Encoding AV1 with Open Source Alternatives

Jan Ozer
janozer@gmail.com

Agenda

- Producing Libaom-AV1
- Producing SVT-AV1
- How they compare

FFmpeg libaom-AV1

- Overview
- Encoding basics
  - Encoding string starting point
  - Passes
  - Presets
  - Threads

Overview

FFmpeg uses the libaom-av1 codec from the Alliance for Open Media.

FFmpeg documents the basic switches here (bit.ly/ffmpeg_av1). You can find basic command strings at the FFmpeg Wiki.

Encoding Basics

FFmpeg can accept almost any file input; no need to convert to YUV/Y4M

Two-line encoding string for target bitrate (bit.ly/av1_guide)

```
ffmpeg -i input.mp4 -c:v libaom-av1 -b:v 2M -pass 1 -an -f null /dev/null && \
ffmpeg -i input.mp4 -c:v libaom-av1 -b:v 2M -pass 2 -c:a libopus output.mkv
```

- `-c:v libaom-av1` - use the libaom-av1 codec
- `-b:v` - bitrate video
- `-an` - No audio
- `-pass 1 -f null` - first pass no output (Linux)
- `-pass 2` - second pass
- `output.mkv` - output file

Encoding Basics

Setting maximum bitrate, and I-frame parameters

```
ffmpeg -y -i Football_10.mp4 -c:v libaom-av1 -b:v 1500K -g 60 -
keyint_min 60  -pass 1 -f matroska NUL & \

ffmpeg -y -i Football_10.mp4 -c:v libaom-av1 -b:v 1500K -maxrate 3000K
-g 60 -keyint_min 60  -pass 2 Football_1.mkv
```

- `-g 60` - GOP size (I-frame interval). Setting to 60 frames (2-seconds for 30 fps file).
- `-keyint_min 60` - set minimum keyframe interval at GOP size (essentially telling FFmpeg not to insert I-frames at scene changes)
- `-f matroska` - Identifying format of final file (typical approach)
- `-maxrate 3000K` - Setting maximum rate
- `-pass 2` - second pass
- `output.mkv` - output file

Decision: One or Two Passes

1-Pass

```
ffmpeg -y -i input.mp4 -c:v libaom-av1 -b:v 1500K -maxrate 3000K -g 60
-keyint_min 60 -cpu-used 4 output_1pass.mkv
```

2-Pass

```
ffmpeg -y -i input.mp4 -c:v libaom-av1 -b:v 1500K -g 60 -keyint_min 60
-cpu-used 8 -pass 1 -f matroska NUL & \

ffmpeg -y -i input.mp4 -c:v libaom-av1 -b:v 1500K -maxrate 3000K -g 60
-keyint_min 60 -cpu-used 4 -pass 2 output_2pass.mkv
```

Decision 1: One Pass or Three-Pass

Feature Analysis:

- Encode two/three files using different alternatives (here 1pass/2pass)
- Measure
  - Encoding time
  - Bitrate/bitrate accuracy
  - VMAF (overall quality)
  - Low-frame VMAF (transient issues)
  - Standard deviation (variability)
  - Green is good/yellow bad

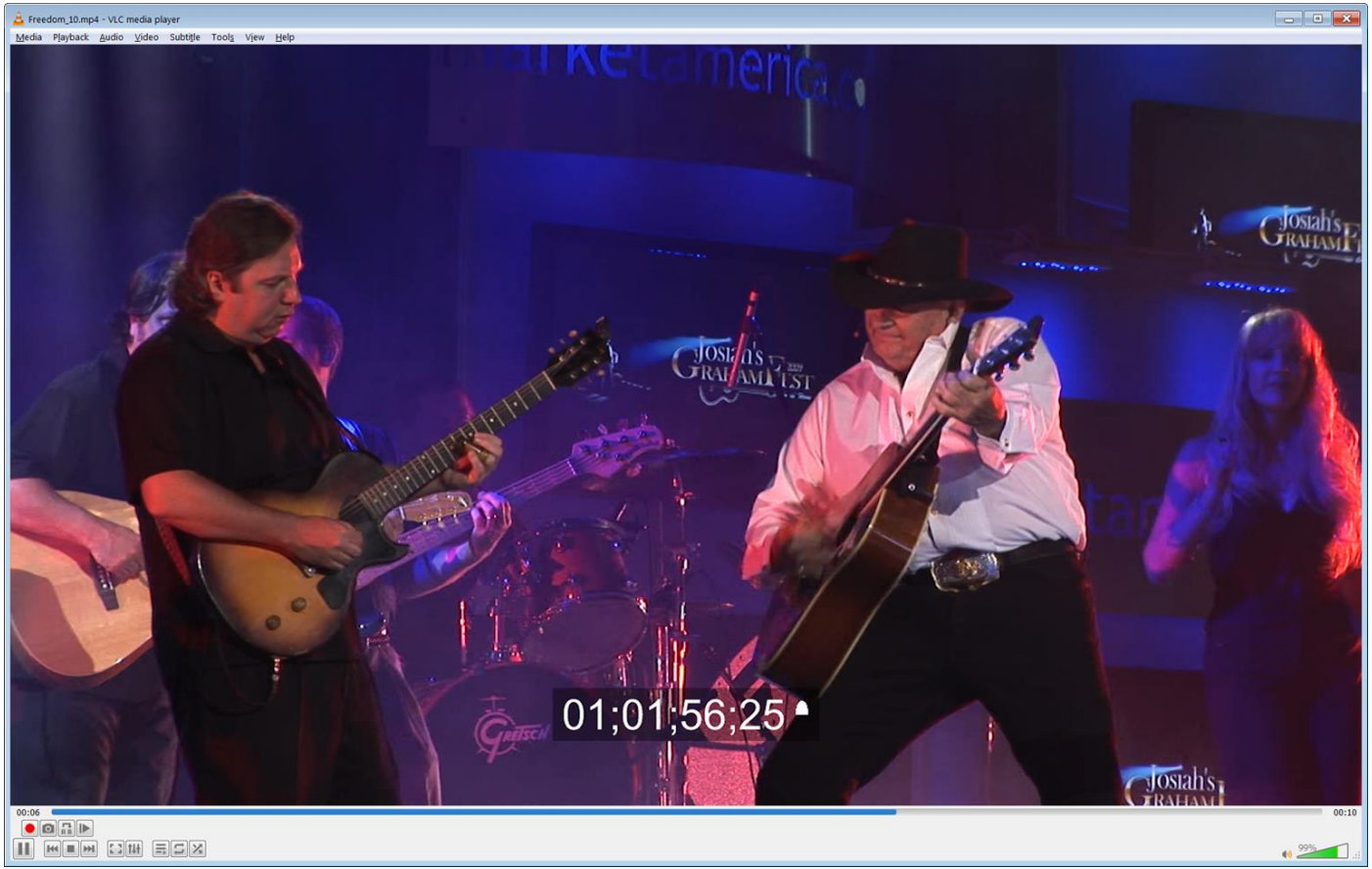
Single/Two Pass

Encoding time	1-pass	2-pass	Delta
Freedom	0:05:38	0:05:14	7.10%
Football	0:03:55	0:03:56	0.42%
Easy Hard	0:06:57	0:06:29	6.71%
Average	0:05:30	0:05:13	5.15%
bitrate	**1-pass**	**2-pass**	**Delta**
Freedom	1,478	1,479	0.07%
Football	1,494	1,528	2.23%
Easy Hard	1,513	1,610	6.02%
Average	**1,495**	**1,539**	**2.86%**
Delta from 1500	**-0.33%**	**2.60%**	
VMAF	**1-pass**	**2-pass**	**Delta**
Freedom	84.81	85.15	0.39%
Football	82.18	85.10	3.43%
Easy Hard	83.02	87.30	4.91%
Average	**83.34**	**85.85**	**2.93%**
Low Frame	**1-pass**	**2-pass**	**Delta**
Freedom	76.91	77.58	0.87%
Football	58.55	66.41	11.83%
Easy Hard	54.91	67.92	19.15%
Average	**63.46**	**70.64**	**10.16%**
Standard Deviation	**1-pass**	**2-pass**	**Delta**
Freedom	3.69	3.23	12.42%
Football	10.47	6.74	35.59%
Easy Hard	13.29	9.51	28.45%
Average	**9.15**	**6.49**	**29.02%**

Football

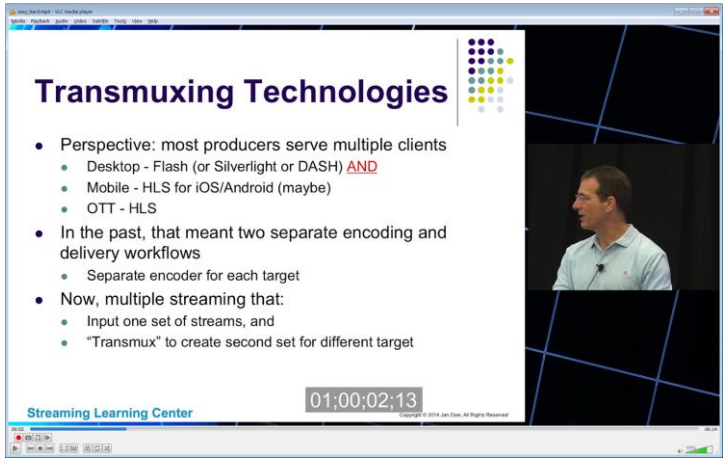
Source: Harmonic Inc

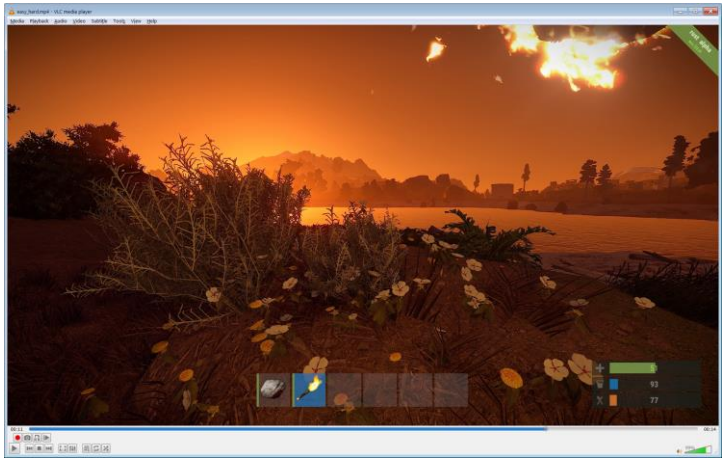
Freedom



Source: Josiah Weaver concert

Easy Hard





- 6 seconds tutorial, 6 seconds Rust computer game (very hard to encode)
- Why? Test bitrate control

Decision 1: One Pass or Three-Pass

Feature Analysis:

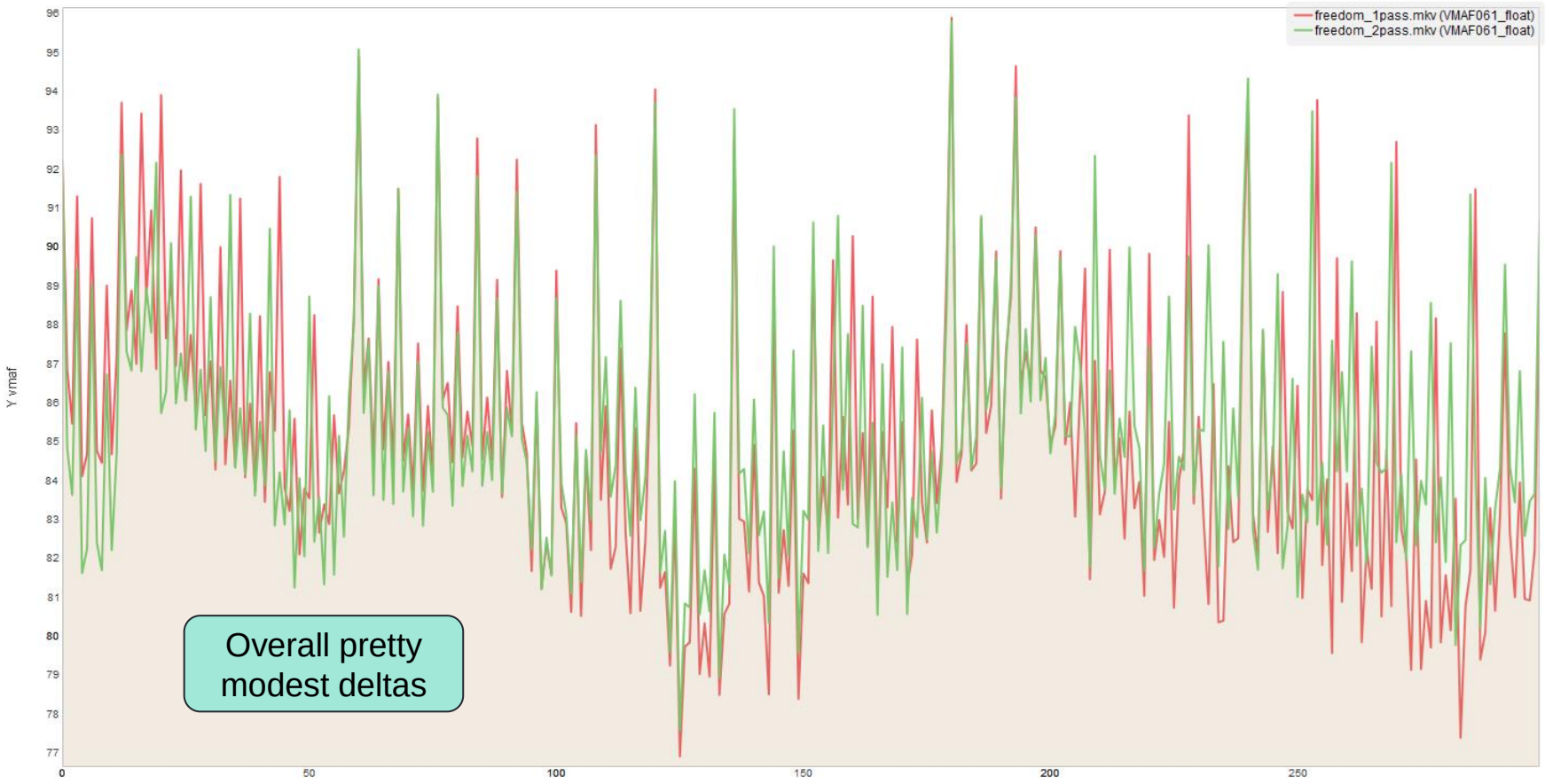
- Time:
  - One-pass is 5% longer
- Bitrate:
  - 1-pass does a better job achieving target bitrate (lower delta from 1500 target)
    - Had to adjust Football to lower bitrate
- VMAF - average – two-pass is 2.5 higher
- Low-frame (transient issues)
  - 2-pass is meaningfully lower on all clips, particularly easyhard.
- Standard deviation (quality variability)
  - 1-pass substantially higher

Single/Two Pass

Encoding time	1-pass	2-pass	Delta
Freedom	0:05:38	0:05:14	7.10%
Football	0:03:55	0:03:56	0.42%
Easy Hard	0:06:57	0:06:29	6.71%
Average	0:05:30	0:05:13	5.15%
bitrate	**1-pass**	**2-pass**	**Delta**
Freedom	1,478	1,479	0.07%
Football	1,494	1,528	2.23%
Easy Hard	1,513	1,610	6.02%
Average	**1,495**	**1,539**	**2.86%**
Delta from 1500	**-0.33%**	**2.60%**	
VMAF	**1-pass**	**2-pass**	**Delta**
Freedom	84.81	85.15	0.39%
Football	82.18	85.10	3.43%
Easy Hard	83.02	87.30	4.91%
Average	**83.34**	**85.85**	**2.93%**
Low Frame	**1-pass**	**2-pass**	**Delta**
Freedom	76.91	77.58	0.87%
Football	58.55	66.41	11.83%
Easy Hard	54.91	67.92	19.15%
Average	**63.46**	**70.64**	**10.16%**
Standard Deviation	**1-pass**	**2-pass**	**Delta**
Freedom	3.69	3.23	12.42%
Football	10.47	6.74	35.59%
Easy Hard	13.29	9.51	28.45%
Average	**9.15**	**6.49**	**29.02%**

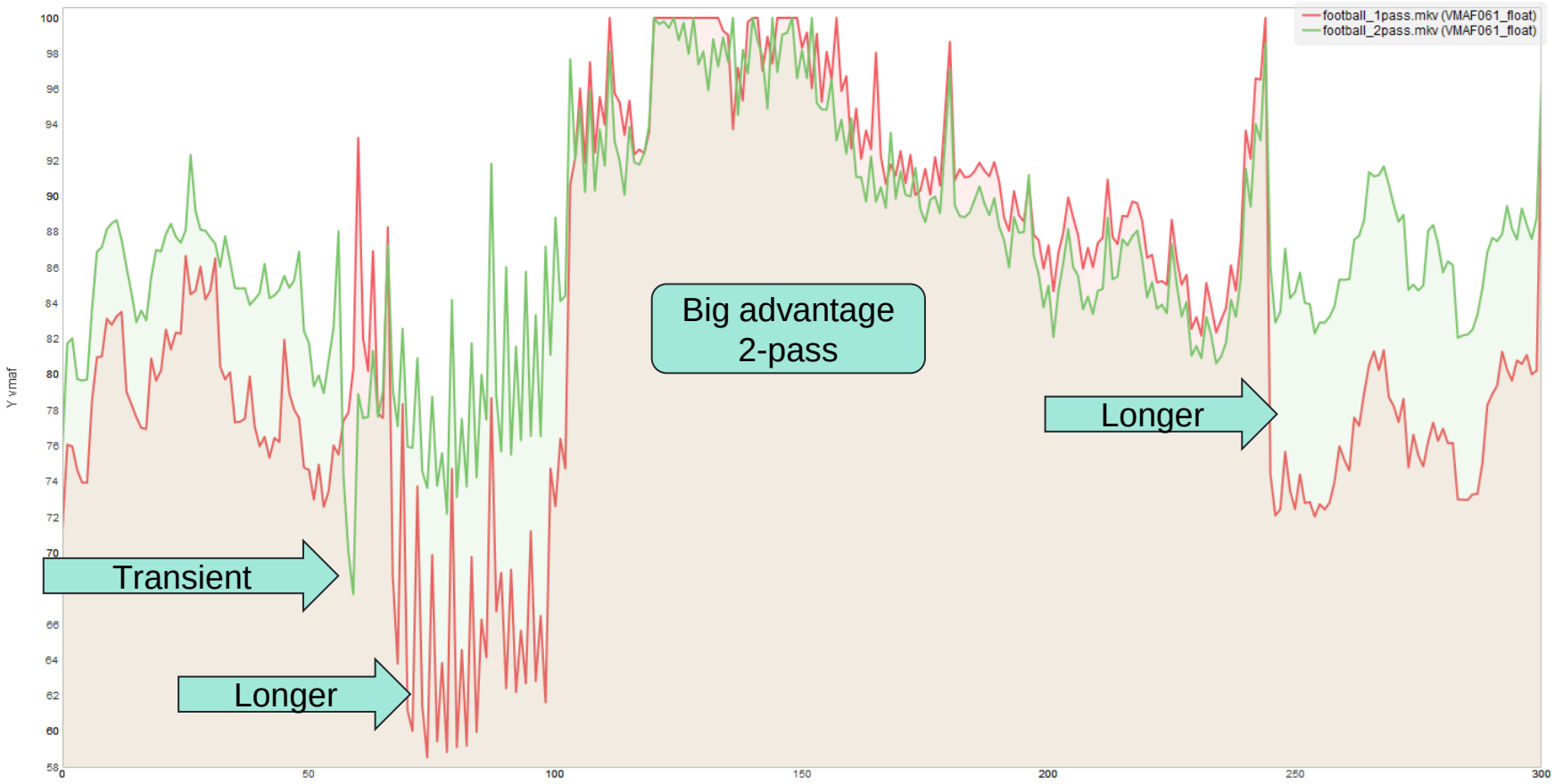
Results Plots and Frames - Freedom

Red = 1-Pass
Green = 2-pass



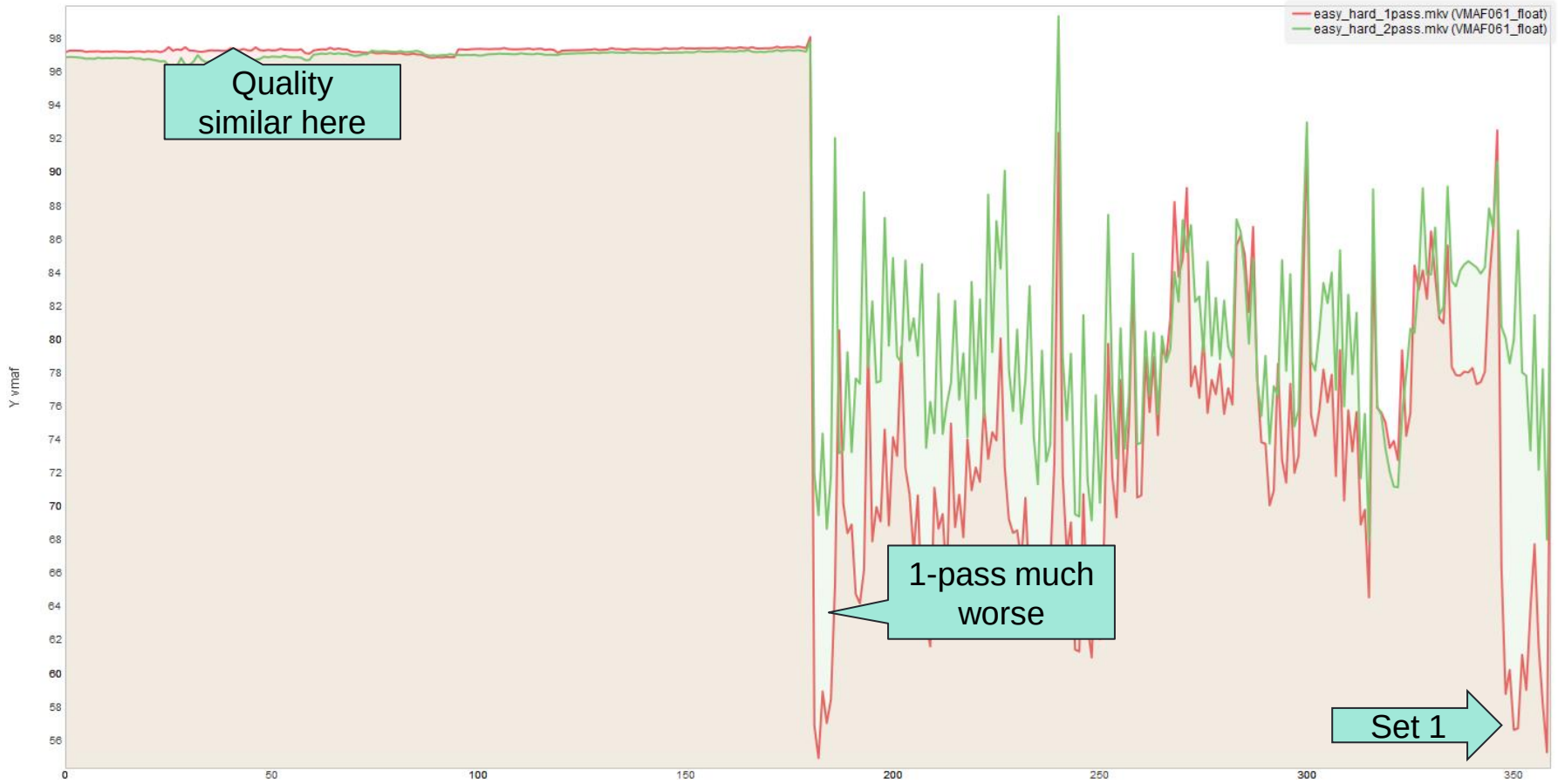
Results Plots and Frames - Football

Red = 1-Pass
Green = 2-pass



Results Plots and Frames - Easy_Hard

Red = 1-Pass
Green = 2-pass



Data Rate Allocation

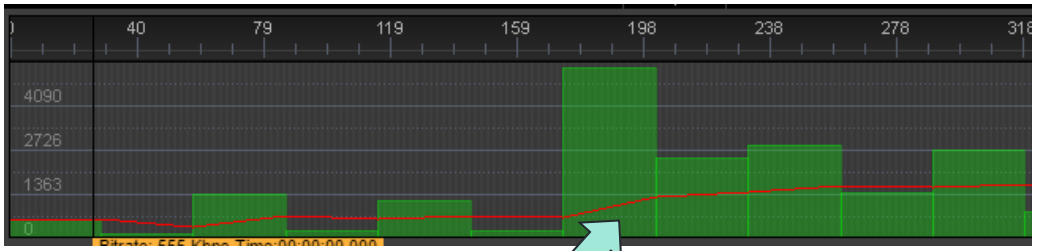
- Classic 1-pass vs. multiple-pass
- 1-pass - don't know where difficult sections are
- Multiple-pass
  - Scan once, ID easy and hard regions
  - Then encode

1-Pass

2571

Substantial data wasted here

2-Pass



4090

Data allocated to challenging sections


E:\archives\AV1_Trials\AV1_FFmpeg_2022\single_multiple\easy_hard2.mp4
1920x1080
Source
Y
Netflix VMAF VMAF061_float 1-st proc 56.603668
Netflix VMAF VMAF061_float 2-nd proc 79.980904
00:00:11.20+360
x: 267
y: 213
Y: 128
V: 129


E:\archives\AV1_Trials\AV1_FFmpeg_2022\single_multiple\easy_hard_1pass.mkv
1920x1080
1-Pass
Y
Netflix VMAF VMAF061_float 1-st proc 56.603668
Netflix VMAF VMAF061_float 2-nd proc 79.980904
00:00:11.20+360


E:\archives\AV1_Trials\AV1_FFmpeg_2022\single_multiple\easy_hard_2pass.mkv
1920x1080
2-Pass
Y
Netflix VMAF VMAF061_float 1-st proc 56.603668
Netflix VMAF VMAF061_float 2-nd proc 79.980904
00:00:11.20-350

Decision 1: One Pass or Two-Pass

Feature Analysis:

- Time:
  - One-pass is 5% higher
- Bitrate:
  - 2-pass does a better job achieving target bitrate (lower delta from 1500 target)
- VMAF - average - 2.5 higher (mostly easyhard)
- Low-frame (transient issues)
  - 2-pass is meaningfully lower on all clips, particularly easyhard.
- Standard deviation (quality variability)
  - 1-pass substantially higher

Conclusion: 2-Pass where available (not live)

Single/Two Pass

Encoding time	1-pass	2-pass	Delta
Freedom	0:05:38	0:05:14	7.10%
Football	0:03:55	0:03:56	0.42%
Easy Hard	0:06:57	0:06:29	6.71%
Average	0:05:30	0:05:13	5.15%
bitrate	**1-pass**	**2-pass**	**Delta**
Freedom	1,478	1,479	0.07%
Football	1,494	1,528	2.23%
Easy Hard	1,513	1,610	6.02%
Average	**1,495**	**1,539**	**2.86%**
Delta from 1500	**-0.33%**	**2.60%**	
VMAF	**1-pass**	**2-pass**	**Delta**
Freedom	84.81	85.15	0.39%
Football	82.18	85.10	3.43%
Easy Hard	83.02	87.30	4.91%
Average	**83.34**	**85.85**	**2.93%**
Low Frame	**1-pass**	**2-pass**	**Delta**
Freedom	76.91	77.58	0.87%
Football	58.55	66.41	11.83%
Easy Hard	54.91	67.92	19.15%
Average	**63.46**	**70.64**	**10.16%**
Standard Deviation	**1-pass**	**2-pass**	**Delta**
Freedom	3.69	3.23	12.42%
Football	10.47	6.74	35.59%
Easy Hard	13.29	9.51	28.45%
Average	**9.15**	**6.49**	**29.02%**

Key Performance/Quality-Related Parameters

aomenc Command	Function	Strategy	FFmpeg Equivalent	Default?
`-cpu-used`	Encoding time/ quality tradeoff	Choose best option (3 appears to be it)	`-cpu-used`	8
`-auto-alt-ref`	Enable automatic alt reference frames	Ensure enabled	`-auto-alt-ref`	On by default
`-threads`	Set number of threads used	Set desired limit	`-threads`	Not specified (seems like up to 8)
`-tile-columns 1 --tile-rows 0`	Divides frame into sections for faster encoding	Use if helpful	same	Not deployed
`-row-mt`	Enable row based multi-threading	Ensure enabled	`-row-mt`	on by default
`-lag-in-frames`	Number of frames to look ahead at for alternate reference frame selection	2	`-lag-in-frames`	listed as -1 Appears to be 25

Decision: Choosing a Preset

- Preset - controls quality/encoding time tradeoff
- Tests
  - Encode 3 clips to all presets
  - Record encoding time, VMAF and low-Frame VMAF
- Encoding parameters (results will change if you use different parameters)
  - Two-pass (significant difference)
  - VBR
  - 2-second keyframe
- Test files will also make a difference
- So, customize for your files and encoding parameters as much as possible before performing these test

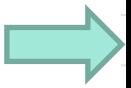
```
ffmpeg -y -i input.mp4 -c:v libaom-av1 -b:v 1500K -g 60 -keyint_min 60  -
cpu-used 8 -pass 1 -f matroska NUL & \

ffmpeg -y -i input.mp4 -c:v libaom-av1 -b:v 1500K -maxrate 3000K -g 60 -
keyint_min 60 -cpu-used 4 -pass 2 output_2pass.mkv
```

Fastest first pass

Cycle all second pass

Presets

- Total time for all three files (other tests done with 2 ten-second files)
  - Time
  - VMAF
- Max delta
  - 2.39% for VMAF
  - 3.57% for Low-Frame
- SVT-AV1 has much greater range of scores and times
- At fastest (preset 8), it takes 96:58 minutes to encode 8:25 (min:sec), so at best ~11.5x real time
  - SVT AV1 offers multiple presets that are faster than real time, albeit with much greater quality swings



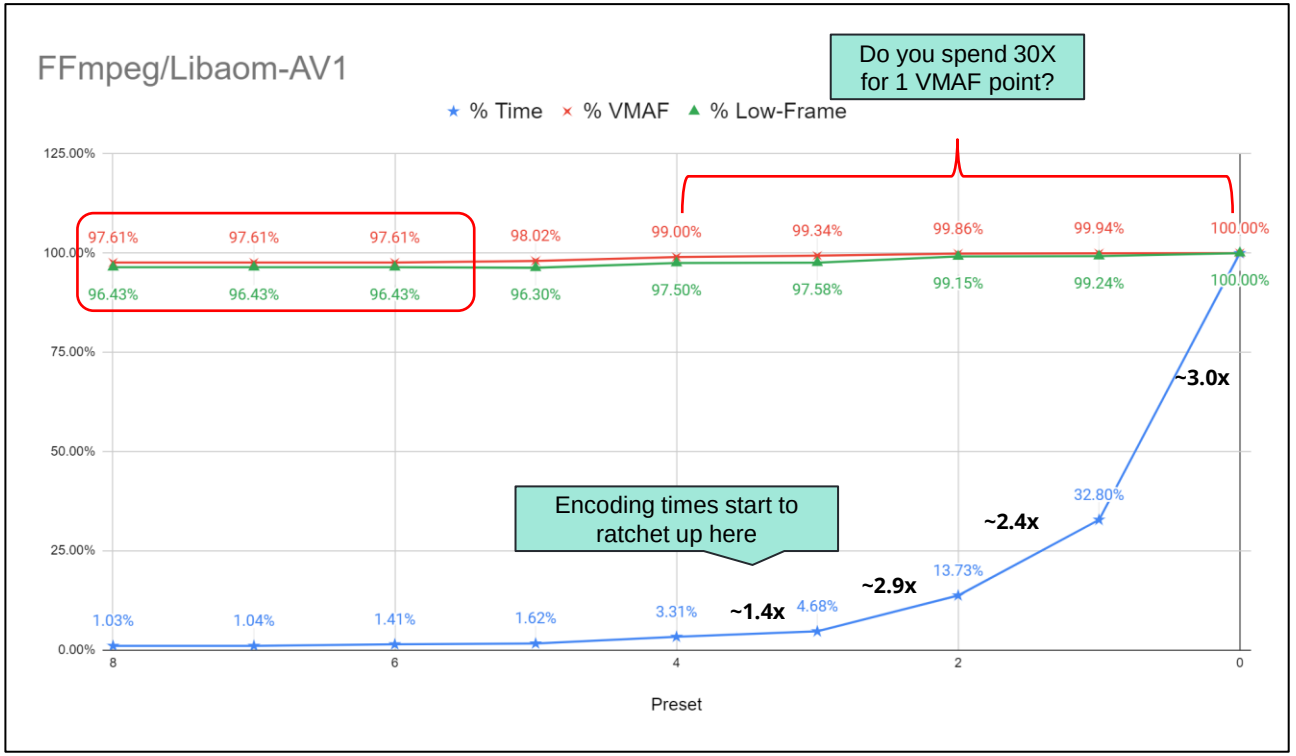
Preset	Encode Time 8:25 minutes	VMAF	Low-Frame
0	140:08:44	94.46	82.70
1	45:57:43	94.40	82.07
2	19:14:17	94.33	82.00
3	6:33:49	93.83	80.69
4	4:38:34	93.51	80.63
5	2:16:32	92.59	79.64
6	1:58:20	92.20	79.74
7	1:27:28	92.20	79.74
8	1:26:58	92.20	79.74
Delta	138:41:46	2.26	2.95
% Delta	**98.97%**	**2.39%**	**3.57%**

Delta - SVT-AV1	5.96%	14.79%

Which Preset? (Default =1)

Analysis:

- Convert time and scores to % of 100%
- Really tight range of quality
  - 97.61% is lowest quality
  - Bottom 3 are identical
- Encoding times blow up after 4
  - Between 3/4 looks like a sweet spot
- Used 4 for my comparisons

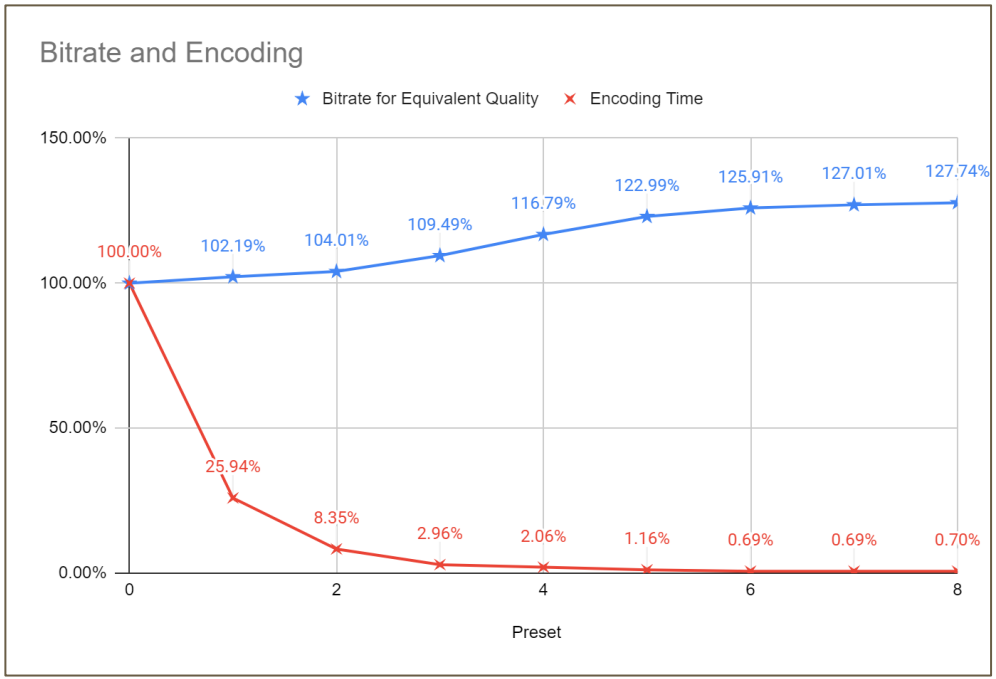


Presets - Cost Per Hour

- Cost per hour - single file
  - Multiply by ~3-4 for full ladder
  - Lower resolutions will encode more quickly
- I used 4 for testing
  - Cost/hour FFmpeg - $9.00
  - Cost/hour SVT-AV1 - $0.44
    - Preset 7

Preset	Minutes per minute	Minutes Per hour (times 60)	Hours of encoding/per video hour	Cost per hour - c6g.2xlarge	Cost per Hour
0	999.1	59,944	999.1	$0.272	$271.74
1	389.6	23,373	389.6	$0.272	$105.96
2	137.1	8,228	137.1	$0.272	$37.30
3	46.8	2,807	46.8	$0.272	$12.72
4	33.1	1,986	33.1	$0.272	$9.00
5	16.2	973	16.2	$0.272	$4.41
6	14.1	844	14.1	$0.272	$3.82
7	10.4	624	10.4	$0.272	$2.83
8	10.3	620	10.3	$0.272	$2.81

Preset and Bitrate for Equivalent Quality



High volume streaming

Preset	Bitrate	Encoding Time
0	100.00%	100.00%
1	102.19%	25.94%
2	104.01%	8.35%
3	109.49%	2.96%
4	116.79%	2.06%
5	122.99%	1.16%
6	125.91%	0.69%
7	127.01%	0.69%
8	127.74%	0.70%

Low volume streaming

- Assuming deliver at certain quality level, how much do you need to increase bandwidth to maintain quality
  - Preset 8 – boost bandwidth by 27.74% for same quality as preset 0
- Trade-off encoding cost vs. bandwidth cost

Decision: Threads

Threads limit the number of CPUs used during encode (encoding speed/quality)

```
ffmpeg -y -i Football_10.mp4 -c:v libaom-av1 -b:v 1500K -g 60 -
keyint_min 60  -cpu-used 8 -auto-alt-ref 1 -threads 8 -pass 1 -f
matroska NUL & \

ffmpeg -y -i Football_10.mp4 -c:v libaom-av1 -b:v 1500K -maxrate 3000K
-g 60 -keyint_min 60  cpu -used 4 -auto-alt-ref 1 -threads 8 -pass 2
Football_1.mkv
```

- `-threads` threads integer (decoding/encoding,video). Set the number of threads to be used, in case the selected codec implementation supports multi-threading. Possible values: ‘auto, 0’ automatically select the number of threads to set.
- **Default** value is ‘auto’.

Threads

What command does: Number of logical processors to be used

Default: Auto for codec / looks to be all in this case but would be less on a system with fewer cores (test station is 40-core system)

Impact:

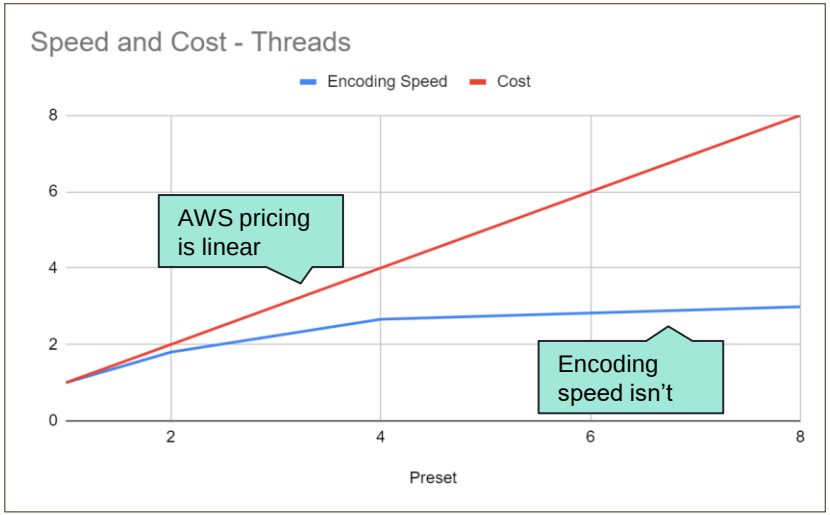
- **Encoding time -** significant reduction from 1-8; nothing thereafter
- **Quality** - nothing really

Discussion: Best option depends upon:

- Cost

	Baseline	1 thread	2 threads	4 threads	8 threads	16 threads	32 threads	Delta
Freedom	0:05:14	0:16:02	0:08:43	0:05:51	0:05:18	0:05:16	0:05:09	67.88%
Football	0:03:56	0:11:11	0:06:25	0:04:23	0:03:49	0:03:45	0:03:57	66.47%
Average	0:04:35	0:13:37	0:07:34	0:05:07	0:04:34	0:04:31	0:04:33	**66.87%**
Bitrate	**Baseline**	**1 thread**	**2 threads**	**4 threads**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	1,479	1,447	1,479	1,479	1,479	1,479	1,479	2.16%
Football	1,528	1,523	1,528	1,528	1,528	1,528	1,528	0.33%
Average	**1,504**	**1,485**	**1,504**	**1,504**	**1,504**	**1,504**	**1,504**	**1.23%**
VMAF	**Baseline**	**1 thread**	**2 threads**	**4 threads**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	85.15	85.09	85.15	85.15	85.15	85.15	85.15	0.07%
Football	85.10	85.00	85.10	85.10	85.10	85.10	85.10	0.12%
Average	85.13	85.04	85.13	85.13	85.13	85.13	85.13	**0.10%**
Low Frame	**Baseline**	**1 thread**	**2 threads**	**4 threads**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	77.58	78.57	77.58	77.58	77.58	77.58	77.58	1.25%
Football	66.41	66.21	66.41	66.41	66.41	66.41	66.41	0.30%
Average	72.00	72.39	72.00	72.00	72.00	72.00	72.00	**0.54%**
Standard Devia	**Baseline**	**1 thread**	**2 threads**	**4 threads**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	3.23	3.25	3.23	3.23	3.23	3.23	3.23	0.65%
Football	6.74	6.77	6.74	6.74	6.74	6.74	6.74	0.41%
Average	4.99	5.01	4.99	4.99	4.99	4.99	4.99	**0.49%**

Threads

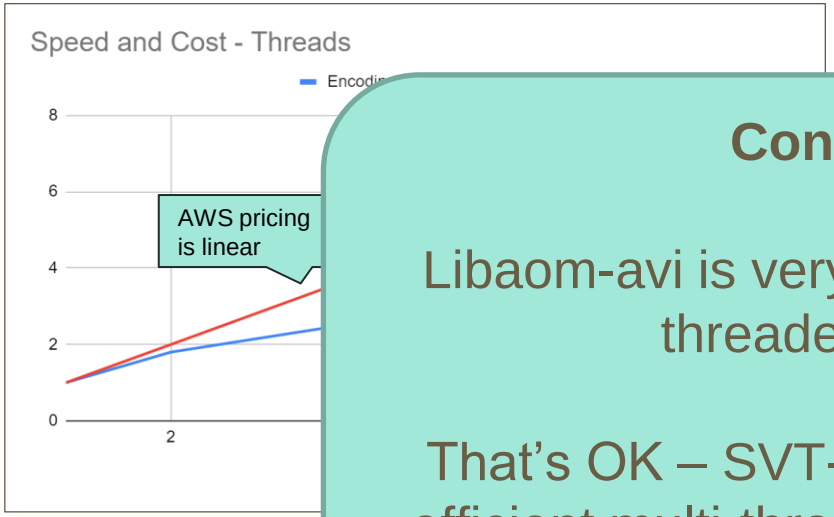


Optimal encoding station	1 thread	2 threads	4 threads	8 threads
Time to encode 1 hour (in hours)	82	46	31	28
Hourly cost (C6g.med/large)	$0.034	$0.068	$0.136	$0.272
Total cost per hour	**$2.79**	**$3.13**	**$4.22**	**$7.62**

- C6 Medium/large series on-demand pricing
- Green considerations mitigate against single thread

- Ability to split encoding task among multiple machines is huge priority
- In encoding farm, thread counts beyond 2 don't make sense unless running multiple parallel encodes on the same system
- Used 8 on my 40-core computer, but different in production scenario

Threads



reads	4 threads	8 threads
46	31	28
.068	$0.136	$0.272
3.13	**$4.22**	**$7.62**

Conclusion:

Libaom-avi is very inefficient with multi-threaded operation

That's OK – SVT-AV1's raison d'etre is efficient multi-threaded operation (to sell more multi-threaded CPUs)

- C6 Medium/la… pricing
- Green considerations mitigate against single thread

…task among …huge priority

- In encoding farm, thread counts beyond 2 don't make sense unless running multiple parallel encodes on the same system
- Used 8 on my 40-core computer, but different in production scenario

Final Strings

Entries in green are default values;

```
ffmpeg -y -i Football_10.mp4 -c:v libaom-av1 -b:v 1500K -g 60 -keyint_min 60
-cpu-used 8 -auto-alt-ref 1 -threads 8 -tile-columns 1 -tile-rows 0 -row-mt
1 -lag-in-frames 25 -pass 1 -f matroska NUL & \

ffmpeg -y -i Football_10.mp4 -c:v libaom-av1 -b:v 1500K -maxrate 3000K  -g
60 -keyint_min 60 -cpu-used 3  -auto-alt-ref 1 -threads 8 -tile-columns 1 -
tile-rows 0 -row-mt 1 -lag-in-frames 25 -pass 2 Football_1.mkv
```

Observations About libaom-AV1

- Regarding the future of libaom-AV1, Andrey explained that though both Libaom-AV1 and SVT-AV1 will be supported going forward, they serve different functions.
- Specifically, Libaom-AV1 is “the **reference software”** designed to “cover the whole AV1 standard.”
- SVT-AV1 “is more of a **production code base** or the code base that is optimized to work … multi core machines.”

Netflix's Andrey Norkin Talks Future of AV1

https://bit.ly/Norkin_AV1

- Libaom is more accessible (because it’s in FFmpeg), but less functional
- SVT-AV1 has its warts as well

SVT-AV1

- Overview
- Encoding basics
  - Encoding string starting point
  - Passes
  - Presets
  - Threads

Overview

- The SVT-AV1 encoder was developed by Netflix and Intel and launched at NAB 2019
- The codec is available on Github here (**https://bit.ly/get_svt**)
- You can find the user guide here (**https://bit.ly/learn_SVT)**
    - and you should definitely download and print a copy if you plan to work with SVT-AV1. Unfortunately, there's not a lot of guidance there
- In August 2020, the Alliance for Open Media announced that they were forming a working group to "aid the development of AOMedia AV1 products and services." The group will use SVT-AV1 as the basis for these efforts
- Here's the help file from the latest version of the encoder (https://bit.ly/svt_help)
- The SVT-AV1 encoder is SvtAv1EncApp.exe

Encoding Basics

Convert source file to Y4M (the encoder only accepts Y4M or YUV input)

```
ffmpeg -i input.mp4 -pix_fmt yuv420p  input.y4m
```

Multiple-pass string from documentation (https://bit.ly/svt_docs)

```
SvtAv1EncApp -i input.y4m -w 1920 -h 1080 --fps 24 --rc 1 --tbr 1000
--preset 0 --passes 2 --stats stat_file.stat -b output.ivf
```

Items in red not needed if use Y4m and don’t need stat file

Encoding Basics

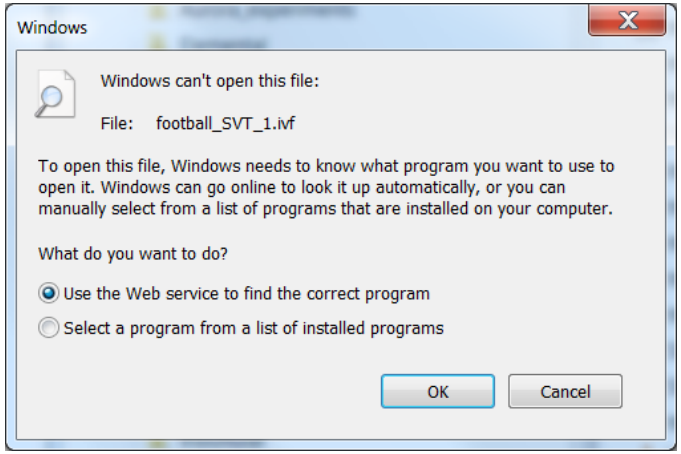
My starting string

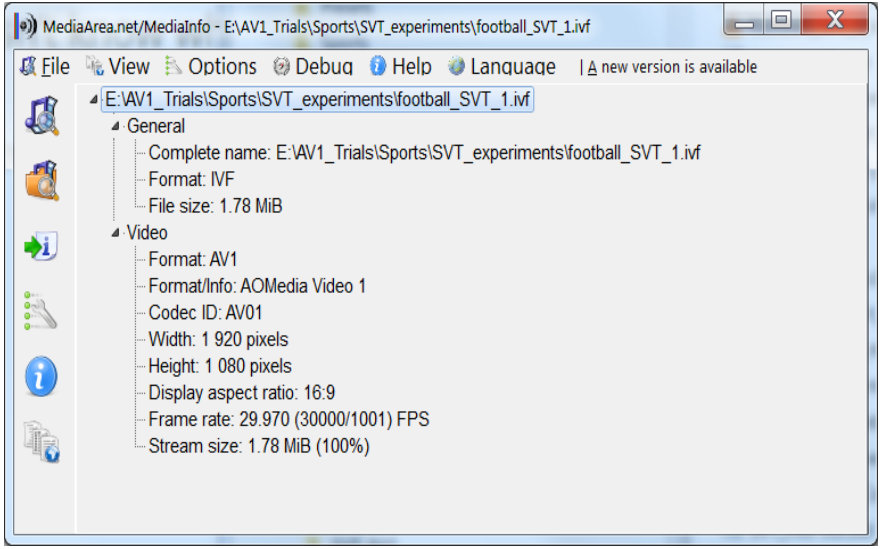
```
SvtAv1EncApp -i input.y4m --rc 1 --tbr 1500 --mbr 3000 --keyint 2s --preset 4 --passes 2  -b output.ivf
```

- `-i` = input file (I prefer Y4m over YUV because you can eliminate -w/-h/-fps)
- `--rc` = Rate control mode (0 = CQP, 1 =VBR , 2 = CBR)
- `--tbr` = target bitrate
- `--mbr` = max bitrate
- `--preset` = Preset (more later, default is 10)
- `--passes 2` = Number of passes (2=3 passes but very, very fast quick)
- `--keyint 2s` = sets keyframe at 2 seconds
- `-b` = Output file

Output Format - IVF, MKV, WebM

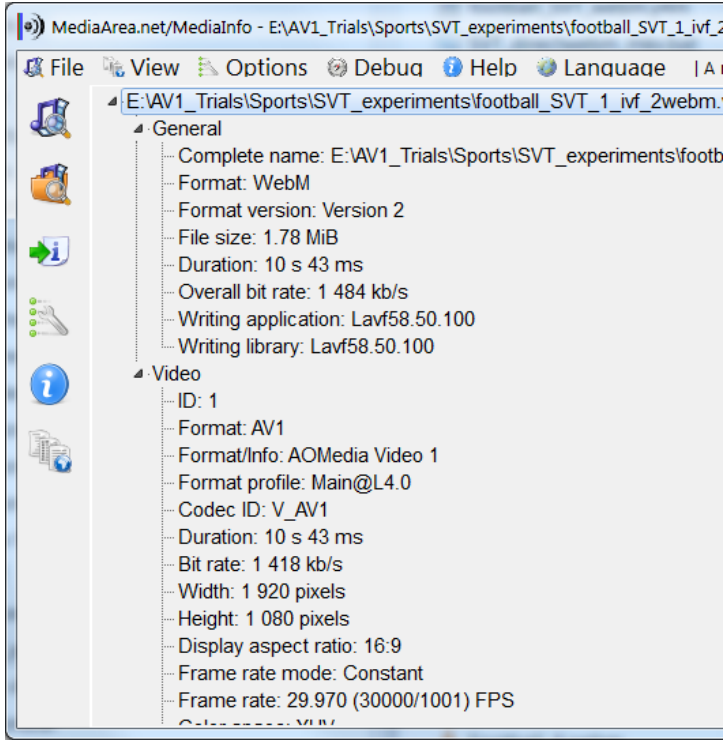
- .ivf is default output
- .ivf is a funky file format.
  - Windows doesn't know what player to assign
  - MediaInfo knows but provides limited information





Output Format - IVF, MKV, WebM

- During experimentation phase
  - Output as .ivf
  - Convert in FFmpeg to web/mkv
- File read by any AV1 compatible player and analysis tool




```
ffmpeg -i input.ivf -vcodec copy -acodec copy output.webm
```

One or Three Passes

1-Pass

```
SvtAv1EncApp -i input.y4m --rc 1 --tbr 1500 --mbr 3000 --keyint 2s
--preset 4  -b output.ivf
```

2-pass/3-Pass

```
SvtAv1EncApp -i input.y4m --rc 1 --tbr 1500 --mbr 3000 --keyint 2s
--preset 4 --passes 2  -b output.ivf
```

Decision 1: One Pass or Three-Pass

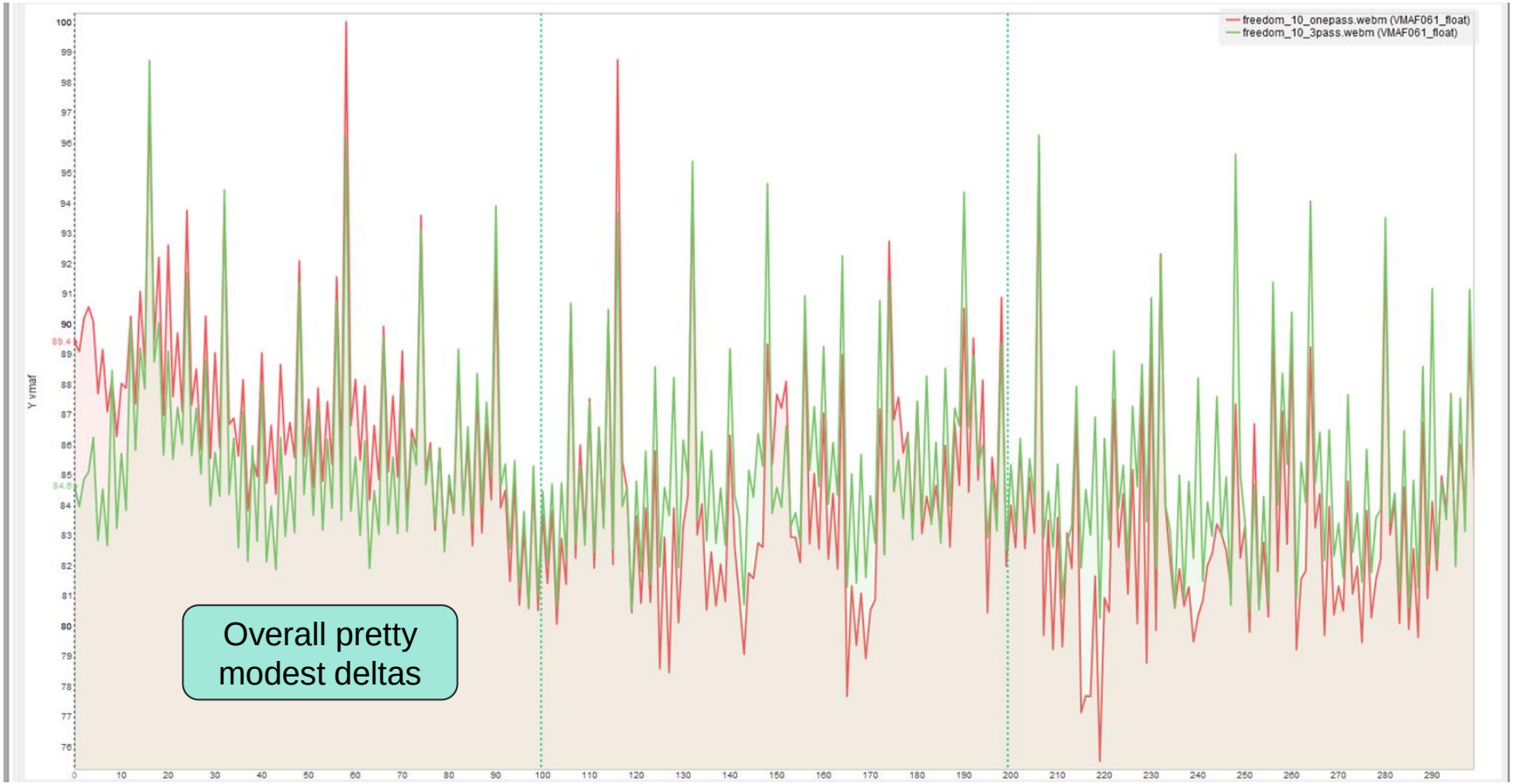
Feature Analysis:

- Time:
  - One-pass is 7% faster
- Bitrate:
  - 3-pass does a better job achieving target bitrate (lower delta from 1500 target)
- VMAF - average - 2.5 higher (mostly easyhard)
- Low-frame (transient issues)
  - 3-pass is meaningfully lower on all clips, particularly easyhard.
- Standard deviation (quality variability)
  - 1-pass substantially higher

Single/Three Pass			
Encoding time	1-pass	3-pass	Delta
Freedom	0:01:13	0:01:19	7.59%
Football	0:01:25	0:01:31	6.59%
Easy Hard	0:02:55	0:03:08	6.91%
Average	0:01:51	0:01:59	6.98%
bitrate	1-pass	3-pass	Delta
Freedom	1,448	1,438	0.69%
Football	1,413	1,398	1.06%
Easy Hard	1,348	1,456	7.42%
Average	1,403	1,431	1.93%
Delta from 1500	-6.47%	-4.62%	
VMAF	1-pass	3-pass	Delta
Freedom	84.59	85.18	0.70%
Football	82.90	81.08	2.20%
Easy Hard	69.79	78.32	10.90%
Average	79.09	81.53	2.99%
Low Frame	1-pass	3-pass	Delta
Freedom	75.53	80.26	5.90%
Football	64.40	67.32	4.34%
Easy Hard	33.94	55.05	38.35%
Average	57.96	67.55	14.20%
Standard Deviation	1-pass	3-pass	Delta
Freedom	3.83	3.29	14.26%
Football	6.36	5.53	13.17%
Easy Hard	17.84	11.05	38.05%
Average	9.35	6.62	29.15%

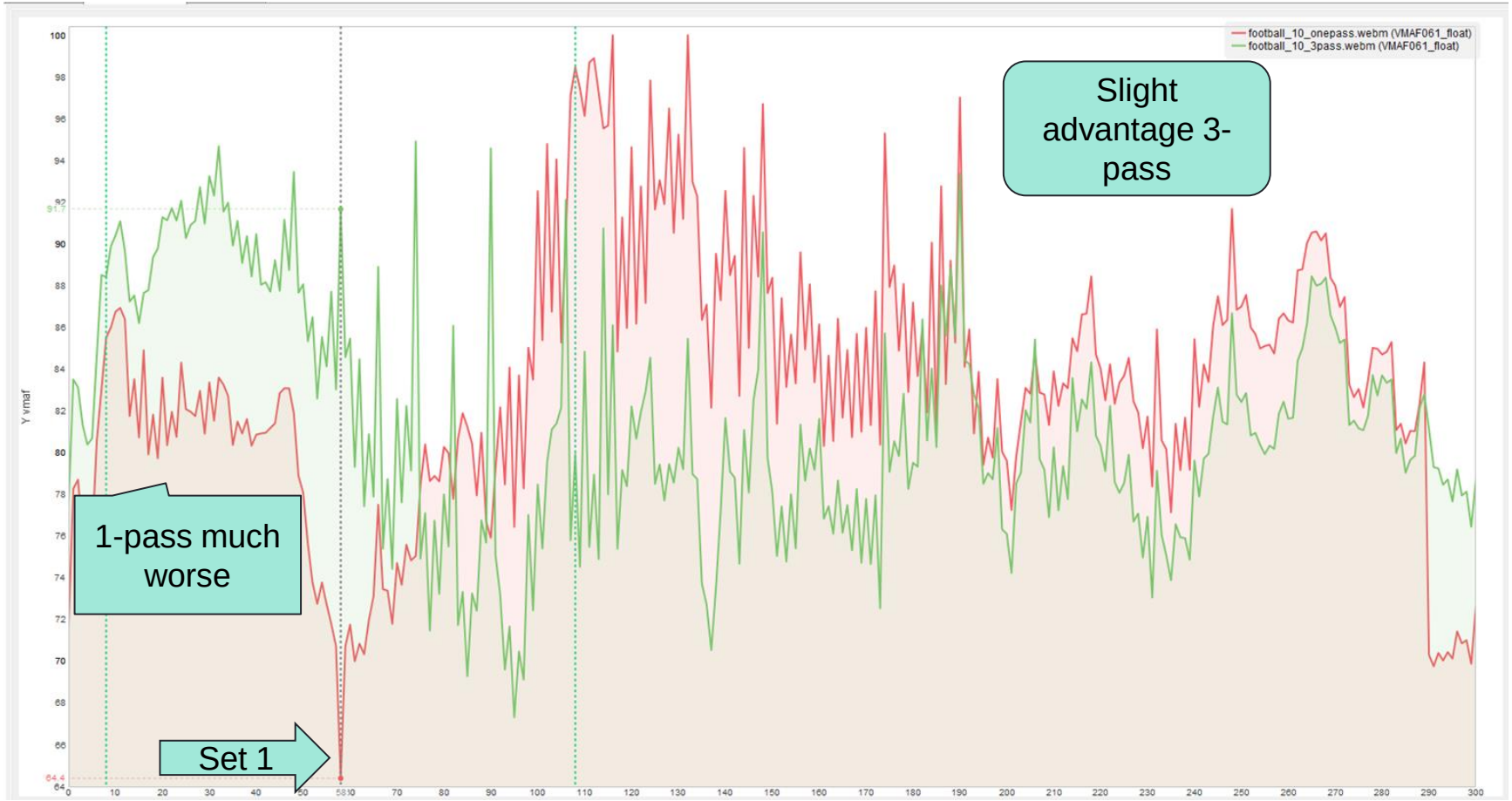
Results Plots and Frames - Freedom

Red = 1-Pass
Green = 3-pass

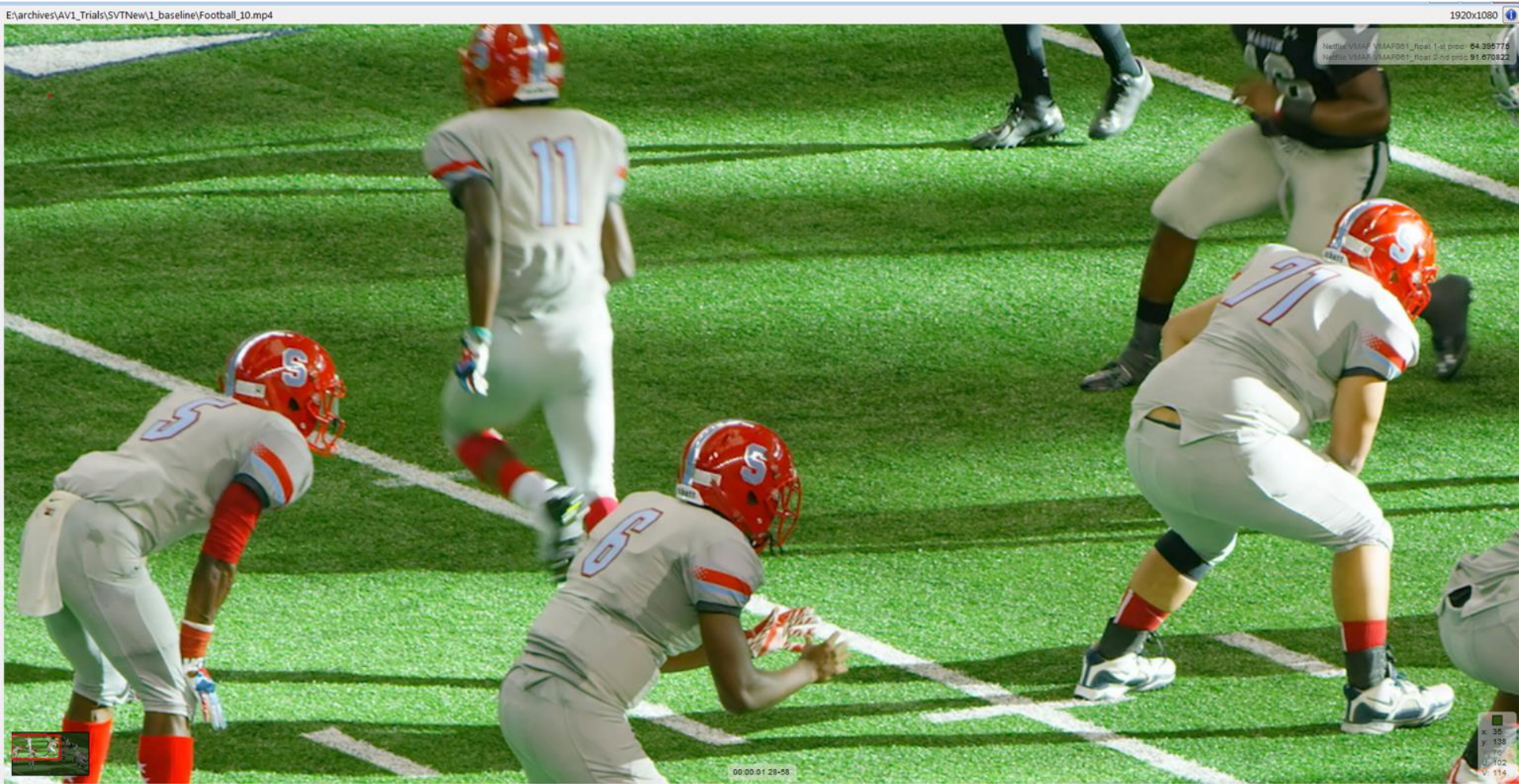


Results Plots and Frames - Football

Red = 1-Pass
Green = 3-pass



Source


E:\archives\AV1_Trials\SVTNew\1_baseline\Football_10.mp4
1920x1080
11
71
6
5

1-Pass


E:\archives\AV1_Trials\SVTNew\1_baseline\football_10_onepass.webm
1920x1080
Netflix VMAF_VMAF061_float 1-st proc 64.395775
Netflix VMAF_VMAF061_float 2-nd proc 91.670822
00:00:01:28+58

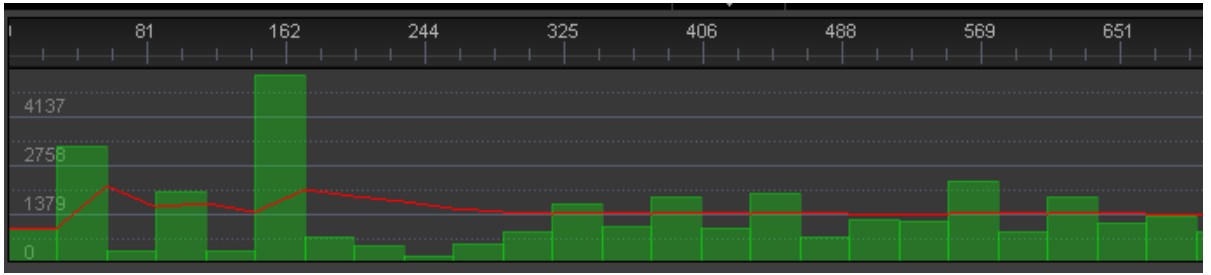
3-Pass


E:\archives\AV1_Trials\SVTNew\1_baseline\football_10_3pass.webm
1920x1080
Netflix VMAF VMAF061_float 1-st proc 64.395775
Netflix VMAF VMAF061_float 2-nd proc 91.670822
00:00:01.28+58

Data Rate Allocation – Easy Hard Clips

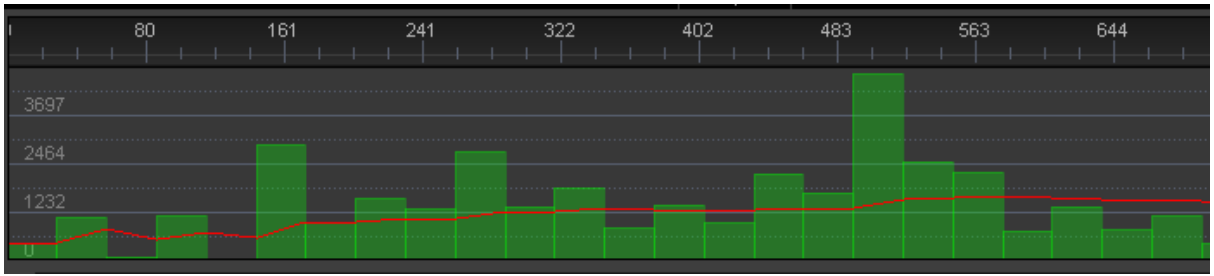
- Classic 1-pass vs. multiple-pass
- 1-pass - don't know where difficult sections are
- Multiple-pass
  - Scan once, ID easy and hard regions
  - Then encode

1-Pass



Substantial data wasted here

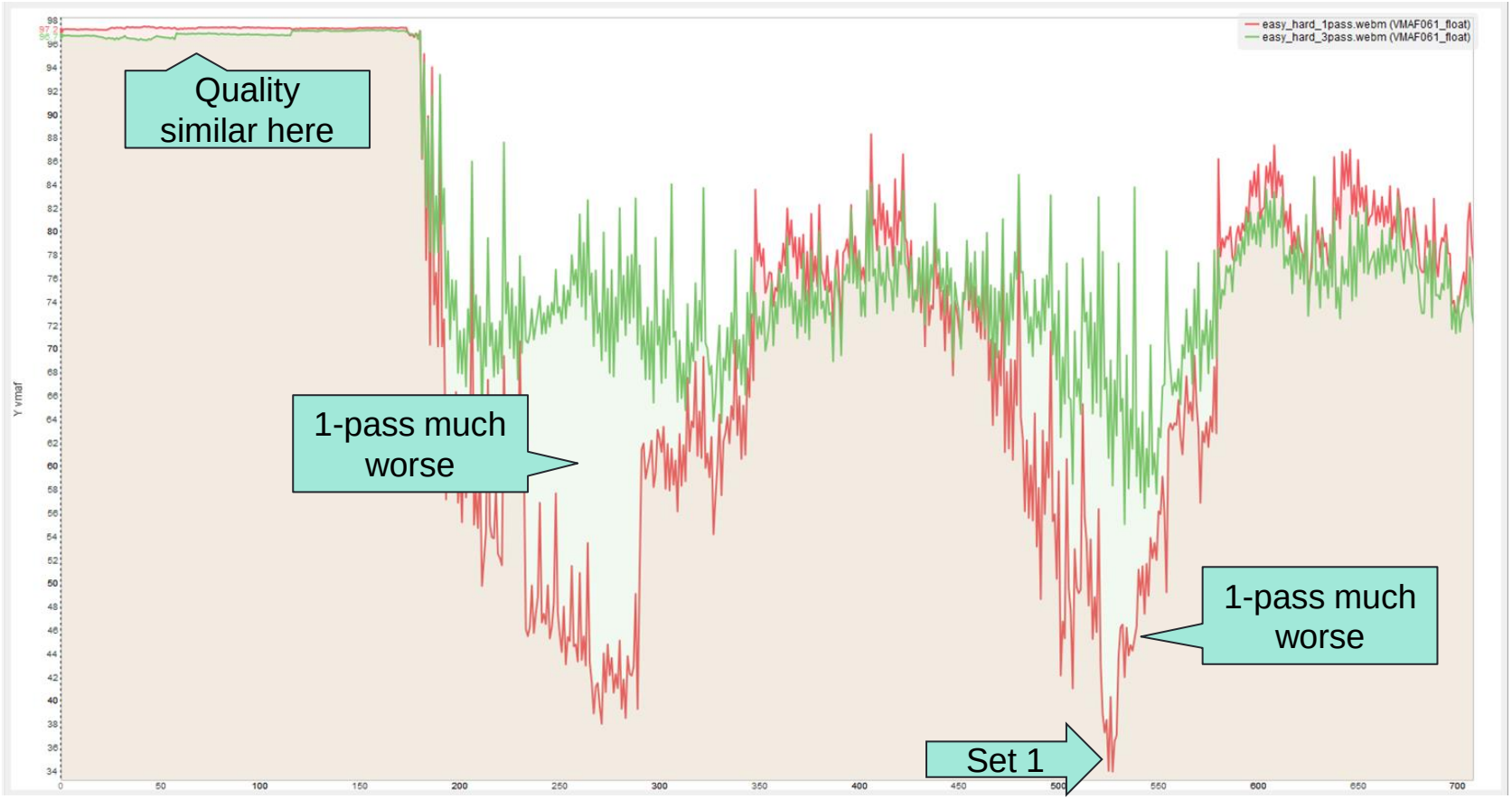
3-Pass



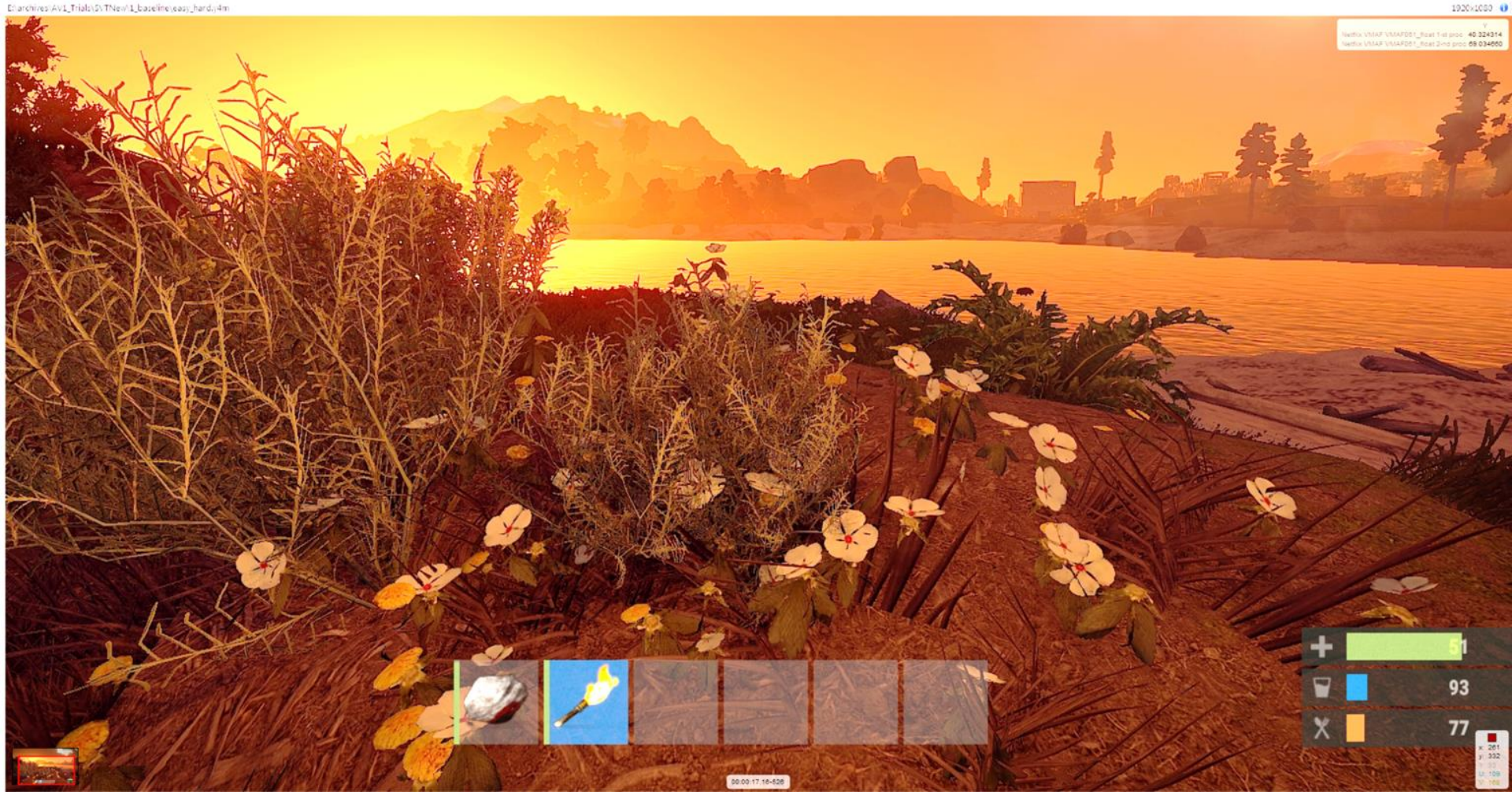
Data allocated to challenging sections

Results Plots and Frames - Easy_Hard

Red = 1-Pass
Green = 3-pass



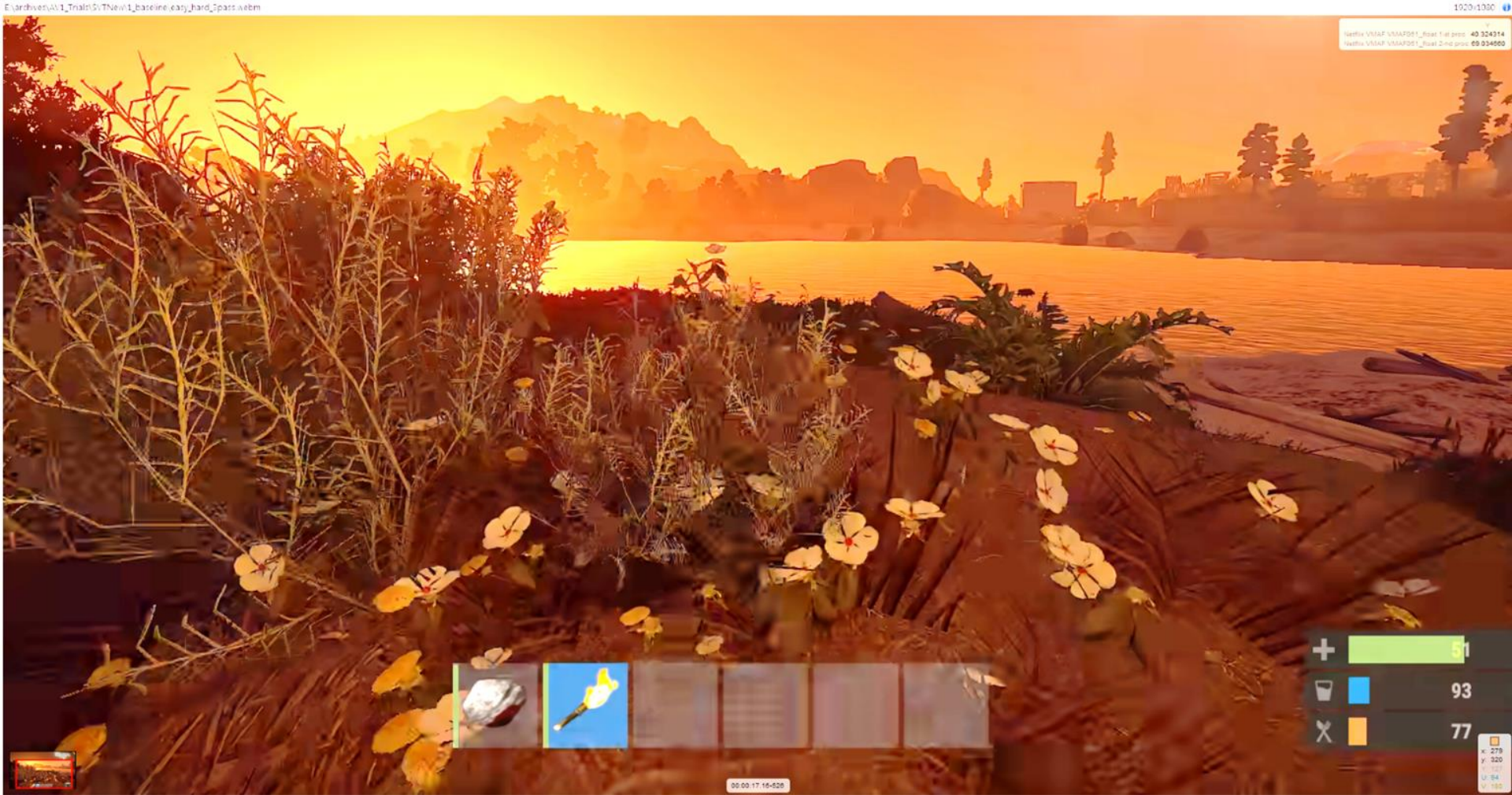
Source


E:\archives\AV1_Trials\OVTNew\1_baseline\easy_hard_4m
1920x1080
93
77

1-Pass


E:\archive\AV1_Trials\SVTNew\1_baseline\easy_hard_1pass.webm
1920x1080
93
77

3-Pass


E:\archives\AV1_Trials\S\TNew\1_baseline\cary_hard_2pass.webm
1920x1080
93
77

Decision 1: One Pass or Three-Pass

Feature Analysis:

- Time:
  - One-pass is 7% higher
- Bitrate:
  - 3-pass does a better job achieving target bitrate (lower delta from 1500 target)
- VMAF - average - 2.5 higher (mostly easyhard)
- Low-frame (transient issues)
  - 3-pass is meaningfully lower on all clips, particularly easyhard.
- Standard deviation (quality variability)
  - 1-pass substantially higher

- **Conclusion: 3-Pass where available (not live)**

Single/Three Pass			
Encoding time	**1-pass**	**3-pass**	**Delta**
Freedom	0:01:13	0:01:19	7.59%
Football	0:01:25	0:01:31	6.59%
Easy Hard	0:02:55	0:03:08	6.91%
Average	0:01:51	0:01:59	**6.98%**
bitrate	**1-pass**	**3-pass**	**Delta**
Freedom	1,448	1,438	0.69%
Football	1,413	1,398	1.06%
Easy Hard	1,348	1,456	7.42%
Average	**1,403**	**1,431**	**1.93%**
Delta from 1500	**-6.47%**	**-4.62%**	
VMAF	**1-pass**	**3-pass**	**Delta**
Freedom	84.59	85.18	0.70%
Football	82.90	81.08	2.20%
Easy Hard	69.79	78.32	10.90%
Average	**79.09**	**81.53**	**2.99%**
Low Frame	**1-pass**	**3-pass**	**Delta**
Freedom	75.53	80.26	5.90%
Football	64.40	67.32	4.34%
Easy Hard	33.94	55.05	38.35%
Average	**57.96**	**67.55**	**14.20%**
Standard Deviation	**1-pass**	**3-pass**	**Delta**
Freedom	3.83	3.29	14.26%
Football	6.36	5.53	13.17%
Easy Hard	17.84	11.05	38.05%
Average	**9.35**	**6.62**	**29.15%**

SVT-AV1 in FFmpeg

- You can access SVT-AV1 in FFmpeg (if you compile a special version)
- Only single-pass is supported

Key Performance/Quality-Related Parameters

aomenc Command	Function	STV Equivalent	Default?
`--cpu-used=`	Encoding/quality tradeoff	`--preset`	8
`--threads=`	Set number of threads used	`--lp` (logical processors)	Not specified (seems like all available cores)
`--tile-columns=1 --tile-rows=0`	Divides frame into sections for faster encoding	`same`	Not deployed
`--auto-alt-ref=1`	Enable automatic alt reference frames	`--enable tf`	On by default
`--row-mt=`	Enable row based multi-threading	None	NA
`--lag-in-frames=`			“When Rate Control is set to 1 (VBR) it's best to set this parameter to be equal to the Intra period value”

Choosing a Preset

- Preset - controls quality/encoding time tradeoff
- Encoding parameters (results will change if use different)
  - Two-pass (significant difference)
  - LP 8 (makes a significant difference - see why later)
  - VBR
  - 2-second keyframe
- Test files will also make a difference
- Run your own tests to be sure for your settings and clips

```
SvtAv1EncApp -i input.y4m  --rc 1 --tbr 1500 --mbr 3000 --keyint 2s --passes
2 --preset 1 --passes 2 -b input.ivf
```

Presets

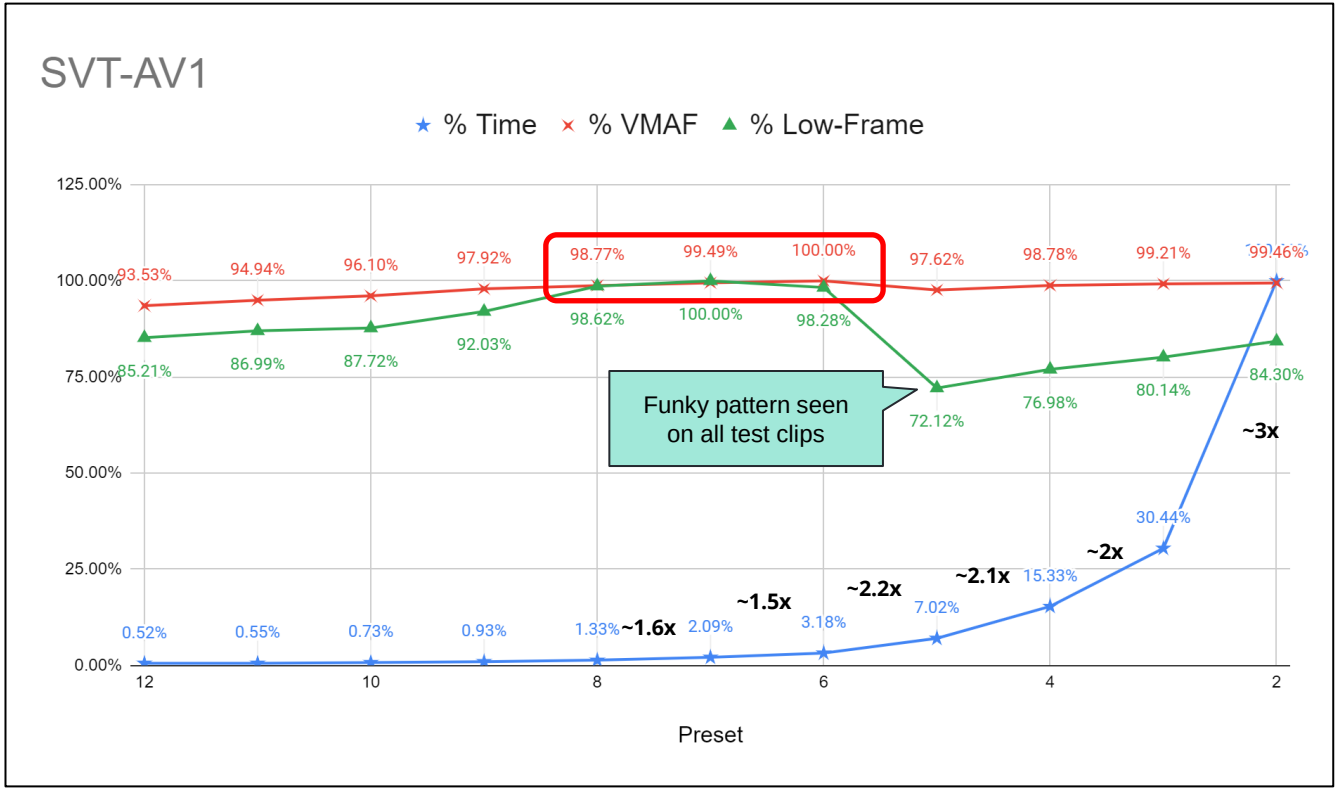
- Combined for all three test files (total ~ 8.5 min)
- Max delta
  - 5.96% for VMAF
  - 24.79% for Low-Frame
- Single file live (origination) possible at around 8 or lower
  - Minimal quality drop

Preset	Combined Time	VMAF	Low-Frame
2	10:56:02	92.63	70.76
3	3:19:40	92.40	67.27
4	1:40:34	92.00	64.61
5	0:46:03	90.92	60.53
6	0:20:53	93.14	82.49
7	0:13:44	92.67	83.94
8	0:08:45	92.00	82.78
9	0:06:08	91.20	77.24
10	0:04:48	89.51	73.63
11	0:03:35	88.43	73.02
12	0:03:26	87.11	71.52
Delta	10:52:36	5.56	12.42
% Delta	99.48%	5.96%	14.79%

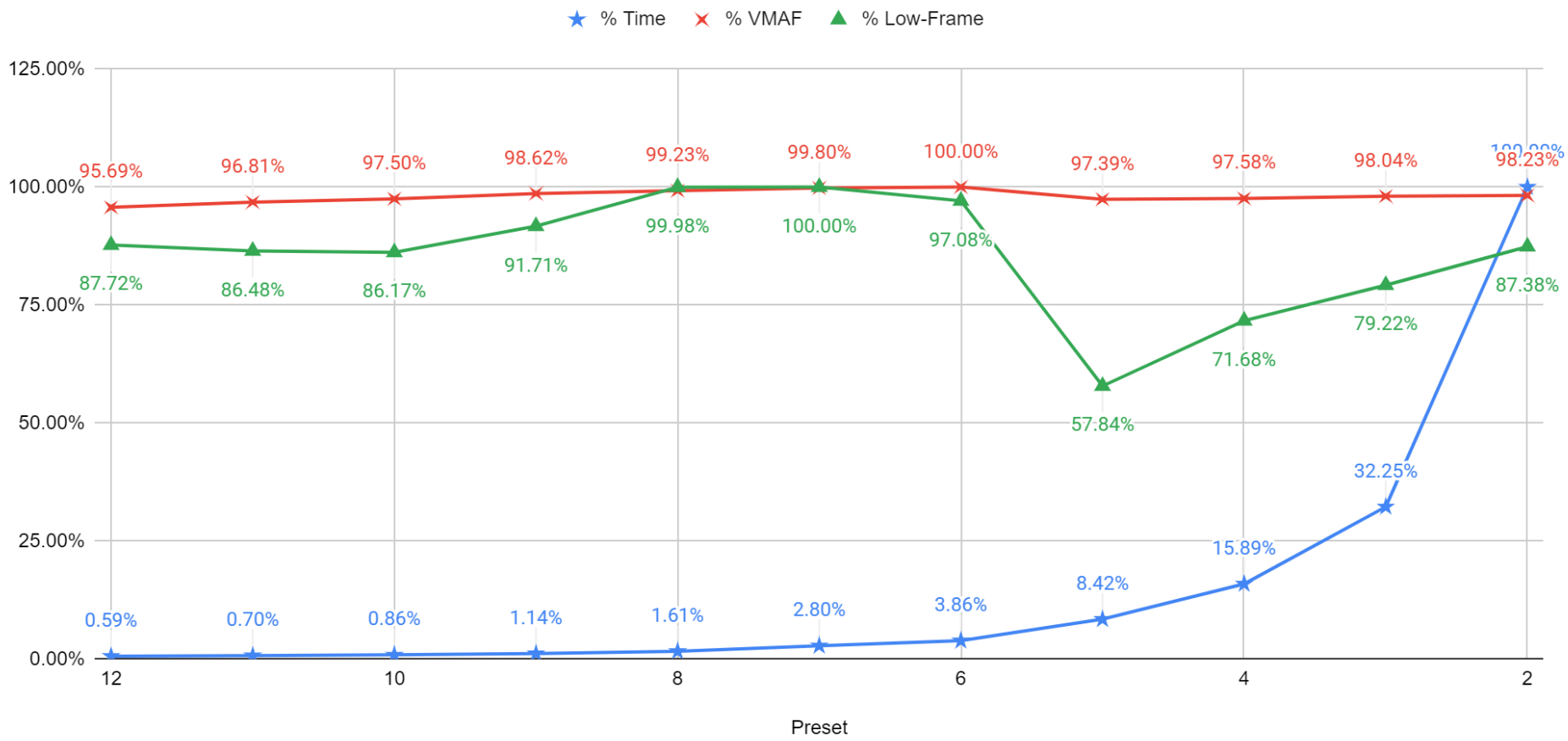
Presets – Default is 10

Analysis:

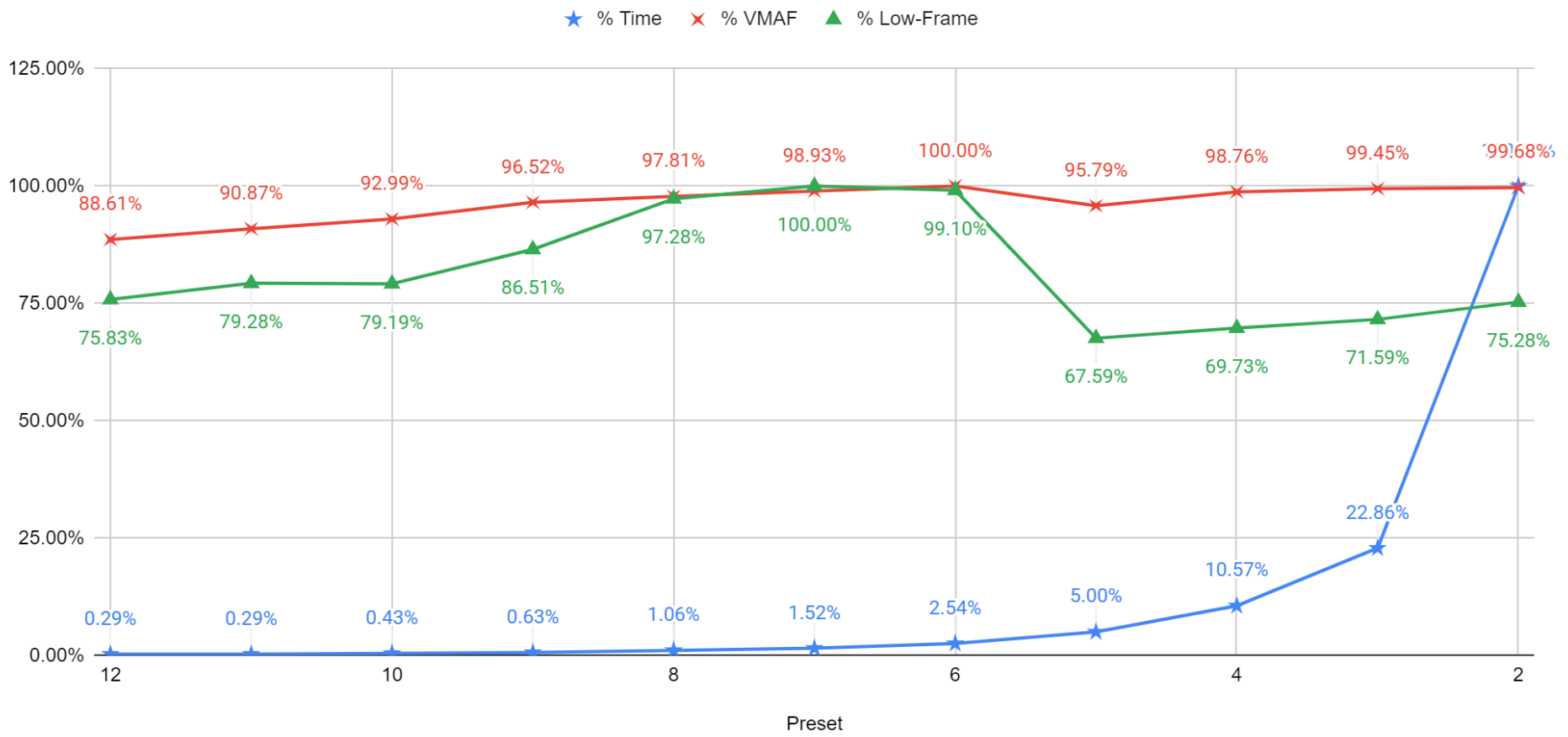
- Convert time and scores to % of 100%
- Funky pattern, very much influenced by Football clip
- On average, 7 appears the best choice
  - 32% lower encoding time than 6
  - .5 VMAF
  - Best low frame



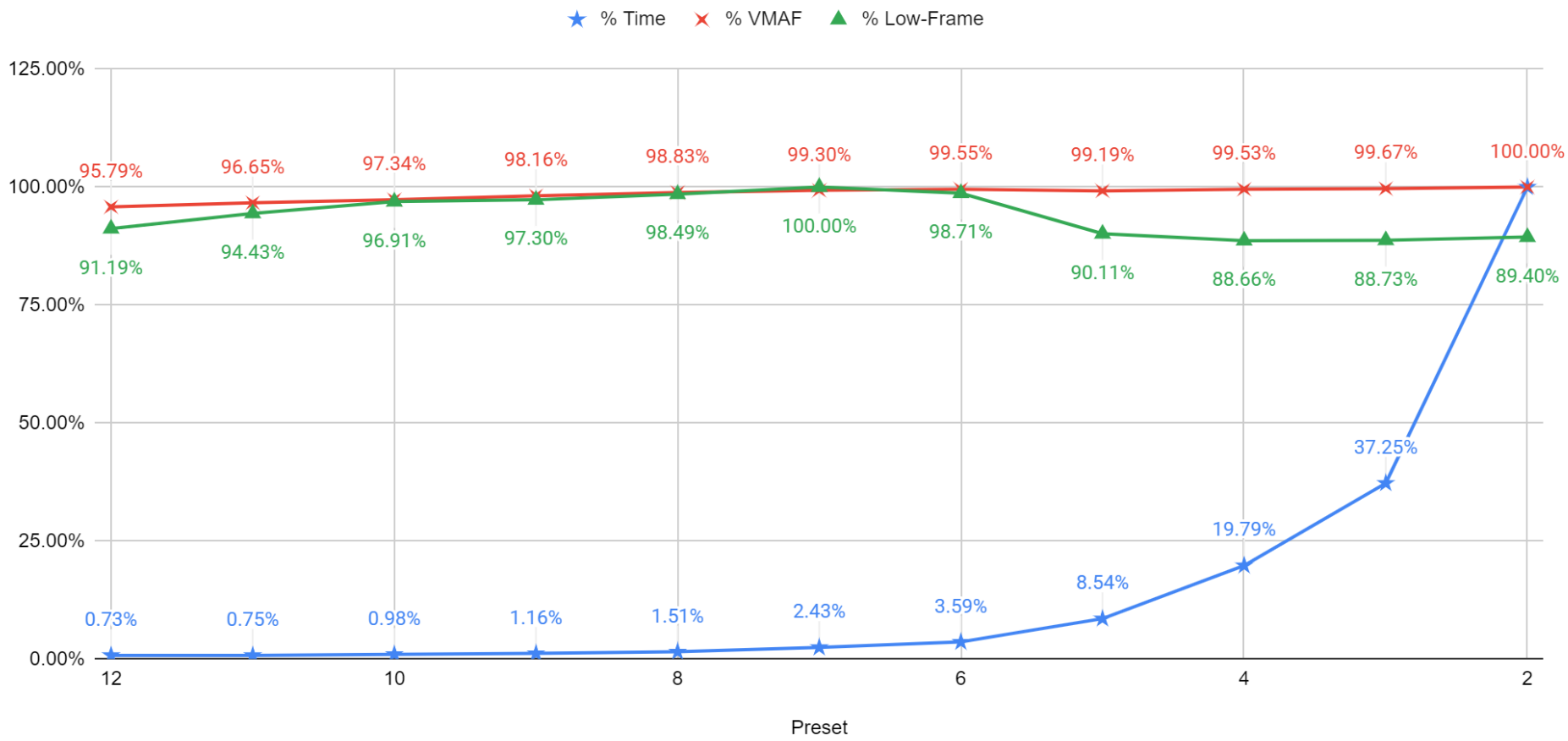
Orchestra


% Time
% VMAF
% Low-Frame
125.00%
100.00%
75.00%
50.00%
25.00%
0.00%
95.69%
96.81%
97.50%
98.62%
99.23%
99.80%
100.00%
97.39%
97.58%
98.04%
98.23%
87.72%
86.48%
86.17%
91.71%
99.98%
100.00%
97.08%
57.84%
71.68%
79.22%
87.38%
0.59%
0.70%
0.86%
1.14%
1.61%
2.80%
3.86%
8.42%
15.89%
32.25%
12
10
8
6
4
2
Preset

Football


% Time
% VMAF
% Low-Frame
125.00%
100.00%
75.00%
50.00%
25.00%
0.00%
88.61%
90.87%
92.99%
96.52%
97.81%
98.93%
100.00%
95.79%
98.76%
99.45%
99.68%
75.83%
79.28%
79.19%
86.51%
97.28%
100.00%
99.10%
67.59%
69.73%
71.59%
75.28%
0.29%
0.29%
0.43%
0.63%
1.06%
1.52%
2.54%
5.00%
10.57%
22.86%
12
10
8
6
4
2
Preset

Freedom


% Time
% VMAF
% Low-Frame
125.00%
100.00%
75.00%
50.00%
25.00%
0.00%
95.79%
96.65%
97.34%
98.16%
98.83%
99.30%
99.55%
99.19%
99.53%
99.67%
100.00%
91.19%
94.43%
96.91%
97.30%
98.49%
100.00%
98.71%
90.11%
88.66%
88.73%
89.40%
0.73%
0.75%
0.98%
1.16%
1.51%
2.43%
3.59%
8.54%
19.79%
37.25%
12
10
8
6
4
2
Preset

Presets - Cost Per Hour

- Cost per hour - single file
  - Multiply by ~3-4 for full ladder
  - Lower resolutions will encode more quickly
- 7 seems like optimal choice
  - $0.44 per hour
  - Compared to $9.00 for libaom
- I used 7 for testing

Preset	minutes per minute	Minutes Per hour (times 60)	Hours to encode video hour	Cost per hour - c6g.2xlarge	Cost per Hour
2	78.39	4,703	78.4	$0.272	$21.32
3	23.49	1,409	23.5	$0.272	$6.39
4	11.83	710	11.8	$0.272	$3.22
5	5.42	325	5.4	$0.272	$1.47
6	2.46	148	2.5	$0.272	$0.67
7	1.62	97	1.6	$0.272	$0.44
8	1.03	62	1.0	$0.272	$0.28
9	0.72	43	0.7	$0.272	$0.20
10	0.56	34	0.6	$0.272	$0.15
11	0.42	25	0.4	$0.272	$0.12
12	0.35	21	0.4	$0.272	$0.10

Something is Flawed Here

- Either my test clips
- My analysis, or
- SVT-AV1 preset selection
- Run a similar analysis before choosing an SVT-AV1 preset

Number of Threads

Threads limit the number of CPUs used during encode (encoding speed/quality)

```
SvtAv1EncApp -i input.y4m  --rc 1 --tbr 1500 --mbr 3000  --
keyint 2s  --preset 4 --passes 2 --lp 8 -b output.ivf
```

- `--lp =` Target (best effort) number of logical cores to be used. 0 means all. Refer to Appendix A.1 of the user guide, default is 0 [0, core count of the machine] (from help file)

Decision: Number of Threads

Impact:

- Encoding time - adding threads decreases encoding time significantly (after 1)
- Very minor quality impact

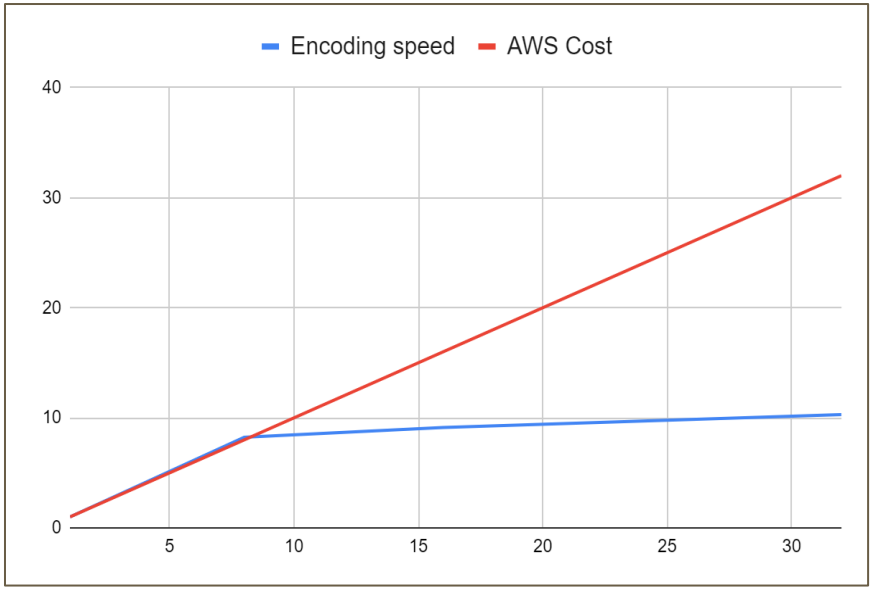
Discussion: Best option depends upon:

- Cost and density

	Baseline	1 thread	8 threads	16 threads	32 threads	Delta
Freedom	0:01:23	0:14:27	0:01:41	0:01:32	0:01:24	90.43%
Football	0:01:11	0:12:19	0:01:34	0:01:24	0:01:12	90.39%
Average	**0:01:17**	**0:13:23**	**0:01:38**	**0:01:28**	**0:01:18**	**90.41%**
Bitrate	**Baseline**	**1 thread**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	1,440	1,436	1,446	1,437	1,438	0.69%
Football	1,398	1,389	1,380	1,398	1,397	1.29%
Average	**1,419**	**1,413**	**1,413**	**1,418**	**1,418**	**0.46%**
VMAF	**Baseline**	**1 thread**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	85.17	85.13	85.23	85.14	85.20	0.11%
Football	81.06	81.47	80.96	81.21	81.09	0.64%
Average	**83.11**	**83.30**	**83.09**	**83.18**	**83.14**	**0.25%**
Low Frame	**Baseline**	**1 thread**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	80.20	80.46	80.51	80.46	80.35	0.38%
Football	67.26	65.12	65.32	65.75	67.39	3.37%
Average	**73.73**	**72.79**	**72.92**	**73.11**	**73.87**	**1.46%**
Standard Deviation	**Baseline**	**1 thread**	**8 threads**	**16 threads**	**32 threads**	**Delta**
Freedom	3.29	3.26	3.25	3.26	3.24	1.39%
Football	5.53	5.77	5.78	5.61	5.49	4.98%
Average	**4.41**	**4.51**	**4.51**	**4.44**	**4.36**	**3.33%**

Threads

	1 thread	8 threads	16 threads	32 threads
Time to encode 1 hour (in hours)	81	10	9	8
Hourly cost	0.034	0.272	0.544	1.088
Total cost per hour	**$2.75**	**$2.72**	**$4.90**	**$8.70**



- C6 Medium/large series on-demand pricing
- Encoding speed tracks pricing through 8 threads
  - After that, it's diminishing returns

- 8 is probably sweet spot for most cloud producers

Final Command String

```
SvtAv1EncApp -i input.y4m  --rc 1 --tbr 1500 --mbr 3000  --keyint
2s  --preset 4 --passes 2 --lp 8 --tile-columns 0 --tile-rows 0 --
enable-tf 1  -b output.ivf
```

Compare SVT-AV1/Libaom-AV1

Libaom-AV1

Preset	Encode Time 8:25 minutes	VMAF	Low-Frame
0	140:08:44	94.46	82.70
1	45:57:43	94.40	82.07
2	19:14:17	94.33	82.00
3	6:33:49	93.83	80.69
4	4:38:34	93.51	80.63
5	2:16:32	92.59	79.64
6	1:58:20	92.20	79.74
7	1:27:28	92.20	79.74
8	1:26:58	92.20	79.74
Delta	138:41:46	2.26	2.95
% Delta	**98.97%**	**2.39%**	**3.57%**

SVT-AV1

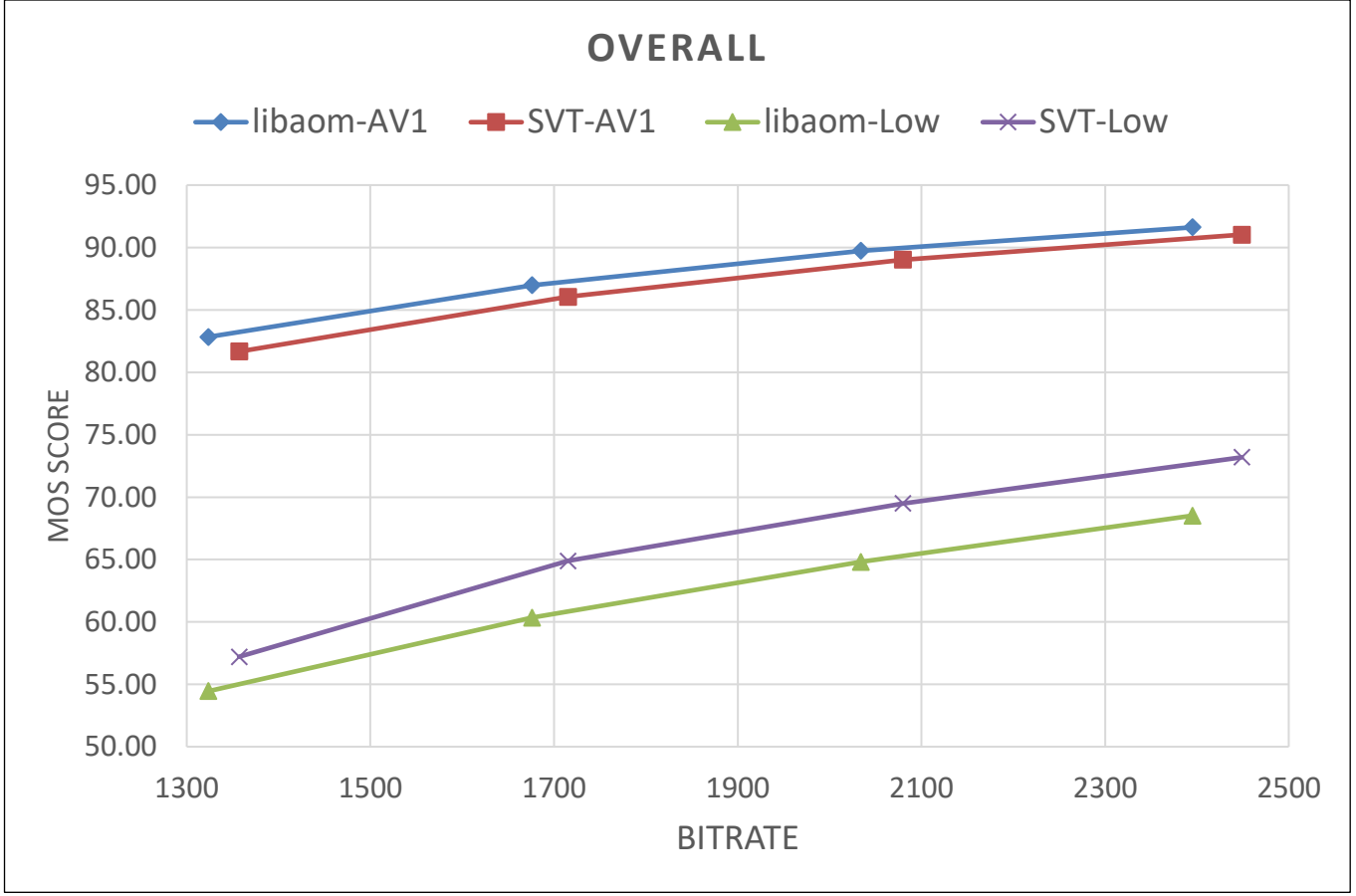
Preset	Time (10 seconds)	VMAF	Low-Frame
2	10:56:02	92.63	70.76
3	3:19:40	92.40	67.27
4	1:40:34	92.00	64.61
5	0:46:03	90.92	60.53
6	0:20:53	93.14	82.49
7	0:13:44	92.67	83.94
8	0:08:45	92.00	82.78
9	0:06:08	91.20	77.24
10	0:04:48	89.51	73.63
11	0:03:35	88.43	73.02
12	0:03:26	87.11	71.52
Delta	10:52:36	5.56	12.42
% Delta	99.48%	5.96%	14.79%

- Not really apples-to-apples
  - SVT-AV1 about 32x faster
  - Much, much cheaper
- But – using recommended presets for each codec

Test

- Entertainment
  - Freedom, Meridian, orchestra, TOS, TOS CG, Zoolander
- Animations
  - BBB, Sintel, El Ultimo
- Sports
  - Football, soccer, hockey
- Games
  - EuroTruckSimulator2, GTAV, Minecraft
- Other
  - Animals, carlot

Overall



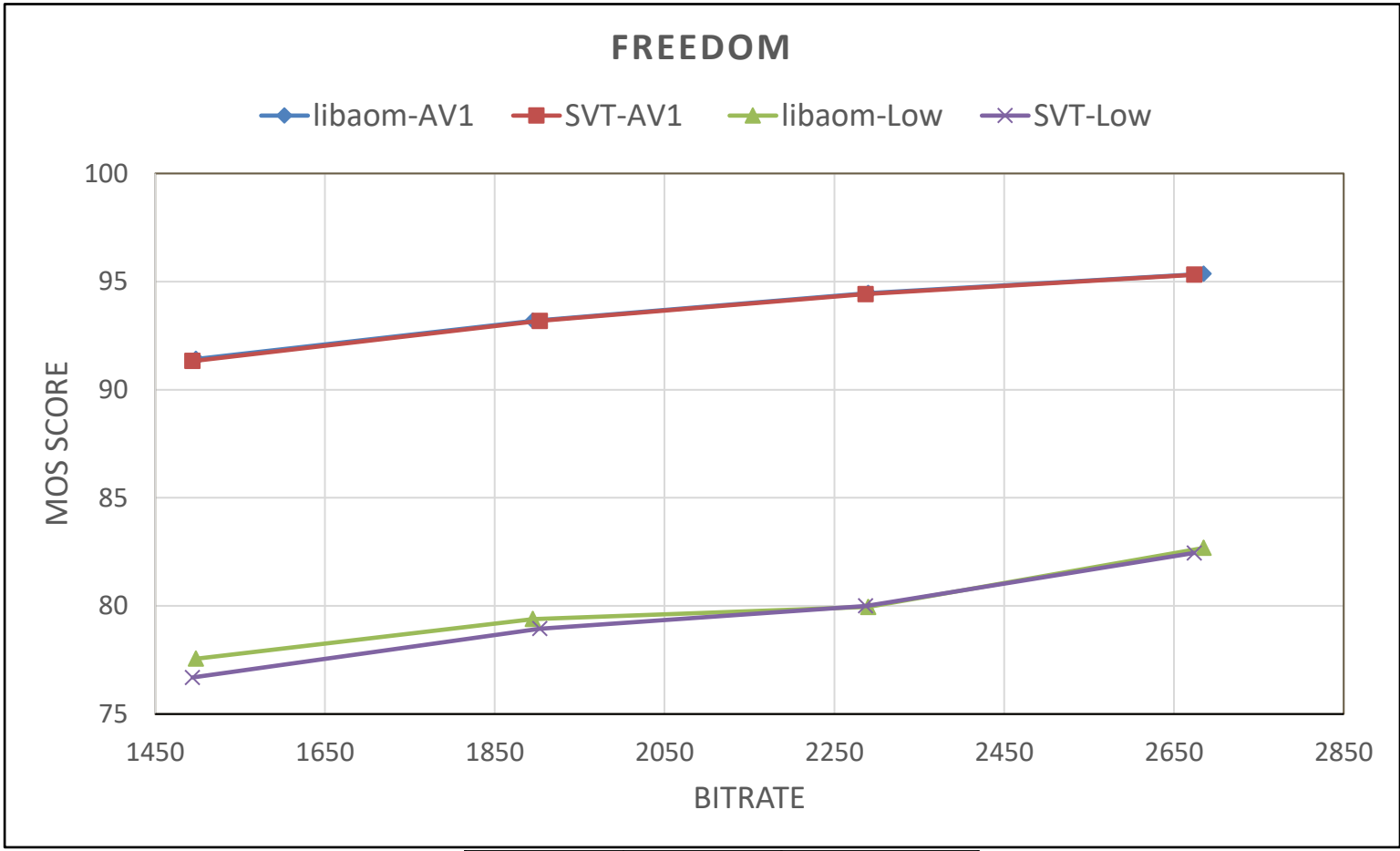
libaom-AV1	SVT-AV1	
-7.68%	8.32%	**VMAF**
16.03%	-13.81%	**Low-Frame**

Analysis

- Quality (mixed)
  - Libaom-av1 slightly better overall
  - Libaom-av1 worse for low frame
- Performance (all SVT)
  - SVT-AV1 much faster but presets are wacky
  - This will likely change over next few releases
    - Higher quality options should be available
- Accessibility (libaom)
  - SVT-AV1 is accessible in FFmpeg currently but single-pass only
  - Should but multiple-pass within FFmpeg soon
- Trending (SVT
  - SVT-AV1 trending towards greater utility
  - Libaom trending towards becoming the reference codec
    - Best overall quality, but limited utility

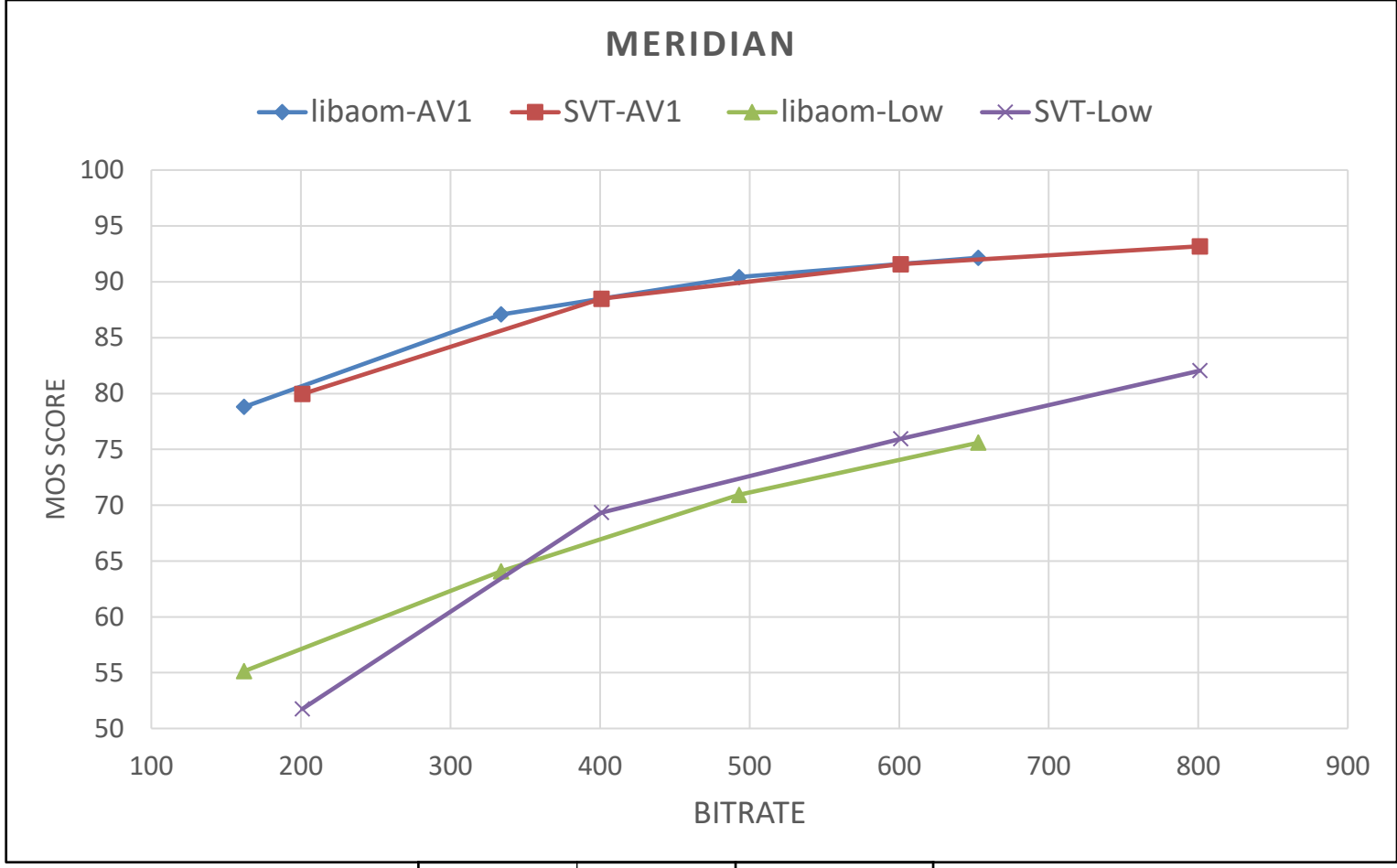
Magic Ball Says

Entertainment



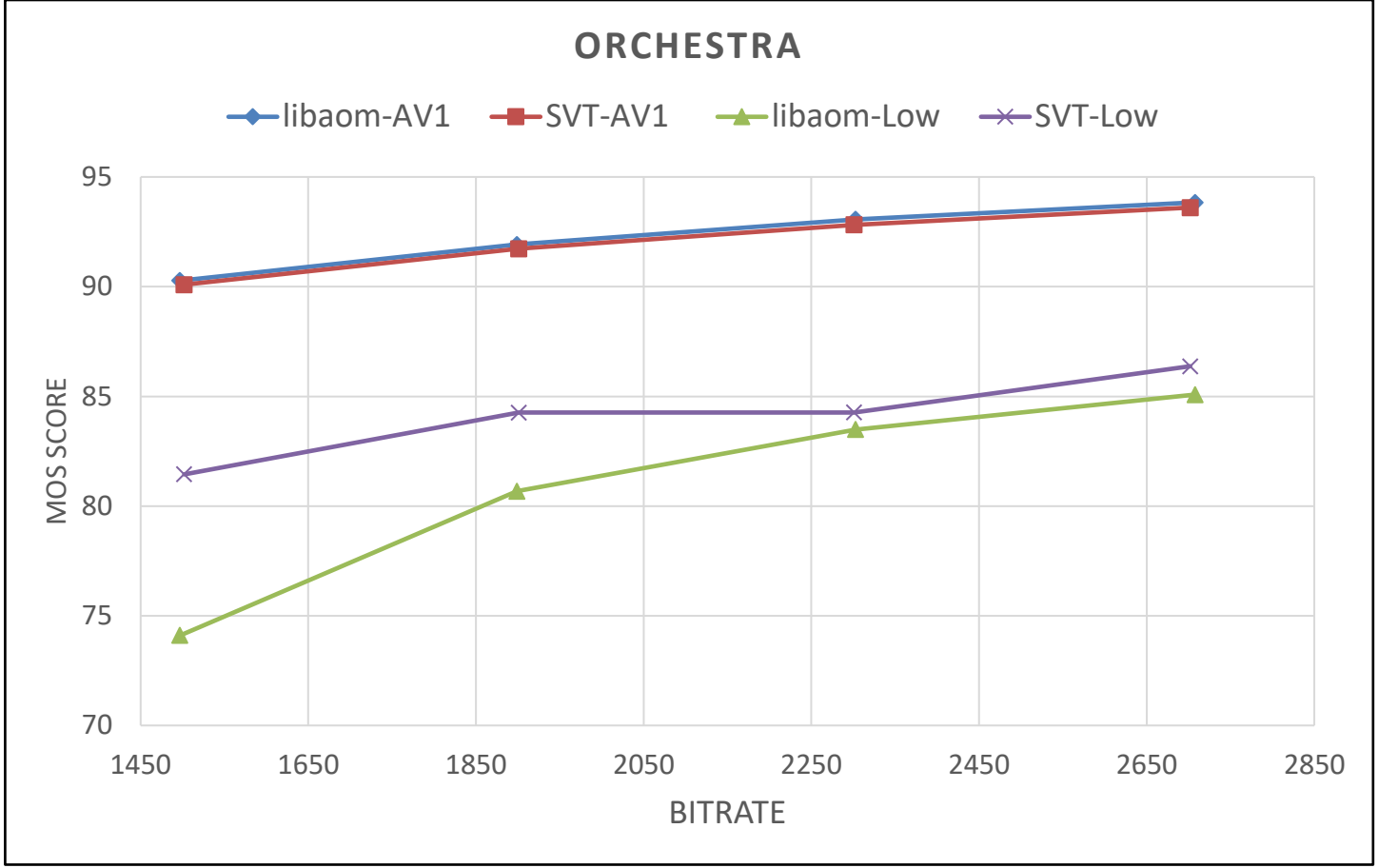
libaom-AV1	SVT-AV1	
-0.62%	0.63%	**VMAF**
-3.56%	3.70%	**Low-Frame**

Entertainment



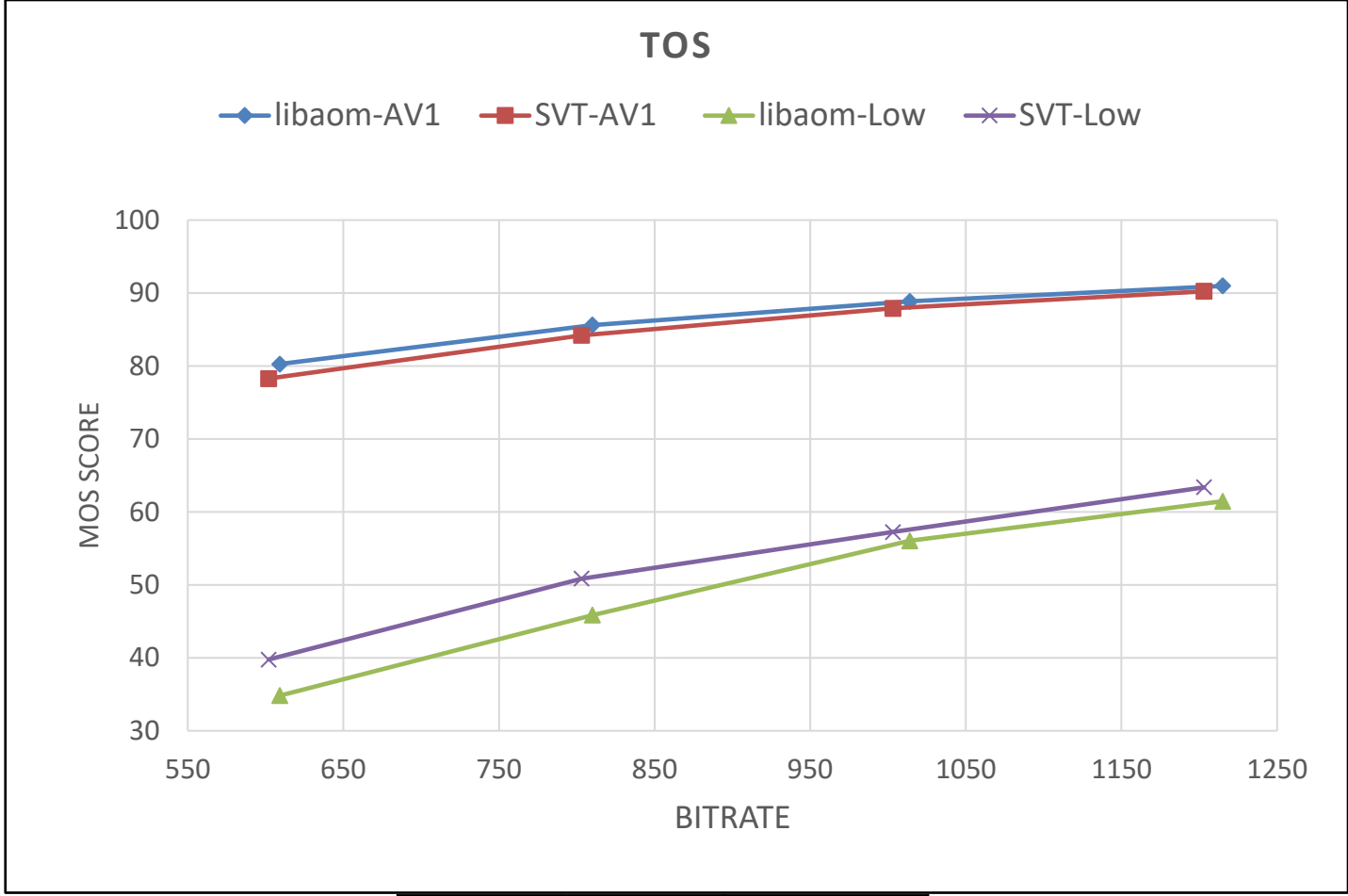
libaom-AV1	SVT-AV1	
-5.32%	5.62%	**VMAF**
2.09%	-2.04%	**Low-Frame**

Entertainment



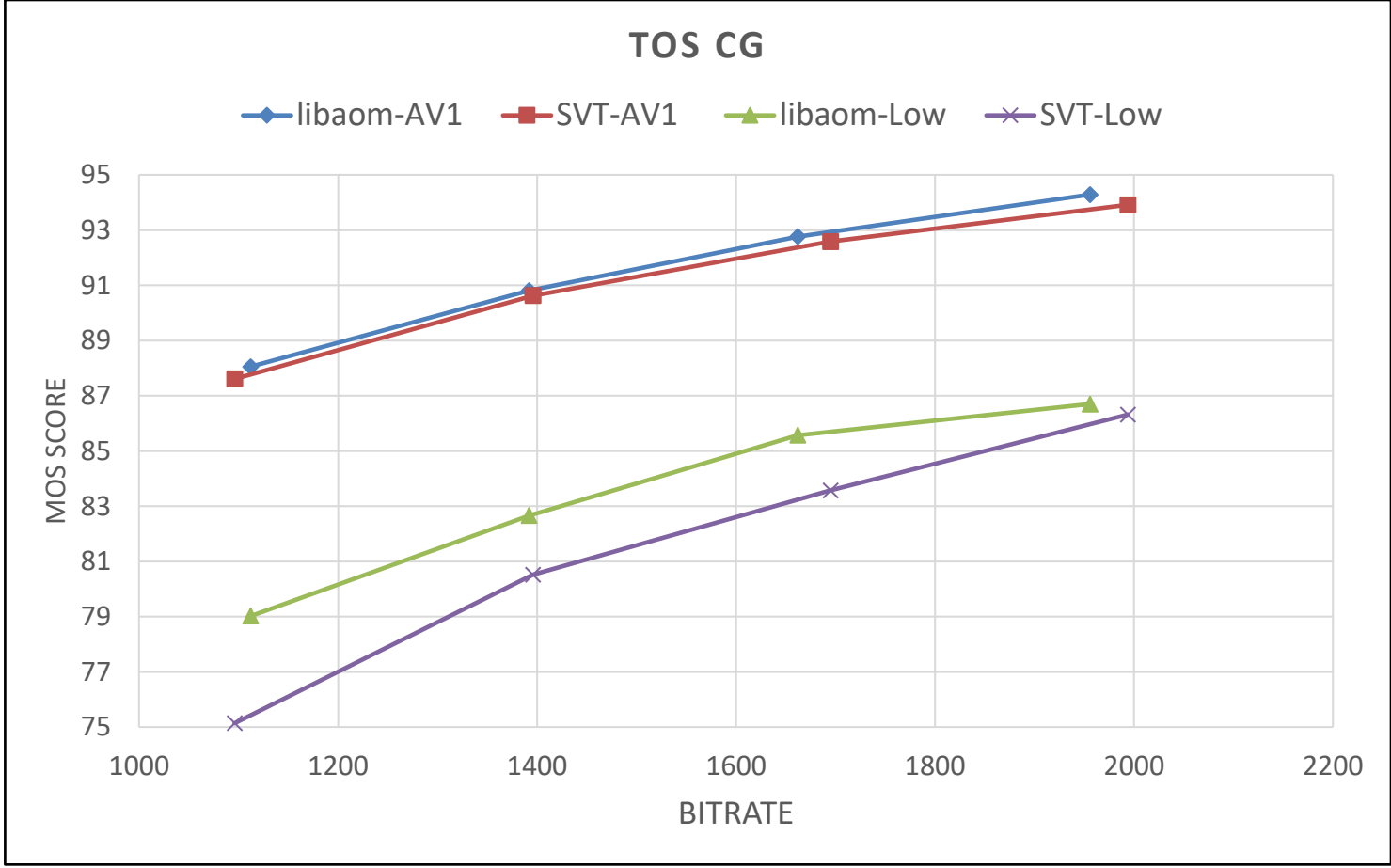
libaom-AV1	SVT-AV1	
-3.59%	3.73%	**VMAF**
29.54%	-22.80%	**Low-Frame**

Entertainment



libaom-AV1	SVT-AV1	
-6.69%	7.17%	**VMAF**
11.26%	-10.12%	**Low-Frame**

Entertainment



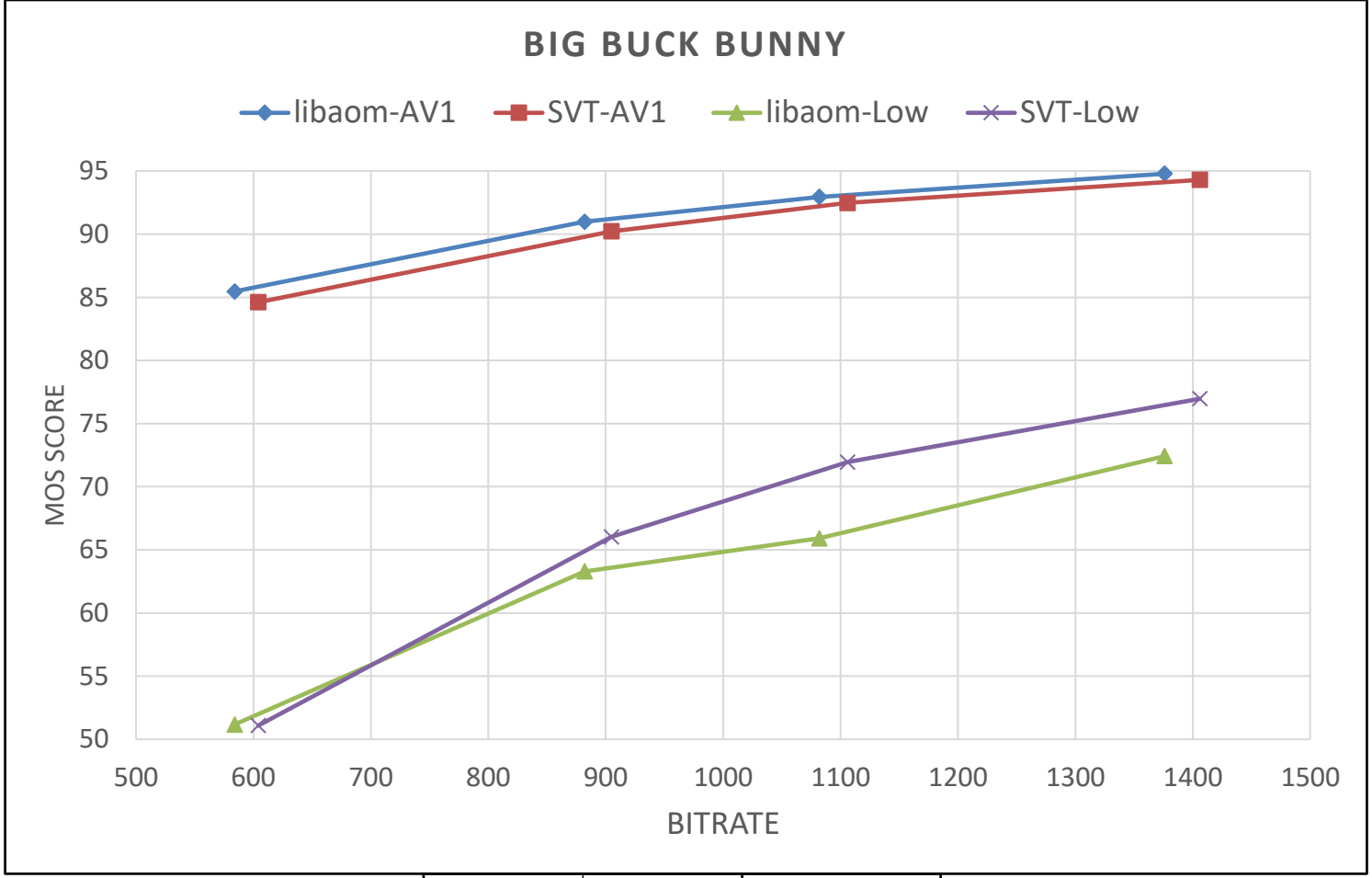
libaom-AV1	SVT-AV1	
-2.63%	2.70%	**VMAF**
-12.99%	14.93%	**Low-Frame**

Entertainment

libaom-AV1	SVT-AV1	Entertainment
-4.27%	4.51%	VMAF
6.02%	-4.21%	Low-Frame

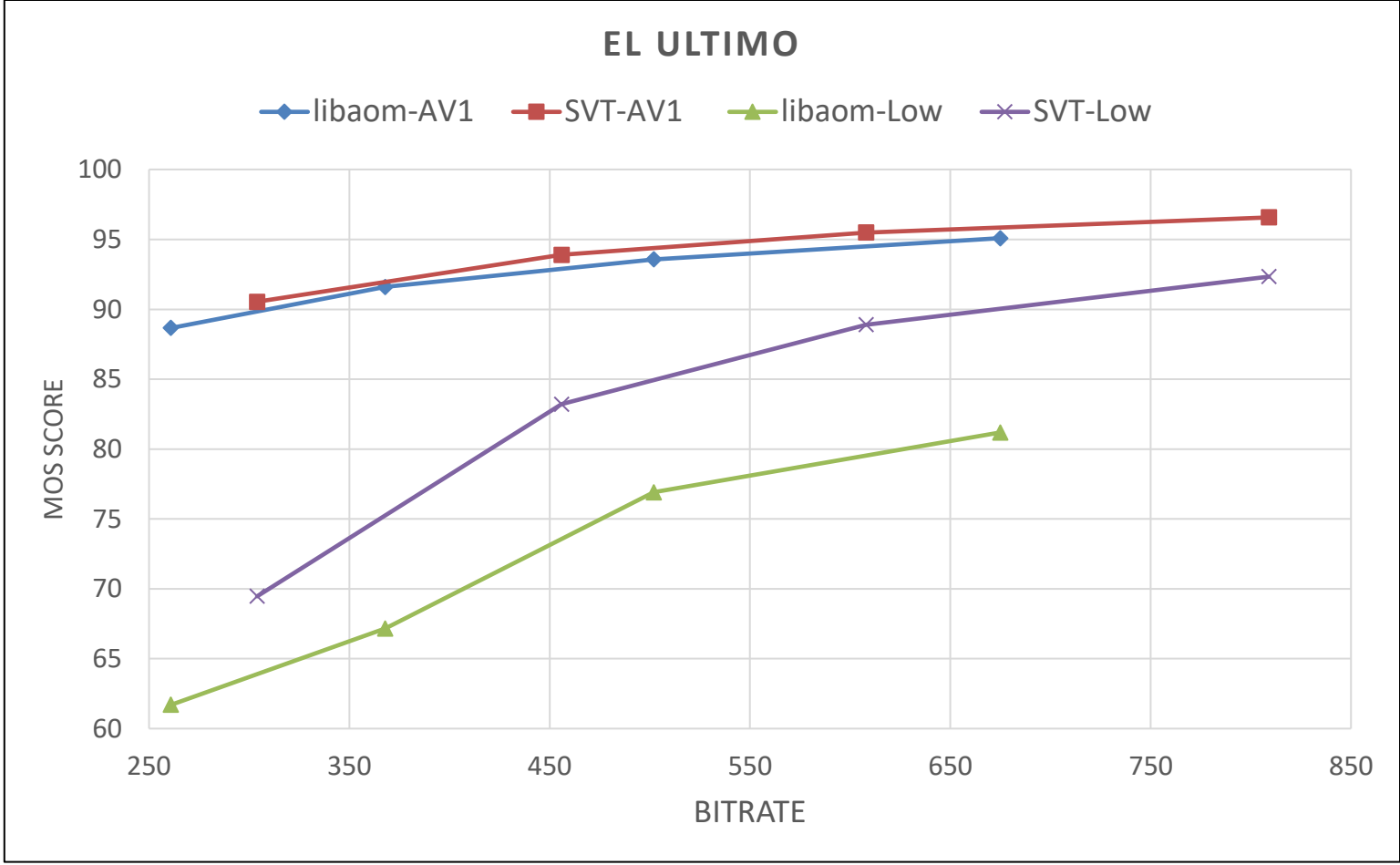
libaom-AV1	SVT-AV1	
-6.74%	7.22%	VMAF
9.81%	-8.94%	Low-Frame

Animations



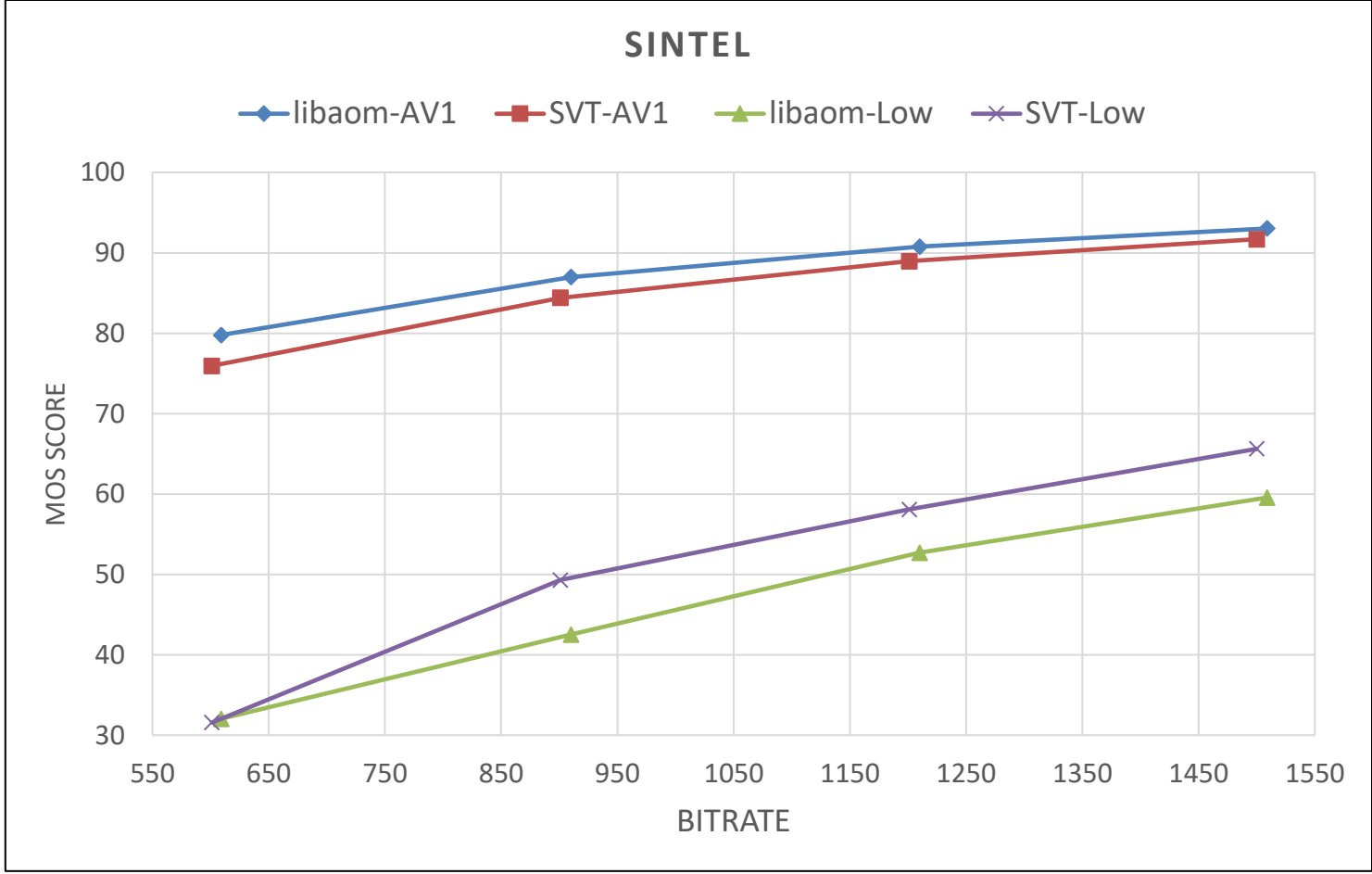
libaom-AV1	SVT-AV1	
-8.60%	9.41%	**VMAF**
6.49%	-6.10%	**Low-Frame**

Animations



libaom-AV1	SVT-AV1	
13.08%	-11.57%	**VMAF**
39.18%	-28.15%	**Low-Frame**

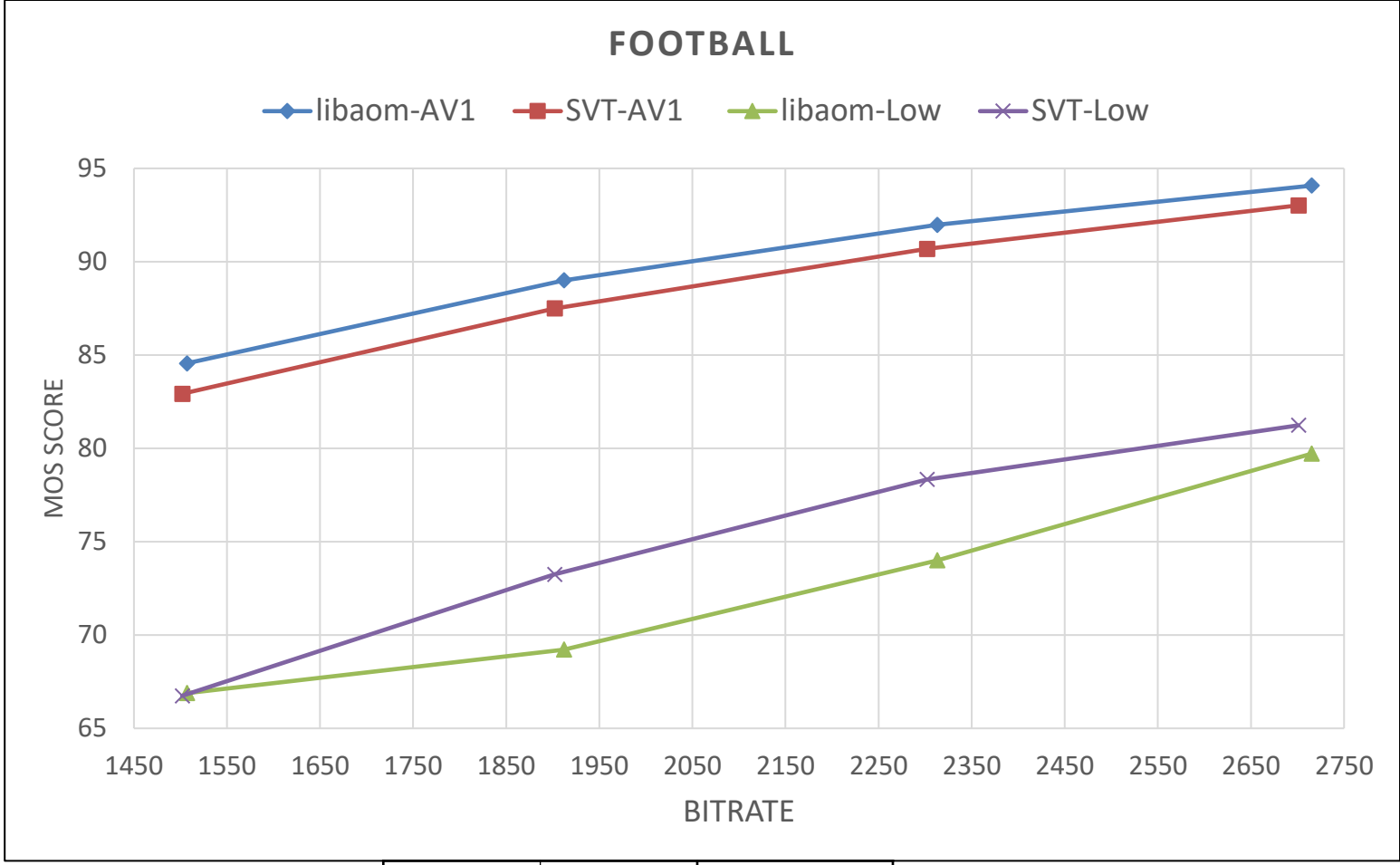
Animations



libaom-AV1	SVT-AV1	Animation
-3.05%	4.54%	VMAF
21.08%	-16.39%	Low-Frame

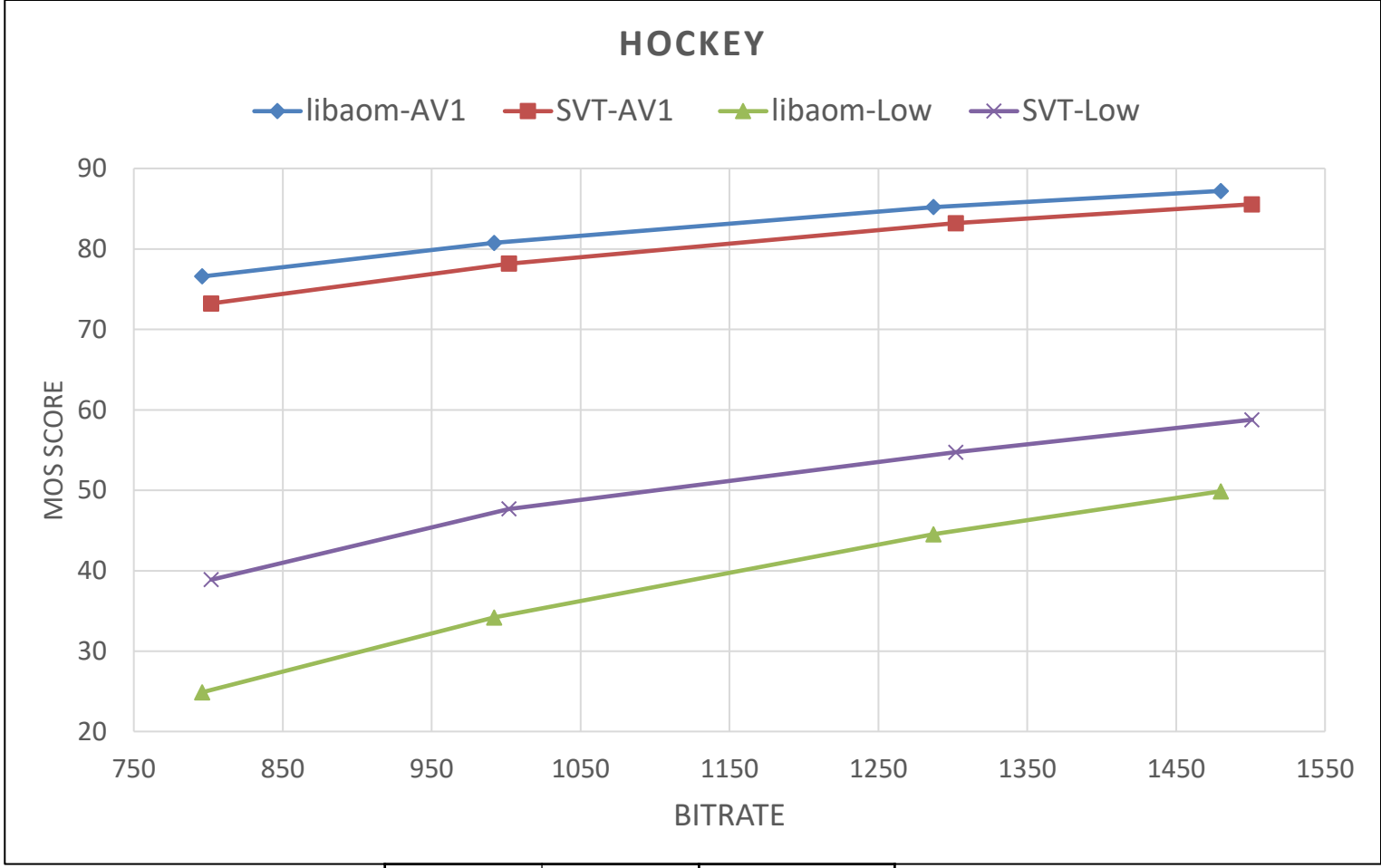
libaom-AV1	SVT-AV1	
-13.62%	15.77%	VMAF
17.55%	-14.93%	Low-Frame

Sports



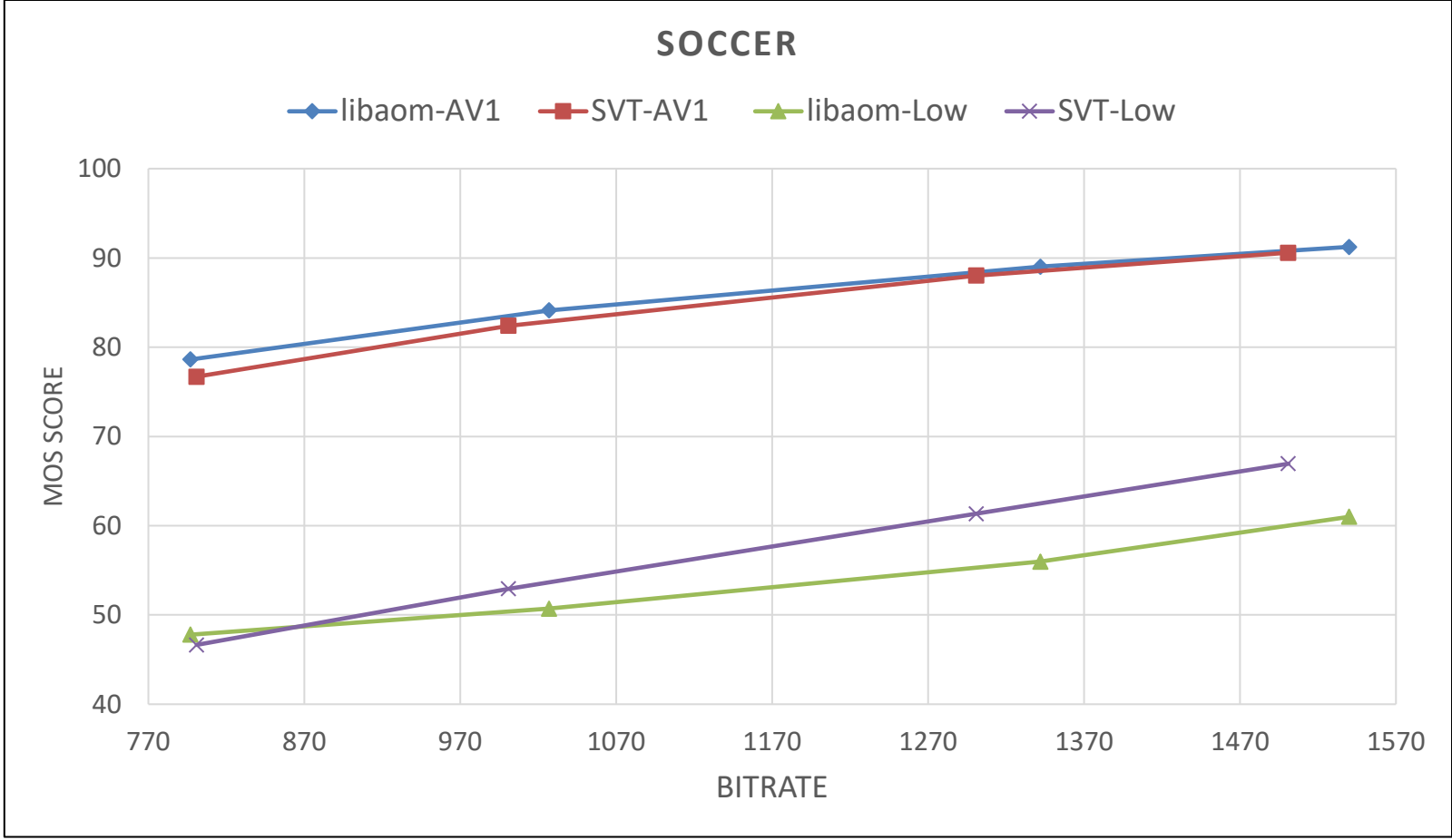
libaom-AV1	SVT-AV1	
-7.76%	8.41%	**VMAF**
15.64%	-13.53%	**Low-Frame**

Sports



libaom-AV1	SVT-AV1	
-13.11%	15.09%	**VMAF**
39.91%	-28.52%	**Low-Frame**

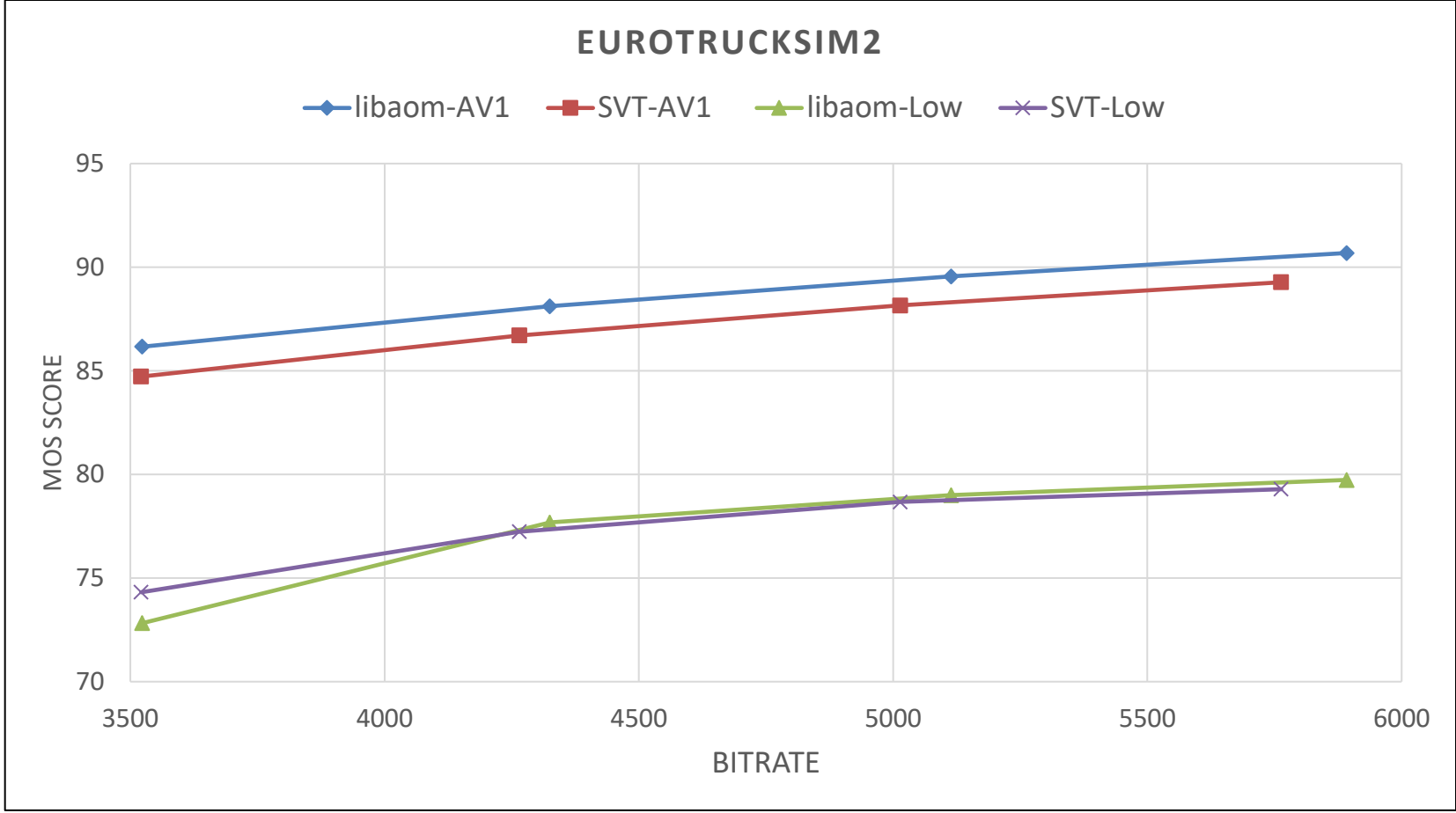
Sports



libaom-AV1	SVT-AV1	Sports
-8.46%	9.41%	VMAF
23.70%	-18.50%	Low-Frame

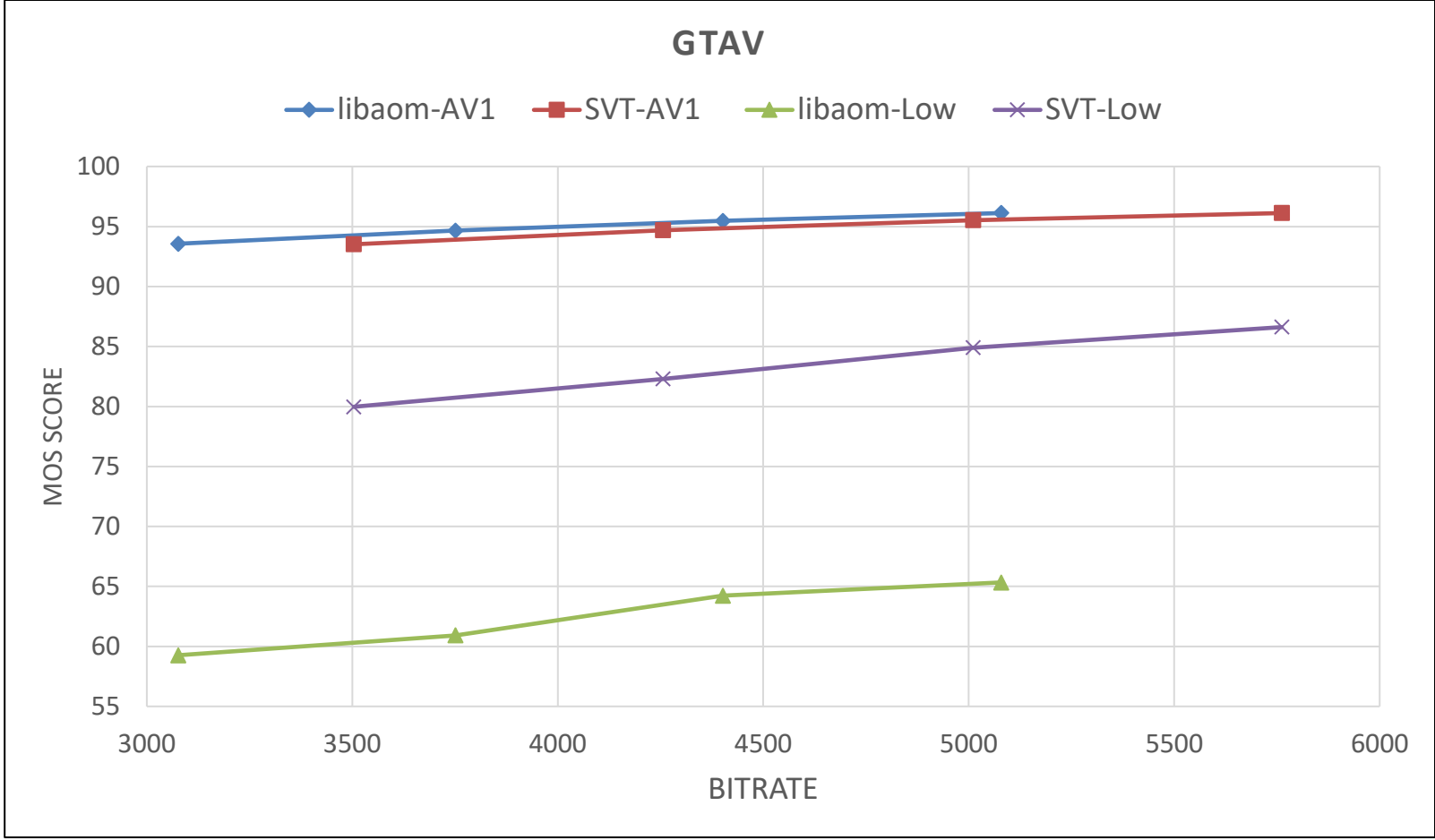
libaom-AV1	SVT-AV1	
-4.52%	4.74%	VMAF
15.56%	-13.46%	Low-Frame

Games



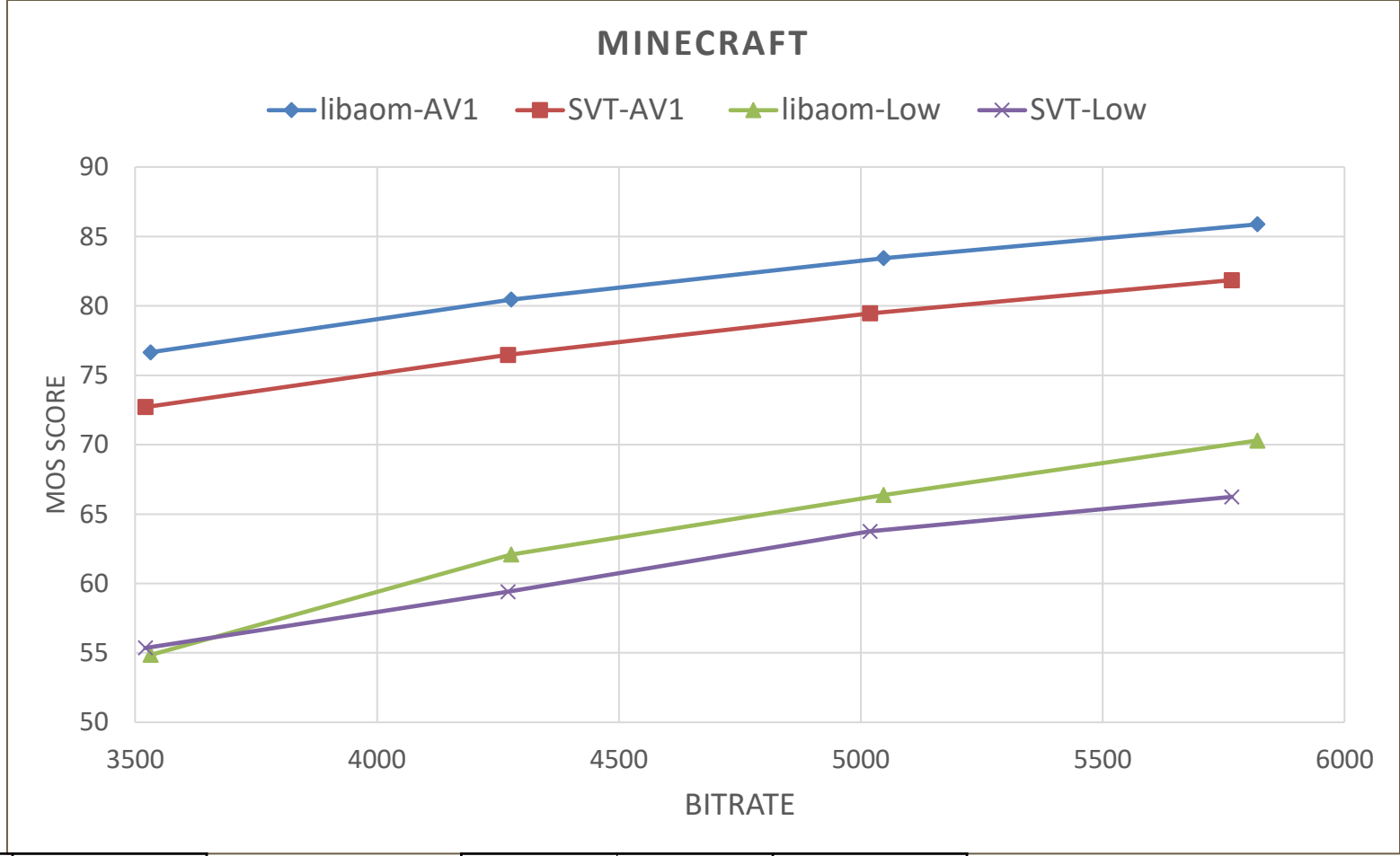
libaom-AV1	SVT-AV1	
-13.23%	15.25%	VMAF
-0.90%	0.90%	Low-Frame

Games



libaom-AV1	SVT-AV1	
-11.45%	12.93%	**VMAF**
100.00%	-100.00%	**Low-Frame**

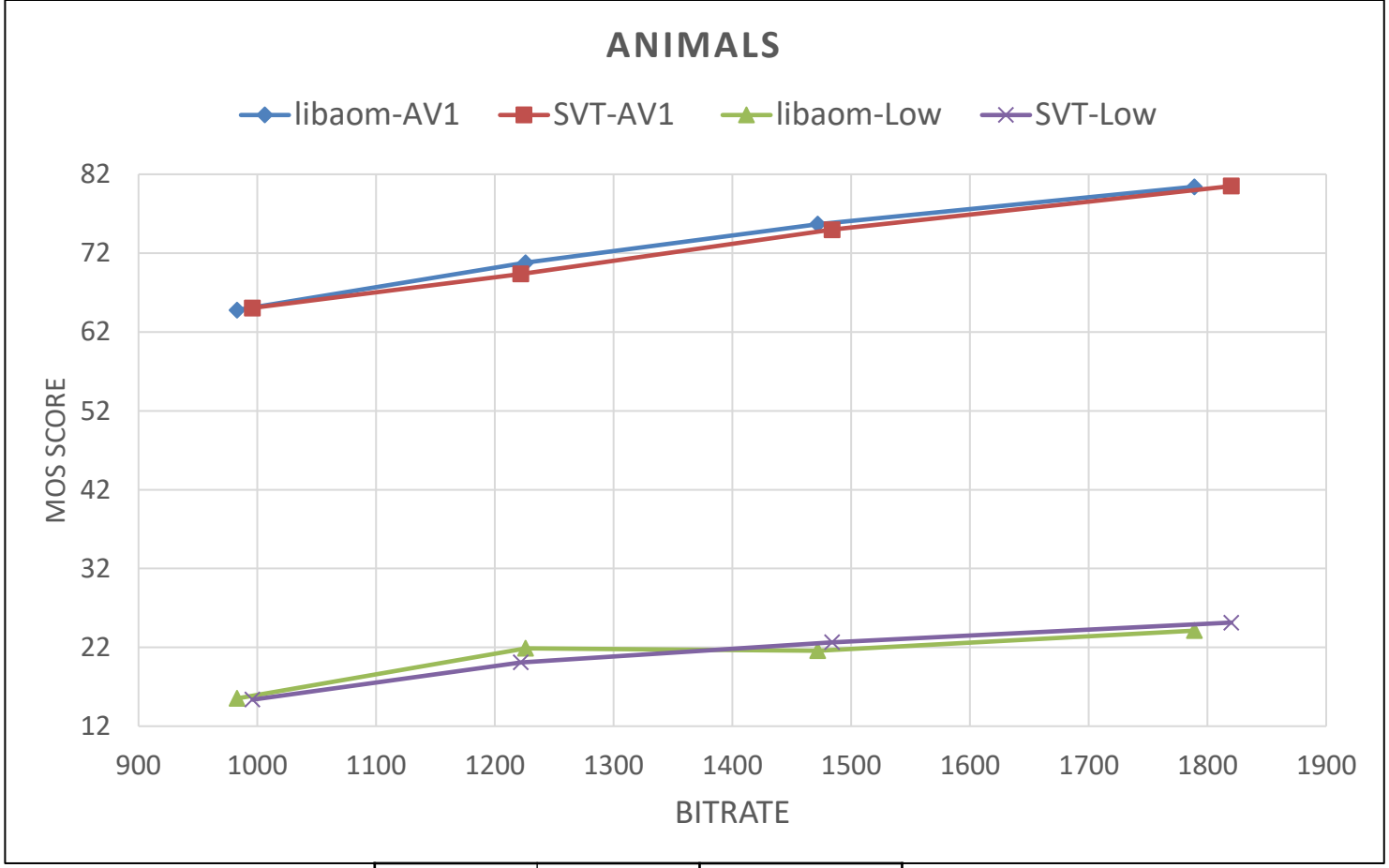
Games



libaom-AV1	SVT-AV1	Games
-14.59%	17.25%	VMAF
30.52%	-30.32%	Low-Frame

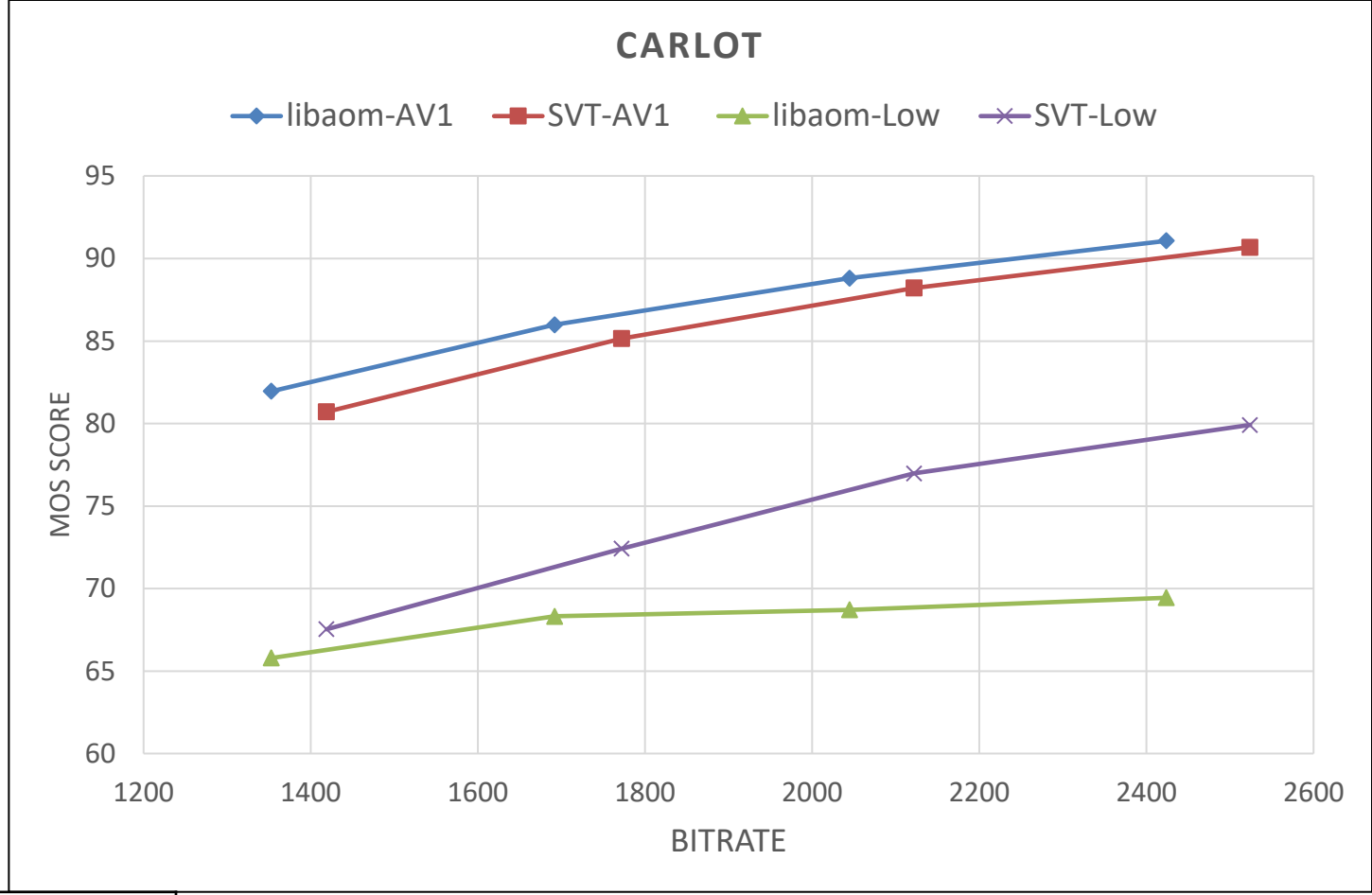
libaom-AV1	SVT-AV1	
-19.07%	23.57%	VMAF
-7.53%	8.15%	Low-Frame

Other



libaom-AV1	SVT-AV1	
-3.38%	3.50%	**VMAF**
3.39%	-3.28%	**Low-Frame**

Other



libaom-AV1	SVT-AV1	Other
-5.97%	6.43%	VMAF
14.98%	-12.14%	Low-Frame

libaom-AV1	SVT-AV1	
-8.56%	9.36%	VMAF
26.57%	-20.99%	Low-Frame