PACKET SCHEDULING OF STREAMING VIDEO WITH FLEXIBLE REFERENCE FRAME USING DYNAMIC PROGRAMMING AND INTEGER ROUNDING

Gene Cheung, Wai-tian Tan

Hewlett-Packard Laboratories

ABSTRACT

Video coding standards like H.264 offer the flexibility to select reference frames during motion estimation for predicted frames. We investigate the packet scheduling problem of streaming video over lossy networks from a real-time encoder with flexible reference frame. In particular, we consider a multi-path streaming setting where each predicted frame of video, in addition to the flexibility to select a reference frame, can schedule one or multiple transmissions on one or multiple delivery paths for the upcoming optimization period. We present an algorithm based on dynamic programming that provides a locally optimal solution with high complexity. We then present a rounding method to reduce computation complexity at the expense of degrading solution quality. Results show that our algorithm performs noticeably better than a greedy scheme, and graceful tradeoff between complexity and solution quality can be achieved.

1. INTRODUCTION

Advances in video coding and networking have created new flexibilities in the design of streaming algorithms. An example is reference frame selection (RFS) of video coding standards like H.264 [1], where each coding block within a predicted frame can choose among several previously encoded frames for motion prediction. This allows a live encoder to avoid using lost frames as references, thereby controlling error propagation. Another example is multi-homing, where a client is equipped with multiple network interfaces, such as 802.11b and WCDMA. Using both interfaces has obvious advantages including higher throughput and less variability.

While it is clear that streaming can be enhanced by exploiting these flexibilities, high complexity is required to jointly select optimal parameters for options afforded by the multiple flexibilities. Specifically, based on feedbacks, at a given optimization instance a live encoder must choose reference frames for predicted frames and transmit frames one or multiple times using one or multiple interfaces. To make such selection, we investigate an algorithm based on dynamic programming that produces a locally optimal solution with high complexity. We then apply an integer rounding technique to the algorithm to reduce the complexity at the expense of degrading solution quality. Results show that our proposed algorithm performs noticeably better than a greedy algorithm derived from the simplified version of RaDiO [2].

The paper is organized as follows. After discussing related work in Section 2, we present our source and network models in Section 3. We present our optimization algorithm in Section 4. In Section 5, we discuss an integer rounding-based procedure to reduce the algorithm complexity at the cost of solution quality. Results and conclusion are presented in Section 6 and 7, respectively.

2. PREVIOUS WORK

Video coding standard H.264 [1] has superior coding performance over previous standards like MPEG-4 and H.263 over a broad range of bit rates. Part of H.264 definition is the flexibility of using any arbitrary frame to perform motion-estimation, originally introduced as Annex N in H.263+ and later as Annex U in H.263++. Early work on enhancing streaming using RFS includes [3, 4]. Our work differs from these works by employing a complexity-scalable optimization procedure and also applying optimization to jointly perform reference frame (RF) and transmission path (TP) selection.

Recently, [5] proposes the reorganization of the prediction structure of a group of pictures (GOP) to minimize the effects of losing a single P-frame. We differ from [5] in two regards: i) we maximize the *expected* performance — the average case, while [5] minimizes the worst case; and, ii) we adapt our scheme according to observed network conditions.

Our recent work [6] has shown that dynamic programming plus integer rounding can be applied to the joint selection of reference frame, QoS and delivery path in a multi-path environment without feedbacks. This paper is a generalization in three important regards: i) we permit retransmissions of lost packets given available client feedbacks; ii) GOP structure is generalized so that any frame can be real-time encoded as an I-frame or a P-frame; and, iii) transmission delay in each path is modeled as a shifted-Gamma-distributed random variable. With these enhancements, we obtain a more general framework for packet scheduling over multiple paths with feedbacks and flexible reference frames.

3. ASSUMPTIONS AND PROBLEM FORMULATION

In our intended application scenario, a sender is jointly optimizing encoding of video and its transport over two network paths via two network interfaces to the client. On the live-encoder side, we choose previously encoded frames to be used for references. On the transport side, we choose the number of transmissions for each encoded frame over the two

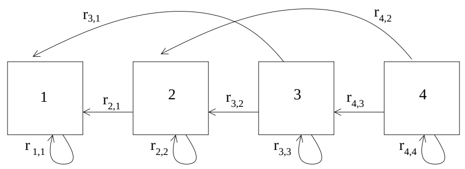


Fig. 1. Example of DAG Source Model

interfaces. We first present the source and network models, then discuss our objective function.

3.1. Directed Acyclic Graph Source Model

Our optimization is run periodically with period P, where during each optimization instance, a window of M consecutive frames of a $M_{\max}$-frame video sequence is under consideration for (re)transmission, and each frame in the window can be coded either as an intra-coded frame (I-frame) or an inter-coded frames (P-frames). For simplicity, we assume frames 1 to M are being optimized, though in reality they can be any M consecutive frames in the $M_{\max}$-frame sequence. Each frame i (F_i) must be delivered to the client by a playback deadline D_i or is rendered useless. At the next optimization instance, the window advances k frames in the sequence, where k is the number of leading frames with playback deadlines having expired at the client. Each F_i has a transmission history $\mathbf{h}_i$ which records the times and numbers of transmissions on each path the optimization had selected for F_i in previous optimization instances (more in Section 3.2).

We model the decoding dependencies of the M frames in the window using a directed acyclic graph (DAG) $G = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}, |\mathcal{V}| = M$, and edge set $\mathcal{E}$, similar to [2]. Specifically, each frame F_i, $i \in \{1, \dots, M\}$, represented by a node $i \in \mathcal{V}$, has a set of outgoing edges $e_{i,j} \in \mathcal{E}$ to nodes j's. F_i can use F_j as reference iff $\exists e_{i,j} \in \mathcal{E}$. We define $x_{i,j}$ to be the binary variable indicating whether F_i uses F_j as reference. Equivalently, given i, we define $x_{i,j}$ as:

$$x_{i,j} = \begin{cases} 1 & \text{if } F_i \text{ uses } F_j \text{ as RF} \qquad \forall j \in \mathcal{V} \mid e_{i,j} \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

H.264 syntax generally allows different coding blocks to use different reference frames. Instead, we restrict all blocks in a frame to use the same reference frame to reduce optimization complexity. We now have the following *RF constraint*:

$$\sum_{\forall j | e_{i,j} \in \mathcal{E}} x_{i,j} = 1 \qquad \forall i \in \mathcal{V} \mid i \neq 1 \tag{2}$$

We assume that only frames in the past are used for reference, i.e. $\forall e_{i,j} \in \mathcal{E}, i \geq j$. Further, we limit the number of candidate reference frames for any predicted frame F_i to be $E_{\max} \ll M$. An example of a DAG of a 4-frame sequence is shown in Figure 1 with $E_{\max} = 2$. We denote by $r_{i,j}$ the bit count of F_i when F_j is used as reference. We assume a sparse *rate matrix* $\mathbf{r}$ of size $\mathcal{O}(M^2)$ is computed *a priori* (to be discussed in Section 6) as input to the optimization algorithm (sparse because each row has at most $E_{\max}$ entries).

3.2. Network Model

We first assume that the network imposes a maximum transport unit of size MTU bytes, so that frame of size larger than MTU will be fragmented. For transmission of packet of size $\leq MTU$ bytes on path k, we assume a time-invariant packet erasure channel with random delay as in [2]. More specifically, let π_k be the packet erasure probability of path k, and $g_k(\gamma)$ be the shifted Gamma distribution with parameters α_k, λ_k and κ_k that describes the probability distribution of delay random variable γ_k of path k:

$$g_k(\gamma_k) = \frac{\lambda_k^{\alpha_k} (\gamma_k - \kappa_k)^{\alpha_k - 1} e^{-\lambda_k(\gamma_k - \kappa_k)}}{\Gamma(\alpha_k)} \qquad \kappa_k < \gamma_k < \infty \tag{3}$$

where $\Gamma(\alpha_k)$ is the *Gamma function*. This means that a packet sent on path k at time t_o will have probability of correct transmission by time T, $\delta_k(t)$, $t = T - t_o$, defined by:

$$\delta_k(t) = (1 - \pi_k) \int_{\kappa_k}^{t} g_k(\gamma) d\gamma \tag{4}$$

We allow each F_i to select two transmission levels (QoS) $q0_i, q1_i \in \mathcal{Q} = \{0, 1, \dots Q\}$ indicating the number of (re)transmissions until the next optimization instance on path 0 and 1, respectively. At a given optimization instance t_o, selection of QoS level $q0_i$ and $q1_i$ and frame size $r_{i,j}$, together with F_i's transmission history $\mathbf{h}_i$, will induce a *frame delivery success probability* $p(\mathbf{h}_i, q0_i, q1_i, r_{i,j}) \in \mathcal{R}$.

We derive $p(\mathbf{h}_i, q0_i, q1_i, r_{i,j})$ given our network model. We first define F_i's history $\mathbf{h}_i$ of length l_i as:

$$\begin{aligned} \mathbf{h}_i \;\; = \;\; & \{(f_i, q0_i^{(1)}, q1_i^{(1)}, t_i^{(1)}), (q0_i^{(2)}, q1_i^{(2)}, t_i^{(2)}), \dots, \\ & (q0_i^{(l_i)}, q1_i^{(l_i)}, t_i^{(l_i)})\} \end{aligned} \tag{5}$$

where the reference frame F_j selected for F_i is denoted by f_i, and QoS selections and transmission time of instance k are denoted by $q0_i^{(k)}$, $q0_i^{(k)}$ and $t_i^{(k)}$, respectively. Let $n_i = \left\lceil \frac{r_{i,f_i}}{8*MTU} \right\rceil$ be the number of packets required to encode F_i using reference frame f_i. The frame delivery failure probability of using QoS $q0_i$ at time t_o is then $\zeta_0(q0_i, n_i, D_i - t_0)$:

$$\zeta_0(q0_i, n_i, D_i - t_0) = (1 - \delta_0(D_i - t_0)^{n_i})^{q0_i} \tag{6}$$

If F_i is not ACKed, $p(\mathbf{h}_i, q0_i, q1_i, r_{i,j})$ can be written as:

$$\begin{aligned} p(\mathbf{h}_i, q0_i, q1_i, r_{i,j}) = 1 - \zeta_0(q0_i, n_i, D_i - t_o)\zeta_1(q1_i, n_i, D_i - t_o) \\ \prod_{k=1}^{l_i} \zeta_0(q0_i^{(k)}, n_i, D_i - t_i^{(k)})\zeta_1(q1_i^{(k)}, n_i, D_i - t_i^{(k)}) \end{aligned} \tag{7}$$

$p(\mathbf{h}_i, q0_i, q1_i, r_{i,j}) = 1$ if F_i is ACKed.

3.2.1. Network Resource Constraint

We impose network constraints on the amount of resource we can use per optimization period. Accordingly, the constraints for path 0 and path 1 are respectively:

$$\begin{aligned} \sum_{i=1}^{M} \sum_{\forall j | e_{i,j} \in \mathcal{E}} x_{i,j} \, q0_i \, r_{i,j} \;\; & \leq \;\; \bar{R}_0 \\ \sum_{i=1}^{M} \sum_{\forall j | e_{i,j} \in \mathcal{E}} x_{i,j} \, q1_i \, r_{i,j} \;\; & \leq \;\; \bar{R}_1 \end{aligned} \tag{8}$$

Constraint parameters $\bar{R}_0$ and $\bar{R}_1$ are network available bandwidth scaled by optimization period P. Network bandwidth can be estimated, for example, using congestion control algorithms like TCP-friendly rate control (TFRC).

3.3. Objective Function

The objective function we selected is the expected number of correctly decoded frames at the decoder for the M frames in the optimization window. Each frame F_i is correctly decoded iff F_i and all frames F_j's it depends on are delivered on-time and drop-free. We write $j \preceq i$ if frame F_i depends on frame F_j. Mathematically, maximizing this objective function means computing:

$$\max_{\{x_{i,j}\},\{q0_i\},\{q1_i\}} \left\{ \sum_{i=1}^{M} \prod_{\forall j \preceq i} \sum_{\forall k | e_{j,k} \in \mathcal{E}} x_{j,k}\, p(\mathbf{h}_j, q0_j, q1_j, r_{j,k}) \right\} \tag{9}$$

The problem is then: given pre-computed rate matrix $\mathbf{r}$ and delivery success probability function $p(\mathbf{h}_i, q0_i, q1_i, r_{i,j})$, find variables $\{x_{i,j}\}$, $\{q0_i\}$ and $\{q1_i\}$ that maximize (9) while satisfying the RF constraint (2) and the network resource constraints (8). This formally defined optimization is called the *RF / QoS / Path scheduling problem* (RQP scheduling).

4. DYNAMIC PROGRAMMING ALGORITHM

RQP scheduling can be easily shown to be NP-hard. To tackle the problem, we first construct a *weakly* exponential algorithm using dynamic programming to produce a locally optimal solution. Second, we introduce a rounding technique that allows multiple complexity-quality tradeoff points.

The algorithm composes of two recursive functions: $Sum()$ and $Prod()$. $Sum(i, R_0, R_1)$ returns the locally optimal expected number of correctly decodable frames for frame F_1 to F_i given available network resources R_0 and R_1 for path 0 and 1. $Prod(j, i, R_0, R_1)$ returns the probability that F_j is *decoded* correctly given R_0 and R_1 for path 0 and 1 are locally optimally distributed from F_1 to F_i. A call to $Sum(M, \bar{R}_0, \bar{R}_1)$ yields the locally optimal solution. $Sum(i, R_0, R_1)$ and $Prod(j, i, R_0, R_1)$ are shown in Figure 2 and 3, respectively.

The recursive case (line 5-15) of $Sum(i, R_0, R_1)$ locally tests every combination of RF j and QoS $q0$ and $q1$ for F_i for the maximal expected number of decodable frames. For a given selection of RF j and QoS $q0$ and $q1$, it induces a resource expenditure of $q0\ r_{i,j}$ and $q1\ r_{i,j}$ for path 0 and 1 respectively, and hence a decoding probability for F_i of $p(\mathbf{h}_i, q0, q1, r_{i,j}) * Prod(j, i-1, R_0 - q0\ r_{i,j}, R_1 - q1\ r_{i,j})$. That is added to the expected sum for F_1 to F_{i-1} — the recursive term $Sum(i-1, R_0 - q0\ r_{i,j}, R_1 - q1\ r_{i,j})$.

The results of the local search are stored in the $[i, R_0, R_1]$ entries of the four dynamic programming (DP) tables (line 16-17). DP tables are lookup tables so that if the same subproblem is called again, the already computed result can be simply looked up and returned (line 1-2).

The complexity of $Sum(M, \bar{R}_0, \bar{R}_1)$ is bounded by the time required to populate the DP tables, which can be easily shown to be $\mathcal{O}(M E_{\max} Q^2 \bar{R}_0 \bar{R}_1)$.

```
function $Sum(i, R_0, R_1)$
1. if ( $DPsum[i, R_0, R_1]$ is filled )    // DP case
2. {   return $DPsum[i, R_0, R_1]$;   }
3. if ( $R_0 < 0$ ) or ( $R_1 < 0$ )          // base case
4. {   return $-\infty$;   }
5. $S := 0$;                                  // recursive case
6. for each $j$ such that $e_{i,j} \in \mathcal{E}$,
7. {   for each $q0, q1 \in \mathcal{Q}$,
8.   { $s := Sum(i-1,\ R_0 - q0\, r_{i,j},\ R_1 - q1\, r_{i,j})$;
9.     if ( $j = i$ )                         // I-frame
10.      {   $s := s + p(\mathbf{h}_i, q0,\ q1,\ r_{i,j})$;   }
11.    else                                   // P-frame
12.      {   $s := s + p(\mathbf{h}_i, q0,\ q1,\ r_{i,j}) *$
             $Prod(j,\ i-1,\ R_0 - q0\, r_{i,j},\ R_1 - q1\, r_{i,j})$;   }
13.    if ( $s > S$ )
14.      {   $(S, X, Y, Z) := (s, j, q0, q1)$;   }
15. } }
16. ( $DPsum[i, R_0, R_1]$, $DPind[i, R_0, R_1]$ ) := ( $S, X$ );
17. ( $DPqos0[i, R_0, R_1]$, $DPqos1[i, R_0, R_1]$ ) := ( $Y, Z$ );
18. return $S$;
```

Fig. 2. Locally Optimal Algorithm $Sum(i, R_0, R_1)$

```
function $Prod(j, i, R_0, R_1)$
1. if ( $R_0 < 0$ ) or ( $R_1 < 0$ )      // base case 1
2. {  return 0;   }
3. if ( $j = i = 1$ )                     // base case 2
4. {  return $DPsum[1, R_0, R_1]$;   }
5. $X := DPind[i, R_0, R_1]$;
6. ( $Y, Z$ ) := ( $DPqos0[i, R_0, R_1]$,  $DPqos1[i, R_0, R_1]$ );
7. if ( $j < i$ )                          // recursive case
8. {  $P := Prod(j, i-1, R_0 - Y\, r_{i,X}, R_1 - Z\, r_{i,X})$; }
9. else                                   // j = i
10. {  $P := p(\mathbf{h}_i, Y, Z, r_{i,X}) *$
       $Prod(X, i-1, R_0 - Y\, r_{i,X}, R_1 - Z\, r_{i,X})$;  }
11. return $P$;
```

Fig. 3. Companion Function $Prod(j, i, R_0, R_1)$

4.1. Companion Recursive Function $Prod(j, i, R_0, R_1)$

From line 12 of Figure 2, we assume that $Prod(j, i, R_0, R_1)$ is called after $Sum(i, R_0, R_1)$ has been called, so we will assume entries $[i, R_0, R_1]$ of the DP tables are available during execution of $Prod(j, i, R_0, R_1)$.

The recursive case has two sub-cases: i) when $j < i$ (line 8), we recurse on $Prod(j, i-1, .)$ given we know resources $Y\ r_{i,X}$ and $Z\ r_{i,X}$ on path 0 and 1 are optimally used for node i; and, ii) when $j = i$ (line 10), we know term i of the product term — $p(\mathbf{h}_i, Y, Z, r_{i,X})$. The maximum product will be this term times the recursive term $Prod(Y, i-1, R_0 - Y\ r_{i,X}, R_1 - Z\ r_{i,X})$.

5. ROUNDING-BASED COMPLEXITY SCALING

Having the dynamic-programming based, weakly exponential $Sum(i, R_0, R_1)$, we now perform integer rounding to trade off complexity with solution quality. The rounding technique *DP dimension rounding* [6] reduces the size of the DP tables, hence reduces complexity.

We first scale and round down budgets $\bar{R}_0$ and $\bar{R}_1$ by factor $K_{DR} \in \mathcal{R}$ — i.e. $\left\lfloor \frac{\bar{R}_0}{K_{DR}} \right\rfloor$ and $\left\lfloor \frac{\bar{R}_1}{K_{DR}} \right\rfloor$ — as input to the optimization. We then scale and round up costs of transmitting predicted frame F_i, $q_i\ r_{i,j}$'s, by the same factor K_{DR} — i.e. $\left\lceil \frac{q_i\ r_{i,j}}{K_{DR}} \right\rceil$. Implementationally, we accordingly rewrite line 8 and 12 of $Sum(i, R_0, R_1)$ of Figure 2:

```
8. $s := Sum(i-1, R_0 - \left\lceil \frac{q0\ r_{i,j}}{K_{DR}} \right\rceil, R_1 - \left\lceil \frac{q1\ r_{i,j}}{K_{DR}} \right\rceil)$;
```

```
12. $s := s + \ p(\mathbf{h}_i, q0, q1, r_{i,j}) *$
    $Prod(j, i-1, R_0 - \left\lceil \frac{q0\ r_{i,j}}{K_{DR}} \right\rceil, R_1 - \left\lceil \frac{q1\ r_{i,j}}{K_{DR}} \right\rceil);$
```

Cost terms in line 8 and 10 of $Prod(j, i, R_0, R_1)$ in Figure 3 are similarly modified. Scaling down $\bar{R}_0$ and $\bar{R}_1$ means scaling down the dimension of the DP tables, hence the complexity is reduced by a factor of K_{DR}^2 at the cost of decreasing solution quality. Using $Sum(i, R_0, R_1)$ and $Prod(j, i, R_0, R_1)$ with the rewritten lines, the complexity of algorithm is now: $\mathcal{O}(ME_{\max}Q^2\bar{R}_0\bar{R}_1K_{DR}^{-2})$.

6. EXPERIMENTATION

We developed a network simulator called multiple-path network simulator (`muns`). Each transmission path k is implemented as a queue of constant service rate μ_k, followed by an identical and independently distributed (iid) packet erasure channel with shifted-Gamma-distributed delay. Upon each packet arrival, client informs the server of its status using ACK with zero client feedback delay. Parameters for the delay distributions for path 0 and 1 are (α_0,λ_0,κ_0) = (3, 0.1, 80) and (α_1,λ_1,κ_1) = (2, 0.1, 80), respectively. Packet loss rates are set at $(\pi_0, \pi_1) = (0.06, 0.10)$ for trial 1, and $(\pi_0, \pi_1) = (0.04, 0.08)$ for trial 2.

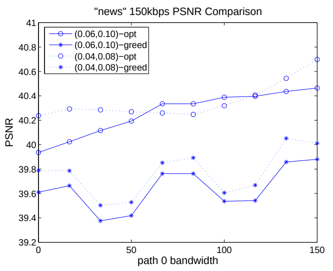
We use H.264 version JM8.4 [7] to encode a 300-frame QCIF MPEG sequence `news` sub-sampled in time by 2, at quantization 25 and 20 for I-frames and P-frames respectively, resulting in coding rate $140.38kbps$ if each P-frames F_i is coded using its previous frame F_{i-1}. To get rates $r_{i,j}$'s, we iteratively force each frame F_i to use reference F_{i-t} for motion prediction during iteration $t = \{0, 1, 2, 3, 4, 5\}$. The resulting coding rate is $r_{i,i-t}$. We assume $E_{\max} = 5$. The optimization period is $P = 300ms$, and window size is $M = 8$.

6.1. Results 1: RQP Scheduling Comparison

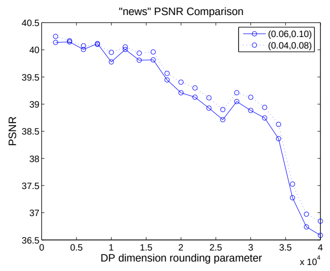
For the first set of experiment, we show that our optimization has practical merits and out-performs a competing greedy scheme. We first fixed the combined bandwidth of the two paths, $\bar{R}_0 + \bar{R}_1$, at $150kps$. Rounding parameter K_{DR} was kept constant at 1000. By varying the share of $150kbps$ bandwidth to the first path bandwidth $\bar{R}_0$, we tracked the corresponding performance at the client in PSNR, where if a frame is not correctly decoded, the correctly decoded frame closest in the past is used to replace it. The sequence was replayed 300 times for an averaging effect.

We compared our algorithm `opt` to a scheme `greedy` that contains elements of the simplified RaDiO [2]. Specifically, given budgets $\bar{R}_0$ and $\bar{R}_1$ in the two paths, it incrementally selects the best combination of reference frame, QoS and delivery path(s) that maximizes the benefit-to-cost ratio, where benefit is the increase in objective value (9), and cost is the increase in bit expenditure. `greedy` proceeds until both budgets are expended.

PSNR versus path 0 bandwidth $\bar{R}_0$ for both schemes under both trials are shown in Figure 4a. We see that PSNR increased with $\bar{R}_0$. This is expected since path 0 has lower



a) PSNR vs. path 0 bandwidth



b) PSNR vs. rounding parameter

Fig. 4. Streaming Performance for `news`

loss rate. Further, we see that `opt` outperformed `greedy` by $0.32dB$ to $0.85dB$ in the trial 1 and $0.36dB$ to $0.78dB$ in trial 2. This provides evidence that `opt` is an effective scheme for RQP scheduling.

6.2. Results 2: Performance / Complexity Tradeoff

We next show that graceful tradeoff of performance and complexity can be achieved. We held bandwidths of the two paths $\bar{R}_0$ and $\bar{R}_1$ constant at $(50kps, 100kps)$. K_{DR} was varied to observe the tradeoff between performance and complexity. PSNR versus K_{DR} for both trials are shown in Figure 4b.

We see that as K_{DR} increased, solution quality suffered due to rounding and PSNR decreased for both trials. More importantly, we see that PSNR decreased gradually over a wide range of rounding parameters. This suggests that a very wide range of useful complexity scaling can be realized using the rounding parameter K_{DR}.

7. CONCLUSION

We study packet scheduling of streaming video with flexible reference frames in a multi-path network scenario with feedbacks. In particular, we first presented an algorithm based on dynamic programming with high complexity. We then presented a rounding technique to allow multiple complexity / quality tradeoff points. Results showed that graceful complexity / quality tradeoff can be achieved over a wide range.

8. REFERENCES

[1] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003, vol. 13, no.7, pp. 560–576.

[2] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," Tech. Rep. MSR-TR-2001-35, Microsoft Research, February 2001.

[3] T. Wiegand, N. Farber, and B. Girod, "Error-resilient video transmission using long-term memory motion-compensated prediction," in *IEEE J. Select. Areas. Comm.*, June 2000, vol. 18, no.6, pp. 1050–1062.

[4] Y.J. Liang, M. Flieri, and B. Girod, "Low-latency video transmission over lossy packet networks using rate-distortion optimized reference picture selection," in *IEEE International Conference on Image Processing*, Rochester, NY, September 2002.

[5] C.-M. Huang, K.-C. Yang, and J.-S. Wang, "Error resilience supporting bi-directional frame recovery for video streaming," in *IEEE International Conference on Image Processing*, Singapore, October 2004.

[6] G. Cheung and W. t. Tan, "Reference frame optimization for multi-path video streaming using complexity scaling," in *International Packet Video Workshop*, Irvine, CA, December 2004.

[7] "The TML project web-page and archive," http://kbc.cs.tu-berlin.de/ stewe/vceg/.