to periodically re-compute

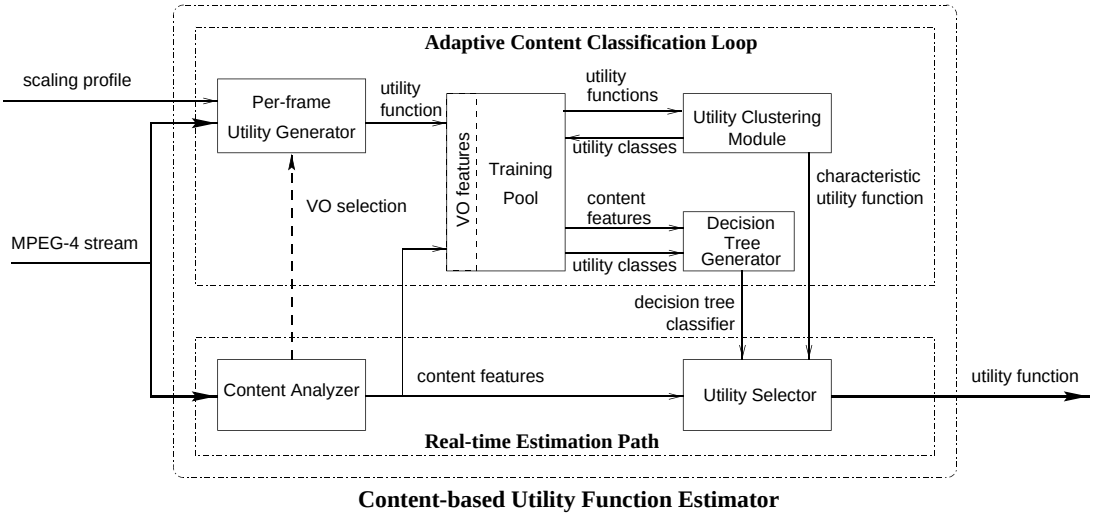


Figure 4: Architecture of Content-based Utility Estimator

decision tree parameters used by the utility selector. Explicit computation of the utility function is performed at this point for the selected video objects. In this manner, the architectural model decouples the adaptation loop processing that is computationally intensive from the real-time estimation path. The system operate in real-time by avoiding explicit per-object generation of utility functions.

B. Real-time Estimation Path

The real-time estimation path illustrated in Figure 4 supports the estimation of utility functions on a frame-by-frame basis where estimation is based on the content features extracted by the content analyzer. Incoming MPEG-4 video streams are demultiplexed and individual video object streams dynamically extracted by the content analyzer. The analyzer processes individual video object streams extracting video content information in real-time [11]. Content information comprises visual features and encoder-specific features. Visual features describe video object characteristics (e.g., video object size, speed, etc.) that do not change if an alternative encoding technique is applied. In contrast, encoder-related features are sensitive to specific encoding technique and encoder parameters (e.g., frame type, DCT values, number of bits for various encoder-specific stream components, etc.). Most content features can be extracted directly from compressed video streams (e.g., object size, picture type). However, some encoder-related features can be only obtained from the encoder itself (e.g., PSNR). Therefore, the extraction of content features is somewhat dependent on the location of the content analyzer in the system (e.g., at the video server or base station). When content analyzer is not co-located with at the video server additional information not readily available in the network can be carried in the extension fields of the MPEG-4 Object Descriptor. In this case, the content analyzer will be able to extract this information directly from the stream's Object Descriptor and can be placed at any point in the system.

Accurate estimation of video object features is complex and requires substantial processing power. It typically requires evaluation of the original video sequence in the spatial domain. Typically, high estimation accuracy of features is not crucial for networking applications discussed in this paper. Content features can be estimated with a good level of accuracy directly from compressed video streams minimizing the need for computationally intensive resources needed to power such algorithms. In addition, the content analyzer selects some video objects for further evaluation in the adaptation loop. The selected video objects can be (i) directly evaluated to speed the future computing of utility functions and (ii) stored with their associated content feature information in the training pool. The algorithm that selects which video object to use during the adaptation loop is out of the scope of this paper. However, the video object indicating "substantial change" in its visual content will be likely selected.

Assuming that the system has been initialized and the adaptation loop has estimated the decision tree parameters [2] (discussed later) then the real-time estimation of utility functions is realized in the following manner. First, content features are extracted in real-time from the compressed video stream by the content analyzer. Second, the utility selector uses these features to determine the utility class of the current video object. This selection is based on the decision tree [2] that is periodically updated during the adaptation loop. Finally, once a utility class has been determined, a characteristic utility function for that class (which is determined initially during the training period) is selected for the video object. The long-term prediction of utility functions is realized using a long-range utility function predictor discussed in Section V.

C. Adaptive Content Classification Loop

The operation of the adaptation loop is computationally intensive and difficult to apply on a per-video object basis. Therefore, the adaptation loop is only invoked intermittently and is functionally independent of the real-time estimation path. The adaptation loop uses a subset of the previously evaluated video objects for the computation of the decision tree. The set of previously selected and evaluated video objects is stored in the training pool. The selection of these objects is based on heuristics. Video objects are selected periodically or when their content features have substantially changed. For example, video objects that are selected from different scenes tend to have content features substantially different from each other. The selection of video objects included in the training pool is managed by the content analyzer. The selected video object is evaluated and its utility function and content features placed in the training pool. The per-frame utility function generator estimates bandwidth utility functions based on the encoding method and scaling profile used by each video object. Object selection supports smooth adaptability to the various video content. Smooth adaptation is assured by continuously replacing and re-clustering new video objects in the training pool [2] as they appear.

A utility clustering module performs automated clustering of objects using unsupervised classification algorithms [4] operating on selected features of the utility function [2]. The module clusters utility functions of different video objects stored in the training pool into a small set of utility classes. When clustering is complete, all the video objects in training pool are marked according to the class they belong to. The decision-tree generation algorithm then uses the marked video object's to form the decision tree used by the utility selector. The characteristic utility function is estimated from utility functions within the same utility class and is used as a utility function estimator for its class. The decision-tree generator starts its operation after re-clustering is complete. The operation is based on machine learning techniques; in particular, supervised classification algorithms [12]. The decision-tree generator determines the decision tree by using utility classes derived by the utility clustering module and content features are extracted by the content analyzer. Note that decision-tree generator does not use parameters describing utility functions. Once a decision tree is formed, the utility class of a particular video object can be determined by using the

video object's content feature. This operation is performed by the utility selector, which uses the decision tree to estimate the utility function.

V. The Long-range Utility Function Predictor

Utility functions are not generated once and expected to remain valid over the lifetime of a video stream. Rather, they are time varying due to their sensitivity to changes in the stream. Typically, network adaptation operates over much longer time-scale, potentially in the order of hundreds of milliseconds to tens of seconds and is conditioned by the signaling/control system capacity, traffic load and the round trip delay between a source encoder and receiving decoder. There is a need to reconcile the mismatch between these two distinct time-scales. We propose to push the utility function generation interval as close to the network adaptation time-scale thereby attempting to keep the generated utility function "accurate" over longer time-scales. The long-range utility function predictor is designed to address this challenge. The predictor operates at the last stage of the utility function generator process as illustrated in Figure 2. The predictor applies an adaptive filtering algorithm [10] based on instantaneously generated utility curves provided by the utility estimator as discussed in the previous section. The adaptive filtering algorithm dynamically adjusts the degree of relaxation for the utility function (i.e., it over-estimates a stream's bandwidth requirement thereby reducing the need for updating existing utility functions). A drawback of this approach is that over-estimation of a utility function can lead to over-allocation of bandwidth which should be avoided.

In what follows, we outline the operation of the prediction algorithm. As utility functions are constructed from sample points, a utility curve is represented by its corresponding sample rate vector R. The prediction algorithm uses an *expanding factor* $\boldsymbol{e}$ that "inflates" the bandwidth demand hedging that a predicted utility function will remain at the upper bound of the bandwidth demand of its instantaneous utility function over the next *utility duration T*. The prediction algorithm aims to dynamically adjust this expanding factor by closely tracking the dynamics of instantaneously generated utility functions. The prediction algorithm operates in normal and exception modes. In the normal mode, the prediction algorithm generates one predicted utility function every *T* interval. In this case, the duration *T* is defined as the utility duration and should ideally be in the order of the network adaptation time-scale. During this interval, the prediction algorithm continuously uses instantaneous utility functions $\boldsymbol{R}^k$ to update its internal measurement $\boldsymbol{R}^{avg}$, following a moving average formula: $\boldsymbol{R}^{avg} = \alpha * \boldsymbol{R}^{avg} + (1-\alpha)\, \boldsymbol{R}^k$, where α is a control parameter. When the *T* interval expires, the algorithm multiplies the $\boldsymbol{R}^{avg}$ with an expanding factor *e* to generate a new predicted utility function for the next *T* interval. The expanding factor *e* is decreased by a small decrement e^{dec} until *e* reaches 1.

In the exception mode, the prediction algorithm manages violations. A violation is defined when the currently predicted utility function has to be changed to accommodate a newly generated instantaneous utility function that exceeds the bandwidth demand of the currently predicted utility function. When violations occur the utility predictor moves to exception mode processing. The algorithm then increases $\boldsymbol{R}^{avg}$ corresponding to the utility curve that registered the violation. The expanding factor is increased by an increment e^{inc}. The prediction algorithm then generates a new predicted utility function using the updated values and interacts with the bandwidth allocator (as illustrated in Figure 1) to re-negotiate bandwidth on an end-to-end basis. The pseudo-code for the prediction algorithm can be found in [10].

The network adaptation time-scale may also change over time. For example, if the network adaptation time-scale is coupled to the round trip delay for networks with time-varying end-to-end delay. Therefore the utility duration *T* needs to be programmable. The utility predictor provides a utility generation trigger interface for this purpose as illustrated in Figure 2. This interface activates the asynchronous generation of instantaneous utility functions on-demand complementing the periodic generation based on the content adaptation time-scale. This interface can be programmed by the network to force a refresh of the current utility functions.

VI. Evaluation

In this section, we present experimental results from the implementation of the proposed framework focusing on the performance of the content-based utility estimator and long-range utility predictor components of the framework. The experimental results demonstrate the viability of our approach. The integration of these mechanisms into the MPEG-4 transport and content scaler is for future work.

A. Experiment Setup

In comparison to continuous-rate scaling methods, discrete-rate scaling methods are straightforward to model (e.g., by measuring the rate of the dropped components and the associated drop in utility value based on the scaling profile). In the following experiments we are primarily concerned with how to stabilize utility functions and without loss of generality we focus on continuous-rate scaling methods (e.g., dropping DCT coefficients). We have selected the dynamic rate shaping algorithm as a means of implementing the dropping of DCT coefficients because it minimizes distortion under any given rate. For evaluation purposes, the scaling profile is statically configured during experimentation to contain only dynamic rate shaping scaling methods. In this case, the profile selector simply defines the maximum (R_{max}) and minimum (R_{min}) scalable rate. The selector generates 50 scaling rate samples that are evenly spaced in the range (R_{min} , R_{max}). The per-frame utility generator is implemented using the dynamic rate shaping source code [3]. For simplicity, the utility metric is defined based on the Signal-to-Noise Ratio (SNR)[1]. The

[1] The bandwidth utility function u(r) is defined as: $u(r) = 1 - err^2(r) / sig^2$, where $sig^2 = \Sigma_{i=1,N} X_i^2 / N$ and $err^2(r) = \Sigma_{i=1,N} (X_i - Y_i(r))^2 / N$. The sig^2 component is the mean energy each pixel has in the original picture

investigation of more sophisticated objective utility metrics incorporating human vision systems is the subject of on-going research.

B. *Content-based Utility Estimation Experiment*

In what follows, we discuss the implementation issues associated with the content-based utility function estimator. In our experiments, we have used an MPEG-2 video stream consisting of 3000 frames created from the movie "Forrest Gump" using the Columbia University MPEG-2 software encoder. Our classification experiment is based on a subset of this trace, namely 734 frames consisting of P-frames only.

1) *Implementation*

We have simulated the operation of all the functional modules within the content-based utility estimator as illustrated in Figure 4. In our simulation experiments, one set of video objects was used for training the utility clustering module and the generation of a decision-tree. A second set of video objects was used to obtain the classification results used by the utility selector. We randomly select half of the video objects and placed them in the training pool thus bypassing the video object selection algorithm discussed in Section V. We have implemented an MPEG-2 content analyzer that extracts content features in the compressed domain. The content analyzer was implemented using the Columbia University MPEG-2 software decoder and broadly operates as follows. First the content analyzer conducts video scene detection then video object detection and finally, content feature extraction. Because the analyzer operates in the compressed domain, the original MPEG-2 decoder was simplified to contain only parts necessary for scene detection, video object detection and visual feature estimation. In particular, the computationally intensive inverse DCT function was fully omitted. This simplification results in real-time performance on a general-purpose workstation. For example, on SUN SPARCstation 5 it is possible to analyze the content of each video frame in less than 10 ms (i.e., before the next frame needs to be processed).

The following content features are extracted directly from the encoded MPEG-2 video stream: current frame size, number of objects, object size (in macroblocks), average and variance of motion vectors, number of forward-predicted macroblocks, number of DCT-encoded macroblocks, camera operation parameters (viz. translation, zoom and divergence speed), and average energy of AC DCT coefficients. We have found that classification accuracy can be greatly increased by including the PSNR in the content information where the PSNR is obtained directly from the encoder.

Functions associated with the utility clustering module and decision-tree generator were simulated using publicly available machine learning tools. In particular, the utility clustering module was realized using the Autoclass III [4] and decision-tree generator was based on the OC1 software [12]. Autoclass III is Bayesian unsupervised classifier that predicts class membership given unlabeled test cases. Autoclass III was configured to automatically select a fixed number of 17 classes used during classification. OC1 is a supervised machine learning system based on oblique decision trees. Decision trees of this form consists of linear combination of the attributes at each internal node and can be viewed as simply a more general form of axis-parallel univariate decision trees. Detailed description of these tools can be found in [4] [12].

2) *Results*

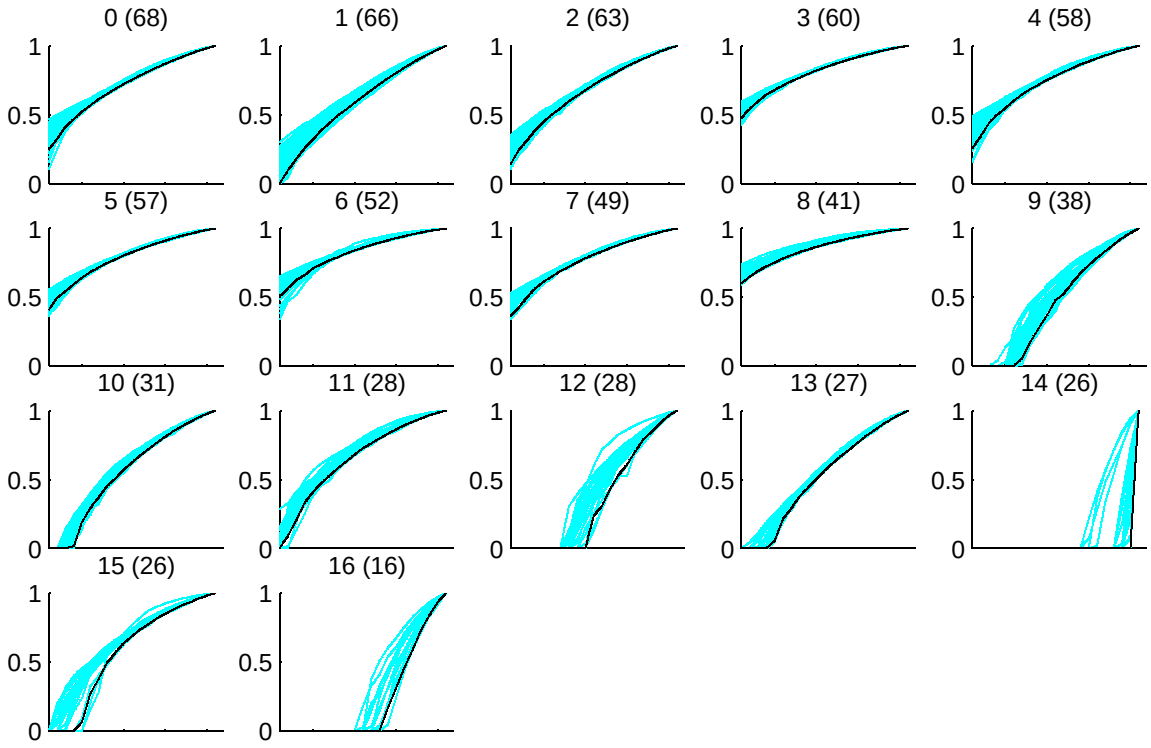
Each per-frame utility function was constructed from 21 rate samples. These samples were evenly distributed in the range 0% to 100% of the original stream bit-rate. Individual sampling points were obtained using the dynamic rate shaping (DRS) system [3]. Utility functions were then processed by the utility clustering module. Figure 5 illustrates one snapshot of the classification results. In this example, 17 classes are formed out of total 734 per-frame utility functions. Each sub-graph illustrates all the individual utility functions within a single class. These are shown as shaded curves in Figure 5. The single dark curve represents the characteristic utility function of the class, which is obtained, in this case, as 10 percentile of the utility curves belonging to each class.

The classes are numbered from 0 to 16. The number in parenthesis indicates the number of utility functions in each class. In this case, the number of utility functions in a single class ranges from 16 to 67. The figure shows the good classification performance of the Autoclass III software, as the utility functions of similar shapes are clustered into the same utility class.

After utility classification is complete, the decision tree generator forms the decision tree from the extracted content features serving as feature points in supervised classification. The decision tree represents a hierarchical mapping between content feature vectors and utility classes. Figure 6 illustrates the classification results based on decision tree using the same presentation as Figure 5. The decision tree accuracy is crucial to the real-time performance of the system. The decision tree enables estimation of the utility function given a set of content features that are easy to obtain from compressed video streams. However, since content features do not contain direct information regarding utility functions, mismatches between content features and utility classes can occur. In other words, at some instances, a decision tree is not able to correctly identify a utility class based on analyzed content features alone. This effect can be observed in Figure 6. For example, comparing classes 9 in Figures 5 and 6, one can find several utility curves that are incorrectly classified into class 9 by decision tree. In practice, this will lead to the utility selector selecting the wrong characteristic utility function. Table 1 summarizes the accuracy of the decision tree for each of the 17 utility classes obtained during the simulation experiments. The overall classification accuracy of the whole set of utility functions was found to be 91 %;

---

and err^2 the mean square error between the distorted image and the original image. N represents the number of pixels in the picture, X_i and Y_i are the ith DCT coefficients in the original and distorted images, respectively. The metric is then normalized into the range of 1 to 5 to correspond to the MOS 5-level quality index.



that is to say, among the total 734 video objects, 91 % of them were classified correctly using content features. This high level of classification accuracy demonstrates the viability of the approach.

C. *Long-range Utility Predictor Experiment*

In what follows, we present the evaluation of the utility predictor that keeps generated utility functions meaningful over longer time-scales. The experiments used two MPEG-2 video traces notably the Chef (1 minute TV interview) and TrueLies (3.5 minutes action movie excerpt) video clips, where the Chef video sequence has relatively slow scene changes and TrueLies relatively fast changes. The prediction algorithm is designed to operate over the quantized instantaneous utility functions. We quantize the instantaneous utility functions easing the processing and reducing the number of states required since quantization reduces the number of rate samples representing a utility function.

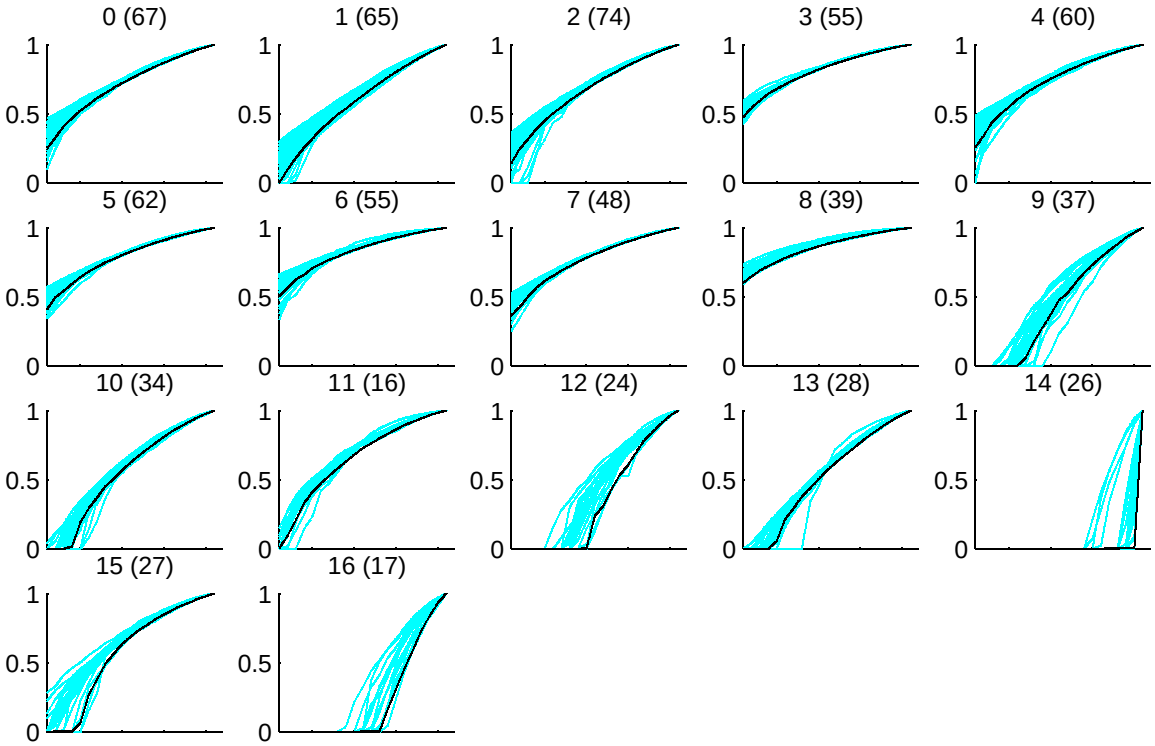
In this experiment, we use the 5-level MOS quantization scale [6]. Utility levels 0, 1, . . ., 4 are mapped into discrete utility values 0.1, 0.3, 0.5, 0.7, and 0.9, respectively. Let r_k denote the rate value corresponding to the *kth* utility level. r_k is calculated via linear interpolation to locate the rate that causes *u(R)* to cross the *kth* discrete utility value. The resulting 5 sample points (r_0 , 0), (r_1 , 1), (r_2 , 2), (r_3 , 3), and (r_4 , 4), together with two end points (R_{min} , 0) and (R_{max} , 4), support the construction of a piecewise-linear utility function whose utility range is [0, 4].

1) *Adaptive Prediction Algorithm*

We apply the prediction algorithm to both traces, with an α = 0.8 and an initial expanding factor e_0 = 1.2. In addition, we set e^{inc} = 0.1 and e^{dec} = 0.01 so that the expanding factor slowly decreases when there is no violation and quickly increased when violations occur. Figure 7 illustrates the generated utility functions for T = 30 seconds. Once generated, each curve is used to represent the stream's adaptation capability to bandwidth variation for the next 30 seconds. In Figure 7(a) and 7(b), we observe that half of the curves do not have shapes like step functions at the maximum scalable rate. Rather, they span the range of scalable rates. This demonstrates that the derived utility functions track the scalability of the video streams rather well. The number of step shaped utility functions in Figure 7(b) result from a single violation at time equal to 19 seconds. When an increment is added to the expanding factor, the next three curves generated follow step functions simply because of the large expanding factor. However, the prediction algorithm corrects its operation by continuously reducing the expanding factor until no violations are observed during the utility duration. Observe that at time 141 seconds into the operation, the generated curve takes on

Table 1: Decision Tree Accuracy

Class	1	2	3	4	5	6	7	8	9
Accuracy	100%	100%	93.88%	90.00%	93.10%	100%	100%	98.08%	92.65%
Class	10	11	12	13	14	15	16	17	
Accuracy	93.65%	92.59%	92.42%	90.32%	81.58%	53.57%	76.92%	78.57%	



a non-step function once again.

The total number of violations observed during the experiment is shown in Table 2. The data was collected for experiments with *e* set at 1.1 to 1.3 with a timeout interval *T* ranging from 10 to 50 seconds. The algorithm generally performs well because the number of violations remains small. This is due to adaptive nature of the prediction algorithm, which dynamically increase or decrease the expanding factor according to the degree of "relaxation" on measurement and the occurrence of violations. Since the observed number of violations does not change significantly when T varies, it indicates that the violations may occur in bursts, which could be caused by a sequence of fast scene changes. In addition, we observe that the number of violations do not increase as T increases. One reason for this is that when the timeout interval is large, the chance of reducing the expanding factor is smaller because the expanding factor is only adjusted after a timeout has occurred. This result implies that the algorithm will perform as well for large and small T.

2) Prediction Error Analysis

To quantitatively analyze the measurement error introduced by the prediction algorithm, we define two error metrics that measure the maximum distance between a predicted utility function $u^*(R)$ and an instantaneous utility function $u_j(R)$ generated over the subsequent utility duration for a predicted curve: (i) the over-estimation error err^+, which tracks the maximum amount of bandwidth over-estimation between a predicted utility function and an instantaneous curve; and (ii) the under-estimation error err^-, which captures the maximum amount of bandwidth under-estimation. In mathematical definition, $u^*(R)$'s error of over-estimation on $u_j(R)$ is given by

$$err^+ = \max_{i = 0, \ldots, 4} \{ u^{*-1}(i) - u_j^{-1}(i) , 0 \} / R_{max}$$

and $u^*(R)$'s error of under-estimation on $u_j(R)$ is given by

$$err^- = \min_{i = 0, \ldots, 4} \{ u^{*-1}(i) - u_j^{-1}(i) , 0 \} / R_{max}$$

Note that $u^{*-1}(.)$ denotes the inverse function of $u^*(.)$. Both metrics are defined as the percentage of over or under allocation relative to the maximum rate R_{max}. In Figure 8, we illustrate the measured errors after applying our prediction algorithm to the TrueLies video clip, with an initial value of 1.2 for the adaptive expanding factor. The utility duration *T* is set to 10 seconds in Figure 8(a) and 50 seconds in 8(b). The top curve in both figures represents the over-estimation error that oscillates frequently within the range of 5% and 20% over-estimation. The 20% over-estimation results from an expanding factor of 1.2, which essentially inflates the rate estimation by 20%. The under-

Table 2: Number of Violations in Adaptive Prediction Algorithm

	Chef Clip					TrueLies Clip				
	T=10	T=20	T=30	T=40	T=50	T=10	T=20	T=30	T=40	T=50
e=1.1	4	2	2	2	2	4	5	5	5	5
e=1.2	2	3	0	2	0	2	1	1	1	1
e=1.3	0	0	0	0	0	1	0	0	0	0

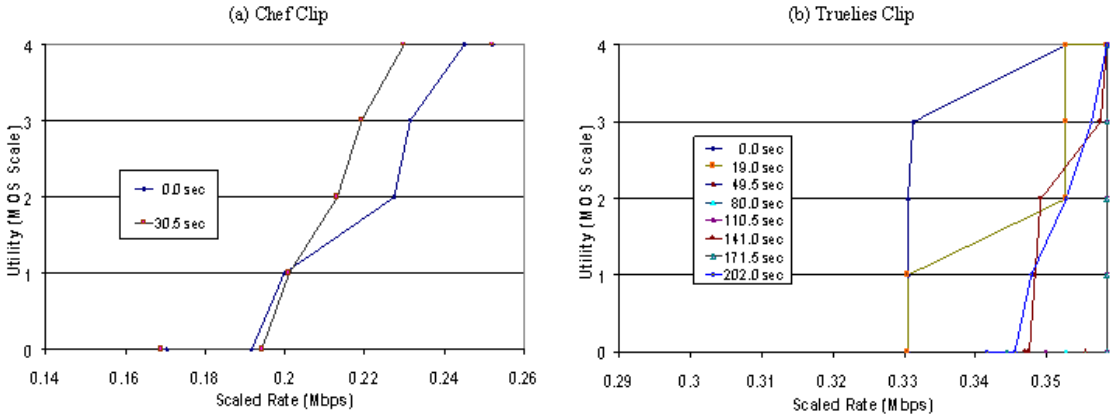


Figure 7: Predicted Utility Function with e_0 =1.2

estimation error mostly remains zero and generates a negative spike of less than -3% only when there are violations.

We performed utility prediction offline to determine the best possible performance achievable. The offline utility predictor stores all the instantaneous utility functions generated during a target utility duration to accurately derive the predicted utility function. The resulting system becomes noncausal as it uses posteriori data to generate predicted utility functions. Figure 9 illustrates the best estimation error achieved by the offline algorithm for the TrueLies video clip with a utility duration of 50 seconds. The under-estimation error is constantly zero. The over-estimation error has a similar shape to Figure 8 but is shifted to the range of 0% and 15%. In this case the over-estimation error is predominantly caused by the intrinsic scene changes of the video content and not by the utility generation procedure.

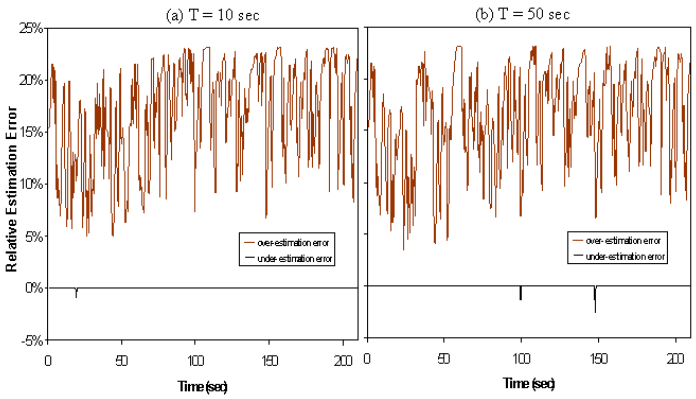


Figure 8: Utility Estimation Error (e_0 =1.2, TrueLies Clip)

After comparing the over-estimation errors in both Figure 9 and 8, we observe that: (i) the two over-estimation curves in Figure 8 are of similar shape indicating that the proposed prediction algorithm is not sensitive to the length of the utility duration; and (ii) the over-estimation error in Figure 9 is of the similar shape to Figure 8 but is smaller by an amount of 5%. This 5% is caused by the difference between the 20% expanding factor and the 15% peak error shown in Figure 9. We argue that the proposed prediction algorithm performs well in tracking the intrinsic scalability of video streams. The extra degree of over-estimation (e.g., 5% in Figure 8) is necessary for a causal system and represents a tradeoff between the tightness of utility estimation and stability of utility duration.

VII. Conclusion

In this paper, we have proposed a utility-based network adaptation framework that enables content-aware adaptive MPEG-4 video delivery over time-varying networks. The design of the scaling profile exploits MPEG-4 object-level scalability and is closely coupled with the media scaling techniques employed by network adaptation mechanisms. The utility generator employs a video content classification algorithm to speedup processing and an adaptive prediction algorithm to keep dynamically generated utility functions meaningful over network adaptation time-scales. Our experimental results demonstrate that the proposed framework represents a viable approach to delivering scalable MPEG-4 media over time-varying networks.

For future work, we plan to complete the integration of the utility generation, content scaling and bandwidth renegotiation mechanisms into the MPEG-4 system and its DMIF transport. Once we are satisfied with the broader performance of the system we plan to port it to our programmable mobile networking environment [17] to manage scaling of media from the base-to-mobile. Also, we plan to investigate the introduction of more sophisticated utility metrics that consider human vision systems capturing perceptual visual.

Acknowledgements

Andrew T. Campbell would like to thank the National Science Foundation (under CAREER Award ANI-9876299) and the Intel Corporation for supporting this research in part. In addition, we would like to thank our colleagues Stephen Jacobs and Alexandros Eleftheriadis, Columbia University, for providing the dynamic rate shaping source code and for taking time to answer our numerous questions.

REFERENCES

[1] P. Bocheck and S.-F. Chang, "Content Based Dynamic Resource Allocation for VBR Video in Bandwidth Limited Networks", *Proc. of the IEEE/IFIP International Workshop on Quality of Service (IWQoS'98),* Napa, California, May 18-20, 1998.

[2] P. Bocheck, Y. Nakajima, and S.-F. Chang, "Real-time Prediction of Subjective utility functions for MPEG-4 Video Objects", *Proc. of PacketVideo'99,* New York City, Apr. 26-27, 1999.

[3] A. Eleftheriadis and D. Anastassiou, "Dynamic Rate Shaping of Compressed Digital Video", *Proc. 2nd IEEE Intl. Conf. on Image Processing*, Arlington, VA, Oct. 1995.

[4] R. Hanson, J. Stutz, and P. Cheeseman, "Bayesian Classification Theory", NASA Technical Report FIA-90-12-7-01, 1990.

[5] ISO/IEC 14496-1, "Information Technology - Coding of Audio-visual Objects, Part 1: Systems", ISO/IEC JT1/SC 29/WG 11 Draft International Standard, Dec. 1998.

[6] ITU-R BT.500-7, "Methodology for the Subjective Assessment of the Quality of Television Pictures", ITU-R Recommendations, Geneva, Oct. 1995.

[7] C. Lambrecht and O. Verscheure, "Perceptual Quality Measure Using a Spatio-Temporal Model of the Human Visual System", *Proc. of IS&T/SPIE*, San Jose, Feb. 1996.

[8] K. Lee, "Adaptive Network Support for Mobile Multimedia", *Proc. of ACM Mobicom'95*, Berkeley, CA, Nov. 1995.

[9] R. Liao and A. Campbell, "On Programmable Universal Mobile Channels", *Proc. of ACM Mobicom'98*, Dallas, TX, Oct. 1998.

[10] R. Liao, P. Bouklee, and A. Campbell, " Online Generation of Bandwidth Utility Function for Digital Video", *Proc. of PacketVideo'99*, New York City, Apr. 26-27, 1999.

[11] J. Meng and S.-F. Chang, "Tools for Compressed-Domain Video Indexing and Editing", *Proc. of SPIE Conference on Storage and Retrieval for Image and Video Database*, Vol. 2670, San Jose, February 1996.

[12] S. K. Murthy, S. Kasif, and S. Salzberg, "A System for Induction of Oblique Decision Trees", *Journal of Artificial Intelligence Research*, 1994

[13] D. Reininger, R. Izmailov, "Soft Quality of Service with VBR + Video", *Proc. of Intl. Workshop on Audio-Visual Services over Packet Networks (AVSPN'97)*, Aberdeen, Scotland, UK, Sept. 15-16, 1997, pp. 207-211.

[14] S. Schenker, "Some Fundamental Design Decisions for the Future Internet", *IEEE Journal on Selected Areas in Communications*, Vol. 13, pp. 1176-1188, 1995.

[15] A. Webster, C. Jones, M. Pison, S. Voran, and S. Wolf, "An Objective Video Quality Assessment System Based on Human Perception", *SPIE Human Vision, Visual Processing, and Digital Display IV*, Vol. 1913, pp. 15-26, San Jose, CA, February 1993.

[16] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, "Filters: QOS Support Mechanisms for Multipeer Communications", *IEEE Journal on Selected Areas in Communications,* Special Issue on Distributed Multimedia Systems and Technology, Vol. 14, No. 7, Sept. 1996, pp 1245-1262.

[17] A. T. Campbell, Kounavis M.E. and R. R.-F. Liao, "Programmable Mobile Networks", *Computer Networks and ISDN Systems*, April 1999.

[18] A. Balachandran, Campbell, A.T., Kounavis, M.E, "Active Filters: Delivering Scalable Media to Mobile Devices", *Proc. Seventh International Workshop on Network and Operating System Support for Digital Audio and Video*, St Louis, May 1997.