Zoomable Video Playback on Mobile Devices by Selective Decoding

Wei Tsang Ooi and Feipeng Liu

Department of Computer Science,
National University of Singapore,
13 Computing Drive Singapore 117417,
{ooiwt, liuf0005}@comp.nus.edu.sg

Abstract. Many mobile devices are not capable of playing high definition video smoothly. Moreover video playback on mobile devices drains the battery quickly. We propose a method named selective decoding to increase video playback frame rate and reduce power consumption. Selective decoding is based on the idea that users are usually interested in only part of a video scene. It will be more efficient if only the user's Region of Interest (ROI) is decoded. To enable selective decoding, we analyzed various dependency relationships among macroblocks in MPEG4 Part 2 Simple Profile codec, traced the macroblocks that are needed to decode the ROI, and modified standard decoding process to decode macroblocks selectively based on the trace. Our experiments on Android platform show that selective decoding can improve playback frame rate by up to 237.7% and reduce energy consumption by up to 64.5%.

Keywords: selective decoding, zoomable video, energy saving, Region of Interest playback

1 Introduction

Video playback is supported in most of today's typical mobile devices. A recent study shows 20% people watch videos on their mobile devices[1].

Despite its popularity, several issues of mobile video playback remain unaddressed. Firstly, because its heavy processing requirement, High Definition (HD) video playback is not supported on many mobile devices. Secondly, video playback is energy demanding, and the fast pace discharge of battery makes the mobile devices ununsable quickly.

The screen on mobile devices is usually not large, despite its high resolution. Pinch zoom and pan gestures have been widely adopted to help users to view photos, maps and web pages. Recent research indicates zoom and pan are helpful for users when watching videos[2]. However, the examination of existing mobile video players on iPhone and Android platforms tell us many of them do not support zoom and pan. And for those supporting these gestures, they simply scale the video up.

When a video is scaled up, only part of the video scene are displayed. The visible part is referred as Region-of-Interest (ROI). The simple scale and display approach for video playback is inefficient since the entire video scene is decoded but only part of it is displayed. This observation presents an opportunity to achieve more efficient video playback on mobile devices.

We propose a software approach named selective decoding that improves the efficiency of zoomable video playback. As its name suggests, this approach decodes video selectively based on the ROI captured from user interactions. With the improved efficiency, it is hoped this approach can help to address the two issues mentioned previously.

Our work is based on MPEG4 Part 2 Simple Profile (SP) codec, however this approach should be applicable for other Discrete Cosine Transform (DCT) based video codec. We focus our work on the decoder side of the codec in the local playback context, but the principles and techniques could be extended to encoder and video playback in network streaming context. A typical MPEG4 SP decoder is illustrated as below, The encoded bitstream is demultiplexed. The

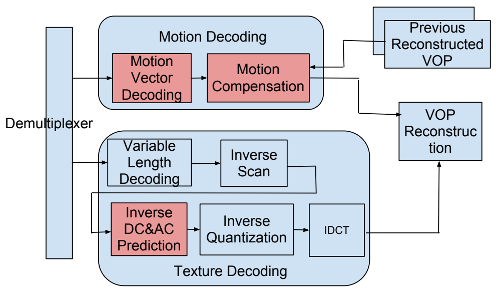


Fig. 1. A Typical MPEG4 Part 2 Decoder

coded texture goes through the texture decoding and coded motion is processed by motion decoding. The decoder finally reconstructs the Video Object Plane (VOP). Note that the figure only depicts the core idea of MPEG4 SP decoder, lots of complexities are not covered.

Various dependencies exist among macroblocks of a MPEG4 SP coded video. From the decoder's pespective, Inverse DC&AC Prediction phase at Texture Decoding, and Motion Vector (MV) Decoding and Motion Compensation at Motion Decoding of a macroblock are carried out with reference to other macroblocks. Specifically, Inverse DC&AC Prediction and Motion Vector Decoding are done with reference to other macroblocks in the current VOP, while motion compensation requires macroblocks from previous reconstructed VOP. Because of these dependencies, decoding the MBs of a ROI will require macroblocks outside of ROI to be decoded.

The selective decoding approach preprocesses the video to obtain the dependencies and save them into files. At decoding, it computes a bitmask based on the dependencies for every VOP on the fly, where the selected macroblocks are marked as '1' and others are marked as '0'. The modified decoder then decodes the selected macroblocks accordingly to the bitmask. In this manner, the macroblocks that are necessary and sufficient to present a clear scene at ROI are decoded.

In the rest of this paper, section 2 reviews related works. section 3 analyzes the dependencies in detail and discusses the offline computation of selective decoding. Online computation is covered in section 4, including the bitmask generation and modified decoder. Section 5 briefly describes different approaches that validates selective decoding. The proposed approach is implemented in Android platform and described in section 6. We evaluate selective decoding for both video playback frame rate and energy consumption in section 7 and finally conclude at section 8.

2 Related Work

H.264/MPEG4 Advanced Video Coding (AVC) standard includes an extension supporting Scalable Video Coding (SVC)[3]. SVC allows embedding of one or more subset bitstream into a high quality bitstream. It enables transmission and decoding of partial bit streams to provide video of lower resolution and/or lower frame rate. SVC requires changes to both encoder and decoder and is not widely adopted yet.

Supporting zoomable video through encoders has been explored by several research groups. Mavlankar etc. studied the optimal slice size for zoomable video in a network streaming context[4]. Feng etc. presented how to produce a video stream with ROI cropping support by constraining the video compression process[5].

To enhance video viewing experience on small display, zoomable video with both manual [6] and automatic ROI tracking [7] are studied. Their focus is to produce adaptive network video stream for mobile devices with different display sizes.

More works exist on zoomable video in network context, each with its own focus. Utilize zoomable video to save bandwidth by refining the encoding process is studie[8]; Zoomable video on peer-to-peer streaming is explored[9]; ROI prediction for streaming zoomable video is examined[10]; multiple ROIs support is investigated[11].

In summary, extensive research has been done to enable zoomable video in network streaming context, with focus on video encoding process. To the best of our knowledge, no material has been found on enabling zoomable video from decoder in local playback context through the decoding process.

3 Offline Computation

The offline computation of selective decoding retrieves the dependencies by partially decoding a video. It is essential to understand various dependencies in order to comprehend offline computation.

For all subsequent disscussions in this paper, we assume the video is in YCbCr420 color space, which means a macroblock contains four luminance blocks and two chrominance blocks.

3.1 Dependencies

Two categories of dependencies can be identified, namely intra-VOP dependency and inter-VOP dependency. Dependencies are the reason why some macroblocks outside of ROI need to be decoded. By saving certain information at offline computation, we can reduce the dependencies and improve the efficiency of selective decoding. Below we analyze the dependencies and introduce the methods to reduce them.

Intra-VOP Dependency Intra-VOP dependency refers to the dependencies among macroblocks within a single VOP. There are two sources of intra-VOP dependency, including DC&AC Prediction and MV coding.

DC&AC Prediction is performed for I-macroblock when the header field short_video_header is set to '0' in MPEG4 SP bistream. DC&AC Prediction decoding consists of two steps, reference block selection and prediction decoding. The reference block selection step can be illustrated by figure below, There

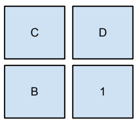


Fig. 2. DC&AC Prediction Reference Block Selection

are two candidate reference blocks for each block, the immediate left block and the immediate upper block. In addition, the upper left block is also needed in order to determine the reference block. To determine the reference block for block 1 in figure above, the following rule is applied,

```
if (|F(B)[0][0] - F(C)[0][0]| < |F(C)[0][0] - F(D)[0][0]|)
   predict from block D
else
   predict from block B
```

Note that F(B)[0][0], F(C)[0][0] and F(D)[0][0] refer to inverse quantized DC value of block B, C and D respectively. The selection rule indicates one block is dependent on three neighboring blocks. Once the reference block is selected, prediction coding can be carried out, where the decoding block depends only on the selected reference block.

Two out of three blocks are only needed to determine the prediction direction, which is either up or left. A single bit is enough to record this information, with '0' indicating up and '1' referring to left. This reduces the dependency for a block from three to one.

MVs of P-VOP is differentially coded, which is the other source of intra-VOP dependency. MPEG4 SP allows one MV for the entire macroblock, or one MV for each non-transparent block. At motion decoding, the decoder recovers the MV values based on the decoded base values and residue values obtained from three predicators. The dependency is introduced by obtaining the residue values, which is shown as figure below, The positions of the candidate predicators

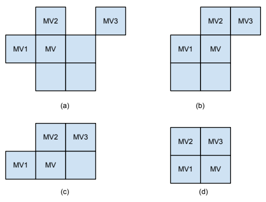


Fig. 3. Different Situations of Motion Vector Prediction

depend on where the current decoding block locates at its macroblock. Note that the MV candicate predicators follow Fig 3(a) for chrominance blocks.

The MVs are recorded and used directly by decoder so that no MV prediction decoding is needed. Therefore, the dependency due to MV prediction decoding is eliminated completely. There is another reason to save MV values, which is discussed in inter-VOP dependency.

Inter-VOP Dependency Inter-VOP dependency refers to the dependencies among macroblocks at different VOPs. It is caused by motion compensation coding.

In MPEG4 SP, motion compensation decoding only occurs at P-macroblock at P-VOP. Considering the case there is one MV per macroblock, the inter-VOP dependency can be illustrated as below, Motion compensation decoding

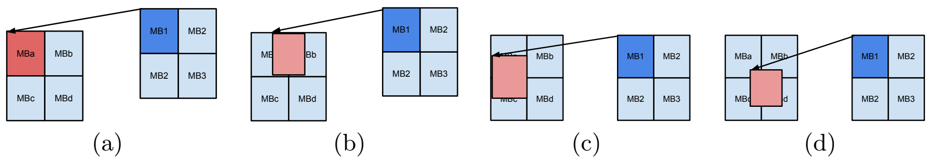


Fig. 4. Different Cases of Motion Compensation Decoding

is performed on MB1 of the current frame, with reference to a 16x16 region of a previous frame. Because the reference region does not have to align with the macroblock boundary, the number of dependent macroblocks differ at different cases shown above. At Fig 4(a), MB1 depends on one macroblock. At (b) and (c), it depends on two macroblocks. And at (d), it depends on four macroblocks.

Khiem etc. proposed an approach to reduce dependency due to motion compensation[8]. We adopted their technique in this research. Using the case in Fig 4(d) as an example, this is illustrated as below, To decode MV1 at Frame 1,

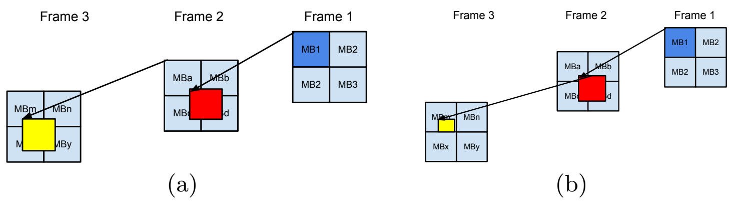


Fig. 5. Dependency Analysis for Motion Compensation

four macroblocks at Frame 2 are needed. For one of the four macroblocks MBa at Frame 2, another four macroblocks at Frame 3 are needed. Dependency can be reduced by tracing it at pixel level. The dependency between Frame 1 and Frame 2 remains the same. But we do not really need all pixels at Frame 2 MBa. By tracing the dependency at pixel level, the region needed at MBa depends on only part of MBm at Frame 3. Therefore, the dependency for MBa between Frame 2 and 3 are reduced. In the example, it is reduced from four to one for macroblock MBa.

3.2 Dependency Files

The offline computation partially decodes a video and records down the dependency information into a set of files named dependency files. The dependency files are generated for each Group of VOP (GOP). This avoids the issue of dealing with big files when dependency information is stored in a single set of files for the entire video. It also reduces the overhead of switching between files when dependency information is stored on a frame by frame basis.

For every GOP, the dependency files include the following,

1. GOP record file: this file contains the start and end frame numbers of a GOP.
2. MB start position file: this file stores the macroblock start bit position in the video bitstream for every MB of all frames in the GOP. The file is in binary format and each position is stored in four bytes.
3. MB end position file: this file contains the macroblock end bit position. The start position and end position files are needed for the decoder to seek the bits for a macroblock.

4. DC&AC Prediction direction file: this file contains the DC&AC Prediction direction. A single bit is used to store the direction for each block. The direction is read directly by the decoder to avoid decoding the macroblocks used in DC&AC Prediction reference selection but not in actual prediction decoding.
5. MV file: this file records the MV values for every macroblock of each P-VOP in the GOP and the number of bits used to encode the MVs. The selective decoder reads the MV values from this file and skip the encoded bits. This file does not only eliminates the MV decoding dependency, but also allows the online computation to trace the inter-VOP dependency. The file is in binary format.
6. Intra frame dependency file: this file contains the positions of macroblocks that a macroblock directly depends on when only intra-frame dependency is considered. Since MV dependency is eliminated, the only source for intra-VOP dependency is DC&AC prediction decoding. Note that it is possible to compute this information online based on DC&AC Prediction direction file, but we obtain this information at offline computation for simplicity. The file is in binary format.

We modified the standard MPEG4 SP decoder to partially decode a video in order to generate the above files.

4 Online Computation

Offline computation is carried out once and the dependency files are saved. Every time the video is played, the online computation computes a selective mask for each frame and decodes according to the mask.

4.1 Dependency File Accessing

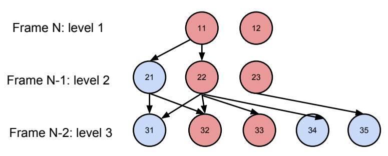
The standard file I/O is slow for random access in a large file because of the seeking overhead. The online computation utilizes the memory mapped I/O for fast random access of dependency files. Before accessing the content, the dependency file is mapped to memory using Linux mmap system call [webmmap] and a pointer to the first record is returned. Memory mapped I/O requires that each record occupies the same number of bytes. Suppose each record is 4 bytes and we need to access the sixth record, the pointer is moved by (6-1)*4=20 bytes. This approach facilities the fast random access of dependency file, with the price of padding the shorter records so that they can be aligned.

4.2 Selective Mask Computation

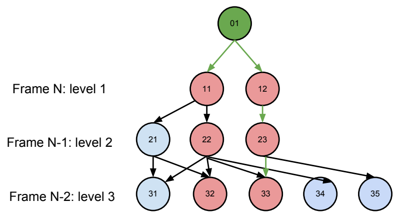
Selective mask indicates the macroblocks that the decoder needs to decode as '1' and the rest as '0'. It considers both inter-VOP and intr-VOP dependencies. Note that the inter-VOP dependency has to be computed first. If intra-VOP dependency is computed first, when computing inter-VOP dependency, the computation will select some new P-macroblocks and I-macroblocks. Since the intra-VOP dependency for the newly selected I-macroblocks is not computed, this leads to decoding errors at those I-macroblocks. The error will subsequently affect the motion compensation decoding at other macroblocks using those I-macroblocks as reference. By contrast, if inter-VOP dependency is computed first, the intra-VOP computation will select only I-macroblocks because DC&AC prediction coding only applies to I-macroblock. Since inter-VOP dependency does not apply to I-macroblocks, the newly selected I-macroblocks won't introduce errors.

Inter-VOP Dependency Computation Inter-VOP dependency is caused by motion compensation decoding. For a MPEG4 SP GOP, every I VOP is followed by a sequence of P VOPs. In a GOP, the P-macroblocks of every P frame are motion compensated with reference to macroblocks of its previous frame. This means every P frame is dependent on its previous frame and subsequently the dependency is one direction within a GOP. Based on this observation, we compute the inter-VOP dependency from last frame back to the first frame of the GOP.

The dependencies are shown Fig 6(a). Suppose frame N is the last frame



(a) Inter-VOP Graph



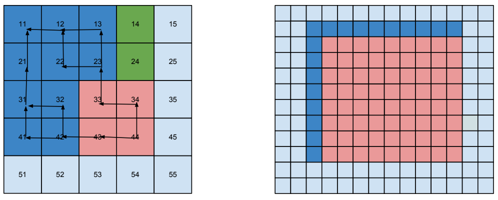
(b) Modified Inter-VOP Graph

Fig. 6. Inter-VOP Dependency Abstraction

of the GOP, the figure shows the dependency relationships among macroblocks in three VOPs. The macroblocks and the dependencies form a graph. With slight modification of adding a pseudo root node and edges connecting the ROI macroblocks, the graph is transformed to a weakly connected directed acyclic graph shown as Fig 6(b).

With this modification, the graph traversal algorithm Depth-First Traversal (DFT) or Breadth-First Traversal (BFT) can be applied. The macroblocks that are visited by the graph traversal are selected, while the rest are not needed.

Intra-VOP Dependency Computation Simiar to Inter-VOP dependency, Intra-VOP dependency computation can be abstracted to graph traversal problems. In Fig 7(a), suppose macroblock 33, 34, 43 and 44 are ROI macroblocks.



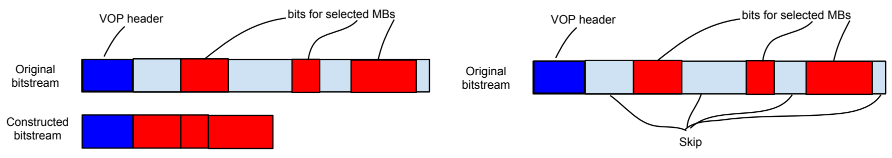
(a) Graph (b) Optimization

Fig. 7. Intra-VOP Dependency and the Optimization

The dependency graphs due to DC&AC prediction coding is depicted. Because the dependency direction is always towarding up or left, the graph rooted at a macroblock is always formed by one of the graphs rooted at the first row and first column macroblocks of the ROI and some macroblocks within the ROI. Therefore, an optimization is to apply graph traversal algorithms only on the macroblocks at upper and left edges of the ROI, and select all macroblocks within ROI.

Select the Bits The standard MPEG4 SP decoder is modified to decode selectively according to the selective mask. Because the decoder does not decodes all the bits, a mechanism is needed for the decoder to select the bits for selected macroblocks.

Two approaches can be used to select the bits, which are depicted in figures below. Fig 8(a) illustrates the bitstream reconstruction approach, where



(a) Reconstruction (b) Skip

Fig. 8. Two different Approches of Selecting the Bits

a new video bitstream is constructed according to the selective masks and the macroblock start and end positions. Thew newly constructed bitstream consists of only the macroblocks selected in selective masks. The second approach is to instruct the decoder to seek to the start position of next selected macroblock at decoding, which is shown in Fig 8(b). The second approach avoids the additional memory allocation for the new bistream therefore it is the preferred approach in this project. However, for applications in the next streaming context, the first approach may provide some benefits in terms of bandwidth as the shorter reconstructed bitstream can be transmitted.

4.3 Modifications of Decoder

The online computation requires modification for both motion decoding and texture decoding of standard MPEG4 SP decoder. For motion decoding, the MV values are loaded directly from MV dependency file. No MV prediction decoding is carried out. For texture decoding, the DC&AC prediction direction is read from DC&AC prediction direction file. The capability of bits seeking is also added for the decoder to decode selectively.

5 Decoding Verification

Selective decoding is a complicated process with lots of details despite its simple idea and principles. It is important to validate the selective decoding is done correctly. Several approaches can be used.

The naive approach is simply watch the selective decoded video to see if there is any visible blemishes. A better approach is to compare the selectively decoded ROI pixel values with the pixels decoded by standard MPEG4 SP decoder. A third approach is to compare the output values of each stage of the selective decoding each the values produced by standard decoding. This approach does not only verify if selective decoding is done correctly, but also facilites finding the phase that causes problems if there is any.

6 Implementation

We implemented the techniques and processes described in previous sections on Android platform[12] as a zoomable video player. The architecture of the player is depicted as figure below. Based on the open source MPEG4 SP codec from

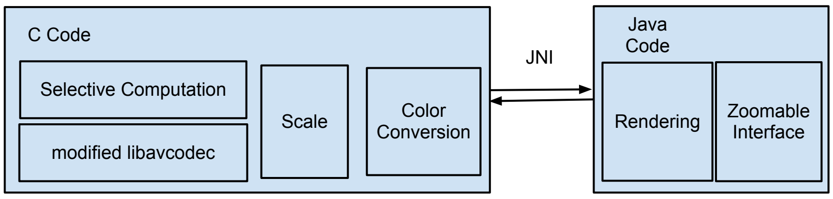


Fig. 9. Zoomable Video Player Implementation

libavcodec of ffmpeg 0.7[13], we added the selective decoding functions. At the time of implementation, the ffmpeg codec is not optimized to run on Android platform, especially the scale and color conversion process. We adopts the code from another two open source projects, with scale from libyuv[14] and color conversion from Google Chromimum project[15].

We implemented the rendering and zoomable interface in Java with Android SDK. The gesture dectector detects a user's pinch or pan gesture, and translates the gesture detected as zoom scale and pan scale. We then compute a ROI from these scales based on what user can see on the phone screen.

7 Evaluation

The purpose of selective decoding is to achieve more efficient zoomable video playback. We evaluate the efficiency of selective decoding from two aspects, video playback frame rate and energy consumption.

We used two test phones for evaluation with the specifications shown below, The description thereafter refers HTC Incredible S as Phone A, and Samsung Galaxy S2 as Phone B.

Table 1. Test Phones for Evaluation

Phones	CPU Clock Rate	RAM	CPU Info	Display	Android Version
HTC Incredible S	1GHz	768MB	ARMv7 rev1, with NEON	S-LCD, 480x800	2.3.7
Samsung Galaxy S2	Dual Core 1200 MHz	1024 MB	ARMv7 rev1, with NEON	Super Amoled Plus 480x800	2.3.6

Two test video sequences are used for evaluation. The first video sequence is a lecture recording with little motion, and the second video sequence contains constnt object motion. We refer them as video sequence 1 and video sequence 2 respectively in subsequent description. Both are 1080p HD videos.

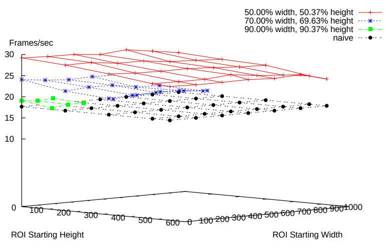
7.1 Frame Rate

Video playback frame rate is affected by both ROI sizes and ROI positions. We test the influence of each. We also measure the overhead by selective decoding in terms of frame rate. All the test for frame rate are done three times and the average value is recorded for analysis.

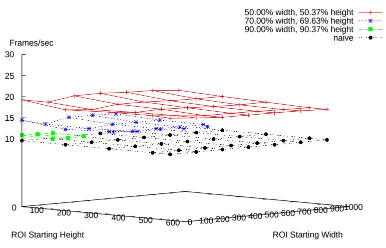
Different ROI Positions Different regions of a video frame may contain different amount of motions and dependencies, therefore the ROI position can affect the amount of processing at decoding and subsequently the video playback frame rate.

We fix the ROI size, and then move the ROI from the upper left corner to the lower right of the video frame.

The test are carried out for three different ROI sizes, with the ROI width and height set as 50%, 70%, and 90% of the original video's width and height. Below are the results, In figures above, each 3D interface indicatest the frame rate for a ROI size. The intersection in a surface indicate the start position of a particular ROI size.



(a) Phone B Video 1



(b) Phone B Video 2

Fig. 10. Frame Rate with 90%, 70%, and 50% of ROI at Different Positions

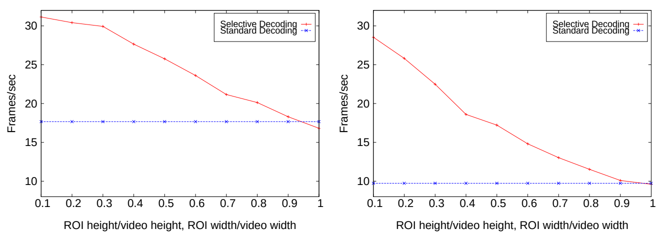
Looking at a single 3D interface at each figure, the frame rate tends to decrease when ROI starting witdth and/or starting height increases. This is because the intra-VOP dependencies increases towarding the lower right, and more dependencies cause more macroblocks to be selected and decoded, and thus the frame rat decreases. However, there are exceptions to this decrease trend, which are probably caused by different amount of motions and motion dependencies at different ROI positions.

Compare different 3D interfaces at a single figure, it is clear that the ROI size has a more significant effect than ROI positions. At 3 out of 4 test combinations, when ROI size is smaller than 90%, the frame rate of selective decoding is higher than standard MPEG4 SP decoding.

Compare figures for a single phone, the frame rate for video sequence 1 is higher than video sequence 2. This is expected because video sequence 1 has less amount of motion than video sequence 2. Compare figures for a single video sequence, the frame rate for phone A is lower than phone B. This is because phone B has more powerful hardware.

Different ROI Size Previous test already reveals that ROI size has significant influcence on video playback frame rate at selective decoding. This experiment examines the affection of ROI size more closely. We position the ROI at the center of the video frame and vary the ROI height and width from 10% to 100% of original video height and width. Below shows the experiment results, Looking at each figure, frame rate decreases as ROI size increases. At figure ??, selective decoding achieves higher frame rate at ROI size at around 80% or less. At the rest of the figures, selective decoding decodes faster at ROI size smaller than 90%. At 10% ROI, the frame rate on phone A is improved by 57.5% for video sequence 1, and 237.7% for video sequence 2, while the frame rate on phone B is improved by 76.3% and 193.3% respectively.

Compare figures for a single phone or a single video, we can draw similar conclusions to the ROI position experiment. In addition, the curves at different



(a) Phone B Video 1 (b) Phone B Video 2

Fig. 11. Frame Rate with 10% to 100% ROI Centered

figures differ due to different amount of motion at center of the video frame and ifferent phone processing capabilities.

Selective Decoding Overhead To evaluate the overhead of selective decoding, we set the ROI as the entire video frame.

Table 2. Full Frame Selective Decoding VS. Standard Decoding

Decoding Methods	Phone A Video 1	Phone A Video 2	Phone B Video 1	Phone B Video 2
Selective Decoding	10.57	6.30	16.81	9.60
Standard Decoding	11.82	6.50	17.67	9.73

Because of selective mask computation, selective decoding has lower playback frame rate than standard decoding. The overhead, however, is not signficant. In the four test combinations, the overhead is 10.58%, 3.08%, 4.87%, and 1.34% respectively. In practice, we can switch between selective decoding and standard decoding dynamically based on ROI sizes to achieve higher frame rate.

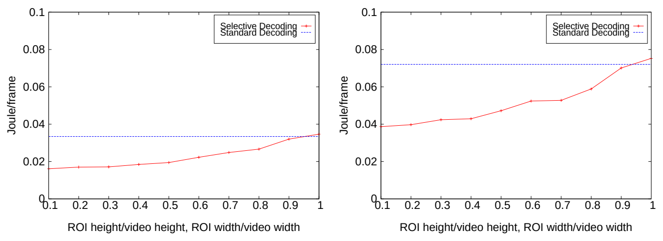
In summary, the experiments show that selective decoding can effectively increase frame rate at ROI sizes around 80% 90% or less.

7.2 Energy Consumption

Energy consumption is the other important asepct of our evaluation. Two experiments are done with different focus. The first experiment utilizes a third-party Android application PowerTutor[Zhang:2010:AOP:1878961.1878982] to measure the power consumption at hardware component level. The second experiment measures the percentage of battery power discharged when playing the two different video sequences repeatedly for certain number of times.

PowerTutor Measurements Battery energy is mainly consumed by display and CPU at video playback. PowerTutor, an Android power measurement tool, is capable of measuring power consumed by different hardware components of a specific app. In the experiment, we vary the ROI sizes from 10% to 100%. The frame rate is controlled so that both selective decoding and standard decoding always play at same frame rate. This is mainly because the CPU display power is dependent on time and display content and controlling the time is essential for a fair comparison. Each test is done three times and the average value is recorded.

For display measurements, we found the curves for selective decoding and standard decoding almost overlap, which means they consume similar amount of energy. However, this is not the case for CPU power consumption, which are shown as below. Looking at each figure individually, CPU power consumption



(a) Phone B Video 1 (b) Phone B Video 2

Fig. 12. CPU Power Consumption Per Frame

per frame at selective decoding increases when ROI size increases. At ROI size 90% or less, CPU power consumption at selective decoding is lower than standard decoding at all four test combinations. At figure ???, selective decoding has lower power consumption at 100%, this is probably because the MV decoding and DC&AC prediction coding avoided by selective decoding online computation weigh more than selective mask computation, which is the overhead due to selective decoding.

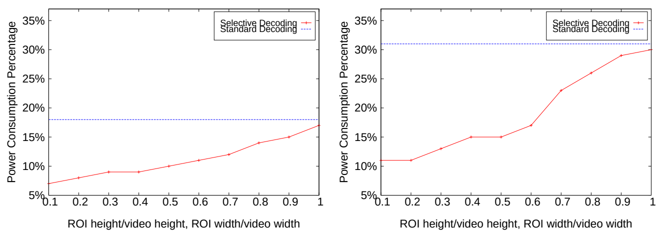
Compare figures for a single phone, video playback for video with heavy motion tends to consume more power because of more motion compensation decoding. Selective decoding tends to save more energy for the heavy motion video by reducing number of macroblocks decoded, and subsequently amount of motion compensation decoding.

Compare figures for a single video sequence, the amount of energy consumed by different phones differ. This is probably due to different hardware specifications.

The overall power consumption measurements follow the similar trend as CPU power consumption, therefore it is not shown here.

Power Drain Experiment PowerTutor measurments show selective decoding can save energy, mainly by reducing CPU power consumption. However, PowerTutor measurement is obtained through offline-training models[16] and may not be accurate for all phones. In addition, the measurements does not take into RAM and SDCARD access into consideration.

We designed and carried out another experiment to measure the power consumption. In this experiment, we place ROI of different sizes at the video frame center and play the video repeatedly for certain number of times while keeping the frame rate same for a single test combination. Based on the processing power and video sequence, we set the frame rate for phone A as 9 FPS and 6FPS, and frame rate for phone B as 15 FPS and 8FPS. The percentage of power drained is recorded for comparison. Before each test, we fully charge the phone battery to 100% and disable all background acitities including WiFi, Bluetooth, GPS, etc. All tests are repeated twice and the average is taken for plotting the graphs below, Looking at each figure individually, it is clear that selective de-



(a) Phone B Video 1 (b) Phone B Video 2

Fig. 13. Percentage of Power Drained

coding consumes less power when ROI sizes are below certain size. At 10% ROI size, the battery consumption on phone A is reduced by 27.8% and 24.8% for video sequence 1 and 2 respectively, while the battery consumption on phone B is reduced by 61.1% and 64.5%. Because we set the frame rate differently for different phones and video sequences, the comparison for different phones and video sequences are not fair and therefore avoided.

In summary, selective decoding proves to be efficient in terms of achiving higher frame rate and saving battery energy consumption when ROI size is below certain threshold.

8 Future Work

Selective decoding presented in this paper illustrates the idea of achieving more efficient zoomable video playback by tracing the dependencies among macroblocks. There are many possible future work can be done based on this simple idea.

8.1 Minimize Storage

The offline computation of selective decoding prodcues dependency files. The dependency files are plain values stored in binary format. To facilitate memory mapped I/O, heavy padding is introduced to make the records aligned. The table below shows the storage overhead for the two test video sequences. One

Table 3. Video Sizes and Dependency File Sizes

Test Video Sequences	Video File Sizes	Dependency File Sizes
Video 1	69.0MB	385.9MB
Video 2	88.3 MB	88.4 MB

simple idea to reduce storage overhead is to find the limits for stored values and allocate exact number of bits for storing those values. A more complicated but worth exploring approach is to find a better storage scheme to avoid the heavy padding but still be able to access the file quickly in a random fashion. Furthermore, compression can be applied to dependency files. Variable length codes and prediction codes can be used to encode the values. This introduces new computation overhead because the values are compressed at offline computation and de-compressed at online computation. However, it is possible this approach can save a fair amount of space.

Finally, if we expand the work to encoder and use encoder to generate dependency files, the video file size can be reduced since the duplicated information on dependency files are not needed in video files any more.

8.2 Optimize Dependency File Generation

Our work focuses on optimizing the online computation of selective decoding. The offline computation is carried out once and dependency files are saved. However, the ideal approach is to generate dependency files on the fly when the user is playing the video for the first time. We have worked on a prototype of this approach and it proves working. Because this process is not optimized, we exclude it from our evaluation.

In addition, as mentioned above, we can generate the dependency files when encoding the video to remove the dependency file generation overhead completely from decoder side.

8.3 Encoder Assisted Selective Decoding

Besides the advantages described above, expanding selective decoding to encoder can bring other research opportunities. The encoding parameters can be adjusted to control the amount of dependenices among macroblocks. However, this could also affect the compression efficiency. Can we configure the encoding parameters to produce videos that achieve an optimal balance bewtween compression efficiency and dependency reduction?

8.4 Selective Decoding in Network Streaming Context

Our work focuses on selective decoding in local playback context. However, it is possible to apply selective decoding in network streaming context. The server side could generate dependency files, compute the seletive masks based on ROI sent from client side, and reconstruct the video bitstream for the client side. Note that this differs from Khiem's [8] in the way that both encoder and decoder are controlled to work corporately. It is possible that this approach can achieve better bandwidth efficiency than Khiem's work because the decoder can skip the MBs accordingly to the selective masks while Khiem's work assumes a standard decoder.

9 Conclusion

HD video playback on mobile devices needs to be improved. Battery power expenditure is fast at HD video playback and many low end devices suffer from low frame rate.

Based on the fact that users are only interested in part of a video scene at some cases, we designed a software approach named selective decoding to reduce the battery power consumption and increase the frame rate for HD video playback on mobile devices.

Selective decoding is based on analyzing and tracing various intra- and inter-frame dependency relationships among macroblocks. By doing so, we compute a selective mask which indicates the macroblocks needed to present a clear scene in a user requested ROI. The selective mask instructs a modified decoder to decode the macroblocks selectively. The ROI can be captured using a multi-touch gesture interface, which is common in todays mobile devices.

References

1. businessinsider.com: Here's what people are actually doing with their cellphones. http://articles.businessinsider.com/2010-07-07/tech/30063182_1_google-android-phones-mobile-device-cellphone (07 2010)
2. Khiem, N.Q.M., Ravindra, G., Ooi, W.T.: Towards understanding user tolerance to network latency in zoomable video streaming. In: Proceedings of the 19th ACM international conference on Multimedia. MM '11, New York, NY, USA, ACM (2011) 977–980
3. Schwarz, H., Marpe, D., Wieg, T.: Overview of the scalable video coding extension of the h.264/avc standard. In: IEEE Transactions on Circuits and Systems for Video Technology In Circuits and Systems for Video Technology. (2007) 1103–1120
4. Mavlankar, A., Baccichet, P., Varodayan, D., Girod, B.: Optimal slice size for streaming regions of high resolution video with virtual pan/tilt/zoom functionality. In: Proc. of 15th European Signal Processing Conference (EUSIPCO. (2007)
5. Feng, W.C., Dang, T., Kassebaum, J., Bauman, T.: Supporting region-of-interest cropping through constrained compression. ACM Trans. Multimedia Comput. Commun. Appl. **7**(3) (September 2011) 17:1–17:16

6. Shimoga, K.B.: Region-of-interest based video image transcoding for heterogenous client displays. In: Packet Video 2002. (2002)
7. Fan, X., Xie, X., qin Zhou, H., ying Ma, W.: Looking into video frames on small displays. In: In Proc. of ACM Multimedia 2003, Press (2003) 247–250
8. Ngo, K.Q.M., Guntur, R., Ooi, W.T.: Adaptive encoding of zoomable video streams based on user access pattern. In: Proceedings of the second annual ACM conference on Multimedia systems. MMSys '11, New York, NY, USA, ACM (2011) 211–222
9. Mavlankar, A., Noh, J., Baccichet, P., Girod, B.: Peer-to-peer multicast live video streaming with interactive virtual pan/tilt/zoom functionality. In: in Proc. of IEEE International Conference on Image Processing (ICIP
10. Mavlankar, A., Varodayan, D., Girod, B.: Region-of-interest prediction for interactively streaming regions of high resolution video. In: In Proc. International Packet Video Workshop. (2007)
11. Bae, T.M., Thang, T.C., Kim, D.Y., Ro, Y.M., Kang, J.W., Kim, J.G.: Multiple region-of-interest support in scalable video coding. ETRI Journal (2006) 239–242
12. Android: Android official website. http://www.android.com/ (05 2012)
13. ffmpeg: Ffmpeg. http://ffmpeg.org/index.html (05 2012)
14. libyuv: libyuv. http://code.google.com/p/libyuv/ (5 2012)
15. chromium: chromium. http://code.google.com/p/chromium/issues/detail?id=71403 (5 2012)
16. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R.P., Mao, Z.M., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. CODES/ISSS '10, New York, NY, USA, ACM (2010) 105–114