PEER-TO-PEER VIDEO STREAMING
WITH INTERACTIVE REGION-OF-INTEREST

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Aditya Mavlankar
April 2010
© 2010 by Aditya Atul Mavlankar. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: http://purl.stanford.edu/dj419jq8315
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

        Bernd Girod, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

        Robert Gray

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

        Jatinder Pal Singh

Approved for the Stanford University Committee on Graduate Studies.

        Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

Increasing spatial resolution offered by digital imaging sensors and growing capacities of storage devices are helping the rise of high-spatial-resolution digital video. Such videos offer the possibility of viewing an arbitrary region-of-interest (RoI) interactively. The user can pan/tilt/zoom while watching the video. This allows watching user-selected portions of high-resolution video even on displays of lower spatial resolution. In the case of remote streaming, it avoids transmitting the entire field-of-view in high resolution, thus reducing required data rate.

The first main contribution of this thesis is spatial-random-access-enabled video compression. The goal is to encode the video content once such that arbitrary RoIs corresponding to different zoom factors can be extracted from the compressed bit-stream, thus avoiding a dedicated encoder for each user. We explore the trade-off between storage space and mean transmission bit-rate. We show how to choose the slice size for video coding for minimizing the mean transmission bit-rate.

Recently, peer-to-peer (P2P) video streaming has shown good potential for enabling large-scale video multicast at much lower cost than currently applied techniques. The second main contribution of this thesis is the design of a P2P live multicast streaming system that allows each user to independently control pan/tilt/zoom while still exploiting the commonalities in their RoIs for delivering content in a P2P manner. We present a distributed P2P protocol, which is run at each client and allows each client to receive and relay relevant data in real time. The protocol is designed by keeping in mind that the overlaps among users’ RoIs are highly transient. Our findings indicate that to support the same number of clients as traditional client-server unicast, our P2P approach requires much less server uplink bandwidth. We show how to allocate the server bandwidth for streaming regions that vary in popularity and rate-distortion operating points.
Acknowledgments

Foremost, I would like to thank Prof. Bernd Girod, whose support could always be counted on. His ideas and enthusiasm played a key role in shaping my style of conducting and presenting research. Prof. Robert M. Gray and Dr. Jatinder Pal Singh provided many thoughtful comments as readers of this dissertation while Prof. Leonid Kazovsky graciously chaired the oral defense committee.

With my colleagues in the Image, Video and Multimedia Systems (IVMS) group, I shared the knowledge of data compression/communication as well as life at Stanford. I enjoyed various insightful discussions not only with fellow graduate students, but also with visiting researchers, who were hosted by the group. I learnt several facets of video compression through discussions with David Varodayan, Dr. Pierpaolo Baccichet, Prof. Markus Flierl, Keiichi Chono and Mina Makar. My key collaborators in the area of peer-to-peer video streaming were Jeonghun Noh, Dr. Pierpaolo Baccichet and Dr. Sachin Agarwal. It was fun to invent and deploy the ViewXtreme (now called ClassX) lecture video recording and publishing system together with Piyush Agrawal and Prof. Bernd Girod. It is rewarding to see students using the system and I am sure the new recruits, Sherif Halawa and Derek Pang, will continue improving the system and furthering its deployment. I also enjoyed building demos to showcase my work as part of a collaboration with Deutsche Telekom Research Laboratories, Berlin, Germany. Throughout my stint with the IVMS group, Kelly Yilmaz made administrative things as easy as possible.

Several friends in the San Francisco Bay area can be credited with giving good company over the years, be it barbecue, skiing, hiking, dining, or just drinks: Parimal Puranik, Hung Tran, Suyog Kulkarni, Kailash Kothari, Vignesh Ganapathy, Omkar Mate, Vishal Sankhla, Khushboo Sancheti, Taral Joglekar, Karthik Vijayaraghavan, Fernando Amat, just to name few. It is a joy to share updates with several friends.
and relatives all over the globe, who stay in touch over phone and email, and more recently over Facebook. Back in India, I can always count on my sister, Dhanashree, and her husband, Amit, for providing a sounding board and good company as well as for telling my little nephew, Soham, stories about me. Amruta, my fiancé, is patient and fun-loving and reminds me to take a break from work. My grandmother, Charulata, and my parents, Shalaka and Atul, inculcated good qualities in me from day one, and I have deepest gratitude for their love and support.
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract</strong></td>
<td>iv</td>
</tr>
<tr>
<td><strong>Acknowledgments</strong></td>
<td>v</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Research Contributions</td>
<td>5</td>
</tr>
<tr>
<td>1.2 Organization</td>
<td>6</td>
</tr>
<tr>
<td><strong>2 Background</strong></td>
<td>9</td>
</tr>
<tr>
<td>2.1 H.264 Video Coding</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Coding for Random Access</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1 Image Coding for Random Access</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 Video Coding for Random Access</td>
<td>12</td>
</tr>
<tr>
<td>2.2.3 Multi-View Images/Videos</td>
<td>14</td>
</tr>
<tr>
<td>2.3 One-to-Many Transmission Approaches</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1 IP Multicast</td>
<td>15</td>
</tr>
<tr>
<td>2.3.2 Content Delivery Networks</td>
<td>17</td>
</tr>
<tr>
<td>2.3.3 Application-Layer P2P Multicast</td>
<td>17</td>
</tr>
<tr>
<td><strong>3 Spatial-Random-Access-Enabled Video Coding</strong></td>
<td>21</td>
</tr>
<tr>
<td>3.1 Coding Scheme Employing Upward Prediction and Slices</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1 Observations on Employing Current Video Compression Standards</td>
<td>24</td>
</tr>
<tr>
<td>3.1.2 Computational Complexity</td>
<td>25</td>
</tr>
</tbody>
</table>
3.2 Minimization of Mean Transmission Bit-Rate . . . . . . . . . . . . . . 26
  3.2.1 Pixel Overhead . . . . . . . . . . . . . . . . . . . . . . . . . . 27
  3.2.2 Optimal Slice Size . . . . . . . . . . . . . . . . . . . . . . . . 30
  3.2.3 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . 31
3.3 Background Extraction . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  3.3.1 Storage Reduction . . . . . . . . . . . . . . . . . . . . . . . . 41
  3.3.2 Transmission Bit-Rate Reduction . . . . . . . . . . . . . . . . . 41
  3.3.3 Modeling Mean Transmission Bit-Rate . . . . . . . . . . . . . . . 45
3.4 Chapter Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

4 P2P Live Multicast of IRoI Video . . . . . . . . . . . . . . . . . . . . . 51
  4.1 Overview of System Architecture . . . . . . . . . . . . . . . . . . . . 51
  4.2 Distributed P2P Protocol . . . . . . . . . . . . . . . . . . . . . . . . 54
    4.2.1 Subscription of Multicast Trees . . . . . . . . . . . . . . . . . . 55
    4.2.2 Soft Handoff . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
    4.2.3 Tree Maintenance . . . . . . . . . . . . . . . . . . . . . . . . . 57
  4.3 Analysis of Server Load . . . . . . . . . . . . . . . . . . . . . . . . . 58
    4.3.1 Server Load for Client-Server Unicast . . . . . . . . . . . . . . 58
    4.3.2 Server Load for Peer-to-Peer Multicast . . . . . . . . . . . . . 60
  4.4 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
    4.4.1 Experimental Setup . . . . . . . . . . . . . . . . . . . . . . . . 66
    4.4.2 Single Tree Per Tile . . . . . . . . . . . . . . . . . . . . . . . 68
    4.4.3 Multiple Trees Per Tile . . . . . . . . . . . . . . . . . . . . . . 69
  4.5 Chapter Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74

5 Optimal Server Bandwidth Allocation . . . . . . . . . . . . . . . . . . . . 79
  5.1 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
  5.2 Solution of Optimization Problem . . . . . . . . . . . . . . . . . . . . 82
    5.2.1 Parametric Approximation of Losses . . . . . . . . . . . . . . . 82
    5.2.2 Casting into a Knapsack Problem . . . . . . . . . . . . . . . . . 84
    5.2.3 Special Case . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
    5.2.4 General Scope of the Optimization Framework . . . . . . . . . . 86
5.3 Experimental Results ........................................ 87
5.4 Chapter Summary ........................................... 91

6 Conclusions .................................................. 93
   6.1 Lessons Learned .......................................... 94
   6.2 Open Problems ........................................... 95

A High-Spatial-Resolution Video Sequences .................. 97
   A.1 Cardgame .................................................. 97
   A.2 Brainstorming ............................................ 98
   A.3 Soccer ..................................................... 100
   A.4 Café ....................................................... 102
   A.5 Panel 1 ..................................................... 102
   A.6 Panel 2 ..................................................... 103

B RoI Prediction and Pre-Fetching .......................... 105
   B.1 Related Work ............................................ 105
   B.2 Extrapolating the Navigation Trajectory ............... 106

Bibliography .................................................... 112

ix
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Six high-spatial-resolution video sequences are used for the experiments. Each sequence is coded into a thumbnail and two high-resolution layers with the shown dimensions. Further information about the video sequences can be found in Appendix A.</td>
</tr>
<tr>
<td>3.2</td>
<td>Bit-rate and PSNR for encoding the thumbnail videos. PSNR values corresponding to the high-resolution layers of the six video sequences.</td>
</tr>
<tr>
<td>3.3</td>
<td>Slice sizes employed for experiments in Figs. 3.12 and 3.13 for comparing coding efficiency with and without the improvement based on background extraction. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.</td>
</tr>
<tr>
<td>3.4</td>
<td>Slice sizes chosen for experiments in later chapters. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.</td>
</tr>
<tr>
<td>4.1</td>
<td>Average tile subscription durations for six high-resolution video sequences. The values are computed from user interaction trajectories of 100 peers. Peer lifetimes themselves are exponentially distributed with an average of 90 seconds.</td>
</tr>
<tr>
<td>4.2</td>
<td>Total tiles and the corresponding bit-rate of the coded representation of six high-resolution sequences. Average number of tiles needed per peer and corresponding download data rate, obtained by averaging over 100 user interaction trajectories.</td>
</tr>
<tr>
<td>4.3</td>
<td>Server load for client-server unicast and P2P live multicast with 100 clients.</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Screenshots of a video player supporting interactive pan/tilt/zoom. Apart from displaying the RoI, the video player can display a thumbnail overview to aid navigation in the scene. The player could also offer to track certain objects, for example, the soccer ball or the soccer players. In the tracking mode, the user is not required to control pan and tilt, but could still control the zoom factor. ........................................ 2

2.1 The shaded “slices” belong to the same resolution layer and occur in the same location in each video frame, thus constituting a “tile”. .... 12

3.1 Graphical user interface: The client’s display shows the thumbnail and the RoI. The effect of changing the zoom factor can be seen by comparing the two screenshots. Each screenshot shows a frame of the panoramic Cardgame video sequence. ................................. 22

3.2 Video coding scheme: The thumbnail video constitutes a base layer and is coded with H.264/AVC using I, P and B pictures. The reconstructed base layer video frames are upsampled by a suitable factor and used as prediction signal for encoding video corresponding to the higher resolution layers. Higher resolution layers are coded using P slices. ... 23

3.3 Depending on the slice size and the location of the RoI within the given resolution layer, there is an overhead of pixels that are transmitted but not used for rendering the client’s display. The shaded portion depicts the pixel overhead in this example. ................................. 26
3.4 A sequence of pixels is divided into 1-D “slices”. In this example, the
length of each slice is $s = 4$. The length of the 1-D “region-of-interest”
is $R = 3$.

3.5 Model prediction versus empirical values for pixels transmitted per
rendered pixel, $\psi(s_w, s_h)$, shown for the Cardgame, Brainstorming and
Soccer sequences. The empirical values are obtained by averaging over
100 user interaction trajectories for each sequence. The second Y-axis
shows the bits per pixel for coding the residual of the high-resolution
layer, $\eta(s_w, s_h)$. The slice width and slice height in number of pixels
are denoted by $s_w$ and $s_h$ respectively.

3.6 Model prediction versus empirical values for pixels transmitted per
rendered pixel, $\psi(s_w, s_h)$, shown for the Café, Panel 1 and Panel 2 se-
quences. The empirical values are obtained by averaging over 100 user
interaction trajectories for each sequence. The second Y-axis shows
the bits per pixel for coding the residual of the high-resolution layer,
$\eta(s_w, s_h)$. The slice width and slice height in number of pixels are
denoted by $s_w$ and $s_h$ respectively.

3.7 Model prediction versus empirical values for bits transmitted per ren-
dered pixel, shown for the Cardgame, Brainstorming and Soccer se-
quences. The empirical values are obtained by averaging over 100 user
interaction trajectories for each sequence. The slice width and slice
height in number of pixels are denoted by $s_w$ and $s_h$ respectively.

3.8 Model prediction versus empirical values for bits transmitted per ren-
dered pixel, shown for the Café, Panel 1 and Panel 2 sequences. The
empirical values are obtained by averaging over 100 user interaction
trajectories for each sequence. The slice width and slice height in
number of pixels are denoted by $s_w$ and $s_h$ respectively.
3.9 Model prediction versus empirical values for zoom-adjusted relative pixel overhead, $\phi(s_w, s_h)$, shown for the *Brainstorming* and *Panel 1* sequences. The empirical values are obtained by averaging over 100 user interaction trajectories. The second Y-axis shows the bits per pixel for coding the residual of the high-resolution layer, $\eta(s_w, s_h)$. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.

3.10 Improvement based on background extraction: Each high-resolution layer frame has two references to choose from, the frame obtained by upsampling the reconstructed thumbnail frame and the background frame from the same layer in the background pyramid.

3.11 Background frame for layer 1 of *Cardgame*, *Brainstorming*, and *Soccer* sequences.

3.12 Coding bit-rate reduction through background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP), shown for the *Cardgame*, *Brainstorming*, and *Soccer* sequences. The slice sizes employed for the experiment are shown in Table 3.3.

3.13 Coding bit-rate reduction through background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP), shown for the *Café*, *Panel 1*, and *Panel 2* sequences. The slice sizes employed for the experiment are shown in Table 3.3.

3.14 Number of I and P slices transmitted over the streaming session, when background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP) are employed. The data are plotted for a single user interaction trajectory. Slice sizes are shown in Table 3.3.

3.15 Number of I and P slices transmitted over the streaming session, when background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP) are employed. The numbers are averaged over 100 user interaction trajectories. Slice sizes are shown in Table 3.3.
3.16 Transmission bit-rate is reduced after employing background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP); here shown for the two layers of Cardgame, Brainstorming and Soccer sequences. The slice width and slice height in number of pixels are denoted by \( s_w \) and \( s_h \) respectively. Transmission bit-rate values are obtained by counting bits required to transmit relevant high-resolution slices. The values are averaged over 100 user interaction trajectories.

3.17 Transmission bit-rate is reduced after employing background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP); here shown for the two layers of Café, Panel 1 and Panel 2 sequences. The slice width and slice height in number of pixels are denoted by \( s_w \) and \( s_h \) respectively. Transmission bit-rate values are obtained by counting bits required to transmit relevant high-resolution slices. The values are averaged over 100 user interaction trajectories.

3.18 Model prediction versus empirical values for bits transmitted per rendered pixel, shown for the Cardgame, Brainstorming and Soccer sequences, encoded using background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP). The empirical values are obtained by averaging over 100 user interaction trajectories. The slice width and slice height in number of pixels are denoted by \( s_w \) and \( s_h \) respectively.

3.19 Model prediction versus empirical values for bits transmitted per rendered pixel, shown for the Café, Panel 1 and Panel 2 sequences, encoded using background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP). The empirical values are obtained by averaging over 100 user interaction trajectories. The slice width and slice height in number of pixels are denoted by \( s_w \) and \( s_h \) respectively.
4.1 Example illustrating RoIs of three users within the multi-resolution video representation. The shaded slices represent tiles that are currently needed by more than one client and represent the “overlaps” exploited by the IRoI P2P system. ................. 52

4.2 The server distributes the thumbnail overview and the high-resolution tiles on different multicast trees. Peer-hosts attach to the appropriate multicast trees depending on their individual RoIs. ................. 52

4.3 The server advances the transmission of the base layer by $S$ seconds. When the client renders frame $n$ (at time $T_{X,n} + C$), it predicts the RoI in frame $n + d$ and initiates subscription to new trees. The parameters of the design help to render the RoI instantly upon user input. The diagram is drawn for $S = D + B_{c2e}$. ................. 54

4.4 Cumulative distribution function (cdf) of tile subscription durations computed for six high-resolution video sequences. One hundred user interaction trajectories representing as many peers were recorded for each video sequence. Each trajectory is 1000 seconds long, whereas peer lifetimes themselves are exponentially distributed with an average of 90 seconds. After switching off for a random interval, a peer can join the session again. The off-duration is exponentially distributed with an average of 10 seconds. Note that switching tiles due to RoI change is the dominant factor governing the distribution of tile subscription durations. ................. 56

4.5 Number of tiles streamed by the server shown for the (a) Cardgame and (b) Soccer sequences. For the P2P delivery of tiles, it is assumed that the server has at most one direct child per tile. ................. 63

4.6 Server load shown for the (a) Cardgame and (b) Soccer sequences. For the P2P delivery of tiles, it is assumed that the server has at most one direct child per tile. ................. 63
4.7 Number of tiles streamed by the server shown for the *Cardgame* sequence. (a) For P2P delivery, the server supports at most 2 direct children per tile. (b) For P2P delivery, the server supports at most 3 direct children per tile. 

4.8 Server load shown for the *Cardgame* sequence. (a) For P2P delivery, the server supports at most 2 direct children per tile. (b) For P2P delivery, the server supports at most 3 direct children per tile. 

4.9 Example of tree topology used for simulating the network. 

4.10 Cumulative distribution function (cdf) of peer uplink bandwidth. 

4.11 Traces of the aggregate numbers of received, required and missing slices over all 100 peers shown for the six high-resolution video sequences. 

4.12 Server load for supporting 100 peers shown for the six high-resolution video sequences. The first Y-axis shows the number of direct children supported by the server and the number of slices streamed to at least one direct child. The second Y-axis shows the server load in terms of bit-rate. 

4.13 Each tile distributed using two dedicated multicast trees. One tree delivers slices associated with even frames and the other tree delivers slices associated with odd frames. 

4.14 Percentage of missing slices and percentage of error-concealed pixels in the rendered RoI shown for the six high-resolution video sequences. The numbers are compared for having either one or two trees per tile. For either case, the server supports up to three direct children per multicast tree.
4.15 Luminance PSNR shown for five randomly chosen peers as well as the average over 100 peers. Despite the fact that the percentage of missing slices was comparable, the PSNR with two trees per tile was observed to be slightly better than with a single tree per tile. Also shown are the lower bound and the upper bound for the PSNR. The lower bound corresponds to the case in which all pixels are error-concealed using the base layer. The upper bound corresponds to the case in which all required high-resolution slices are received in time.

4.16 Volume of control traffic shown as percentage of the total traffic. For all six high-resolution sequences, the control traffic increases when more trees are built per tile.

5.1 Example with 14 peers subscribing tile $j$. On the left, $\alpha_j = 1$ direct child is supported and on the right, $\alpha_j = 2$ direct children are supported. To compare the impact of peer churn, imagine that the peer marked with the cross leaves suddenly. More peers are affected for the tree configuration shown on the left.

5.2 Example with the server hosting two tiles. The configurations shown on the left and on the right differ in the number of direct children supported for the two tiles.

5.3 The expected number of peers that fail to contribute to the total distortion reduction associated with tile $j$ can be modeled according to (5.4). The model parameters $(a, b, c)$ used for both plots (a) and (b) are $(a = 0.33, b = 0.9, c = 0.3)$. We observed that for a wide range of $(m_j, \tilde{p}_j)$, the approximation of $E(\hat{M}_j)$ by the parametric model is close to $E(\hat{M}_j) = \sum_{i \in P_j} \Pr\{Y_{ij} = 1\}$, obtained by assuming a tree structure for the tile determined by $\alpha_j$ and a common peer-node out-degree.

5.4 Comparison of the four rate allocation approaches when the server bandwidth is 10 Mbps. The results are averaged over 100 peers.
5.5 Comparison of the four rate allocation approaches when the server bandwidth is 5 Mbps. The results are averaged over 100 peers.

5.6 Comparison of the four rate allocation approaches when the server bandwidth is 3 Mbps. The results are averaged over 100 peers.

5.7 Snapshots of the audience sizes and the numbers of direct children associated with the different tiles, shown for the Brainstorming and Panel 1 sequences. Server bandwidth is 3 Mbps, whereas the data rates corresponding to the encoded Brainstorming and Panel 1 sequences are 3.8 Mbps and 5.2 Mbps respectively.

A.1 Sample frame and sample RoIs from various time-instants shown for the Cardgame sequence.

A.2 Sample frame and sample RoIs from various time-instants shown for the Brainstorming sequence.

A.3 Sample frame and sample RoIs from various time-instants shown for the Soccer sequence.

A.4 Sample frame and sample RoIs from various time-instants shown for the Café sequence.

A.5 Sample frame and sample RoIs from various time-instants shown for the Panel 1 sequence.

A.6 Sample frame and sample RoIs from various time-instants shown for the Panel 2 sequence.

B.1 Traces of numbers of received, required and missing slices shown aggregately for all 100 peers for the Cardgame, Brainstorming, Panel 1 and Panel 2 sequences. Left: No excess slices around the predicted RoI are pre-fetched. Right: Width and height of the predicted RoI are extended by 30% each while retaining the center location.

B.2 Percentage of missing slices and percentage of RoI pixels that have to be error-concealed.
B.3 Trace of percentage of error-concealed pixels in the RoI shown for a randomly sampled peer. Here shown for the Cardgame, Brainstorming, Panel 1 and Panel 2 sequences. .......................... 110

B.4 Luminance PSNR of the rendered RoI. The results are averaged over 100 peers. ................................................................. 111
Chapter 1

Introduction

High-spatial-resolution digital video will be widely available at low cost in the near future. This development is driven by increasing spatial resolution offered by digital imaging sensors and growing capacities of storage devices. Furthermore, there exist algorithms for stitching a comprehensive high-resolution view from multiple cameras [32, 61, 65, 79, 112, 175, 218, 219]. Some currently available video-conferencing systems stitch a large panoramic view in real time [5]. Also, image acquisition on spherical, cylindrical or hyperbolic image planes via multiple cameras can record scenes with a wide field-of-view while the recorded data can be warped later to the desired viewing format [204]. An example of such an acquisition device is [3].

Imagine that a user wants to watch a high-spatial-resolution video that exceeds the resolution of his display screen. If the user were to watch a downsampled version of the video that fits the display screen, then he might not be able to view local regions with the recorded high resolution. A possible solution to this problem is a video player that supports interactive pan/tilt/zoom. The user can thus choose to watch an arbitrary region-of-interest (RoI). We refer to this functionality as interactive region-of-interest (IRoI). Screenshots of a video player supporting IRoI can be seen in Fig. 1.1. The screenshots illustrate interactive viewing of a soccer game. Such a video player could also offer to track certain objects, whereby the user is not required to control pan and tilt, but could still control the zoom factor.
CHAPTER 1. INTRODUCTION

Figure 1.1: Screenshots of a video player supporting interactive pan/tilt/zoom. Apart from displaying the RoI, the video player can display a thumbnail overview to aid navigation in the scene. The player could also offer to track certain objects, for example, the soccer ball or the soccer players. In the tracking mode, the user is not required to control pan and tilt, but could still control the zoom factor.
CHAPTER 1. INTRODUCTION

Some practical scenarios where this kind of interactivity is well-suited are: interactive playback of high-resolution video from locally stored media, interactive TV for watching content captured with very high detail (e.g., interactive viewing of sports events), providing virtual pan/tilt/zoom within a wide-angle and high-resolution scene from a surveillance camera, and streaming instructional videos captured with high spatial resolution (e.g., lectures, panel discussions). A video clip that showcases interactive viewing of soccer in a TV-like setting can be seen here [8].

This thesis focuses on the scenario where a large audience is simultaneously watching a high-spatial-resolution video, yet each user can independently control his RoI. However, some contributions of this thesis, for instance, those in the area of video coding, also apply to scenarios like IRoI video playback from locally stored media and asynchronous IRoI video streaming to multiple users. Hereunder, we briefly describe challenges arising in these scenarios.

Consider IRoI video playback from locally stored media. In this case, the video content is encoded offline before storing it on the relevant media, for example, a high-capacity portable disk. Note that the RoI trajectory is not known while encoding the content. An RoI trajectory is determined each time a user watches the video with interactive pan/tilt/zoom. This leads us to two design choices; 1) the video player can be designed to decode the entire high spatial resolution while displaying only the RoI or 2) the adopted compression format could allow decoding only relevant regions, possibly with some overhead. Depending on the resolution of the video and the capability of the player, the first design choice might be prohibitive. Other applications mentioned above entail streaming from a remote source. In most cases, streaming the full spatial extent of the video to a user can be ruled out due to prohibitive bandwidth requirement. If RoI-specific portions can be streamed to the remote user, the RoI dimensions could be adapted to suit the available data rate for communication apart from the user’s display screen as noted above.

Consider the difficulty of employing a standard video encoder in the streaming scenario. A standard video encoder generally does not provide efficient spatial random access, i.e., the ability to extract regions from the compressed bit-stream. The video streaming can be for live content or for pre-stored content. For live content, the
server can crop out an RoI sequence on-the-fly considering the user’s pan/tilt/zoom commands and compress it as a video sequence using standard video encoding. The load of encoding might get prohibitively large with increasing number of users. For pre-stored content, unless the video is stored without compression, the server has to first decode the high-spatial-resolution video and then crop the RoI sequence. Not only does the load of encoding increase, but if multiple users watch the content asynchronously, then even the decoding load at the server increases. On the other hand, if a spatial-random-access-enabled video coding scheme is employed, the server needs to encode the recorded field-of-view only once, possibly with multiple resolution layers to support different zoom factors. The encoding load can thus be upper-bounded both for live content as well as for pre-stored content irrespective of the number of users.

In addition to limiting the encoding load, if the streaming bandwidth\(^1\) required from the server can also be limited, then the streaming system can scale to large numbers of users. If several users simultaneously watch arbitrary regions of a high-spatial-resolution video, there exist portions of the field-of-view that are commonly required by more than one user at a time. If the transmission system enables clients\(^2\) to relay commonly required data to each other, it could drastically reduce the amount of bandwidth required from dedicated servers. The main challenge is that user interaction determines in real time which regions are commonly required by which clients. The network overlay formed by the clients needs to adapt quickly and in a distributed manner, i.e., clients take most of the action necessary for acquiring the data they need, without much central intervention. Dependence on central intervention represents another hurdle in scaling. Another challenge is that clients can switch off randomly, taking away the resources they bring with them.

---

\(^1\)With respect to the server, we use the terms “bandwidth” and “uplink capacity” interchangeably, unless noted otherwise. Both terms imply the maximum uplink data rate provisioned at the server.

\(^2\)The term “client” encompasses both hardware and software employed at the user’s end.
CHAPTER 1. INTRODUCTION

1.1 Research Contributions

The major contributions of this thesis are:

- **Design of video coding scheme:** We present a spatial-random-access-enabled video coding scheme that allows encoding the entire high-spatial-resolution video once such that relevant portions of the bit-stream can be served to multiple clients, either synchronously or asynchronously, depending on their individual RoIs. As hinted above, this contribution has merit also in the local playback scenario, since a different RoI trajectory has to be accommodated each time a user views the video.

- **Analysis and optimization of video coding scheme:** The proposed coding scheme employs multiple resolution layers and multiple slices for encoding each resolution layer except the base layer. We analyze the trade-off between coding efficiency and transmission bit-rate governed by slice size. The analysis helps identify the slice size that minimizes mean transmission bit-rate without recording user interaction trajectories and preparing long bit-streams encoded with different slice sizes.

  We also present an improvement of the coding scheme based on background extraction. The proposed improvement reduces both storage as well as transmission bit-rate significantly while retaining efficient random access.

- **Application-layer peer-to-peer multicast of IRoI video:** Assuming that several remote users concurrently participate in watching an IRoI video, we present a transmission system that employs application-layer peer-to-peer (P2P) multicast for delivering the streams to the users. The proposed design exploits the commonalities in the clients’ regions such that they relay data to each other in real time. This allows limiting the bandwidth required at the server by making use of the forwarding capacities of the clients.

---

3 “Slice” refers to a rectangular portion of a video frame, whereas “tile” refers to the sequence of slices from the same resolution layer and at the same position in each video frame.
The design is optimized for low-delay forwarding of data as well as robustness to constant change in overlapping interests among the clients. The former is achieved by building different multicast trees corresponding to different tiles and propagating the data packets downstream along the tree, starting from the server, which is the root of all trees, to interested clients that attach themselves at various levels in the respective multicast trees. The distributed P2P protocol is designed such that it can reduce the disruption in received data due to the clients’ changing memberships of the multicast trees.

- **Server bandwidth allocation for IRoI P2P:** The tiles hosted by the server constitute a collection of data streams, each distributed using the P2P approach. The streams generally vary in popularity. The uplink rate allocated at the server for distributing the tiles should be adapted accordingly. We propose an optimization framework for rate allocation that takes into account the audience sizes, rate-distortion operating points as well as the rates at which clients join and leave the multicast trees. The proposed framework also has merit in other practical scenarios, for example a P2P server simultaneously hosting multiple video channels without IRoI functionality.

### 1.2 Organization

The thesis is organized as follows.

- Chapter 2 begins by reporting the state-of-the-art in video coding standards. It then presents a sampling of interactive streaming systems found in the literature. The goal of surveying prior systems is to highlight the challenges and earlier proposed approaches for providing random access and P2P design for other interactive applications that are similar in spirit to IRoI video. Section 2.2 discusses coding for interactive image browsing, approaches for RoI video (either interactive RoI or pre-defined RoI) and interactive free navigation based on novel-view generation. Section 2.3 provides background on one-to-many transmission approaches like IP multicast, content delivery networks
and application-layer P2P multicast. The same section also reviews receiver-driven layered multicast (RLM) that leverages IP multicast while encoding the multimedia signal in hierarchical layers. The concept of application-layer P2P multicast is contrasted against IP multicast. P2P multicast streaming systems can be roughly categorized into tree-based push systems and mesh-based pull systems. Section 2.3 discusses differences between the two approaches and argues that the tree-based push approach is better suited for interactive streaming applications.

- Chapter 3 presents a spatial-random-access-enabled video coding scheme that encodes the video with multiple resolution layers and also employs slices. The proposed scheme employs building blocks from a state-of-the-art video coding standard, called H.264/AVC, but itself is not standard-compliant. Section 3.1.1 highlights difficulties of standard-compliance. Section 3.1.2 comments on the computational complexity of the proposed scheme. The trade-off between storage and transmission bit-rate, associated with choosing the slice size, is studied in Section 3.2. The same section also shows how to choose the slice size that minimizes mean transmission bit-rate. An improvement of the coding scheme based on background extraction is presented in Section 3.3. It is shown that both storage as well as transmission bit-rate can be significantly reduced.

- Chapter 4 starts by presenting the overall system architecture for P2P live multicast of IRoI video. The distributed P2P protocol, which enables each client to obtain relevant video data, is presented in Section 4.2. Section 4.3 estimates the server load, in terms of uplink data rate, for traditional client-server unicast as well as P2P multicast delivery of tiles. Experimental results presented in Section 4.4 analyze the efficiency of the IRoI P2P system. The advantages of building two or more multicast trees per tile instead of a single tree per tile are also investigated in the same section.

- Chapter 5 begins by formulating the optimization problem of allocating rate at the server for distributing tiles in a P2P manner. Sections 5.2.1 and 5.2.2 solve the optimization as a knapsack problem. Section 5.2.3 investigates the influence
of audience size on rate allocated to a tile. Section 5.2.4 discusses modifications that fall within the scope of the optimization framework and other practical scenarios that could benefit from similar optimization.

Finally, in Chapter 6, we summarize the key findings of our work and discuss future research directions. This thesis also includes two appendices. Appendix A provides information about the video sequences used for experiments throughout the thesis. Appendix B analyzes performance of the IRoI P2P system after employing practical techniques for RoI prediction and pre-fetching.
Chapter 2

Background

This chapter reviews interactive streaming systems found in the literature. We briefly survey challenges in designing such systems as well as solutions proposed in the literature. The survey of prior systems will set the stage for later chapters discussing IRoI video streaming with the aim of serving large numbers of users. Before we discuss interactive streaming systems, we briefly report the state-of-the-art in video compression standards, since the scheme proposed in Chapter 3 employs building blocks from the latest standard, H.264/AVC.

2.1 H.264 Video Coding

The latest video coding standard, H.264/AVC, was standardized in March 2003 as a joint effort of ITU-T and ISO/IEC standardization bodies [95]. The standard is formally known as ITU-T Recommendation H.264 or ISO/IEC 14496 (MPEG-4) Part 10. The basic design, hybrid video coding, involving motion-compensated prediction and transform coding, has been retained in H.264/AVC, similar to its predecessors H.261, H.262 (MPEG-2), H.263 and MPEG-4 Visual (MPEG-4 Part 2) [85–88]. The principles of modern video coding are explained, e.g., in [25, 78, 162–164, 182, 197, 215, 227, 231]. An overview of video coding techniques with a focus on H.264/AVC can be found in [139, 171, 216, 237, 238].
The coding gains of H.264/AVC over earlier standards like H.263 and MPEG-2 are analyzed in [100]. H.264/AVC can achieve similar video quality as H.263 baseline profile while requiring about 50% of the bit-rate. Although the basic design is similar to earlier standards, the gain results from various new features like: better motion-compensated prediction involving more reference frames [67, 213, 235], varying block sizes down to $4 \times 4$ pixels [38, 143, 240, 245], spatial prediction of intracoded frames, improved entropy coding [145], and an improved deblocking filter [133]. A reference software implementation of the standard is freely available [6]. Recently, two extensions were added to the standard. One extension, called fidelity range extension (FRExt), is for high-resolution video signals [214]. It supports, for example, increased bit depth per sample and higher resolution for chrominance including sampling structures YUV 4:2:2 and YUV 4:4:4. The second extension, scalable video coding (SVC), supports spatial resolution scalability, temporal resolution scalability and quality scalability [96, 185]. For example, for spatial resolution scalability, SVC allows extracting portions of the bit-stream corresponding to lower spatial resolutions, thus avoiding transcoding for reducing the spatial resolution.

2.2 Coding for Random Access

Interactive streaming systems benefit from coding schemes that provide efficient random access. As the name suggests, this enables accessing relevant portions of the encoded data on-the-fly for serving the data to a user.

2.2.1 Image Coding for Random Access

Remote image browsing with interactive pan/tilt/zoom is similar in spirit to IRoI video. It can be used, e.g., for high-resolution archaeological images, aerial or satellite images, images of museum exhibits, online maps, etc [89, 105, 180, 246]. The scale for viewing online maps can be selected by controlling the zoom level. The image corresponding to each zoom level is subdivided into slices\(^1\) which are coded.

\(^1\)We note the distinction between “slice” and “tile”. The sequence of slices from the same resolution layer and at the same position in each video frame constitutes a tile.
independently. Also, the images corresponding to different zoom levels are coded independently. The compressed representation does not exploit redundancy across zoom levels but provides easy random access. The server accesses the slices intersecting the selected view and sends these slices to the user. Generally, after a zoom in operation, the relevant part from the current zoom level is interpolated to quickly render the newly desired view. As the slices from the new zoom level arrive, the graphics become crisper. Upon a zoom out operation, the relevant part from the old zoom level is resampled to render part of the display while uncovered regions might appear blank. As the new slices arrive, the rendered view becomes crisper.

Interactive browsing of images using JPEG2000 is explored in [94, 132, 170, 226]. The multi-resolution representation of an image using wavelets is leveraged to provide pan/tilt/zoom. The wavelet representation in JPEG2000 is not overcomplete unlike the Gaussian and Laplacian pyramids [30] that generate more coefficients than there are pixels in the high-resolution image. JPEG2000 encodes blocks of wavelet transform coefficients independently. Each coded block influences the reconstruction of a limited number of pixels of the image. Moreover, the coding of each block results in an independent, embedded bit-stream. This makes it possible to stream any given block with a desired degree of fidelity. The JPIP (JPEG2000 over Internet protocol) transmission protocol has been developed for communication between a client and a server to support remote interactive browsing of JPEG2000 coded images [225]. The server can keep track of the RoI trajectory of the client as well as the parts of the bit-stream that have already been streamed to the client. Given a rate of transmission, the server solves an optimization problem to determine which parts of the bit-stream to send and the order to transmit these parts for maximizing the quality of the rendered RoI. This is similar to packet scheduling algorithms proposed in [48] for streaming of video. It should be noted, however, that an accurate model for the distortion reduction due to successful delivery of any particular packet is necessary.


Figure 2.1: The shaded “slices” belong to the same resolution layer and occur in the same location in each video frame, thus constituting a “tile”.

2.2.2 Video Coding for Random Access

The distinction between slice and tile is illustrated in Fig. 2.1. The slices, shown shaded in Fig. 2.1, belong to the same resolution layer and occur in the same location in each video frame, thus constituting a tile.

The video compression standard H.264/AVC includes tools like flexible macroblock ordering (FMO) [119] and arbitrary slice ordering (ASO). These tools were primarily created for error resilience\(^2\), but can also be used to define an RoI prior to encoding [59, 120, 121]. The RoI can either be defined through manual input or through automatic content analysis. Slices corresponding to the RoI (or multiple RoIs) can be encoded with higher quality compared to other regions. Optionally, H.264/SVC, can be used for adding fine or coarse granular fidelity refinements for RoI slices. The user experiences higher quality for the RoI if the refinement packets are received. The RoI encoding parameters can be adapted to the network or the user [19]. Note that these systems transmit the entire picture while delivering the RoI with higher quality. Among the class of such systems, some employ JPEG2000 with RoI support and conditional replenishment for exploiting correlation among successive frames [58]. Parts of the image that are not replenished can be copied from the previous frame or a background store.

In our own work, we have proposed a video transmission system for interactive pan/tilt/zoom [142]. This system crops the RoI sequence from the high-resolution video and encodes it using H.264/AVC. The RoI cropping is adapted to yield efficient

\(^2\)An overview of error-resilient coding and transmission techniques can be found in [113,232,233].
motion compensation in the video encoder. The RoI adjustment is confined to ensure that the user does not notice the manipulation and experiences accurate RoI control. The normal mode of operation for this system is streaming live content but we also allow the user to rewind and play back older video. Note that in the second mode of operation, the high-resolution video is decoded prior to cropping the RoI sequence. Although efficient in terms of transmitted bit-rate, the drawback is that RoI video encoding has to be invoked for each user, thus limiting the system to few users. This system targets remote surveillance, which typically entails fewer simultaneous users than applications like interactive TV.

Video coding for spatial random access presents a special challenge. To achieve good compression efficiency, video compression schemes typically employ motion-compensated interframe prediction for exploiting correlation among successive frames [71–73]. However, the coding dependencies among successive frames make it difficult to provide random access for spatial browsing within the scene. The decoding of a block of pixels requires that other reference frame blocks used by the predictor have previously been decoded. These reference frame blocks might lie outside the RoI and might not have been transmitted or decoded earlier.

Coding, transmission and rendering of high-resolution panoramic videos using MPEG-4 is proposed in [77, 82]. A limited part of the entire scene is transmitted to the client depending on the chosen viewpoint. Only intraframe coding is used to allow random access. The scene is subdivided into slices which are coded independently. The authors also consider interframe coding to improve compression efficiency. However, they note that this involves transmitting slices from the past if the current slice requires those for its decoding. A longer intraframe period entails significant transmission overhead for slices from the latter frames in the group of pictures (GOP), as this dependency chain grows. Besides the transmission overhead, the reference frame blocks also entail growing overhead of decoding.
2.2.3 Multi-View Images/Videos

Interactive streaming systems that provide virtual fly-around in the scene employ novel-view generation to render views of the scene from arbitrary viewpoints. With these systems, the user can experience more free interactive navigation compared to pan/tilt/zoom [104, 205–207, 224]. These systems typically employ image-based rendering (IBR) which is a technique to generate the novel view from multiple views of the scene recorded using multiple cameras [123, 141, 159, 200, 201]. Note that in these applications, the scene itself might or might not be evolving in time. Transmitting arbitrary views from the multi-view data-set on-the-fly also entails random access issues similar to those arising for transmitting arbitrary regions in interactive pan/tilt/zoom. Interframe coding for compressing successive images in time as well as from neighboring views can achieve higher compression efficiency but can lead to undesirable dependencies for accessing random views. There exists a large body of works that employs hybrid video coding for compressing multi-view data-sets [22, 23, 68, 109, 116, 135, 136, 160, 208]. These studies highlight the trade-off in storage requirement, mean transmission bit-rate and decoding complexity. Recently, an analytical framework was proposed for optimizing the coding structure for coding multi-view data-sets [40, 41]. The framework allows multiple representations of a picture, for example, compressed using different reference pictures. The optimization not only finds the best coding structure but also determines the best set of coded pictures to transmit corresponding to a viewing path. The framework can accommodate constraints like limited step-size for view switching, permitting view switching only during certain frame-intervals and capping the length of the burst of reference frames that are used for decoding a viewed frame but are not themselves displayed. The framework can minimize a weighted sum of expected transmission bit-rate and storage cost for storing the compressed pictures.

The video compression standard H.264/AVC defines two new slice types, called SP and SI slices [102, 189]. Using these slice types, it is possible to create multiple representations of a video frame using different reference frames. Similar to the solutions described above, the representation to be streamed is chosen according to the reference frames available at the decoder. However, the novelty is that
the reconstruction is guaranteed to be identical. This drastically reduces the total number of representations required to be stored. SP frames have been used for interactive streaming of static light fields [177, 178]. Another solution to the random access problem associated with multi-view data-sets is based on distributed source coding (DSC) [9, 90, 220]. This solution allows coding a frame independently, however, the decoder can utilize available frames as side-information to aid decoding. If a coded frame is represented using enough parity bits it leads to an identical reconstruction irrespective of the frame(s) used by the decoder as side-information. This implies that multiple representations are not required to be stored, however, the number of parity bits is determined by the side-information frame having the least correlation to the frame to be coded. Similar to some prior work based on hybrid video coding for multi-view data-sets mentioned above, recent work based on DSC explores the trade-off between transmission bit-rate and storage requirement [42–45].

2.3 One-to-Many Transmission Approaches

In the following, we discuss different one-to-many transmission approaches, since one of our goals is to design the system for serving large numbers of users watching an IRoI video simultaneously.

2.3.1 IP Multicast

A comprehensive overview of multipoint communication over packet-switched networks can be found in [60]. IP multicast can drastically reduce the bandwidth required from dedicated media servers [56, 57]. IP multicast allows sending an IP datagram to a group of hosts identified by a single IP destination address [14]. Hosts may join and leave a multicast group at any time. This requires multicast-capable routers that replicate packets as required. Even though IP multicast is extremely efficient at distributing data to multiple interested receivers, most routers on the Internet keep this functionality turned off due to reasons related to security, billing and the size of the data-structures to be maintained by the router. Nevertheless, the bandwidth
conservation benefits of IP multicast have resulted in rising deployment for corporate communications [147] and, more recently, IPTV service [97, 107, 161, 168].

The heterogeneity of available throughput from source to the different receivers poses a challenge for multimedia streaming over multicast [46, 129, 134]. The seminal work on receiver-driven layered multicast (RLM) [158] focuses on video streaming without interactive pan/tilt/zoom. The authors propose compressing the multimedia signal in hierarchical layers and letting individual receivers choose the layers to join. Receiving more layers leads to better quality. Each layer is delivered using a different multicast group. Note that if a receiver joins too many layers and creates congestion on a link, then packets can be dropped indiscriminately from all layers affecting received quality, possibly for multiple receivers that share the congested link. A receiver performs regular tests to decide if it should unsubscribe already joined layers or subscribe to new layers. Shared learning among receivers can reduce the number of tests and hence the convergence time.

Multicast solutions allowing different end-points to receive different rates fall under the category called multirate multicast. An overview of such techniques can be found in [125]. Some techniques apply forward error correction (FEC) with or without layered coding to mitigate effects of packet loss [106, 183, 222]. Automatic repeat request (ARQ) in multicast leads to feedback implosion [66]. Some solutions suggest forming groups of receivers with some kind of hierarchy such that feedback all the way to the source can be limited [49, 106].

Recently, the RLM framework was adapted for interactive dynamic light field streaming [114]. Depending on the chosen viewpoint, the client decides which views and consequently which multicast groups to subscribe to. The latency for joining a new multicast group is generally low with IP multicast [63]. As in the case of RLM, it is the client’s responsibility to avoid congestion on intermediate links. The source does not adapt transmission to curtail congestion; it keeps transmitting IP datagrams to the multicast groups’ addresses.
2.3.2 Content Delivery Networks

A single server can serve a limited number of clients simultaneously with client-server unicast. Hence, a common and longstanding approach has been to deploy an overlay of replication or mirror servers such that users can be redirected to a server with available capacity. Akamai, Limelight, VitalStream, and Mirror Image are some commercial implementations of such overlays known as content delivery networks (CDNs) [51, 111, 228, 236]. Optimizing the placement of the replicated content has been studied in [15, 39, 70, 91, 101, 103, 124, 176]. The optimization of a CDN varies depending on whether it delivers streaming media or hypertext transfer protocol (HTTP) web objects [93, 122]. Streaming is data intensive and packets belonging to a streaming object have stricter deadlines. If the content distributed is a live media stream then real-time adaptation is required for feeding copies of the content to the replication servers, as well as directing users to the replication servers [20, 33, 37, 92]. Since CDNs require significant dedicated infrastructure, they are expensive. They have to be provisioned for the rare event of a large audience. The advantage is that it is possible to give some service guarantees. Recent work motivates peering among different CDN operators as well as investigates challenges, like data privacy, content placement on outside CDNs, redirecting requests, incentives and pricing [27, 31].

2.3.3 Application-Layer P2P Multicast

Contrary to network-layer IP multicast, P2P streaming implements the multicast protocol in software at the end-hosts rather than at the routers inside the network [50]. Unlike IP multicast, the application-layer software can be widely deployed with little investment. Although the P2P approach generally results in more duplication of packets and inefficient routing compared to IP multicast, the benefits outweigh the inefficiencies. The source as well as each peer\(^3\) can respond to local retransmission requests as well as perform sophisticated packet scheduling to maximize the experience of downstream peers [188].

\(^3\)The “client” employed at the user’s end is referred to as a “peer” in the P2P context.
CHAPTER 2. BACKGROUND

P2P file sharing systems like BitTorrent, Gnutella, eDonkey, or Kazaa can be looked upon as predecessors of P2P streaming systems. The main challenge in P2P file sharing systems is locating peers that can share the relevant content. A large body of research explores the scalability and efficiency of distributed search algorithms such that a centralized index can be avoided [35,83,131,137,181,209,212,234]. The challenge specific to P2P streaming, on the other hand, is locating peers in real time for supporting the required throughput. An early work on P2P video streaming aimed at building a distributed video-on-demand system [196]. This was followed by other works on asynchronous P2P video streaming [52,53,242,244]. Substantial research effort has been devoted to synchronous P2P video streaming to large audiences. Early work in this area includes [34,50,173]. Several implementations of such systems have been deployed, for example, SopCast, PPLive, PPStream, CoolStreaming, GridMedia, etc. Various studies have assessed the performance of these systems, with some studies analyzing real sessions involving thousands of peers on the Internet [10,80,186,223,248–250]. Since these systems aim to accommodate as many users as possible, it is beneficial to be able to traverse network address translators (NATs) and firewalls [26,69,184,211].

P2P streaming systems can be broadly classified into mesh-pull vs. tree-push systems [74,140]. The design of mesh-pull systems evolved from P2P file sharing systems. In these systems, a peer advertises the chunks of data that it has and complies with requests to relay chunks to other peers [173,186,250]. Tree-push systems, on the other hand, distribute data using one or more spanning trees [34,50,190]. After finding its place inside a distribution tree, a peer generally keeps its association with the parent and its children and relays data without waiting for requests from children. Building multiple spanning trees enables finer aggregation of forwarding capacity and brings robustness to peer churn [190]. In the literature, several techniques have been used for robustness, sometimes in the context of both mesh-pull as well as tree-push systems: FEC, layered coding, multiple description coding, retransmissions, distortion-optimized packet scheduling, etc. [21,172,190,193]. Some prior work employing the tree-push approach focuses on low latency [17,191]. Recent work has investigated a hybrid of CDN and P2P, depending on the CDN to provide
lower latency and higher robustness, while P2P boosts the capacity of the network manyfold [187, 247].

Compared to mesh-pull systems, tree-push systems result in fewer duplicate packets, lower end-to-end delay and lower delay-jitter [11, 148]. These traits are beneficial for interactive streaming systems where sub-streams of the coded content are required on-the-fly. A tree-based P2P protocol has been recently proposed for interactive streaming of dynamic light fields [117, 118]. Early results demonstrate the capability of the system to support many more users with the same server resources as compared to traditional unicast client-server streaming [118].

Before closing this chapter, we discuss the relevance of network coding [12, 28, 127, 128] for video multicast. It has been shown that, to achieve capacity, the relay-nodes might have to process and modify data before forwarding. A special form of network coding called random linear network coding (RNC), also applicable to multi-source multicast\(^4\), is proved to achieve capacity [84]. This can be looked upon as a generalization of routing and has led to studies for improving crossbar switches that can be deployed in a multicast-capable fabric [108, 217]. Note that in application-layer P2P multicast, the intermediate routers are not assumed to perform network coding. The end-hosts can apply network coding, however, it provides no gain in throughput for P2P multicast, as proved in [47]. The proof in [47] assumes that the content at the server is infinitely divisible and is infinitely large so that the server can continuously send content to the peers. The proof also assumes a simple star topology, i.e., the server and all peers are connected to a central node. The central node relays packets without performing network coding. It is also assumed that if a peer relays the same data to multiple peers, it has to push the data on its uplink as many times. The analysis in [47] does not settle the debate whether network coding can provide any benefit in practical P2P streaming systems. Inspired by network coding, recent work proposes using rateless codes such that peers make random linear

\(^4\)The situation dealt with in this thesis involves transmitting relevant portions of video data to multiple sets of receivers, technically falling under the multi-source multicast category.
combinations of packets before relaying them [98, 165]. Rateless codes relieve coordination effort in acquiring data from different parent peers\(^5\). Similar investigations exist in the context of network coding for P2P file sharing [75, 76, 229]. In general, P2P protocols can handle downloading of data from multiple peers, reducing the need for applying rateless codes for this purpose. As pointed out in the seminal paper [62], realizing a mutually beneficial relationship between information theory and practical networking is non-trivial, to say the least. After all, information-theoretic analysis often makes assumptions that conflict with reality or are unreasonable in practical engineering. For example, such analysis typically neglects burstiness of the source, it often neglects delay in transmitting/forwarding packets, it typically assumes block codes of unbounded length, and it often ignores receiver feedback [62].

\(^5\)Rateless codes enable successful decoding of a chunk of data after receiving a certain number of packets, without depending on which specific packets are received [138, 198].
Chapter 3

Spatial-Random-Access-Enabled Video Coding

The previous chapter described how motion compensation poses a challenge in providing spatial random access. We now present solutions that facilitate encoding the high-spatial-resolution video such that relevant portions of the coded bit-stream can be simply served to multiple users, i.e., without employing a dedicated encoder per user.

We have developed a graphical user interface that allows the user to select the region-of-interest (RoI) while watching the video. The RoI location and zoom factor are controlled by operating the mouse. The application supports continuous zoom to provide smooth control of the zoom factor. In addition to the RoI, the client can also display a thumbnail overview with an overlaid rectangle indicating the location of the RoI. Screenshots of the client's display are shown in Figs. 1.1 and 3.1 for two high-spatial-resolution video sequences.

The change in the zoom factor occurs in fine steps, hence it feels to the user as if the zoom change were continuous. For supporting different zoom factors, it is beneficial to be able to access data corresponding to different spatial resolutions in the coded bit-stream. Section 3.1 describes the proposed video coding scheme and discusses its key benefits. The coding scheme was first presented by us in [149]. Section 3.2 shows how to optimize the scheme for minimizing mean transmission bit-rate. Section 3.3
CHAPTER 3. SPATIAL-RANDOM-ACCESS-ENABLED VIDEO CODING

shows how the storage as well as mean transmission bit-rate can be further reduced by employing background extraction while retaining efficient spatial random access. We have presented the improvement based on background extraction in [151] and have recently submitted a detailed article on the proposed coding scheme [150].

3.1 Coding Scheme Employing Upward Prediction and Slices

The coding scheme, illustrated in Fig. 3.2, shows the video being encoded with multiple resolution layers. The thumbnail overview constitutes a base layer video and is coded with H.264/AVC using I, P and B pictures. The reconstructed base layer video frames are upsampled by a suitable factor and used as prediction signal for encoding video corresponding to the higher resolution layers. Each frame belonging to a higher resolution layer is coded using a grid of rectangular P slices. Employing upward
Figure 3.2: Video coding scheme: The thumbnail video constitutes a base layer and is coded with H.264/AVC using I, P and B pictures. The reconstructed base layer video frames are upsampled by a suitable factor and used as prediction signal for encoding video corresponding to the higher resolution layers. Higher resolution layers are coded using P slices.

prediction from only the thumbnail enables efficient random access to local regions within any spatial resolution. For a given frame-interval, the display of the client is rendered by transmitting the corresponding frame from the base layer and few P slices from exactly one higher resolution layer. We transmit slices from that resolution layer which corresponds closest to the user’s current zoom factor. At the client’s side, the corresponding RoI from this resolution layer is resampled to correspond to the user’s zoom factor. Thus, we can avoid storing a large number of high-resolution layers. We typically employ two high-resolution layers and one thumbnail video but the client supports continuous adjustment of the zoom factor\(^1\). If a required enhancement layer P slice is unavailable at the client, for example, due to loss in the network, then error concealment can be performed by upampling portions of the thumbnail video.

In our experiments, reported later in this chapter, the spatial resolution layers stored at the server are dyadically spaced. Hence, the reconstructed thumbnail frame needs to be upsampled by powers of two horizontally and vertically to generate the

\(^1\)A zoom in operation entails multiplying the zoom factor by 1.1, whereas a zoom out operation entails dividing the zoom factor by 1.1. Zoom change might appear too slow to the user if the multiplier is reduced below this value and it might appear too jerky to the user if the multiplier is increased to higher values.
corresponding prediction signals. For upsampling the luminance component, we employ the six-tap filter having the coefficients \((1, -5, 20, 20, -5, 1)/32\) as defined in H.264/AVC. For chroma, we employ a simple two-tap filter with equal coefficients. The upsampling procedure is repeated an appropriate number of times depending on the resolution layer. Although we choose these parameters for our experiments, our design can incorporate arbitrarily spaced resolution layers and also arbitrary procedures for upsampling the reconstructed base layer. Also, at the client’s side, for resampling the corresponding RoI from the chosen resolution layer, any technique can be accommodated. In our experiments that follow later, we use bilinear interpolation.

3.1.1 Observations on Employing Current Video Compression Standards

The coding scheme proposed above uses H.264/AVC building blocks but itself is not standard compliant. State-of-the-art video compression standards, H.264/AVC and SVC, provide tools like slices but no straightforward method for spatial random access since their main focus has been compression efficiency of full-frame video and resilience to losses. SVC supports both slices as well as spatial resolution layers. Alas, SVC allows only single-loop decoding whereas upward prediction from intercoded base-layer frames implies multiple-loop decoding. If the base layer frame is intercoded, then SVC allows predicting the motion compensation residual at the higher resolution layer from the residual at the base layer. However, interframe prediction dependencies across tiles belonging to a high-resolution layer hamper spatial random access. Note that for employing SVC, the motion vectors (MVs) can be chosen to avoid inter-tile dependencies. Also note that instead of SVC, AVC could be employed separately for the high-resolution layers with the MVs similarly restricted to eliminate inter-tile dependencies. This is very similar to treating the tiles as separate video sequences. An obvious drawback is the redundancy between the high-resolution tiles and the base layer. A second drawback is that after RoI change, a newly needed tile can only be decoded starting with an intracoded slice. In contrast, the scheme presented in Fig. 3.2 exploits the redundancy between the base layer and each high-resolution
layer and a newly needed tile can be decoded starting from any frame-interval. Thus, the system can serve a new tile during any frame-interval without waiting for the end of the GOP or having to transmit extra slices from the past.

Prior work on view random access, discussed in Section 2.2.3, employs multiple representations for coding an image. Similarly, we can use multiple representations for coding a high-resolution slice. This will allow us to use interframe coding among successive high-resolution layer frames and to transmit the appropriate representation for a slice depending on the slices that have been transmitted earlier. Some representations will exploit inter-tile correlation, thus lowering the transmission bit-rate. However, more storage will be required for multiple representations. The benefit of the scheme in Fig. 3.2 is that knowing the current RoI is enough to decide which data need to be transmitted unlike the case of multiple representations where the decision is conditional on prior transmitted data.

3.1.2 Computational Complexity

In the proposed scheme, motion compensation among successive frames is performed at the base layer. While coding high-resolution P slices, we employ displacement compensation with a small search range of about four pixels to find the best match relative to the upsampled base layer frame. The total encoding load is determined by the maximum resolution and the number of layers. For example, assume two high-resolution layers and the base layer, all dyadically spaced in terms of resolution. Now consider the load of encoding the highest resolution layer with our scheme compared to the load of encoding the same using standard motion-compensated hybrid video coding. Although the search range is smaller for displacement compensation compared to the search range for motion compensation with standard encoding, let us assume, for simplicity, that the encoding load is comparable. In comparison to the highest resolution layer, the encoding loads corresponding to the other high-resolution layer and the base layer are about one-fourth and one-sixteenth respectively. Thus, the total load of encoding can be estimated to be roughly 1.3 times the load of encoding just the highest resolution layer using standard motion-compensated hybrid video coding.
Figure 3.3: Depending on the slice size and the location of the RoI within the given resolution layer, there is an overhead of pixels that are transmitted but not used for rendering the client’s display. The shaded portion depicts the pixel overhead in this example.

Note that compared to video encoding, creating lower resolution versions of the video through resampling and creating the reference frames via upsampling reconstructed base layer frames entail negligible computational complexity.

3.2 Minimization of Mean Transmission Bit-Rate

For the coding scheme shown in Fig. 3.2, the slice size for each resolution layer can be independently optimized given the prediction residual for that layer. The strategy proposed here can be independently used for all layers. Given a resolution layer, we assume that the slices form a regular rectangular grid, so that every slice is $s_w$ pixels wide and $s_h$ pixels tall. The slices on the boundaries can have smaller dimensions due to the layer dimensions not being integer multiples of the slice dimensions.

The number of bits transmitted to the client, or decoded for local playback, depends on the slice size as well as the user’s RoI trajectory over the interactive viewing session. The quality of the decoded video depends on the quantization parameter (QP) used for encoding the slices. For the same QP, almost the same quality is obtained for different slice sizes, even though the number of bits is different. Hence, given the QP, our goal is to choose the slice size that minimizes the expected number of bits transmitted or decoded per rendered pixel. The smaller the slice size the worse is the coding efficiency. This is because of an increased number of slice headers, lack of context continuation across slice boundaries for context adaptive coding, and inability to exploit inter-pixel correlation across slice boundaries. On the other hand, a smaller slice size entails lower pixel overhead. The pixel overhead consists of pixels
Figure 3.4: A sequence of pixels is divided into 1-D “slices”. In this example, the length of each slice is $s = 4$. The length of the 1-D “region-of-interest” is $R = 3$.

that have to be transmitted or decoded because of the coarse slice division, but are not used to render the client’s display. For example, the shaded pixels in Fig. 3.3 show the pixel overhead for the shown slice grid and location of the RoI.

In the following analysis, we assume that the RoI location can be changed with a granularity of one pixel both horizontally and vertically. Also, every location is equally likely to be selected. Depending on the application scenario, the slices might be put in different transport layer packets. The packetization overhead of layers below the application layer, for example RTP/UDP/IP, has not been taken into account but can be easily incorporated into the proposed optimization framework.

3.2.1 Pixel Overhead

To simplify the analysis, we first consider the 1-D case and then extend it to 2-D.

Analysis in 1-D

Imagine an infinitely long sequence of pixels. This sequence is divided into “slices” of length $s$. For example, in Fig. 3.4, $s = 4$. Also given is the length of the “region-of-interest”, denoted by $R$. Assume $R = 3$ in this example. To calculate the pixel overhead, we are interested in the probability distribution of the number of 1-D “slices” that need to be transmitted. This can be obtained by testing for locations within one slice, since the pattern repeats every slice. For RoI locations $w$ and $x$, we would need to transmit a single slice, whereas for locations $y$ and $z$, we would need to transmit 2 slices. Let $N$ be the random variable representing the number of slices to
be transmitted. Given $s$ and $R$, we can uniquely choose $m, R^* \in \mathbb{N}$ such that $m \geq 0$ and $1 \leq R^* \leq s$ and also the following relationship holds

$$R = ms + R^*.$$

By inspection, we find the p.m.f. of random variable $N$:

$$\Pr\{N = m + 1\} = \frac{s - (R^* - 1)}{s},$$

$$\Pr\{N = m + 2\} = \frac{R^* - 1}{s},$$

and zero everywhere else. From the p.m.f. of $N$,

$$E\{N\} = (m + 1)\frac{s - (R^* - 1)}{s} + (m + 2)\frac{R^* - 1}{s} = (m + 1) + \frac{R^* - 1}{s}. \quad (3.1)$$

Let $P$ be the random variable which denotes the number of pixels that need to be transmitted.

$$E\{P\} = s \times E\{N\} = (m + 1)s + R^* - 1 = R + s - 1. \quad (3.2)$$

The expected pixel overhead is $s - 1$. It increases monotonically with slice length $s$ and surprisingly is independent of the length $R$ of the “region-of-interest”. Alas, the result is that simple only for 1-D.

If $R$ itself is a random variable, then for a given value of $R = r$, (3.2) can be rewritten as:

$$E\{P|R = r\} = r + s - 1. \quad (3.3)$$

\footnote{For the toy example in Fig. 3.4, $N$ has non-zero probability for values 1 and 2. Note that in general, $m$ can be non-zero, thus leading $N$ to have non-zero probability for two consecutive integers $m + 1$ and $m + 2$.}
Analysis in 2-D

We define two new random variables, \( P_w \), the number of columns to be transmitted and \( P_h \), the number of rows to be transmitted. Similarly, \( R_w \) and \( R_h \) are random variables denoting the number of columns and rows (among those transmitted) required to render the RoI respectively. From the 1-D analysis, we obtain

\[
E\{P_w \mid R_w = r_w\} = r_w + s_w - 1,
\]
\[
E\{P_h \mid R_h = r_h\} = r_h + s_h - 1.
\]

The number of transmitted pixels is also a random variable, \( P = P_w P_h \). Since the positioning of the RoI horizontally is independent of the positioning vertically, \( P_w \) and \( P_h \) are conditionally independent given \( R_w, R_h \). Hence,

\[
E\{P \mid R_w = r_w, R_h = r_h\} = (r_w + s_w - 1)(r_h + s_h - 1).
\] (3.4)

While \( R_w \times R_h \) is the number of pixels among those transmitted which are rendered in the RoI window, it is not the size of the RoI window. The array of \( R_w \times R_h \) pixels is resampled to fit the fixed size \( d_w \times d_h \) of the RoI display window. Recall that this allows us to support arbitrary zoom factors with a small number of discretely spaced resolution layers.

Random variable \( Z_C \) denotes the continuous zoom factor controlled by user input. Its value determines the value of the discrete random variable \( Z_D \) which is the zoom factor rounded to a power of two. For example,

\[
Z_D = \begin{cases} 
1, & \text{if } (1 \leq Z_C < 1.5) \\
2, & \text{if } (1.5 \leq Z_C < 4).
\end{cases}
\] (3.5)

To render the RoI at some zoom factor \( Z_C \), we round to discrete zoom factor \( Z_D \) and retrieve the resolution layer \( \log_2(Z_D) + 1 \). The mismatch \( Z_C/Z_D \) is made up by resizing the transmitted video after decoding. For our analysis, we need to model the conditional pdf of \( Z_C \) given the layer number. In our modeling below, we assume that,
given the layer number, $Z_C$ is uniformly distributed. For example, if the optimization is being carried out for the second layer in the example above, then we assume that $Z_C$ is uniformly distributed between 1.5 and 4. Note that the distribution of the user-selected zoom factor in practice might depend on sizes of certain salient objects in the video. Nevertheless, we make the assumption about $Z_C$ without performing any video content analysis\(^3\).

Let $d_w$ and $d_h$ be constants denoting the width and height of the RoI display portion on the client’s display respectively. The random variables $R_w$ and $R_h$ are determined by $Z_C$ as follows:

$$
R_w = d_w \frac{Z_D}{Z_C}, \quad R_h = d_h \frac{Z_D}{Z_C}.
$$

The expected values of $R_w$ and $R_h$ are given by

$$
E\{R_w\} = d_w \times Z_D \times E\left\{\frac{1}{Z_C}\right\},
$$

$$
E\{R_h\} = d_h \times Z_D \times E\left\{\frac{1}{Z_C}\right\},
$$

since the analysis is carried out given the layer number and hence the discrete zoom factor, $Z_D$. Now, we can apply iterated expectations on (3.4) to yield

$$
E\{P\} = (E\{R_w\} + s_w - 1) (E\{R_h\} + s_h - 1). \quad (3.6)
$$

### 3.2.2 Optimal Slice Size

The average number of bits per pixel for coding the prediction residual of a given resolution layer, denoted by $\eta(s_w, s_h)$, is a function of the slice size $\langle s_w, s_h \rangle$. We also define the number of pixels transmitted per rendered pixel as the relative pixel overhead $\psi(s_w, s_h) = \frac{E\{P\}}{d_w d_h}$, where $E\{P\}$ is given by (3.6).

---

\(^3\)Experimental results appearing later will show that the simplifying assumption about the distribution of $Z_C$ does not hurt the accuracy of the model in predicting the optimal slice size.
Table 3.1: Six high-spatial-resolution video sequences are used for the experiments. Each sequence is coded into a thumbnail and two high-resolution layers with the shown dimensions. Further information about the video sequences can be found in Appendix A.

The optimal slice size minimizes the expected number of bits transmitted per rendered pixel and is given by

\[
(s_{w}^{\text{opt}}, s_{h}^{\text{opt}}) = \arg \min_{(s_{w}, s_{h})} \eta(s_{w}, s_{h}) \times \psi(s_{w}, s_{h}) . \tag{3.7}
\]

One way to obtain the function \(\eta(s_{w}, s_{h})\) is through sample encodings of the prediction residual with varying the slice size. Alternatively, \(\eta(s_{w}, s_{h})\) could also be predicted by an analytical model to reduce the number of sample encodings. Either way, (3.7) can be used to find the optimal slice size.

### 3.2.3 Experimental Results

We now present experimental results to demonstrate that our model predicts the optimal slice size accurately without requiring the capture of user interaction trajectories. In our experiments, we obtain \(\eta(s_{w}, s_{h})\) through a sample encoding of about 30 frames for each tested slice size configuration \((s_{w}, s_{h})\).

We use six high-resolution video sequences for the experiments reported in this dissertation. The width \(\times\) height of the sequences as well as the thumbnail and the high-resolution layers are shown in Table 3.1. Further information about the video sequences can be found in Appendix A. The RoI display is 480\(\times\)240 pixels. We encode...
the thumbnail videos with an intraframe period of 15 frames using two consecutive B frames between anchor frames. For the thumbnail video as well as the high-resolution layers, we report the quality of the encoded video in terms of the peak signal-to-noise ratio (PSNR), which is obtained as follows\(^4\). We first compute the mean squared error (MSE) for every encoded frame by comparing the reconstructed frame from the coded bit-stream versus the original uncoded frame. Next we compute the average MSE over the total frames, \(N\), in the sequence,

\[
\text{Average MSE} = \frac{1}{N} \sum_{n=1}^{N} \text{MSE}[n],
\]

where \(\text{MSE}[n]\) is the MSE corresponding to frame \(n\). The PSNR corresponding to the average MSE for the thumbnail or the high-resolution layer is finally given by

\[
\text{PSNR} = 10 \log_{10} \frac{255^2}{\text{Average MSE}}.
\]

The PSNR and the bit-rate for encoding the thumbnail videos are shown in Table 3.2. The same table also shows the PSNR for the high-resolution layers. Note

\(^4\)The PSNR of the RoI video itself, rendered at the user’s end, is not investigated here but in Chapter 4. Nevertheless, higher PSNR values corresponding to the coded thumbnail and the high-resolution layers lead to better quality RoI video.
that for a given layer of a given sequence, once we choose the QP, the PSNR remains roughly the same although both the coding bit-rate and the transmission bit-rate change with the slice size.

The relative pixel overhead, \( \psi(s_w, s_h) \), is shown for the six sequences in Figs. 3.5 and 3.6. We compare the model prediction against empirical values averaged over 100 user interaction trajectories for each sequence. The trajectories were recorded while interactively viewing the sequence using the graphical user interface described at the beginning of this chapter. Each trajectory starts at a random location with a random zoom factor and is 5 minutes long. The set of frames of the original sequence are looped to play for 5 minutes. The user’s zoom factor, \( Z_C \), is allowed to vary between 1 and 4. The thresholds given by (3.5) determine the high-resolution layer for rendering the RoI.

The cost function, bits transmitted per rendered pixel, is plotted for the six sequences in Figs. 3.7 and 3.8. For a given sequence and resolution layer, the comparison in Figs. 3.5 through 3.8 for different slice sizes is made for the same QP and hence similar PSNR. Although the model predicts the optimal slice size fairly accurately, it can underestimate or overestimate the bit-rate. This is because the popular slices that constitute the salient objects in the video could entail high or low bit-rate compared to the average. Also, the location of the objects can bias the pixel overhead to the high or low side, whereas the model uses the average overhead. For certain zoom factors chosen by the user, the “accessed”/transmitted pixels could be less than the number of rendered pixels. This can be seen in Figs. 3.5 and 3.6, where the relative pixel overhead, \( \psi(s_w, s_h) \), goes below one. Hence, we also compute the zoom-adjusted relative pixel overhead, \( \phi(s_w, s_h) = E \left\{ \frac{R_w R_h}{R_w R_h} \right\} \). This quantity is always greater than one,

\[
\phi(s_w, s_h) = \left( s_w - 1 \right) E \left\{ \frac{1}{R_w} \right\} + 1 \left( s_h - 1 \right) E \left\{ \frac{1}{R_h} \right\} + 1
\]

where

\[
E \left\{ \frac{1}{R_w} \right\} = \frac{E \left\{ Z_C \right\}}{d_w Z_D}, \quad E \left\{ \frac{1}{R_h} \right\} = \frac{E \left\{ Z_C \right\}}{d_h Z_D}.
\]
Figure 3.5: Model prediction versus empirical values for pixels transmitted per rendered pixel, $\psi(s_w, s_h)$, shown for the Cardgame, Brainstorming and Soccer sequences. The empirical values are obtained by averaging over 100 user interaction trajectories for each sequence. The second Y-axis shows the bits per pixel for coding the residual of the high-resolution layer, $\eta(s_w, s_h)$. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
Figure 3.6: Model prediction versus empirical values for pixels transmitted per rendered pixel, $\psi(s_w, s_h)$, shown for the Café, Panel 1 and Panel 2 sequences. The empirical values are obtained by averaging over 100 user interaction trajectories for each sequence. The second Y-axis shows the bits per pixel for coding the residual of the high-resolution layer, $\eta(s_w, s_h)$. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
Figure 3.7: Model prediction versus empirical values for bits transmitted per rendered pixel, shown for the Cardgame, Brainstorming and Soccer sequences. The empirical values are obtained by averaging over 100 user interaction trajectories for each sequence. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
Figure 3.8: Model prediction versus empirical values for bits transmitted per rendered pixel, shown for the Café, Panel 1 and Panel 2 sequences. The empirical values are obtained by averaging over 100 user interaction trajectories for each sequence. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
The zoom-adjusted relative pixel overhead, $\phi(s_w, s_h)$, is shown in Fig. 3.9 for the Brainstorming and Panel 1 sequences. Although not shown here, we observed that the model prediction is close to the empirical values for all six sequences. Thus, the analysis presented in Section 3.2 enables estimating various quantities related to “accessed” portions from the scene representation without recording user interaction trajectories and measuring these quantities from long bit-streams coded for various slice sizes. This helps system dimensioning of an interactive video transmission system.

Figure 3.9: Model prediction versus empirical values for zoom-adjusted relative pixel overhead, $\phi(s_w, s_h)$, shown for the Brainstorming and Panel 1 sequences. The empirical values are obtained by averaging over 100 user interaction trajectories. The second Y-axis shows the bits per pixel for coding the residual of the high-resolution layer, $\eta(s_w, s_h)$. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
Figure 3.10: Improvement based on background extraction: Each high-resolution layer frame has two references to choose from, the frame obtained by upsampling the reconstructed thumbnail frame and the background frame from the same layer in the background pyramid.

### 3.3 Background Extraction

The coding scheme proposed in Section 3.1 exploits temporal correlation by performing motion compensation among successive frames of the thumbnail video. Temporal prediction among successive frames of the high-resolution layers is avoided to enable efficient random access. Although it enables efficient random access, upward prediction using the reconstructed thumbnail frames might result in substantial residual energy for high spatial frequencies. In this section, we propose creating a background frame \[64,146\] for each high-resolution layer and employing long-term memory motion-compensated prediction (LTM MCP) \[239\] to exploit the correlation between this frame and each high-resolution frame to be encoded. The background frame\(^5\) is intracoded. As shown in Fig. 3.10, high-resolution P slices have two references to choose from, upward prediction and the background frame. If a transmitted high-resolution P slice refers to the background frame, then relevant I slices from the background frame are transmitted only if they have not been transmitted earlier. This is different from \[24\], in which the encoder uses only those parts of the background for prediction that exist in the decoder’s multi-resolution background pyramid. The encoder mimics

---

\(^5\)There is a large body of literature dealing with motion or object detection based on background subtraction \[54,126,144,169,199,230\]. These works also have the notion of a background frame. However, in many of these works, the background frame is updated in successive frame-intervals.
the decoder in [24], which builds a background pyramid out of all previously received frames. Background extraction algorithms as well as detection and update of changed background portions have been previously studied, for example in [81], and are not the focus of our work.

Since a moving camera might hamper the spatial browsing experience, the camera is static in our sequences. A simple temporal median operator [146] yields a plausible background frame. Out of the first 150 frames, we include every fifth frame for the median operation. Fig. 3.11 shows the result for Cardgame, Brainstorming, and Soccer. Although some stationary objects remain in the background frame, this helps the coding efficiency. In our experiments, the background frame is not updated after its creation at the start. This is typical with a static camera. For example, in a soccer game, the background typically changes due to illumination change, which happens infrequently. The background frame is intracoded with the same slice structure as the other frames from the layer.
Table 3.3: Slice sizes employed for experiments in Figs. 3.12 and 3.13 for comparing coding efficiency with and without the improvement based on background extraction. The slice width and slice height in number of pixels are denoted by \( s_w \) and \( s_h \) respectively.

### 3.3.1 Storage Reduction

We compare the reduction in storage by plotting the number of bits per pixel for coding an enhancement layer with and without the background frame. The slice sizes shown in Table 3.3 are employed for the comparison and the results are plotted in Figs. 3.12 and 3.13. Coding bit-rate reduction of up to 85% was observed. We observed similar percentage bit-rate reduction across several slice sizes. The slice sizes, shown in Table 3.3, are either optimal or close to optimal in terms of mean transmission bit-rate, which we report next.

### 3.3.2 Transmission Bit-Rate Reduction

The improvement based on employing the background frame implies that some background I slices have to be transmitted in addition to relevant high-resolution P slices. For the high-resolution layers, Fig. 3.14 shows the number of transmitted I slices from the background pyramid and the number of transmitted P slices. It shows the numbers for a single user interaction trajectory. For the first frame of the streaming session, roughly equal numbers of I and P slices are transmitted. Subsequently, I slices need to be transmitted sporadically and generally fewer in number than at the start. As shown in Fig. 3.15, when averaged over 100 trajectories, the profiles of the transmitted I and P slices appear smoother; the number of P slices is almost constant.
Figure 3.12: Coding bit-rate reduction through background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP), shown for the Cardgame, Brainstorming, and Soccer sequences. The slice sizes employed for the experiment are shown in Table 3.3.

and matches the expected number of transmitted P slices that can be computed from analysis similar to Section 3.2.1. The average number of transmitted I slices is highest at the start and is about 1–2% of the number of transmitted P slices thereafter. Finally, as shown in Figs. 3.16 and 3.17, the coding bit-rate reduction due to the proposed improvement leads to significant reduction of transmission bit-rate.
Figure 3.13: Coding bit-rate reduction through background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP), shown for the Café, Panel 1, and Panel 2 sequences. The slice sizes employed for the experiment are shown in Table 3.3.

We also performed experiments, where the first frame of the high-resolution layer was stored in the long-term reference buffer rather than computing a background frame as stated above. The performance was found to be close to that reported here for the background frame. Specifically, only about 5–10% less coding or transmission bit-rate reduction was observed compared to the background frame.
Figure 3.14: Number of I and P slices transmitted over the streaming session, when background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP) are employed. The data are plotted for a single user interaction trajectory. Slice sizes are shown in Table 3.3.

Figure 3.15: Number of I and P slices transmitted over the streaming session, when background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP) are employed. The numbers are averaged over 100 user interaction trajectories. Slice sizes are shown in Table 3.3.
<table>
<thead>
<tr>
<th>Sequence</th>
<th>Layer 1 slice size</th>
<th>Layer 2 slice size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardgame</td>
<td>( 4 \times 16 )</td>
<td>( 8 \times 8 )</td>
</tr>
<tr>
<td>Brainstorming</td>
<td>( 4 \times 16 )</td>
<td>( 8 \times 8 )</td>
</tr>
<tr>
<td>Soccer</td>
<td>( 6 \times 4 )</td>
<td>( 6 \times 4 )</td>
</tr>
<tr>
<td>Café</td>
<td>( 6 \times 4 )</td>
<td>( 6 \times 4 )</td>
</tr>
<tr>
<td>Panel 1</td>
<td>( 6 \times 4 )</td>
<td>( 6 \times 4 )</td>
</tr>
<tr>
<td>Panel 2</td>
<td>( 6 \times 4 )</td>
<td>( 6 \times 4 )</td>
</tr>
</tbody>
</table>

Table 3.4: Slice sizes chosen for experiments in later chapters. The slice width and slice height in number of pixels are denoted by \( s_w \) and \( s_h \) respectively.

### 3.3.3 Modeling Mean Transmission Bit-Rate

We model the bits transmitted per rendered pixel as before (as in Section 3.2). However, for simplicity, the cost of transmitting I slices is counted in the coding bit-rate, \( \eta(s_w, s_h) \), but not in the number of pixels transmitted per rendered pixel, \( \psi(s_w, s_h) \). As shown in Figs. 3.18 and 3.19, the model matches closely with the empirical values for all six video sequences. It should be noted that the change in the optimal slice size after employing the background frame is small, and the slice size that is optimal for the earlier scheme still yields a mean transmission bit-rate very close to that corresponding to the new optimal slice size. For our experiments in later chapters, we choose the slice sizes shown in Table 3.4.

### 3.4 Chapter Summary

Spatial-random-access-enabled video compression limits the load of encoding to a one-time encoding of the video irrespective of the number of users to be served, whether synchronously or asynchronously. Random access is beneficial for streaming both live content as well as pre-stored content to multiple users. Even when the video is played back from locally stored media, a different RoI trajectory has to be accommodated each time a user watches the content. Spatial random access allows the video player to selectively decode relevant regions only.
For supporting zoom control, the video is encoded with multiple resolution layers. Since the client can additionally resample received data from a given resolution layer, continuous zoom can be supported despite storing limited number of layers at the server. This chapter presents a spatial-random-access-enabled video coding scheme with the following benefits:

- The scheme allows the receiver to start decoding a new region, with an arbitrary zoom factor, during any frame-interval instead of having to wait for the end of a GOP or having to transmit extra slices from previous frames.

- The slices needed to render the current RoI are fully determined by the current RoI location rather than being conditional on the previous RoI path.

- Unlike the Gaussian pyramid which encodes the resolution layers independently, the proposed scheme exploits redundancy between the base layer and the higher resolution layers.

- Motion compensation, performed on the base layer, exploits redundancy among successive frames, while the relatively small search range associated with coding high-resolution P slices limits the computational complexity.

- Since each high-resolution layer employs the base layer for prediction and no other high-resolution layers, this provides efficient random access, and also the encoding of the high-resolution layers can be parallelized.

This chapter also shows how to estimate the slice size that minimizes the mean transmission bit-rate over an IRoI streaming session without recording any user interaction trajectories. Both the coding bit-rate as well as the transmission bit-rate can be further reduced by employing background extraction and long-term memory motion-compensated prediction. Storage reduction of up to 85% was observed due to the background frame. These benefits can be had while maintaining efficient spatial random access.
Figure 3.16: Transmission bit-rate is reduced after employing background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP); here shown for the two layers of Cardgame, Brainstorming and Soccer sequences. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively. Transmission bit-rate values are obtained by counting bits required to transmit relevant high-resolution slices. The values are averaged over 100 user interaction trajectories.
Figure 3.17: Transmission bit-rate is reduced after employing background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP); here shown for the two layers of Café, Panel 1 and Panel 2 sequences. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively. Transmission bit-rate values are obtained by counting bits required to transmit relevant high-resolution slices. The values are averaged over 100 user interaction trajectories.
Figure 3.18: Model prediction versus empirical values for bits transmitted per rendered pixel, shown for the Cardgame, Brainstorming and Soccer sequences, encoded using background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP). The empirical values are obtained by averaging over 100 user interaction trajectories. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
Figure 3.19: Model prediction versus empirical values for bits transmitted per rendered pixel, shown for the Café, Panel 1 and Panel 2 sequences, encoded using background extraction (BE) and long-term memory motion-compensated prediction (LTM MCP). The empirical values are obtained by averaging over 100 user interaction trajectories. The slice width and slice height in number of pixels are denoted by $s_w$ and $s_h$ respectively.
Chapter 4

P2P Live Multicast of IRoI Video

In order to allow the IRoI video streaming system to scale to large numbers of users, it is important to limit both the encoding load as well as the bandwidth required at the server. The video compression approach presented in Chapter 3 limits the encoding load at the server irrespective of the number of users. The goal of this chapter is to limit the bandwidth required from dedicated servers. We assume that several users concurrently watch the video, however, each user enjoys independent control of the region to watch. In this chapter, we propose a peer-to-peer (P2P) streaming system that can achieve live multicast of IRoI video\(^1\). Section 4.1 provides an overview of the system. We describe the distributed P2P protocol in Section 4.2 and analyze the server load reduction due to this approach in Section 4.3. Experimental results are presented in Section 4.4.

4.1 Overview of System Architecture

We employ the compression scheme proposed in Chapter 3, illustrated in Figs. 3.2 and 3.10, for compressing the thumbnail video and the high-resolution layers. IRoI P2P aims to exploit overlaps among the users’ RoIs. For example, Fig. 4.1 illustrates RoIs of three users leading to the highlighted tiles being concurrently required by more

\(^1\)The IRoI P2P system was first presented by us in [154]. Following this, we presented some improvements in [153,157].
Figure 4.1: Example illustrating RoIs of three users within the multi-resolution video representation. The shaded slices represent tiles that are currently needed by more than one client and represent the “overlaps” exploited by the IRoI P2P system.

Figure 4.2: The server distributes the thumbnail overview and the high-resolution tiles on different multicast trees. Peer-hosts attach to the appropriate multicast trees depending on their individual RoIs.

than one user. Although not highlighted, the thumbnail video, treated as one tile, is required by all users. We propose a distributed P2P protocol that allows peers to relay relevant tiles to each other. This approach harnesses the forwarding capacities of peers, thus drastically reducing the bandwidth required from dedicated servers. The
proposed P2P protocol builds multicast trees rooted at the server for distributing video data corresponding to different tiles. As shown in Fig. 4.2, each high-resolution tile, also called enhancement layer tile, is delivered using a separate multicast tree. Similarly, a dedicated multicast tree delivers the thumbnail video, called the base layer. Each peer subscribes to the base layer at all times and additionally some enhancement layer tiles that are required to render the RoI. Peers also dynamically unsubscribe tiles that are no longer required to render the RoI. The P2P protocol, which allows peers to receive and relay relevant tiles, is described in Section 4.2 below.

Apart from reducing the required server bandwidth, our design strives to provide low latency of interaction. Accordingly, a new RoI is rendered immediately upon user input, without waiting for new data to arrive. In order to render the RoI instantly despite the delay of packets, the client predicts the user’s future RoI d frame-intervals (or D seconds) in advance and, if required, initiates connection to new trees beforehand. Also, the server advances the base layer transmission slightly compared to the transmission of the enhancement layer tiles. This way peers can buffer some base layer frames as well as request retransmissions of lost base layer packets. The stringent latency constraint associated with interactive RoI makes retransmissions of enhancement layer packets difficult. Recall that the base layer can be used to fill in missing parts while rendering the RoI. The error-concealed parts might appear blurry but the user experiences low-latency RoI control.

Let us assume that the base layer transmission is advanced by S seconds compared to the enhancement layer tiles at the source. Let \( T_{B,n} \) denote the time when frame \( n \) of the base layer emanates from the server. For some enhancement layer tile \( X \), let the time when the slice corresponding to frame \( n \) emanates from the server, be denoted by \( T_{X,n} = T_{B,n} + S \). Finally, a pre-roll delay of \( C \) seconds is used to account for delay-jitter in the arrival of the enhancement layer packets; i.e., at time \( T_{X,n} + C \), both the thumbnail and the RoI are rendered for frame \( n \).

The parameters of this design are depicted in Fig. 4.3. For tile \( X \), let \( X_{e2e} \) denote the worst-case end-to-end delay, i.e., the longest time it takes for a packet of tile \( X \) to

\(^2\)Techniques of RoI prediction and pre-fetching are described in Appendix B.
Figure 4.3: The server advances the transmission of the base layer by $S$ seconds. When the client renders frame $n$ (at time $T_{X,n} + C$), it predicts the RoI in frame $n+d$ and initiates subscription to new trees. The parameters of the design help to render the RoI instantly upon user input. The diagram is drawn for $S = D + B_{e2e}$.

reach the client from the server. Notice that $C > X_{e2e}$ helps avoid late-losses. Similarly, let $B_{e2e}$ denote the worst-case end-to-end delay for the base layer. Irrespective of the value of $C$, choosing $S \geq D + B_{e2e}$ ensures that $d$ future frames of the base layer are available in the client’s buffer at the time of displaying the current frame; i.e., up to frame $n+d$ available at (or before) time $T_{X,n} + C$.

Note that when frame $n$ is displayed at time $T_{X,n} + C$, the RoI for frame $n+d$ can be predicted by observing mouse moves up to frame $n$ and possibly also by analyzing the motion of objects in the buffered thumbnail frames up to frame $n+d$ [152, 156]. Consequently, the join request for new tiles required starting with frame $n+d$ cannot be sent earlier than $T_{X,n} + C$ or $T_{X,n+d} + C - D$. The chosen lookahead $d$ should thus be sufficiently large such that it allows joining new trees and receiving the required new slices before their display deadline $T_{X,n+d} + C$. In our experiments, $S$, $C$ and $D$ are set to 2, 0.8 and 1.6 seconds respectively.

### 4.2 Distributed P2P Protocol

The P2P approach hinges on peers being able to relay data to each other in real time. The main challenge is that user interaction determines on-the-fly which slices are needed by which peers. The P2P overlay needs to adapt quickly and in a distributed manner, i.e., peers take most of the action necessary for acquiring the data they need,
without much central intervention. The second challenge is that peers can switch off randomly, taking away the resources they bring with them. The IRoI P2P protocol builds on top of the Stanford peer-to-peer multicast (SPPM) protocol [188,190] which operates in tree-push manner. SPPM was originally developed for P2P video streaming without any pan/tilt/zoom functionality. Nevertheless, we can leverage SPPM for building and maintaining multicast trees in a distributed manner. Hereunder, we explain how the overlay topology is adapted in response to changing RoIs of the users as well as peers turning off abruptly.

### 4.2.1 Subscription of Multicast Trees

The server maintains a database of tiles that each peer is currently subscribed to. Whenever the RoI prediction indicates a change of RoI, the peer sends an RoI-switch request to the server. This consists of the top-left and bottom-right tile IDs of the old RoI as well as the new RoI. In response to the RoI-switch request, the server sends a list of potential parents for every new multicast tree that the peer needs to subscribe to. Corresponding to every multicast tree, there is a limit on the number of peers the server can directly serve, and the server includes itself in the list if this quota is not yet full. The server also updates its database assuming that the peer will be successful in updating its subscriptions. After receiving the list from the server, the peer sends Probe messages to potential parents for every new multicast tree it needs to join. If it receives a positive Probe Reply, it sends an Attach Request for that tree. If it still fails to connect, the peer checks for positive replies from other probed peers and tries attaching to one of them. Once connected to a multicast tree corresponding to a tile, the peer checks if it has previously received the corresponding background I slice. If it has not then the peer obtains the background I slice from one of the peers in the list or the server.

### 4.2.2 Soft Handoff

When RoI prediction indicates a change of RoI, the peer waits a while before sending Leave messages to its parents on trees that its RoI no longer requires. This ensures
Figure 4.4: Cumulative distribution function (cdf) of tile subscription durations computed for six high-resolution video sequences. One hundred user interaction trajectories representing as many peers were recorded for each video sequence. Each trajectory is 1000 seconds long, whereas peer lifetimes themselves are exponentially distributed with an average of 90 seconds. After switching off for a random interval, a peer can join the session again. The off-duration is exponentially distributed with an average of 10 seconds. Note that switching tiles due to RoI change is the dominant factor governing the distribution of tile subscription durations.

that tiles are not unsubscribed prematurely. On the other hand, the peer sends Leave messages to its children immediately but keeps forwarding data as long as it receives data from its parent. Upon receiving Leave messages, the respective children request potential parents’ lists from the server for the respective multicast trees and try finding new parents. The delay in unsubscribing is chosen such that the children experience a smooth handoff from old parent to new parent. In rare cases, a child peer takes longer than the handoff deadline to find a new parent and experiences disruption on that tree.

The cumulative distribution function (cdf) of tile subscription durations, shown in Fig. 4.4, indicates how long peers attach to a multicast tree. The cdf is computed over all six high-resolution video sequences and the average tile subscription duration is 14.9 seconds. The low value of subscription duration indicates high amount churn in the memberships of the multicast trees. The transient memberships of the multicast trees indicate that the associations among peers are highly dynamic, implying that
Table 4.1: Average tile subscription durations for six high-resolution video sequences. The values are computed from user interaction trajectories of 100 peers. Peer lifetimes themselves are exponentially distributed with an average of 90 seconds.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Total tiles</th>
<th>Avg. tile subscription duration [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardgame</td>
<td>141</td>
<td>10.6</td>
</tr>
<tr>
<td>Brainstorming</td>
<td>141</td>
<td>13.1</td>
</tr>
<tr>
<td>Soccer</td>
<td>382</td>
<td>16.5</td>
</tr>
<tr>
<td>Café</td>
<td>431</td>
<td>14.6</td>
</tr>
<tr>
<td>Panel 1</td>
<td>431</td>
<td>18.3</td>
</tr>
<tr>
<td>Panel 2</td>
<td>431</td>
<td>14.4</td>
</tr>
</tbody>
</table>

Soft handoff is critical for avoiding data disruption. Table 4.1 shows the total number of tiles and the average tile subscription duration for each of the six sequences. Each sequence is encoded using two high-resolution layers apart from the thumbnail video. The total number of tiles includes the tiles of the two high-resolution layers and the thumbnail video which is counted as one tile.

### 4.2.3 Tree Maintenance

In addition to leaving multicast trees gracefully, peers can also switch off altogether leading to ungraceful departures. If a child peer does not receive data for a particular tree for a timeout interval, it assumes that the parent is unavailable and tries to rejoin the tree by enquiring about other potential parents. To monitor the online status of parents, peers send *Hello* messages regularly to their parents and the parents reply back. Since most tree disconnections are graceful and occur due to RoI change, the interval for sending *Hello* messages can be large to limit the protocol overhead. Similar to SPPM, a loop-avoidance mechanism on individual distribution trees ensures that a descendant is not chosen as a parent [18, 167, 188, 190, 192, 194]. For additional details on peer state transitions and timeouts associated with sending and receiving control messages, the reader may refer to [195].
4.3 Analysis of Server Load

With traditional client-server unicast, the server itself serves each client. On the other hand, the IRoI P2P approach implies that peers relay tiles to each other, thus reducing dedicated bandwidth required from the server. The goal of this section is to analyze the server load for each approach and estimate the bit-rate savings achievable through the P2P approach compared to traditional client-server unicast.

4.3.1 Server Load for Client-Server Unicast

We first estimate the number of tiles belonging to each resolution layer streamed by the server and then estimate the total bit-rate by considering the average tile bit-rates of each layer. Let us denote the layer-specific average number of tiles required by a client by \( \kappa_{Z_D} \), where the layer is identified by \( Z_D \). The value of \( \kappa_{Z_D} \) depends on the slice size and the size of the RoI portion of the display. It can be computed through analysis similar to Section 3.2. For the given layer, the expected number of tiles required by a client can be written as

\[
\kappa_{Z_D} = \frac{E\{R_w\} + s_w - 1}{s_w} \times \frac{E\{R_h\} + s_h - 1}{s_h},
\]

where,

\[
E\{R_w\} = d_w \times Z_D \times E\left\{ \frac{1}{Z_C} \right\};
\]

\[
E\{R_h\} = d_h \times Z_D \times E\left\{ \frac{1}{Z_C} \right\}.
\]

The symbols used to denote various quantities are as in Chapter 3. For convenience, we repeat their description. As before, \( s_w \) and \( s_h \) denote slice width and slice height respectively. The width and height of the RoI portion of the display are denoted by \( d_w \) and \( d_h \) respectively. The number of columns and the number of rows from the high-resolution layer corresponding to the RoI are denoted by random variables \( R_w \) and \( R_h \) respectively. The continuous zoom factor is denoted by random variable \( Z_C \).
We estimate the average number of tiles per client, \( \kappa \), by assuming that \( Z_C \) is uniformly distributed over the entire zoom range. From (3.5), \( \Pr \{ Z_D = 1 \} = \frac{1}{6} \) and \( \Pr \{ Z_D = 2 \} = \frac{5}{6} \). Assuming that each client receives the thumbnail video in addition to high-resolution tiles,

\[
\kappa = 1 + \frac{1}{6} \kappa_{Z_D=1} + \frac{5}{6} \kappa_{Z_D=2}. \tag{4.2}
\]

The average bit-rate per client, \( R_{\text{client}} \), is given by

\[
R_{\text{client}} = R_{\text{thumbnail}} + \frac{1}{6} \kappa_{Z_D=1} R_{Z_D=1} + \frac{5}{6} \kappa_{Z_D=2} R_{Z_D=2}, \tag{4.3}
\]

where \( R_{Z_D} \) denotes the layer-specific average bit-rate per tile. For example:

- Consider the Cardgame sequence, for which we employ the following slice sizes in our experiments later: \((s_w = 64, s_h = 256)\) pixels for layer 1 and \((s_w = 128, s_h = 128)\) pixels for layer 2. The layer-specific values, \( \kappa_{Z_D} \), are 12.2 tiles and 10 tiles for layer 1 and layer 2 respectively\(^3\). Applying (4.2), the overall average, \( \kappa \), is 11.4 tiles per client. The layer-specific average bit-rates per tile are \( R_{Z_D=1} = 25.0 \) kbps and \( R_{Z_D=2} = 19.7 \) kbps. As shown in Table 3.2, \( R_{\text{thumbnail}} = 162 \) kbps. Applying (4.3), the average bit-rate per client, \( R_{\text{client}} \), is about 380 kbps.

- Consider the Soccer sequence, for which we employ the following slice sizes in our experiments later: \((s_w = 96, s_h = 64)\) pixels for both layer 1 and layer 2. The layer-specific values, \( \kappa_{Z_D} \), are 20.3 tiles and 19.2 tiles for layer 1 and layer 2 respectively\(^3\). Applying (4.2), the overall average, \( \kappa \), is 20.4 tiles per client. The layer-specific average bit-rates per tile are \( R_{Z_D=1} = 43.8 \) kbps and \( R_{Z_D=2} = 35.3 \) kbps. As shown in Table 3.2, \( R_{\text{thumbnail}} = 355 \) kbps. Applying (4.3), the average bit-rate per client, \( R_{\text{client}} \), is about 1.1 Mbps.

The server load, \( R_{\text{unicast}} \), increases linearly with the number of clients. The rate at which the load increases is given by \( R_{\text{client}} \). Next we estimate server load for the IRoI P2P approach.

---

\(^3\)We make the same assumptions about \( d_w, d_h \) and \( Z_C \) as in Chapter 3; i.e., \( d_w = 480, d_h = 240 \), and \( Z_C \) uniformly distributed.
CHAPTER 4. P2P LIVE MULTICAST OF IROI VIDEO

4.3.2 Server Load for Peer-to-Peer Multicast

For simplicity, let us assume that for each tile, the server supports at most one direct child. Thus, peers other than the direct child have to obtain the tile data in a peer-to-peer manner if those data are required for their current RoIs. We first define the following quantities:

- \( S_{Z_D=1} \): Set of high-resolution tiles associated with \( Z_D = 1 \). Let \(|S_{Z_D=1}| = S_{Z_D=1}\).
- \( P_{Z_D=1} \): Set of peers being served high-resolution tiles associated with \( Z_D = 1 \). Let \(|P_{Z_D=1}| = P_{Z_D=1}\).
- \( S_{Z_D=2} \): Set of high-resolution tiles associated with \( Z_D = 2 \). Let \(|S_{Z_D=2}| = S_{Z_D=2}\).
- \( P_{Z_D=2} \): Set of peers being served high-resolution tiles associated with \( Z_D = 2 \). Let \(|P_{Z_D=2}| = P_{Z_D=2}\).
- \( S \): Set of all tiles. Let \(|S| = S_t\). Note that \( S_t = 1 + S_{Z_D=1} + S_{Z_D=2}\) since the thumbnail video constitutes one tile.
- \( P \): Set of all peers. Let \(|P| = P_t\). Note that \( P_t = P_{Z_D=1} + P_{Z_D=2}\).
- \( X_{ij} \): Bernoulli random variable which is 1 if peer \( i \in P_{Z_D=1} \) needs tile \( j \in S_{Z_D=1} \). Let \( \Pr\{X_{ij} = 1\} = p_j, \forall i \in P_{Z_D=1} \).
- \( X_{uv} \): Bernoulli random variable which is 1 if peer \( u \in P_{Z_D=2} \) needs tile \( v \in S_{Z_D=2} \). Let \( \Pr\{X_{uv} = 1\} = p_v, \forall u \in P_{Z_D=2} \).
- \( X_{j}^{Z_D=1} \): Indicator random variable which is 1 if at least one out of \( P_{Z_D=1} \) peers needs tile \( j \).
- \( X_{v}^{Z_D=2} \): Indicator random variable which is 1 if at least one out of \( P_{Z_D=2} \) peers needs tile \( v \).
\( N_S \): Random variable denoting the number of tiles streamed by the server. With the assumption of at most one direct child per tile, \( N_S \) can be at most equal to the total number of tiles in the coded representation.

Assuming each peer subscribes to the thumbnail tile at all times,

\[
N_S = 1 + \sum_{j \in S_{ZD=1}} X_{j}^{ZD=1} + \sum_{v \in S_{ZD=2}} X_{v}^{ZD=2}.
\]

The expected number of tiles streamed by the server is given by

\[
E \{ N_S \} = 1 + \sum_{j \in S_{ZD=1}} E \{ X_{j}^{ZD=1} \} + \sum_{v \in S_{ZD=2}} E \{ X_{v}^{ZD=2} \}
\]

\[
= 1 + \sum_{j \in S_{ZD=1}} \Pr \{ X_j^{ZD=1} = 1 \} + \sum_{v \in S_{ZD=2}} \Pr \{ X_v^{ZD=2} = 1 \}
\]

\[
= 1 + \sum_{j \in S_{ZD=1}} 1 - \Pr \{ X_j^{ZD=1} = 0 \} + \sum_{v \in S_{ZD=2}} 1 - \Pr \{ X_v^{ZD=2} = 0 \}
\]

\[
= 1 + \sum_{j \in S_{ZD=1}} 1 - (1 - p_j)^{pZD=1} + \sum_{v \in S_{ZD=2}} 1 - (1 - p_v)^{pZD=2}. \tag{4.4}
\]

This uses the fact that random variables \( X_{ij}, \forall i \in P_{ZD=1} \) are independent given \( p_j \). They represent independent coin tosses with a coin having bias \( p_j \). For any peer, say peer \( i \), we do not assume independence between random variables \( X_{ij}, \forall j \in S_{ZD=1} \) since the knowledge of \( X_{ij} \) affects the probability of peer \( i \) subscribing tiles neighboring to tile \( j \). Similarly, random variables \( X_{uv}, \forall u \in P_{ZD=2} \) are independent given \( p_v \). They represent independent coin tosses with a coin having bias \( p_v \). For any peer, say peer \( u \), we do not assume independence between random variables \( X_{uv}, \forall v \in S_{ZD=2} \) since the knowledge of \( X_{uv} \) affects the probability of peer \( u \) subscribing tiles neighboring to tile \( v \).

The expected number of high-resolution tiles needed by peer \( i \in P_{ZD=1} \), denoted by \( \kappa_{ZD=1} \), can be obtained as in Section 4.3.1, but it is also given by

\[
\kappa_{ZD=1} = \sum_{j \in S_{ZD=1}} E \{ X_{ij} \} = \sum_{j \in S_{ZD=1}} p_j. \tag{4.5}
\]
Similarly, the expected number of high-resolution tiles needed by peer $u \in P_{ZD=2}$, denoted by $\kappa_{ZD=2}$, can be obtained as in Section 4.3.1, but it is also given by

$$\kappa_{ZD=2} = \sum_{j \in S_{ZD=2}} E \{ X_{uv} \} = \sum_{v \in S_{ZD=2}} p_v. \quad (4.6)$$

Under the constraints $\sum_{j \in S_{ZD=1}} p_j = \kappa_{ZD=1}$ and $\sum_{v \in S_{ZD=2}} p_v = \kappa_{ZD=2}$, it can be shown\footnote{Form a Lagrangian cost function $J = 1 + \sum_{j \in S_{ZD=1}} (1 - p_j) P_{ZD=1} + \sum_{v \in S_{ZD=2}} (1 - p_v) P_{ZD=2} + \lambda \left( \kappa_{ZD=1} - \sum_{j \in S_{ZD=1}} p_j \right) + \nu \left( \kappa_{ZD=2} - \sum_{v \in S_{ZD=2}} p_v \right)$ combining the cost function given by (4.4) and the constraints given by (4.5) and (4.6). Set $\frac{\partial J}{\partial p_j} = 0, \forall j \in S_{ZD=1}$. Similarly, set $\frac{\partial J}{\partial p_v} = 0, \forall v \in S_{ZD=2}$. Also, set $\frac{\partial J}{\partial \lambda} = 0$ and $\frac{\partial J}{\partial \nu} = 0$ to obtain (4.7).} that

$$\max_{p_j, \forall j \in S_{ZD=1}, \ p_v, \forall v \in S_{ZD=2}} E \{ N_S \} = 1 + \sum_{j \in S_{ZD=1}} 1 - \left( 1 - \frac{\kappa_{ZD=1}}{S_{ZD=1}} \right)^{P_{ZD=1}}$$

$$+ \sum_{v \in S_{ZD=2}} 1 - \left( 1 - \frac{\kappa_{ZD=2}}{S_{ZD=2}} \right)^{P_{ZD=2}}$$

$$= 1 + S_{ZD=1} \left[ 1 - \left( 1 - \frac{\kappa_{ZD=1}}{S_{ZD=1}} \right)^{P_{ZD=1}} \right]$$

$$+ S_{ZD=2} \left[ 1 - \left( 1 - \frac{\kappa_{ZD=2}}{S_{ZD=2}} \right)^{P_{ZD=2}} \right] \quad (4.7)$$

The maximum occurs when $p_j = \frac{\kappa_{ZD=1}}{S_{ZD=1}}, \forall j \in S_{ZD=1}$, and $p_v = \frac{\kappa_{ZD=2}}{S_{ZD=2}}, \forall v \in S_{ZD=2}$; i.e., when the tiles corresponding to $Z_D = 1$ are equally popular\footnote{“Equally popular” means equally likely to be required by a client.} and the tiles corresponding to $Z_D = 2$ are equally popular. Finally, the server load, $R_{P2P}$, is estimated as

$$R_{P2P} = R_{thumbnail} + S_{ZD=1} \left[ 1 - \left( 1 - \frac{\kappa_{ZD=1}}{S_{ZD=1}} \right)^{P_{ZD=1}} \right] R_{ZD=1}$$

$$+ S_{ZD=2} \left[ 1 - \left( 1 - \frac{\kappa_{ZD=2}}{S_{ZD=2}} \right)^{P_{ZD=2}} \right] R_{ZD=2}. \quad (4.8)$$
For the Cardgame and Soccer sequences, Fig. 4.5 shows the number of tiles streamed by the server, estimated by theoretical analysis as well as measured from user interaction trajectories. For the same two sequences, the corresponding bit-rate required at the server is shown in Fig. 4.6. For the client-server unicast approach, the server load increases linearly with the number of clients. On the other hand, for the
P2P approach, the server load increases with the number of peers when the peer population is small, however, the server load saturates eventually. In theory, the number of streamed tiles saturates to the total number of tiles in the coded representation, $S_t$. In practice, some tiles are not in demand most of the time and hence the number of streamed tiles saturates to a value less than $S_t$. 

---

**Figure 4.7:** Number of tiles streamed by the server shown for the *Cardgame* sequence. (a) For P2P delivery, the server supports at most 2 direct children per tile. (b) For P2P delivery, the server supports at most 3 direct children per tile.

**Figure 4.8:** Server load shown for the *Cardgame* sequence. (a) For P2P delivery, the server supports at most 2 direct children per tile. (b) For P2P delivery, the server supports at most 3 direct children per tile.
For simplicity, it was assumed above that the server supports at most one direct child per tile. The above analysis can be extended to account for higher number of direct children per tile. The indicator random variables $X_j^{Z_D=1}$ and $X_v^{Z_D=2}$ need to be replaced accordingly. For example, when allowing up to two direct children, $X_j^{Z_D=1}$ should be replaced by an indicator random variable that takes on the following values:

\[
X_j^{Z_D=1} = \begin{cases} 
0 & \text{no peer needs tile } j \\
1 & \text{exactly one peer needs tile } j \\
2 & \text{at least two peers need tile } j.
\end{cases}
\]

If the server supports up to three direct children, the new random variable $X_j^{Z_D=1}$ takes on the following values:

\[
X_j^{Z_D=1} = \begin{cases} 
0 & \text{no peer needs tile } j \\
1 & \text{exactly one peer needs tile } j \\
2 & \text{exactly two peers need tile } j \\
3 & \text{at least three peers need tile } j.
\end{cases}
\]

and so on for higher numbers of direct children. The same modification applies to random variable $X_v^{Z_D=2}$. For the Cardgame sequence, the number of streamed tiles is shown in Fig. 4.7 when the server supports more than one direct child with the P2P approach. As before, the number of streamed tiles increases with the number of peers and saturates eventually. For example, when the server supports up to two direct children, the number of streamed tiles saturates to $2S_i$ slices theoretically. The server load for the Cardgame sequence is shown in Fig. 4.8 when the server supports more than one direct child with the P2P approach.

Although the P2P protocol detects ungraceful departures of parent peers and child peers, the system displays higher robustness against peer churn when the server supports more direct children. However, for supporting more direct children, correspondingly higher bandwidth has to be provisioned at the server. The trade-off
associated with robustness vs. number of direct children is explored later in Chapter 5. In the next section, we fix the number of direct children and gauge how well peers receive and relay content in real time.

4.4 Experimental Results

We evaluate the performance of the IRoI P2P system by carrying out experiments using a network simulator called NS-2 [7]. The P2P protocol is run by the simulated peers and the simulated server. Peers take necessary action according to their individual RoI trajectories.

4.4.1 Experimental Setup

Simulated Network

We create a tree topology for simulating the network. Figure 4.9 illustrates the topology formed by the physical links constituting the network. The peers as well as the source are placed on randomly chosen edge nodes of the backbone network. The backbone links are sufficiently provisioned with high capacity. The propagation delay of each network link is set to 5 ms, thus resulting in propagation delays of about 50 ms between two peers. According to a study conducted in 2003-2004, this roughly corresponds to the peer population spread across a continent [210]. We assume that
the UDP/IP protocol stack is employed underneath the application layer. Our P2P protocol is implemented at the application layer and we ignore any NAT or firewall issue which may limit connectivity or drop this kind of traffic.

**Simulated Host**

Peers have limited uplink and downlink capacities. The distribution of peer uplink capacities was modeled after a recent study \[166\]\(^6\). The average peer uplink capacity was set to 2 Mbps, which is slightly higher than the 1.7 Mbps average reported in [166]. The cdf of peer uplink capacity is shown in Fig. 4.10. We assume that each peer has measured and knows its uplink capacity accurately. The downlink capacity of the peers was set to 2 Mbps. We varied the uplink capacity of the source in our experiments. The downlink capacity of the source was set to 5 Mbps.

Peer arrival and departure were modeled as follows. A flash crowd was simulated at the beginning such that all peers join the system within the first five seconds. After this period, peers leave ungracefully and rejoin the system. Peer lifetimes are exponentially distributed with an average time of 90 seconds. Departed peers rejoin the system after remaining “off” for 10 seconds on average. The “off” time is also exponentially distributed. Prior studies, for example [36, 202], have found that the

\(^6\)The study is based on analyzing user statistics during P2P live video multicast of the Electronic Sports World Cup (ESWC) 2008. ESWC is an international professional gaming championship [4].
distribution of peer lifetimes is close to exponential. The average peer lifetime depends on the content; for example, it is longer for more popular content, and it tends to be shorter when there is a wider choice of channels to choose from. The average peer lifetime of 90 seconds is shorter than the average, for example, reported in [166]. Note that a shorter average peer lifetime represents a more challenging input for testing the P2P protocol.

The prediction lookahead, $D$, is set to 1.6 seconds. Note that due to inaccuracies in the RoI prediction, it is difficult to isolate the losses that peer churn and the P2P protocol alone are responsible for. Hence, we report results by assuming perfect prediction of the RoI $D$ seconds ahead of time. Further results by incorporating realistic RoI prediction techniques are reported in Appendix B.

**User Interaction Trajectories and Slice Size Settings**

As described in Section 4.2.2, user interaction trajectories corresponding to as many peers in the simulation were recorded beforehand by viewing the high-resolution video sequences interactively. As before, two high-resolution layers are encoded apart from the thumbnail video. Table 4.2 shows the total number of tiles in the coded representation of each sequence as well as the average number of tiles required per peer. The same table also shows the total bit-rate of all tiles as well as the average download data rate required per peer.

**4.4.2 Single Tree Per Tile**

We report results that assess the server load as well as the efficiency of the IROI P2P protocol in delivering relevant content to various peers. As described in Section 4.2, each tile is distributed using an independent multicast tree. The server is allowed to support up to three direct children per tile. We observed that not enforcing a limit on the number of direct children per tile exhausts the server’s capacity and the system becomes unable to serve a new tile that no peer currently subscribes to.

Traces of the numbers of received, required and missing slices are shown in Fig. 4.11 for a population of 100 peers. The percentage of missing slices is small, indicating
Table 4.2: Total tiles and the corresponding bit-rate of the coded representation of six high-resolution sequences. Average number of tiles needed per peer and corresponding download data rate, obtained by averaging over 100 user interaction trajectories.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Total tiles</th>
<th>Total bit-rate [Mbps]</th>
<th>Avg. no. of tiles needed at any given time per peer</th>
<th>Avg. download data rate needed per peer [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardgame</td>
<td>141</td>
<td>2.9</td>
<td>10.9</td>
<td>432</td>
</tr>
<tr>
<td>Brainstorming</td>
<td>141</td>
<td>3.8</td>
<td>11.8</td>
<td>467</td>
</tr>
<tr>
<td>Soccer</td>
<td>382</td>
<td>14.1</td>
<td>25.8</td>
<td>1100</td>
</tr>
<tr>
<td>Café</td>
<td>431</td>
<td>11.0</td>
<td>26.0</td>
<td>1230</td>
</tr>
<tr>
<td>Panel 1</td>
<td>431</td>
<td>5.2</td>
<td>28.6</td>
<td>930</td>
</tr>
<tr>
<td>Panel 2</td>
<td>431</td>
<td>4.0</td>
<td>30.6</td>
<td>528</td>
</tr>
</tbody>
</table>

that clients acquire relevant data in a P2P manner despite the challenges noted above. Appendix B shows that realistic RoI prediction increases the number of missing slices, but only slightly.

The server load for a population of 100 peers is shown in Fig. 4.12. The number of tiles streamed by the server to at least one direct child reaches close to the total number of tiles, $S_t$, for the Cardgame and Brainstorming sequences. However, for the other sequences, the number of tiles streamed by the server is well below $S_t$. Two to three times this number was found to be the number of direct children supported by the server. This is due to the limit on the server for supporting up to three direct children per tile. The server load is compared for client-server unicast vs. P2P live multicast in Table 4.3. For client-server unicast, the server load is estimated by simply aggregating the download data rate required by all peers. Also shown in Table 4.3 is the load reduction factor due to the P2P approach.

### 4.4.3 Multiple Trees Per Tile

For simplicity, the above discussion focused on building one independent multicast tree for delivering each tile. However, a given tile can be delivered using multiple complementary trees, for example using two trees as illustrated in Fig. 4.13. In the shown example, if peer ‘F’ leaves the system ungracefully, peer ‘A’ suffers data
Figure 4.11: Traces of the aggregate numbers of received, required and missing slices over all 100 peers shown for the six high-resolution video sequences.

disruption for the even frames but not the odd frames. In general, employing multiple trees per tile avoids the loss of successive slices for a tile.

Figure 4.14 shows the percentage of missing slices as well as the percentage of pixels that are error-concealed on average for a population of 100 peers. The percentage
CHAPTER 4. P2P LIVE MULTICAST OF IROI VIDEO

Figure 4.12: Server load for supporting 100 peers shown for the six high-resolution video sequences. The first Y-axis shows the number of direct children supported by the server and the number of slices streamed to at least one direct child. The second Y-axis shows the server load in terms of bit-rate.
CHAPTER 4. P2P LIVE MULTICAST OF IROI VIDEO

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Server load</th>
<th>Bit-rate savings factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client-server unicast [Mbps]</td>
<td>P2P live multicast [Mbps]</td>
</tr>
<tr>
<td>Cardgame</td>
<td>43.2</td>
<td>7.2</td>
</tr>
<tr>
<td>Brainstorming</td>
<td>46.7</td>
<td>9.3</td>
</tr>
<tr>
<td>Soccer</td>
<td>110.0</td>
<td>13.7</td>
</tr>
<tr>
<td>Café</td>
<td>123.0</td>
<td>22.5</td>
</tr>
<tr>
<td>Panel 1</td>
<td>93.0</td>
<td>9.8</td>
</tr>
<tr>
<td>Panel 2</td>
<td>52.8</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Table 4.3: Server load for client-server unicast and P2P live multicast with 100 clients.

Figure 4.13: Each tile distributed using two dedicated multicast trees. One tree delivers slices associated with even frames and the other tree delivers slices associated with odd frames.

of missing slices is similar for one or two trees per tile. However, with two trees per tile we observed that, for about 65–75% missing slices, the corresponding slice from the previous frame was available. This allows error-concealment using pixels from the previous frame in most cases, thus maintaining high spatial resolution, which is important for virtual pan/tilt/zoom. The quality of the rendered RoI videos was visually better with two trees per tile.
Now consider the computation of the peak signal-to-noise ratio (PSNR) by comparing against the RoI video rendered directly from the original uncoded sequence. Let \( N \) denote the number of frames displayed at a peer over the simulation duration. \( N \) depends on the total on-time of the peer. We first compute the average mean squared error (MSE) between the compared RoI videos,

\[
\text{Average MSE} = \frac{1}{N} \sum_{n=1}^{N} \text{MSE}[n],
\]

where \( \text{MSE}[n] \) is the mean squared error between the rendered version from received data and the rendered version from the original uncoded sequence for frame \( n \). The PSNR corresponding to the average MSE at the peer is given by

\[
\text{PSNR} = 10 \log_{10} \frac{255^2}{\text{Average MSE}}. \tag{4.9}
\]

The PSNR across the peer population is computed by averaging the values given by (4.9) for different peers. While performing error-concealment, we first check if
the corresponding slice from the previous frame was received. If it is not received, we upsample the corresponding portion from the current base layer frame for error-concealment. Note that using high-spatial-resolution data from the previous frame yields a sharper picture at the cost of temporal resolution. This trade-off might degrade PSNR in case of fast motion even though the rendered RoI might be visually better.

As shown in Fig. 4.15, the PSNR is better with two trees per tile for all sequences except the Café sequence. The same figure also shows the lower bound and the upper bound for the PSNR. The lower bound corresponds to the case in which all pixels are error-concealed using the base layer, i.e., none of the required high-resolution slices are received, however, the base layer is received. The upper bound corresponds to the case in which all required high-resolution slices are received in time. We observed that losses of base layer video are very rare. This is because the base layer is transmitted slightly ahead of time compared to the enhancement layers and also retransmissions are employed for the base layer. If a base layer frame is not decodable, we display the immediately previous base layer frame that was successfully decoded and also render the RoI using this frame.

We also experimented with three multicast trees per tile. For a given tile, high-resolution data corresponding to frame number \( n \) was delivered over tree number \((n \text{ modulo } 3) + 1\) corresponding to that tile. The error-concealment algorithm was modified to check for corresponding high-resolution data from the previous frame, followed by the previous to previous frame and finally using the base layer as the last line of defense. No further gain was observed in terms of visual quality or PSNR compared to the case of two trees per tile. However, building more trees resulted in higher percentage of control traffic. The percentage of control traffic is shown in Fig. 4.16 for different numbers of trees per tile.

4.5 Chapter Summary

This chapter presents an IRoI video streaming system that can achieve P2P live multicast. We assume that several users simultaneously watch the high-resolution
video with virtual pan/tilt/zoom functionality. Each user can control his RoI yet the system allows clients to relay relevant data to each other in real time. Harnessing the forwarding capacities of clients allows the system to scale to large numbers of users. In the small peer population regime, the load on the server grows as new peers are likely to request regions that no other peers are currently receiving. However, beyond a certain population size, the diversity among the served slices saturates and so does the load on the server.

It is crucial for the P2P approach that the multicast overlay reacts quickly to the changing RoIs of the peers, thus limiting the disruption due to changing relationships among the peers. The IRoI P2P protocol presented in this chapter makes sure that a child peer experiences a smooth transition from old parent to new parent when the old parent unsubscribes a multicast tree that is no longer required for its RoI. The IRoI P2P protocol is built on top of the Stanford peer-to-peer multicast (SPPM) protocol which operates in tree-push manner. This design choice is based on the insight that the alternative mesh-pull approach is ill-suited for achieving low latency and low delay-jitter which are essential to an interactive application.

The system is designed such that the RoI chosen by the user is instantly rendered on the screen, thus offering accurate and low-latency RoI control. The base layer helps render missing parts of the RoI. The base layer is transmitted slightly ahead of time and retransmissions ensure reception with very low packet loss. The relatively low percentage of missing slices as well as the rendered PSNR being close to the upper bound indicate that peers also obtain most of the required high-resolution slices in time. The control overhead associated with the P2P protocol is around 5–10% of the total traffic. Streaming successive slice of a high-resolution tile over different multicast trees helps avoid the loss of successive slices. However, employing two trees achieves most of the diversity gain and employing more than two trees per tile increases protocol overhead with no additional gain.

It is worth noting that spatial-random-access-enabled video coding plays an important role in the P2P streaming system. It simplifies the peer’s task of choosing which multicast trees to join on-the-fly. The scheme presented in Chapter 3 minimizes the average download rate required by each peer. A scheme based on multiple
representations coding might further reduce the download rate required by each peer. However, such a scheme might reduce the degree of overlaps and affect the gains possible from the P2P approach apart from requiring more storage space at the server. Thus, the tasks of designing the compression scheme and the P2P delivery mechanism are intimately tied.
Figure 4.15: Luminance PSNR shown for five randomly chosen peers as well as the average over 100 peers. Despite the fact that the percentage of missing slices was comparable, the PSNR with two trees per tile was observed to be slightly better than with a single tree per tile. Also shown are the lower bound and the upper bound for the PSNR. The lower bound corresponds to the case in which all pixels are error-concealed using the base layer. The upper bound corresponds to the case in which all required high-resolution slices are received in time.
CHAPTER 4. P2P LIVE MULTICAST OF IROI VIDEO

Figure 4.16: Volume of control traffic shown as percentage of the total traffic. For all six high-resolution sequences, the control traffic increases when more trees are built per tile.
Chapter 5

Optimal Server Bandwidth Allocation

The goal of this chapter is to optimize the limits on the numbers of direct children supported by the server for streaming the high-resolution tiles. Typically, when users choose regions from a high-spatial-resolution video, some regions are more popular than others. The system described in Chapter 4 achieves P2P live multicast of tiles, however, it does not allocate resources considering the unequal popularity of these tiles. Specifically, the experiments reported in Chapter 4 assume that the server enforces a common limit on the number of direct children per tree, and hence also per tile. This simple strategy avoids uncontrolled fan-out and capacity exhaustion at the server, thus ensuring that a new tile that is not subscribed to by any peer can be served when it is demanded.

A more sophisticated strategy could account for differences among the tiles like popularity, rate-distortion operating points, and possibly also the rate at which peers subscribe to and unsubscribe these tiles. Ideally, an optimization algorithm should consider the total uplink capacity available at the server and allocate it among the tiles, irrespective of how much uplink capacity is provisioned at the server. For example, the uplink capacity of the server might be insufficient to stream all tiles in the multi-resolution representation. Even in this case, judicious allocation of whatever rate is available should be performed.
5.1 Problem Formulation

For simplicity, let us assume that a single tree is built per tile, although the approach developed here easily extends to multiple multicast trees per tile. For tile $j$, let us denote the number of direct children of the server by $\alpha_j$. As shown in Fig. 5.1, more direct children lower the average tree-height, thus improving robustness to peer churn. However, to support more direct children the server has to allocate more rate.

Consider the toy example, shown in Fig. 5.2, in which the server is hosting two tiles. Is the configuration on the right, showing more direct children for the more popular tile, better than the configuration on the left? The answer depends on the rate-distortion operating points as well as the peer churn rates associated with the tiles apart from the audience sizes.

We formulate the optimization problem taking into account the tile characteristics mentioned above. As in Chapter 4, let $P$ denote the set of all peers and let $S$ denote the set of all tiles. The server capacity is denoted by $R_S$. Assume that we are given the following quantities related to tile $j$, $\forall j \in S$:

- $m_j$: Number of peers watching tile $j$.

Figure 5.1: Example with 14 peers subscribing tile $j$. On the left, $\alpha_j = 1$ direct child is supported and on the right, $\alpha_j = 2$ direct children are supported. To compare the impact of peer churn, imagine that the peer marked with the cross leaves suddenly. More peers are affected for the tree configuration shown on the left.
• $r_j$: Bit-rate of tile $j$.

• $\Delta d_j$: Distortion reduction associated with successful delivery of tile $j$ to a single peer.

• $\mu_j$: Exponential distribution parameter for tile $j$. The contiguous period for which a peer joins tile $j$ is an exponentially distributed random variable.

Associated with tile $j$, random variable $\Delta D_j$ represents total distortion reduction over all peers that subscribe to tile $j$. The goal of the optimization problem, stated formally in (5.1) below, is to determine $\alpha_j$, the number of direct children of the server for tile $j$, $\forall j \in S$.

$$\max_{\alpha_j, \forall j \in S} \mathbb{E} \sum_{j \in S} \Delta D_j \left( \Delta d_j, m_j, \mu_j, \alpha_j \right)$$

subject to

$$\sum_{j \in S} \alpha_j r_j \leq R_S,$$

$$\alpha_j \in \mathbb{W}, \alpha_j \leq m_j,$$

where, $\mathbb{W}$ is the set of whole numbers; $\mathbb{W} = \{0, 1, 2, \ldots\}$. 

Figure 5.2: Example with the server hosting two tiles. The configurations shown on the left and on the right differ in the number of direct children supported for the two tiles.
5.2 Solution of Optimization Problem

During the multicast session, the optimization problem can be solved repeatedly, each time for a short time-horizon $T$. The probability $\tilde{p}_j$ that a peer leaves tile $j$ in this interval is given by $\tilde{p}_j = (1 - e^{-\mu_j T})$ since the contiguous period for which a peer joins tile $j$ is an exponentially distributed random variable with mean $1/\mu_j$.

The distortion reduction for tile $j$ over all peers that subscribe to tile $j$ can be written as

$$\Delta D_j (\Delta d_j, m_j, \mu_j, \alpha_j) \geq \sum_{i \in P_j} \Delta d_j (1 - Y_{ij}), \quad (5.2)$$

where,

- $P_j$ is the set of peers interested in tile $j$; $|P_j| = m_j$. The framework accommodates that a peer subscribes to multiple tiles.

- $Y_{ij}$ is an indicator random variable which is 1 if peer $i$ suffers disconnection due to an ancestor leaving the tree or if peer $i$ itself leaves the tile in time-interval $T$; it indicates that peer $i$ fails to contribute to the total distortion reduction.

The inequality sign in (5.2) accommodates the fact that peers leaving or suffering disconnection might receive the tile for part of the time-horizon $T$. Assuming that $T$ is short\footnote{Note that although the optimization problem is solved over short time-horizons, the server bandwidth allocation does not change unless some of the input parameters $(m_j, r_j, \Delta d_j, \mu_j), j \in S$ change and induce a change in the optimal solution $\alpha_k, k \in S$.} enough, we can write

$$E [\Delta D_j (\Delta d_j, m_j, \mu_j, \alpha_j)] \approx m_j \Delta d_j - \Delta d_j \sum_{i \in P_j} E (Y_{ij})$$

$$= m_j \Delta d_j - \Delta d_j \sum_{i \in P_j} \Pr \{Y_{ij} = 1\}. \quad (5.3)$$

5.2.1 Parametric Approximation of Losses

Let random variable $\tilde{M}_j$ denote the number of peers that fail to contribute to the total distortion reduction associated with tile $j$; i.e., $\tilde{M}_j = \sum_{i \in P_j} Y_{ij}$. In general, the lower
Figure 5.3: The expected number of peers that fail to contribute to the total distortion reduction associated with tile \( j \) can be modeled according to (5.4). The model parameters \((a, b, c)\) used for both plots (a) and (b) are \((a = 0.33, b = 0.9, c = 0.3)\). We observed that for a wide range of \((m_j, \tilde{p}_j)\), the approximation of \( E(\tilde{M}_j) \) by the parametric model is close to \( E(\tilde{M}_j) = \sum_{i \in P_j} \Pr\{Y_{ij} = 1\} \), obtained by assuming a tree structure for the tile determined by \( \alpha_j \) and a common peer-node out-degree.

the average tree-height for tile \( j \) the lower the value of \( E(\tilde{M}_j) \). Given the structure of the tree for tile \( j \), \( E(\tilde{M}_j) \) can be obtained by summing the probabilities \( \Pr\{Y_{ij} = 1\} \). If peer \( i \) is \( l \) hops away from the server, then \( \Pr\{Y_{ij} = 1\} = 1 - (1 - \tilde{p}_j)^l \). These probabilities have been analyzed in detail, for example in [55], also when forward error correction (FEC) is applied for error-resilient P2P multicast video streaming. Consider the sample tree configurations shown in Fig. 5.1. We can compute \( E(\tilde{M}_j) \) for increasing \( \alpha_j \) assuming that all peer-nodes have the same out-degree. We observed that when all peer-nodes have the same out-degree, for a wide range of \((\tilde{p}_j, m_j)\), the expected number of peers that fail to contribute to the distortion reduction can be modeled as

\[
E(\tilde{M}_j) = m_j e^{-\frac{a(\alpha_j)^c}{(\tilde{p}_j)^{m_j}}}, \tag{5.4}
\]

with a suitably-fit constant triplet of model parameters \((a, b, c)\). An example of the model is shown in Fig. 5.3. \( E(\tilde{M}_j) \) can be computed either as \( E(\tilde{M}_j) = \)
\[ \sum_{i \in P_j} \Pr\{Y_{ij} = 1\} \] by assuming a tree structure for the tile determined by \( \alpha_j \) or simply by approximating with the proposed parametric model. Note that the fraction of peers failing to contribute to the total distortion reduction associated with tile \( j \) remains unchanged if \( \tilde{p}_j \) and the ratio \( \frac{\alpha_j}{m_j} \) are constant. Hence, the parametric model captures the fact that if the population of the sub-tree rooted at each direct child is kept constant, then the fraction \( \frac{E[\tilde{M}_j]}{m_j} \) remains unchanged.

Using the parametric approximation, the expected distortion reduction by supporting \( \alpha_j \) direct children (i.e., allocating rate \( \alpha_j r_j \)) for tile \( j \) is

\[
E[\Delta D_j(\Delta d_j, m_j, \mu_j, \alpha_j)] = m_j \Delta d_j - m_j \Delta d_j e^{-\frac{\alpha_j}{\tilde{p}_j}(m_j)}.
\] (5.5)

5.2.2 Casting into a Knapsack Problem

The expected distortion reduction per unit rate by adding the \( \alpha_j \)th direct child, given that \( \alpha_j - 1 \) direct children have already been added, is given by

\[
\Theta_j(\alpha_j) = \frac{E[\Delta D_j(\Delta d_j, m_j, \mu_j, \alpha_j)] - E[\Delta D_j(\Delta d_j, m_j, \mu_j, \alpha_j - 1)]}{r_j}.
\] (5.6)

Note that \( \Theta_j(\alpha_j) > \Theta_j(\alpha_j + 1) \). Hence our optimization problem (5.1) can be cast as a classic knapsack problem [153]. A greedy solution\(^2\) can be obtained by sorting all \( \Theta_j(1) \cdots \Theta_j(m_j), \forall j \in S \) and allocating rate till the bit-budget is exhausted. The parametric model introduced above dramatically reduces the computational burden for calculating the quantities \( \Theta_j \). This allows the algorithm to scale with the number of tiles hosted by the server.

5.2.3 Special Case

Consider two video tiles with the same rate, distortion and peer-churn characteristics; i.e., \( r_1 = r_2 = r, \Delta d_1 = \Delta d_2 = \Delta d, \) and \( \tilde{p}_1 = \tilde{p}_2 = \tilde{p} \). The audience sizes, \( (m_1, m_2) \), need not be equal. The server bandwidth is given by \( R_S = \alpha_S \times r \). The optimization

\(^2\)not necessarily optimal
problem then becomes,

\[
\min_{\alpha_1, \alpha_2} E\left(\tilde{M}_1\right) + E\left(\tilde{M}_2\right)
\]

subject to

\[
\alpha_1 + \alpha_2 \leq \alpha_S. 
\]  

(5.7)

For simplicity, we relax the constraint that \((\alpha_1, \alpha_2)\) need to be whole numbers. The Lagrangian cost function can be written as

\[
J = E\left(\tilde{M}_1\right) + E\left(\tilde{M}_2\right) + \lambda (\alpha_1 + \alpha_2 - \alpha_S). 
\]

With \(\frac{\partial J}{\partial \alpha_1} = \frac{\partial J}{\partial \alpha_2} = 0\), we obtain

\[
m_1 \times e^{-\frac{a(\alpha_1)^c}{(\tilde{p})^c(m_1)^c}} \times \frac{(\alpha_1)^{(c-1)}}{(m_1)^c} = m_2 \times e^{-\frac{a(\alpha_2)^c}{(\tilde{p})^c(m_2)^c}} \times \frac{(\alpha_2)^{(c-1)}}{(m_2)^c}. 
\]  

(5.8)

The solution \((\alpha_1, \alpha_2) = \left(\alpha_S \frac{m_1}{m_1 + m_2}, \alpha_S \frac{m_2}{m_1 + m_2}\right)\) satisfies (5.8) and other Karush-Kuhn-Tucker (KKT) \([29]\) conditions. Since the objective function is convex in \((\alpha_1, \alpha_2)\), this is the optimal rate allocation. This special case can be easily extended to multiple tiles and provides the following insight: for video tiles with similar rate, distortion, and peer-churn characteristics, the amount of server bandwidth allocated to them should be proportional to their audience sizes. The resulting multicast trees thus have similar tree-height, and assuming the same peer-node out-degree, the sub-trees rooted at the direct children are nearly balanced.

Unlike the special case considered in this section, in practice each \(\alpha_j\) has to be a whole number. As proposed in Section 5.2.2, the greedy solution can be employed. The greedy solution can also account for differences in rate, distortion and peer-churn characteristics.
5.2.4 General Scope of the Optimization Framework

Several modifications fall within the scope of the framework presented above. These modifications might or might not fall within the scope of this thesis, but we briefly discuss these to highlight the generality of the proposed framework. For example, the modifications might entail a change of metric for the optimization, or a simplification facilitating practical implementation, or accommodating a different practical scenario, etc.

- Multiple multicast trees per tile can be accommodated with an optional constraint of forcing equal number of direct children for each tree of the same tile. The additional constraint can be dropped if the data associated with the trees possess unequal rate-distortion importance, e.g., unequal error protection [13, 243].

- The expected number of peers that receive tiles can be maximized by being oblivious of distortion reduction. This can be easily accomplished by letting $\Delta d_j = \Delta d, \forall j \in S$ ($\Delta d$ is an arbitrary constant) in the solution presented in Section 5.2.2. Although the considered metric is oblivious of distortion, maximizing the expected number of successfully delivered slices corresponds to minimizing the percentage of pixels that have to be error-concealed.

- Min-max fairness can be accommodated to minimize the worst disruption suffered for any tile.

- Rate allocation can be performed by a P2P video streaming server that is simultaneously hosting multiple IRoI video sessions. From the optimization problem’s perspective, the overall streamed data comprise portions to be delivered over independent multicast trees. Based on the framework presented above, rate allocation among multiple video channels without any IRoI functionality has been investigated by us in [155].

- Adapting the rate-distortion operating points of the tiles individually might provide additional degrees of freedom for optimal server bandwidth allocation.
among multiple tiles. An additional constraint could limit the quality variation across the tiles. This is similar in spirit to prior work dealing with delivery of multiple video streams over a common wireless channel [99].

The framework proposed in Section 5.2 is similar in spirit to prior work dealing with multiple P2P streaming overlays comprising a common source or common peer-nodes [241]. In [241], a peer that is part of multiple overlays allocates its uplink capacity among the overlays according to the result of an auction game. The framework in [241] can incorporate priorities for different overlays, however, [241] does not cover any server-side algorithm for assigning priorities. Related prior work, e.g., [130, 221], proposes that peers in different overlays help each other by relaying media belonging to other overlays.

For IRoI P2P, allocating available uplink rate at a peer for relaying different tiles poses the following challenges. The set of tiles subscribed to by the peer itself is highly transient. Also, the peer itself cannot control the number of requests from potential child peers for the different tiles; hence, the peer cannot ensure that allocated quotas will be filled up. The server, on the other hand, while providing lists of potential parents corresponding to different multicast trees, can include itself in the lists and acquire child peers to fill up allocated quotas for the different tiles.

5.3 Experimental Results

We set the server bandwidth to 10 Mbps, 5 Mbps and 3 Mbps and compare different rate allocation schemes. Two multicast trees are built per tile for delivering odd and even frames. The two trees associated with tile \( j \) share the same values of audience size, \( m_j \), and average tile subscription duration, \( \mu_j \). The bit-rate, \( r_j \), and distortion reduction, \( \Delta d_j \) are each divided by two before assigning to the two trees. For rate allocation at the server, we compare four approaches. The first three approaches solve the optimization problem as proposed in Section 5.2.2. They differ in the way the parameters \( m_j, \Delta d_j, \) and \( \mu_j \) are updated. The fourth approach simply allocates rate among the multicast trees in a round-robin manner.
- Approach 1: The server monitors *RoI-switch* requests and updates the mean tile subscription durations, $\mu_j$, $\forall j \in S$. Their values are used to compute $\tilde{p}_j$, $\forall j \in S$. The average distortion reduction associated with each tile as well as the corresponding bit-rates are known beforehand since the server encodes the content. For a given tile, say tile $j$, the values of $\Delta d_j$ and $r_j$ are constant over the simulation duration.

- Approach 2: A constant value of $\mu_j = 14.9$ seconds is assumed for all high-resolution tiles. This is motivated by the average tile subscription duration reported in Chapter 4. For the base layer, the average subscription duration is the same as average peer lifetime. The average distortion reduction associated with each tile as well as the corresponding bit-rates are known beforehand since the server encodes the content. For a given tile, say tile $j$, the values of $\Delta d_j$ and $r_j$ are constant over the simulation duration.

- Approach 3: Apart from the simplification of $\mu_j$ as in Approach 2, the distortion reduction associated with all high-resolution tiles is assumed to have the same value. The distortion reduction associated with the base layer is still computed as the distortion increase when the display is frozen for the entire time-horizon instead of playback using base layer only; this is computed by averaging over some sample RoI sequences. The bit-rates $r_j$ are assumed to be known as before.

- Approach 4: This approach sequentially allocates rate to tiles with ascending tile IDs, stopping when the capacity exhausts.

The P2P protocol facilitates measuring the audience sizes for the different multicast trees. In Approaches 1 through 3, the optimization problem is solved as shown in Section 5.2.2 by assuming a time-horizon $T$ of 0.2 seconds. For computing $\Theta_j(1) \cdots \Theta_j(m_j)$, $\forall j \in S$, we assume an out-degree of two per peer-node for each tree. In practice, we did not enforce a limit on peer-node out-degree. We found that if peer-node out-degree is limited, it takes longer to find parent peers, thus leading to unused forwarding capacities.
Server Bandwidth 10 Mbps: The average luminance PSNR across 100 peers is shown for the different approaches in Fig. 5.4 (a). The server bandwidth, 10 Mbps, is less than the data rates of the multi-resolution representations associated with the Soccer and Café sequences, 14.1 Mbps and 11.0 Mbps respectively. Among the tile characteristics considered by the framework proposed in Section 5.2, Approach 4 considers only the tile bit-rates and does not aim to minimize average distortion. Nevertheless, for most sequences, except for Soccer and Café, there is room to allocate at least one direct child per tree and the performance of Approach 4 is not too far below the other approaches. The percentage of error-concealed pixels in the rendered RoI is shown in Fig. 5.4 (b) for the four approaches.

Server Bandwidth 5 Mbps: The average luminance PSNR across 100 peers is shown for the different approaches in Fig. 5.5 (a). The server bandwidth is far below that required by the Soccer and Café sequences, and hence we skip results for those sequences. The aggregate data rate of the encoded Panel 1 sequence is 5.2 Mbps, slightly higher than the server bandwidth. The percentage of error-concealed pixels in the rendered RoI is shown in Fig. 5.5 (b) for the four approaches.

Server Bandwidth 3 Mbps: The average luminance PSNR across 100 peers is shown for the different approaches in Fig. 5.6 (a). The data rates corresponding to the encoded Brainstorming, Panel 1 and Panel 2 sequences are 3.8 Mbps, 5.2 Mbps and 4.0 Mbps respectively. The performance suffers due to Approach 4 for these sequences, whereas the first three approaches have similar performance. The percentage of error-concealed pixels in the rendered RoI is shown in Fig. 5.6 (b) for the four approaches. Snapshots of the audience sizes and the numbers of direct children associated with the different tiles are shown for the Brainstorming and Panel 1 sequences in Fig. 5.7 for Approach 1. Although not shown, the rate allocation results from Approaches 2 and 3 are similar. It can be seen that allocated rate corresponds well with the popularity of tiles.

In general, the results indicate that if the server has enough bandwidth to host all tiles, there is little gain by optimizing rate allocation. This can be attributed to the efficiency of the P2P protocol. However, if the server bandwidth is scarce then it should be allocated judiciously. When server bandwidth is scarce, Approach 4
can lead to very sub-optimal performance, exemplified by the drop of almost 6 dB associated with the Soccer sequence in Fig. 5.4 (a). The results associated with the Brainstorming, Panel 1 and Panel 2 sequences in Fig. 5.6 serve as further examples.
5.4 Chapter Summary

This chapter deals with the problem of allocating server bandwidth among multiple tiles that are distributed simultaneously in a P2P manner. The proposed framework considers characteristics of individual tiles like bit-rate, distortion reduction associated with successful delivery of the tile, audience size and the rate at which peers join and leave the tile. The framework is amenable to a change of metric as well as replacement of the model of number of peers suffering disconnection. The use of a parametric model for approximating the number of peers suffering disconnection lowers the computational complexity. Lower computational complexity allows the algorithm to scale with the number of tiles, and is desirable since the multi-resolution representation considered here comprises hundreds of tiles.

Thanks to the efficiency of the underlying P2P protocol, if enough server bandwidth is available to distribute all tiles then it is not critical to optimize rate allocation. However, if the server bandwidth is not enough to distribute all tiles then the available rate should be allocated judiciously. Compared to a simple round-robin allocation scheme, we observed a gain of up to 6 dB by optimizing the rate allocation.
Figure 5.7: Snapshots of the audience sizes and the numbers of direct children associated with the different tiles, shown for the *Brainstorming* and *Panel 1* sequences. Server bandwidth is 3 Mbps, whereas the data rates corresponding to the encoded *Brainstorming* and *Panel 1* sequences are 3.8 Mbps and 5.2 Mbps respectively.

The proposed framework has broader scope and can be applied to other practical scenarios. A sample scenario is one where the server simultaneously hosts multiple video channels that are distributed in a P2P manner to the corresponding sets of receivers.
Chapter 6

Conclusions

This dissertation explores an IRoI P2P video streaming system, which is unique considering the functionality it provides. The peers relay content to each other in real time while each user independently controls pan/tilt/zoom for watching the video. IRoI video streaming avoids transmitting the entire field-of-view in high resolution to each peer, thus reducing required data rate. IRoI functionality allows watching high-resolution video even on displays of lower spatial resolution. In our treatment of the IRoI P2P video streaming system, we focus on three main aspects:

- A spatial-random-access-enabled video coding scheme that allows encoding the entire field-of-view once with few resolution layers. Relevant portions of the encoded bit-stream can be served to different clients depending on their individual RoIs.

- A distributed P2P protocol that allows peers to receive and relay relevant content in real time.

- A server-side algorithm that allocates rate for streaming regions in a P2P manner.

In the following, we summarize lessons learned from the design, analysis, optimization and simulation study of the IRoI P2P video streaming system, and discuss open research problems.
6.1 Lessons Learned

The primary goal of this dissertation is designing the system such that it scales to large numbers of users. In working towards this goal, we limit the video encoding load at the server and reduce the amount of bandwidth required from dedicated servers.

The spatial-random-access-enabled video coding scheme, presented in Chapter 3, allows serving multiple users having different RoIs, either synchronously or asynchronously. We analyze the tradeoff in choosing the slice size for encoding a high-resolution layer. A bigger slice size leads to better coding efficiency, thus lowering the storage requirement, however, it entails transmitting more pixels falling outside the RoI due to the coarser slice grid. We show how to choose the slice size that minimizes the mean transmission bit-rate, i.e., the mean data rate downloaded by each peer in the P2P context. The improvement of the coding scheme based on background extraction significantly reduces both storage and transmission bit-rates while retaining efficient spatial random access. The optimal slice size depends on the video content as well as the dimensions of the RoI display. The framework proposed in Chapter 3 allows estimating the optimal slice size without recording user interaction trajectories and preparing long bit-streams encoded with different slice sizes.

The design of the P2P multicast streaming system depends on the video compression scheme. The IRoI P2P video streaming system, presented in Chapter 4, delivers video compressed using the above scheme. We employ the tree-based push approach and build a dedicated set of multicast trees for delivering data associated with each video tile. To deliver good spatial browsing experience, it is crucial to optimize the system for low latency and robustness to constantly changing overlaps among the peers’ RoIs. The IRoI P2P protocol makes sure that a child peer experiences smooth transition from old parent to new parent when the old parent unsubscribes a multicast tree that is no longer required for its RoI. The ability of the protocol to quickly find parent peers is crucial for harnessing forwarding capacities of the peers. Although hundreds of multicast trees are built, the P2P protocol can be optimized to restrict the control traffic to 5–10% of the total traffic. Building multiple multicast trees per tile is advantageous, however, building two trees per tile provides the best results.
When users choose regions from a high-spatial-resolution video, some regions are more popular than others. In Chapter 5, we present a framework for server-side rate allocation for P2P multicast streaming of video tiles. Optimizing the rate allocation is not crucial when the server has enough bandwidth to stream all high-resolution tiles. On the other hand, when the server bandwidth is scarce, it should be judiciously allocated, primarily by considering the popularity of the tiles.

6.2 Open Problems

The research reported in this dissertation opens interesting new research directions.

- The proposed IRoI P2P system assumes that peers watch the video synchronously. If peers can watch any time segment, i.e., rewind and fast-forward, then the relevant data could still be retrieved from each others’ cache. Since storage is becoming cheaper, the cache size employed by peers for storing previously received content can be assumed to be reasonably large. Such kind of time-shifted P2P streaming can be looked upon as the temporal counterpart of the spatial freedom provided by pan/tilt/zoom. A system providing both functionalities would be a natural extension of the system presented here, and will raise interesting protocol design issues.

- The spatial random access approach developed in this dissertation is also relevant for systems that employ image-based rendering and manipulate the transmitted imagery further to yield a novel view, e.g., tele-immersive systems and free viewpoint TV. For example, when a light field representation is used, zoom functionality entails transmitting the relevant region from multiple camera views. Providing spatial random access as well as view random access while exploiting redundancy in the multi-view data set will pose interesting challenges. Benefits of application-layer P2P multicast could also be explored.
• The approach of forming separate multicast groups for video tiles also lends itself to IP multicast. The design, analysis and optimization of an IPTV system based on IP multicast while providing interactive region-of-interest will raise interesting questions. Since the user watches video interactively, gauging and improving user-experience for such a system constitutes an interesting problem.
Appendix A

High-Spatial-Resolution Video Sequences

In this thesis, we report experimental results using six high-spatial-resolution video sequences. For each video sequence, the thumbnail video is obtained by spatially downsampling the original by 4 both horizontally and vertically. There are two high-resolution layers; the first layer sequence is obtained by downsampling the original by 2 both horizontally and vertically, while the second layer sequence is simply the original video. In our experiments, the set of frames comprising each sequence is looped to play for the duration of the respective simulation.

A.1 Cardgame

Spatial resolution: $3584 \times 512$ pixels. Temporal resolution: 25 frames per second. Length: 298 frames.

This sequence shows four persons playing a card game. At the start of the sequence, three persons are seated and are looking at their respective cards. The fourth person takes his seat and starts viewing his cards. After a while, one of them draws a card from the deck in front of them. The card players are seen talking to each other while they are looking at their cards.
Figure A.1: Sample frame and sample RoIs from various time-instants shown for the Cardgame sequence.

This sequence is a 360° panorama obtained by stitching videos recorded using five cameras [174]. We thank the Stanford Center for Innovations and Learning (SCIL) for making this sequence available. A sample frame as well as sample RoIs from different time-instants are shown in Fig. A.1.

A.2 Brainstorming

Figure A.2: Sample frame and sample RoIs from various time-instants shown for the Brainstorming sequence.

This sequence is also a 360° panorama obtained by stitching videos recorded using five cameras [174]. The field-of-view covers several people gathered for a brainstorming session in a classroom. The white-boards behind the participants show some text and graphics. One of the boards shows the agenda of the meeting. The participants take turns for speaking during the meeting.

We thank the Stanford Center for Innovations and Learning (SCIL) for making this sequence available. A sample frame as well as sample RoIs from different time-instants are shown in Fig. A.2.
Figure A.3: Sample frame and sample RoIs from various time-instants shown for the Soccer sequence.

A.3 Soccer

Spatial resolution: $2560 \times 704$ pixels. Temporal resolution: 25 frames per second. Length: 598 frames.

This sequence is obtained by stitching videos recorded using two ARRI FLEX D-20 professional digital camcorders [1, 65]. The field-of-view spans the entire playing field and some portion of the audience stands. At the beginning of the sequence, the action is concentrated in one half of the playing field. The action slowly transitions to the other half of the playing field in the latter part of the video. The video covers
Figure A.4: Sample frame and sample RoIs from various time-instants shown for the Café sequence.

part of a soccer match played in the Commerzbank Arena, Frankfurt, Germany. The two teams in the game are Frankfurt Eintracht and 1. FSV Mainz 05.

We thank Fraunhofer Heinrich-Hertz Institut (HHI) for making this sequence available. A sample frame as well as sample RoIs from different time-instants are shown in Fig. A.3.
A.4 Café

Spatial resolution: $1920 \times 1088$ pixels. Temporal resolution: 30 frames per second. Length: 598 frames.

This sequence was recorded using a consumer HD camcorder, Canon VIXIA HF11 [2]. The video, recorded with 1080 lines of pixels, was line-padded so that the resulting number of lines is a multiple of 16. The video shows patrons of Bytes Café in the David Packard Electrical Engineering building at Stanford University. A sample frame as well as sample RoIs from different time-instants are shown in Fig. A.4.

A.5 Panel 1

Spatial resolution: $1920 \times 1088$ pixels. Temporal resolution: 30 frames per second. Length: 598 frames.

This sequence was recorded using a consumer HD camcorder, Canon VIXIA HF11 [2]. The video, recorded with 1080 lines of pixels, was line-padded so that the resulting number of lines is a multiple of 16. The video shows a panel discussion. Seven panelists are visible in the field-of-view. A sample frame as well as sample RoIs from different time-instants are shown in Fig. A.5.
APPENDIX A. HIGH-SPATIAL-RESOLUTION VIDEO SEQUENCES

Figure A.5: Sample frame and sample RoIs from various time-instants shown for the Panel 1 sequence.

A.6 Panel 2

Spatial resolution: $1920 \times 1088$ pixels. Temporal resolution: 30 frames per second. Length: 598 frames.

This sequence was recorded using a consumer HD camcorder, Canon VIXIA HF11 [2]. The video, recorded with 1080 lines of pixels, was line-padded so that
Figure A.6: Sample frame and sample RoIs from various time-instants shown for the Panel 2 sequence.

the resulting number of lines is a multiple of 16. The video shows a panel discussion involving three panelists. A slide is projected in the backdrop. The slide shows the agenda for the Entrepreneurship Week 2009, which took place at Stanford University, CA. A sample frame as well as sample RoIs from different time-instants are shown in Fig. A.6.
Appendix B

RoI Prediction and Pre-Fetching

The rationale behind pre-fetching is lowering the latency of interaction. The design of the interactive streaming system, presented in Section 4.1, assumes that when frame $n$ is rendered on the user’s screen, the client predicts the RoI in frame $n + d$ (i.e., $D$ seconds ahead of time) and initiates connection to new trees if required. As explained in Section 4.1, the prediction lookahead, $d$ frames, could take into account the tree-connection delay, packet delay on the network, and the desired interaction latency. The results presented in Chapters 4 and 5 assumed the value of $D$ to be 1.6 seconds. In this appendix, we assume the same value of $D$ and investigate the performance of the IRoI P2P system by employing practical RoI prediction/pre-fetching rather than assuming perfect knowledge of the future RoI like in earlier chapters.

B.1 Related Work

A simple user-input device, for example a computer mouse, typically senses position. More sophisticated devices like game-controllers can also measure velocity and/or acceleration. Studies on view trajectory prediction have been conducted in the context of Virtual Reality [16] and networked multi-player video games [203]. A common navigation path prediction technique, dead reckoning, predicts the future path by assuming that the user maintains the current velocity. The velocity can be either read from the input device or computed from successive position measurements.
Previous work on interactive streaming of light fields involves predicting the \((x, y)\) mouse co-ordinates based on dead reckoning and translating these into the viewpoint [179]. The use of a Kalman filter for head movement prediction in scenarios where head movements can control the application have been proposed in [110]. In prior work on dynamic light fields, six Kalman filters have been used for predicting the 3-D co-ordinates and the 3 Euler angles that define the viewpoint [114,115]. For light field rendering, the viewpoint and the rendering algorithm together determine the number of views that need to be streamed to the client. The authors note that if the streaming system allows tuning into a view-stream only during certain frame-intervals, one can choose an appropriately long prediction lookahead and tune into new view-streams beforehand to avoid missing the join opportunities. In light of this, it is worth pointing out that since our spatial-random-access-enabled video compression scheme allows us to start decoding a high-resolution video tile starting from any frame-interval, we do not need to account for specific join opportunities.

\section*{B.2 Extrapolating the Navigation Trajectory}

We use an autoregressive moving average (ARMA) model to estimate the velocity of the RoI center:

\begin{equation}
 v_t = \alpha v_{t-1} + (1 - \alpha)(p_t - p_{t-1}), \tag{B.1}
\end{equation}

where, the co-ordinates of the RoI center, observed up to frame \(n\), are given by \(p_t = (x_t, y_t)\) for \(t = 0, 1, \ldots, n\). The predicted RoI center co-ordinates \(\hat{p}_{n+d} = (\hat{x}_{n+d}, \hat{y}_{n+d})\) for frame \(n + d\) are given by

\begin{equation}
 \hat{p}_{n+d} = p_n + dv_n, \tag{B.2}
\end{equation}

suitably adjusted if the RoI happens to veer off the extent of the video frame. The parameter \(\alpha\) above trades off responsiveness to the user’s RoI trajectory and smoothness of the predicted trajectory. The zoom factor for frame \(n + d\) is simply predicted to be the same as the zoom factor in frame \(n\). We have partly presented this approach in [152,156].
We now present results by choosing the same experimental settings as in Chapter 4. In particular, a single tree is built per tile and the server supports up to three direct children per tile. The traces of numbers of received, required, and missing slices are shown in Fig. B.1 for the Cardgame, Brainstorming, Panel 1 and Panel 2 sequences. The traces are shown aggregately for all 100 peers. The plots on the right hand side of the figure show results when an excess margin around the predicted RoI is pre-fetched. This is done by extending both the width and the height of the predicted RoI by 30% while retaining the center location. Note that even for the plots on the left hand side, due to the grid of slices, some pixels outside the predicted RoI are pre-fetched. The percentage of missing slices and the percentage of pixels that have to be error-concealed are shown in Fig. B.2. For the slice size settings, it was observed that pre-fetching excess slices rarely helped. In fact, for the Soccer and Café sequences, it hurt the downloading of the required slices. These two sequences are the most data rate intensive, as can be seen in Table 4.2.

By comparing results shown in Figs. B.1 and 4.11 we observe that the losses due to practical RoI prediction/pre-fetching are only slightly higher than assuming perfect knowledge of future RoI. Note that Chapter 4 assumes the latter to specifically assess the efficiency of the P2P protocol and streaming system. In the presence of practical RoI prediction/pre-fetching, if we focus our attention on a single peer, we find that losses occur mostly when the RoI shifts to a new area. For a given peer, losses are rare when the RoI is stationary. For a 1000-frame interval, we show traces of the percentage of error-concealed pixels for some sample peers in Fig. B.3. Finally, average PSNR of the rendered RoI is shown in Fig. B.4.

Our prior work, presented in [152, 156], explores the improvement in RoI prediction by analyzing motion in the buffered thumbnail video frames in addition to extrapolating moves of the input device. Although we do not report the performance of video-content-aware RoI prediction/pre-fetching here, we did not observe an improvement by employing it. This is because the slice sizes chosen in this dissertation are larger than the slice size in [152, 156]. A larger slice size entails a bigger cushion for error in predicting the RoI location.
Figure B.1: Traces of numbers of received, required and missing slices shown aggregately for all 100 peers for the Cardgame, Brainstorming, Panel 1 and Panel 2 sequences. **Left:** No excess slices around the predicted RoI are pre-fetched. **Right:** Width and height of the predicted RoI are extended by 30% each while retaining the center location.
Figure B.2: Percentage of missing slices and percentage of RoI pixels that have to be error-concealed.
Figure B.3: Trace of percentage of error-concealed pixels in the RoI shown for a randomly sampled peer. Here shown for the Cardgame, Brainstorming, Panel 1 and Panel 2 sequences.
Figure B.4: Luminance PSNR of the rendered RoI. The results are averaged over 100 peers.
Bibliography


[144] A. Manzanera and J. Richefeu, “A robust and computationally efficient motion detection algorithm based on $\Sigma-\Delta$ background estimation,” in Proc. 4th Indian


[151] ——, “Background extraction and long-term memory motion-compensated prediction for spatial-random-access-enabled video coding,” in Proc. Picture Coding Symposium (PCS’09), Chicago, IL, USA, May 2009.


