Adv MM Assignment #2 – x264 with different Motion Estimation (ME) algorithms

Kai Wen Yoke, 2020280598
Khang Hui Chua, 2020280442

December 27, 2020

1 Introduction

In this assignment, we explore the effects on `x264`'s video encoding performance of (1) different Motion Estimation (ME) algorithms on `x264` and (2) different video types. This also gives us greater insight into why the different presets performed as they did in Assignment #1, as the different presets used different ME parameters.

1.1 Background on motion estimation

Motion estimation is a critical part of video encoding and exploits high temporal redundancy between successive frames in videos. During video encoding, motion vectors that describe the transformation of one 2D image frame to a neighbouring frame are estimated, and then during decoding, motion compensation is applied to synthesize the next frame from the estimated motion vectors. Almost all video encoding standards use block-based motion estimation and compensation methods, including H.264. Block-matching motion estimation involves dividing the current frame into macroblocks (typically of size 16 pixels), and finding similar blocks within a predefined search range (typically +-7 pixel search window) in neighbouring frames, and generating motion vectors that model the movement of the macroblock from one location to another. `x264` provides 5 block-matching motion estimation methods: diamond (`dia`) search, hexagon (`hex`) search, uneven multi-hexagon (`umh`) search, exhaustive (`esa`) search and transformed exhaustive (`tesa`) search. `dia, hex, umh` search involves searching for matching blocks in a heuristic pattern corresponding to their names, and are much faster than `esa, tesa` which search the entire motion search space.

2 Methodology

2.1 Parameters used when running `x264`

In `x264`, there are a number of parameters related to ME, including `--me`, `--subme`, `--merange` and `--mvrange`. For this assignment, we only varied `--me` and kept other related parameters to default values. There are 5 `--me` types in total: `MES='dia hex umh esa tesa'`. In theory, the different `--me` methods are ordered in terms of increasing time complexity and performance. We test out the ME methods on two sets of 10 videos (video sources explained later) because we had suspicious results for the first set and we wanted to verify again with a second set. For the first

video set sourced from vimeo/youtube (`.mp4`), since the ten videos had different bitrates, to ensure fairness when comparing across different videos, we set `x264` with a target average bitrate of 0.8 of the original average video bitrate. For the second video set, all ten videos were in uncompressed forms (`.yuv`) and hence had the same bitrate (2986 Mbps), so we set the output average bitrate to a fixed value of 8Mbps that is more reasonable for typical video broadcasting applications. We then evaluate the visual quality of encoded videos using SSIM and PSNR metrics output by `x264`.
In summary, we ran the following commands:
(1) video set 1: `x264 --bitrate $BITRATE_KBPS_TARG --me $ME --no-progress --ssim --psnr -o $FILENAME_OUT $FILE`
(2) video set 2: `x264 --bitrate 8000 --me $ME --no-progress --ssim --psnr -o $FILENAME_OUT --fps 30 --input-res 3840x2160 $FILE`

2.2 Details of sample videos chosen

Here, we describe in greater detail video sets 1 and 2. We got another video set 2 because we were worried the way we selected source videos in set 1 was not correct, as the source videos downloaded from vimeo/youtube were already compressed, and then we are applying more compression using `x264`, which could lead to result distortion. Hence, for video set 2, we found completely uncompressed videos and tested `x264` on it.

2.2.1 Video Set 1: vimeo/youtube

Video set 1 is the same as what we used for AS1. From vimeo and youtube, we selected 10 sample videos to represent a variety of video types, each of approximately 4k resolution, and trimmed to roughly 10s. The trimmed source videos can be found here: https://cloud.tsinghua.edu.cn/d/c4e2cb11d7f349db9315/. However, the videos downloaded from different sources in vimeo/youtube were already in compressed forms (not sure how they were compressed), and have varying bitrate, fps, file sizes etc, which could make it difficult to make a fair comparison of `x264` encoding performance on the different videos. The content variety is shown in Figure 1.

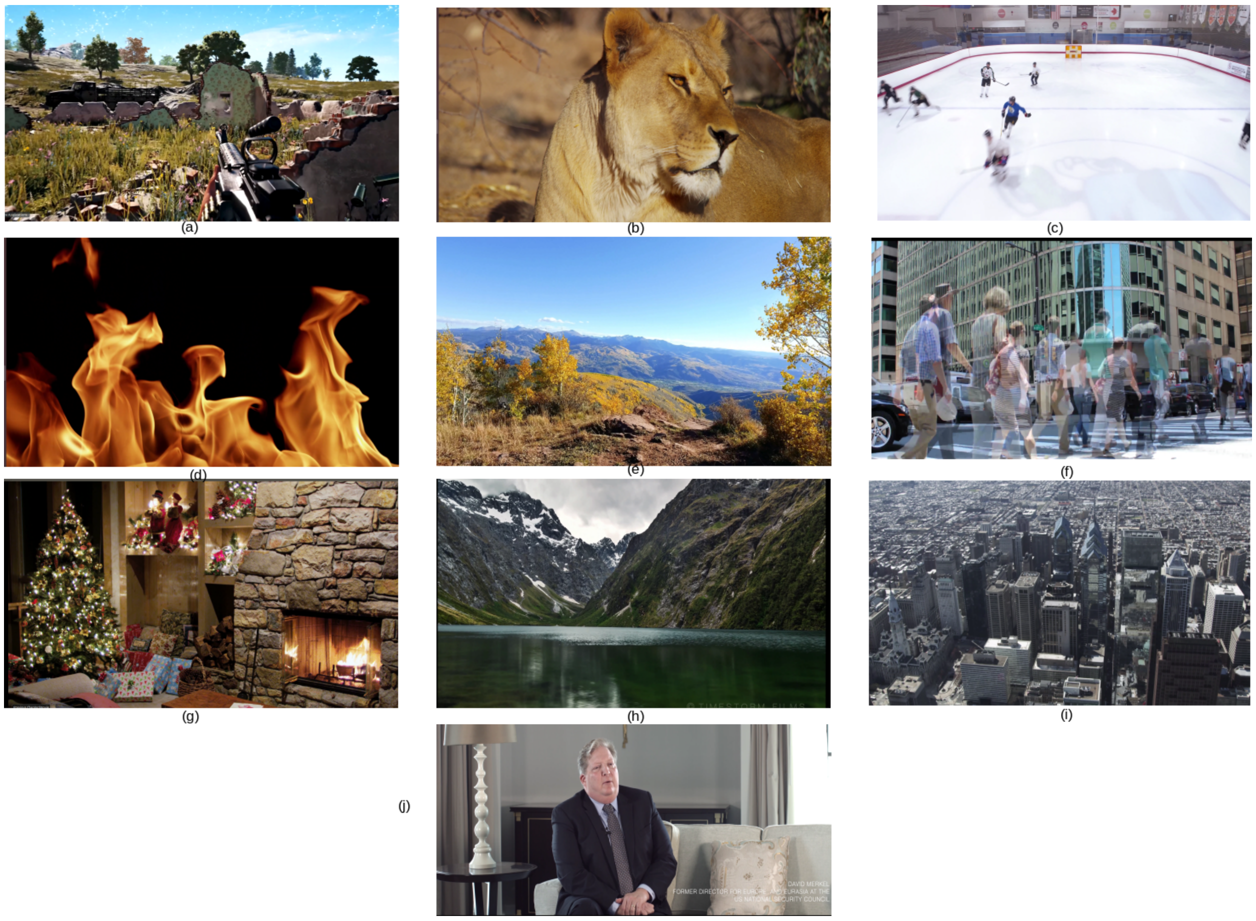
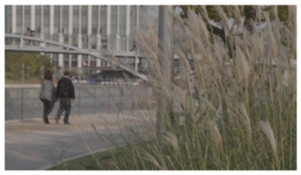


Figure 1: Content range of 10 sample videos: (a) PUBG gameplay; (b) Lion; (c) Hockey; (d) Fire; (e) Drone flying over mountains; (f) City streets timelapse; (g) Christmas at home; (h) Water and clouds in New Zealand; (i) Aerial view of Philadelphia; (j) Interview

2.2.2 Video set 2: Uncompressed videos

Video set 2 is taken from http://mcml.yonsei.ac.kr/downloads/4kuhdvideoquality and is a 4K UHD video database created by researchers from the MCML group at Yonsei University. All the source videos have 4K (3840x2160) resolution, 30 fps, 300 frames. They were saved in uncompressed forms in `.yuv` files, with each video sequence being 3.7GB in size. When running `x264` on these videos, `x264` assumes the files to be in YUV i420 format (i.e. 12 bits per pixels) which gives the video a bitrate of $3840 \times 2160 \times 30 \times 12 \times \frac{1}{10^6} = 2986$ Mbps.


(a) *Park*

(b) *Lake*

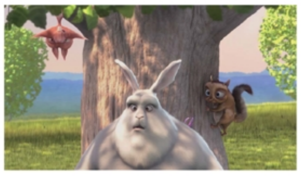
(e) *Construction*

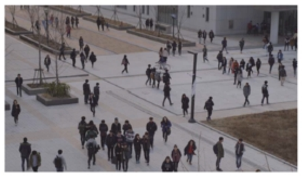
(f) *Maples*

(c) *Basketball*

(d) *Flowers*

(g) *Dolls*


(h) *Bunny*


(i) *Crowd*

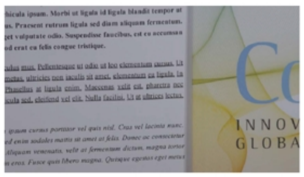

(j) *Characters*

Figure 2: Content range of video set 2

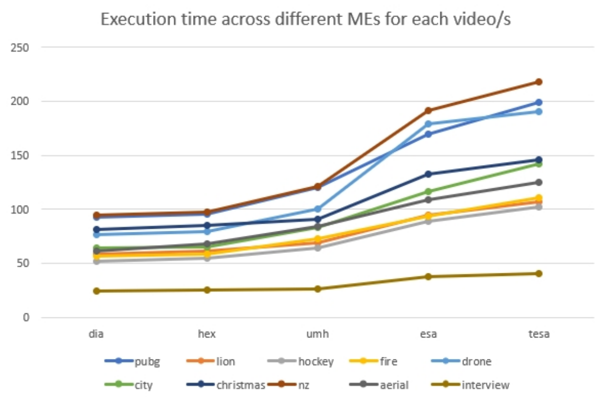
1. Park: Shot at a park on a sunny morning. Reeds sway in the wind and two people walk along the way.
2. Lake: Shot at a park. A small boat on the lake passes and mild waves occur in the wind.
3. Basketball: Shot at an indoor basketball court. Players run and jump to take the ball.
4. Flowers: Shot in a garden in the afternoon. Yellow flowers sway in the wind.
5. Construction: Shot at a construction site. Workers and a forklift move into the building.
6. Maples: Shot in front of a building. The leaves of the maple trees sway in the wind. The camera pans from left to right.
7. Dolls: Shot at a studio with tungsten light. The lion and bear dolls move from right to left.
8. Bunny: This content is a part of “Big Buck Bunny” (animation). The Bunny looks at a flying butterfly and other characters around Bunny.
9. Crowd: Shot at a university campus in a cloudy morning. Many students walk through the street.
10. Characters: Shot at a studio with tungsten light. There are fine text and colorful patterns on a paper. The camera pans from left to right.

3 Comparing encoding performance

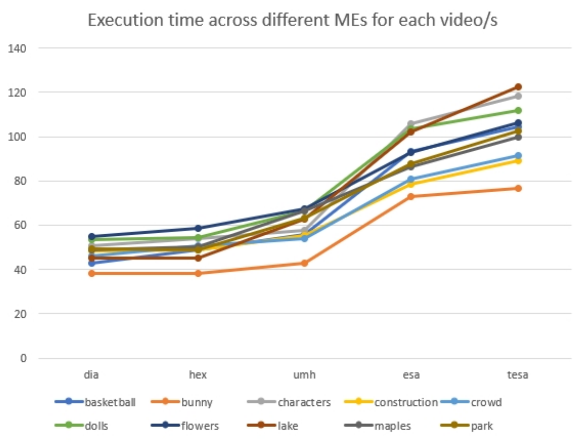
For video set 1, we set target output bitrate to be 0.8 of original bitrate, and for video set 2, we set target output bitrate to 8Mbps. We compare the execution time and visual quality (SSIM, PSNR) of the different ME methods on each video.

3.1 Comparing across MEs

3.1.1 Execution time



(a) Video set 1

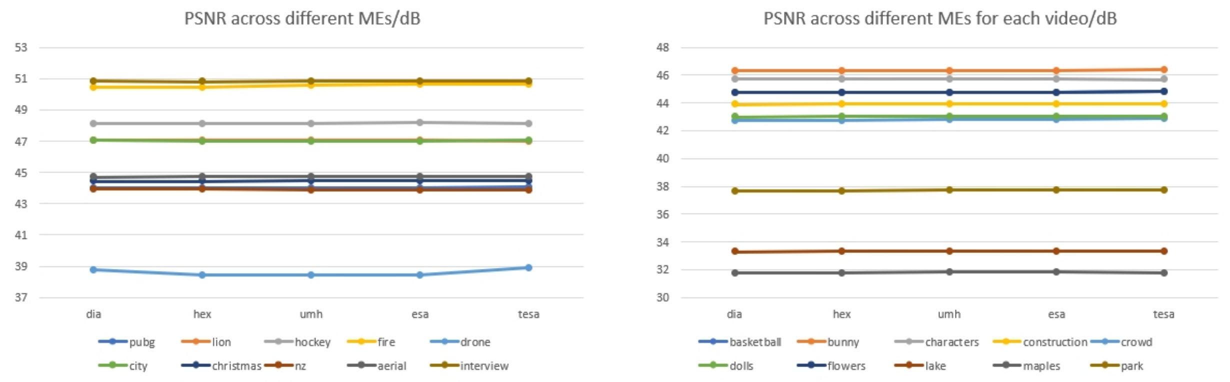


(b) Video set 2

Figure 3: Execution time across different ME methods

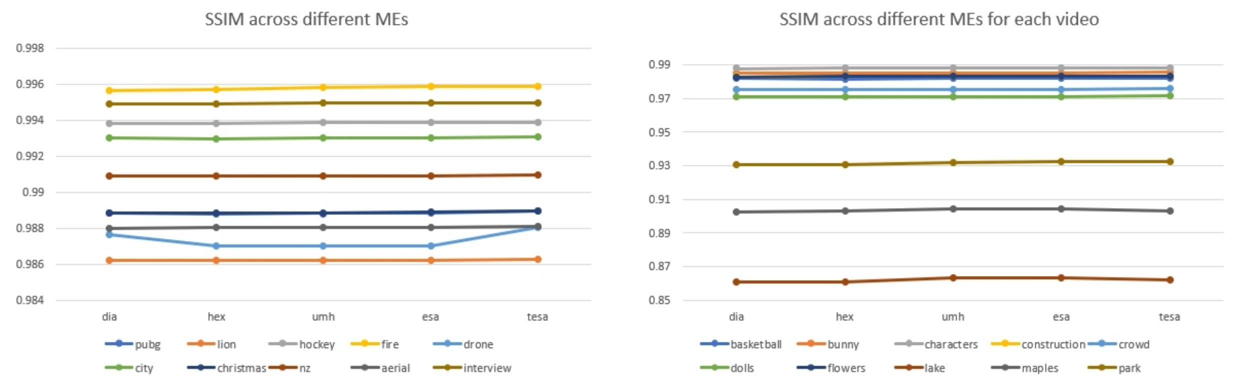
Both video sets show a clear trend of the increasing complexity of each ME method as expected, particularly at the transition between `dia, hex, umh` and `esa, tesa` which marks the transition from requiring only a limited number of search points to fully searching all points in the search window.

3.1.2 Visual quality



(a) Video set 1 (b) Video set 2

Figure 4: PSNR across different MEs



(a) Video set 1 (b) Video set 2

Figure 5: SSIM across different MEs

The visual quality metrics shown in Figure 4 and 5 were very unexpected. Initially, when we noticed no significant improvement in performance for video set 1, we thought it could be because source videos in video set 1 were already compressed beforehand. Then, we performed the same tests on video set 2, and we still got the same result - little improvement in visual quality as the ME method increase in complexity. This is very odd, as we would expect `esa, tesa` to predict motion vectors with greater accuracy than `dia, hex, umh` and thus result in better motion estimation. Instead, this experiment suggests that heuristics based methods that reduce number of search points for motion estimation work as well as exhaustive search, as validated by two video datasets. Hence, there is no point using `esa, tesa` which takes a much longer time and yet does not yield significant performance improvement.

3.2 Comparing across different video content type

Here, we comment mainly on video set 2 which is a more standardized dataset than video set 1, and hence we can more fairly attribute differences in visual quality to the video content type, and not other factors like original bitrate and fps etc.

3.2.1 Amount of motion in each video

We expect ME to work the best when there is higher temporal redundancy between successive frames (i.e. for low motion videos). This is somewhat observed. For both PSNR and SSIM, *Park, Lake, Maples* had far worse visual quality than the rest (see Figures 4, 5). These three videos have the common characteristic of continuously moving small details, such as the reeds in the park, the water in the lake, and the leaves of the maples. Motion of large objects like people playing basketball and large flowers swaying does not seem to cause much problems. Perhaps this is due to the macroblock size not being small enough to capture the motion vectors of fine details, hence we are not able to reduce temporal redundancies in this case. Conversely, for *Characters, Bunny, Flowers*, they exhibit better visual quality scores, because they do not have very fast motion - merely a camera panning slowly over a sheet of printed words, large animated characters moving normally, and large flowers swaying. Hence, in this case, temporal redundancies can be exploited.

4 Conclusion

The number of search points required in a ME algorithm greatly influences the speed of encoding, but does not have a significant impact on the visual quality of encoding, hence using the faster ME methods `dia, hex, umh` is sufficient. Also, the visual quality of the encoded result depends on the amount of motion in each video, with less motion resulting in greater temporal redundancy, and motion of small details being difficult to capture.