GTD-VC

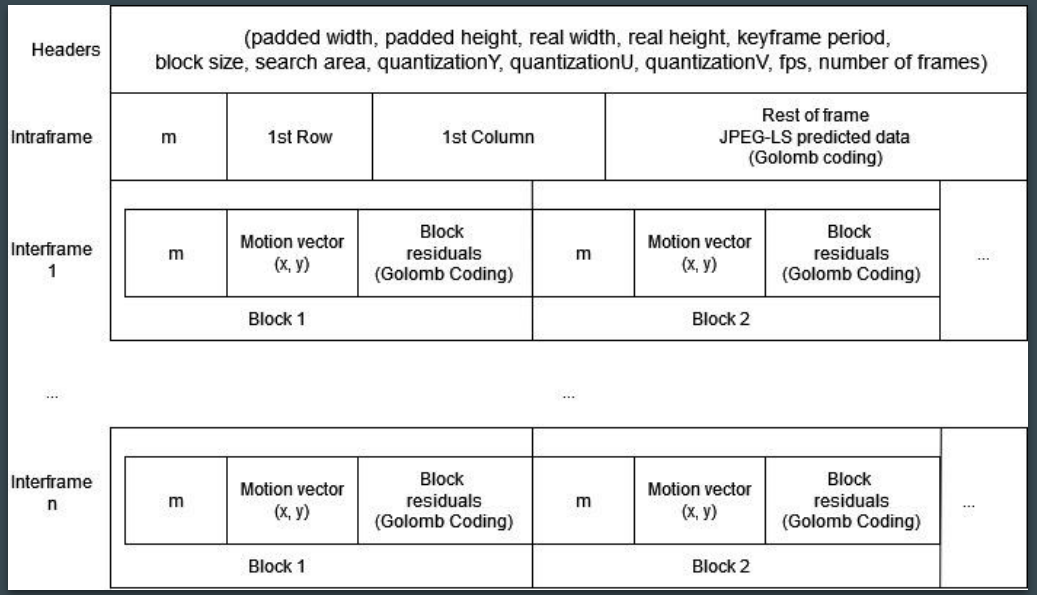
Video Codec

CSLP 2023/24 Project - Video Codec

Made by: João Dourado 108636
Diogo Marto 108298 GROUP 3
Tiago Pereira 108546

Features

- Predictive Coding
- Motion-Based Block-Coding
- Hybrid Coding (Intra + Inter Frame with Predictive/Motion-Block Coding)
- Golomb Coding of Residuals (with **m-parameter estimation**)
- Lossy Coding (Quantization-based)



Methodology

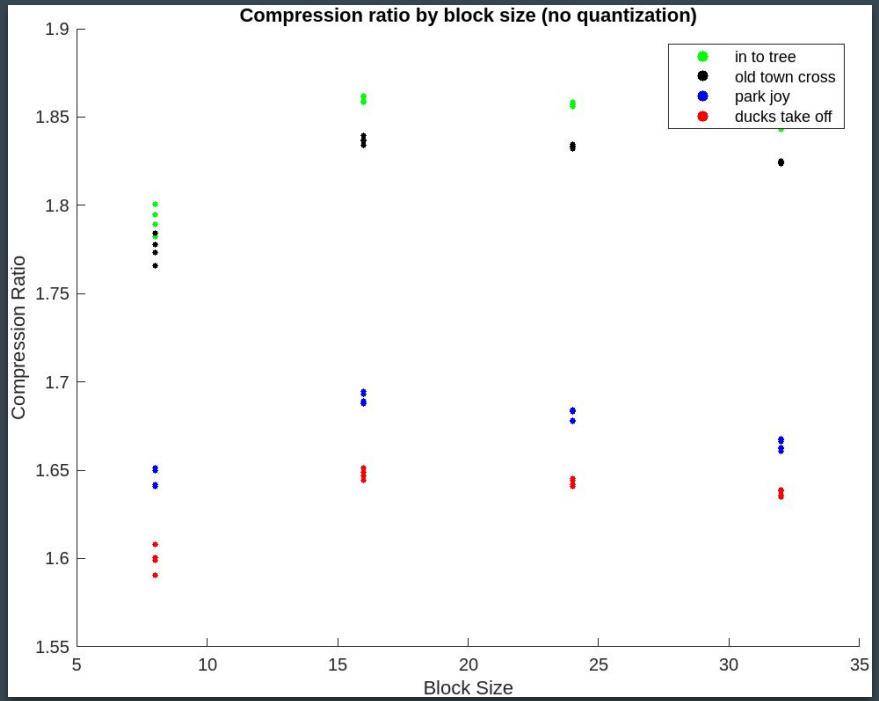
To test our codec, we ran 1440 (+32) different configurations using a script we made.

- 19h24m (Single-Thread) -

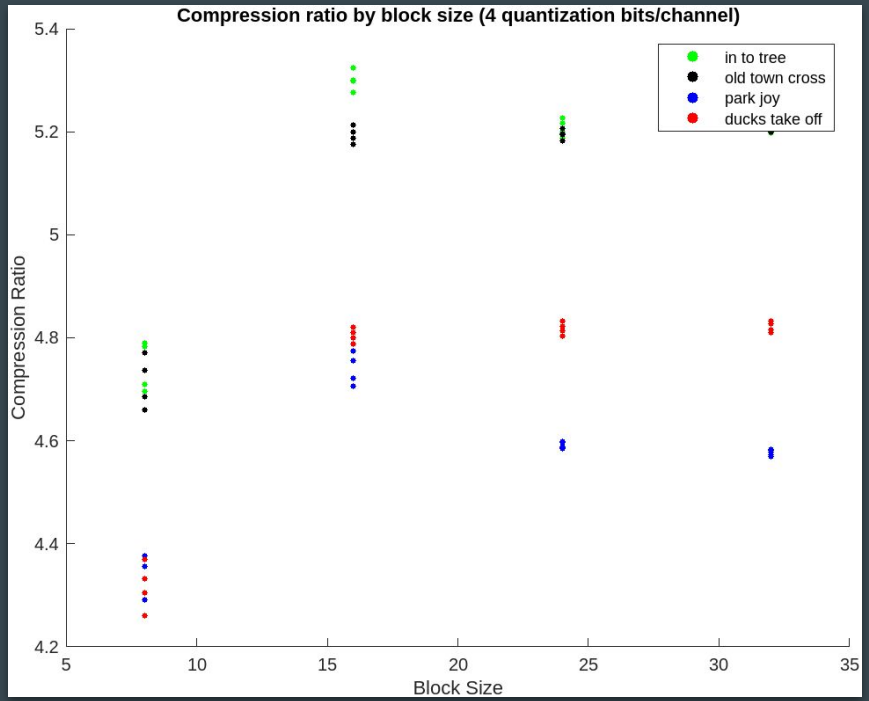
```
tiago@Tigas:~/CLionProjects/GTD-VC$ python3 run.py resources/videos -c out -b 8 33 8 -s 16 33 8 -k 10
101 50 -q 0 5 -r tigas.csv
[INFO] Video list:
[
    ducks_take_off_444_720p50.y4m
   ,park_joy_444_720p50.y4m
   ,in_to_tree_444_720p50.y4m
   ,old_town_cross_444_720p50.y4m
]
[INFO] Input folder: /home/tiago/CLionProjects/GTD-VC/resources/videos/
[INFO] All required files found.
[INFO] Output folder: /home/tiago/CLionProjects/GTD-VC/resources/videos/
[INFO] Report path: /home/tiago/CLionProjects/GTD-VC/resources/videos/tigas.csv
[INFO] Codec: out
[INFO] Auto delete: True
[INFO] Block size: [8, 16, 24, 32]
[INFO] Search area: [16, 24, 32]
[INFO] KeyFrame Period: [10, 60]
[INFO] Quantization: [(0, 0, 0), (0, 1, 1), (1, 1, 1), (0, 2, 2), (1, 2, 2), (2, 2, 2), (0, 3, 3), (1,
 3, 3), (2, 3, 3), (3, 3, 3), (0, 4, 4), (1, 4, 4), (2, 4, 4), (3, 4, 4), (4, 4, 4)]
[INFO] Testing 1440 configurations. (around 1 min per each one)

Do you wish to proceed: (If yes type yes or y)
>
```

Optimal parameters used - block size

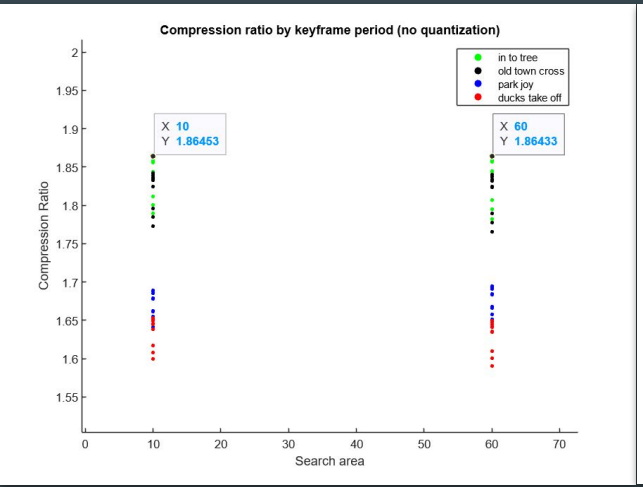


Compression Ratio by block size (no quantization)

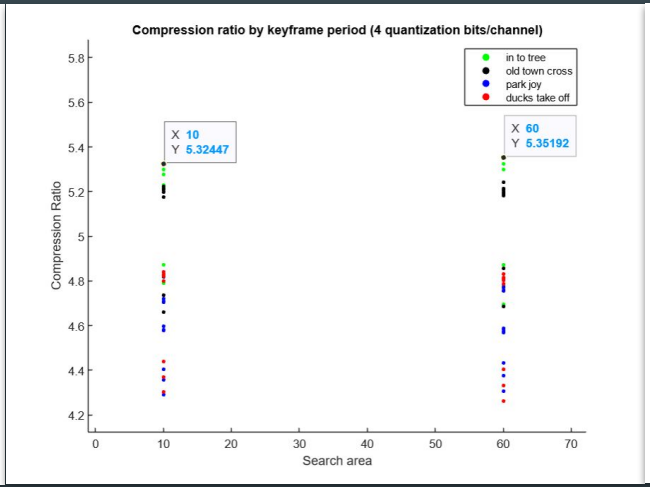


Compression Ratio by block size (lossy, quantization = 4 bits per channel)

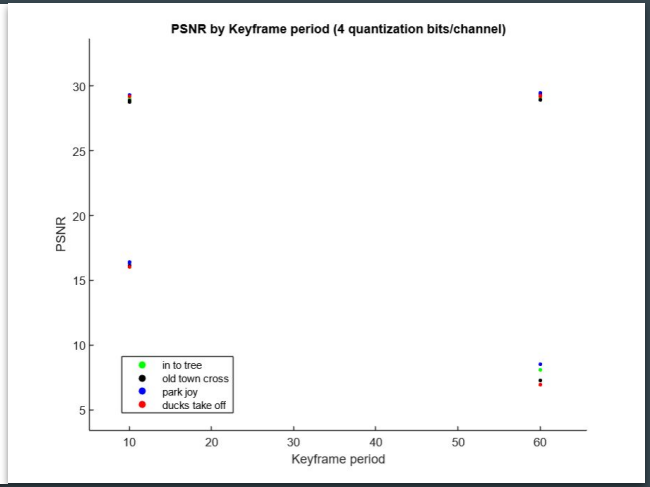
Optimal parameters used - Keyframe Period



Compression Ratio by Keyframe Period (no quantization)

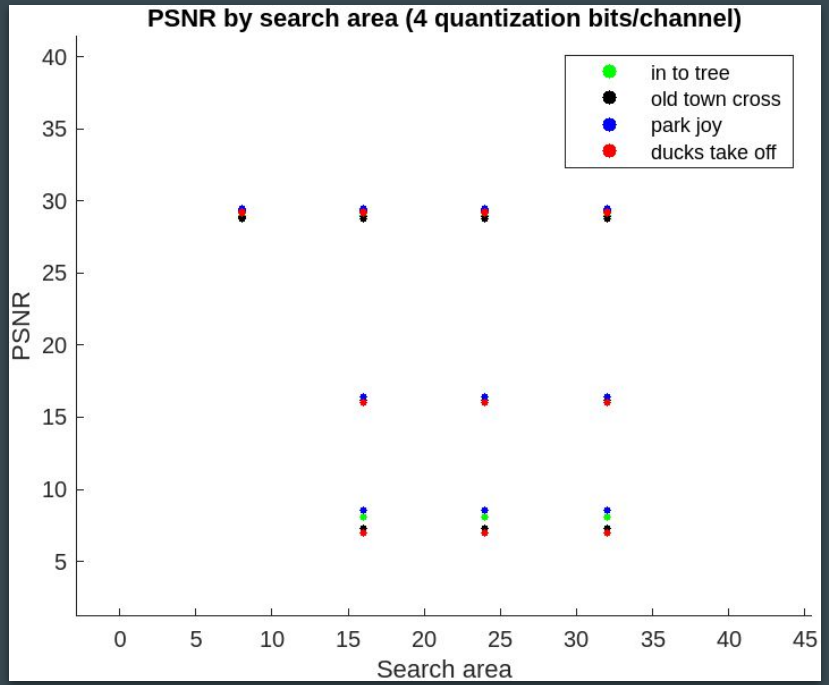


Compression Ratio by Keyframe Period (lossy, quantization = 4 bits per channel)

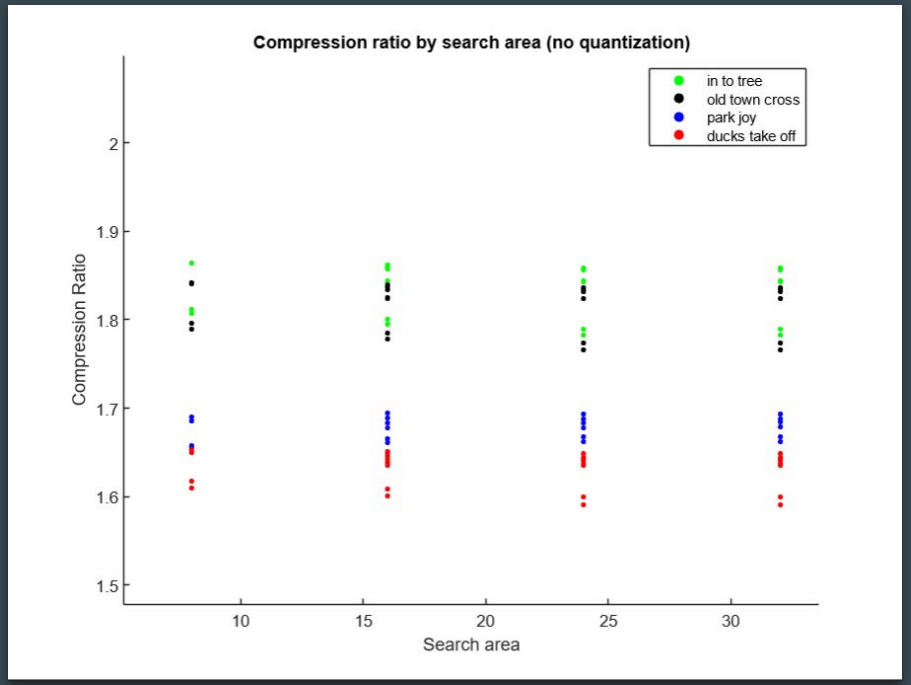


PSNR by Keyframe Period (lossy, quantization = 4 bits per channel)

Optimal parameters used - Search Area (1 of 2)

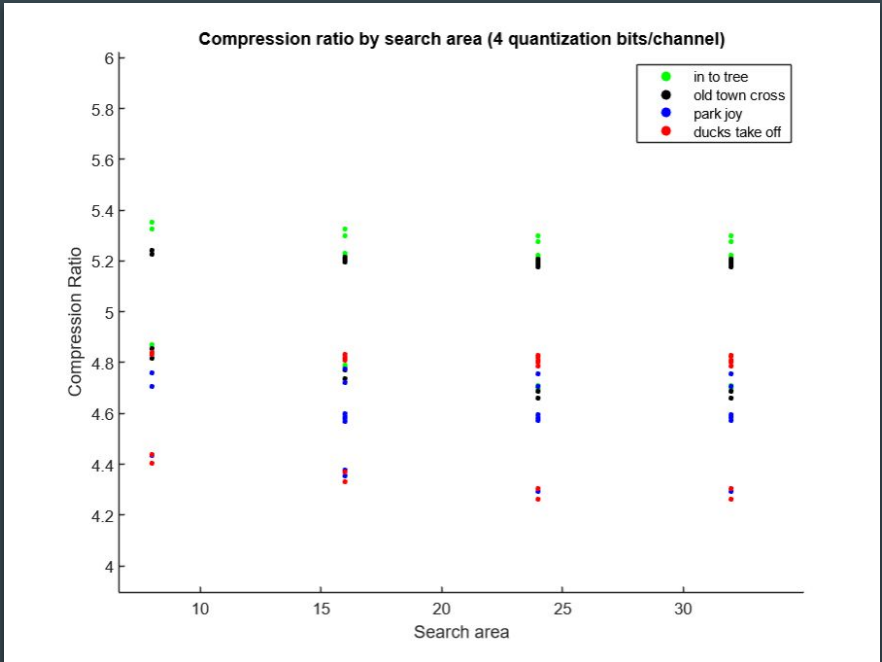


PSNR by search area (4 quantization bits per channel)

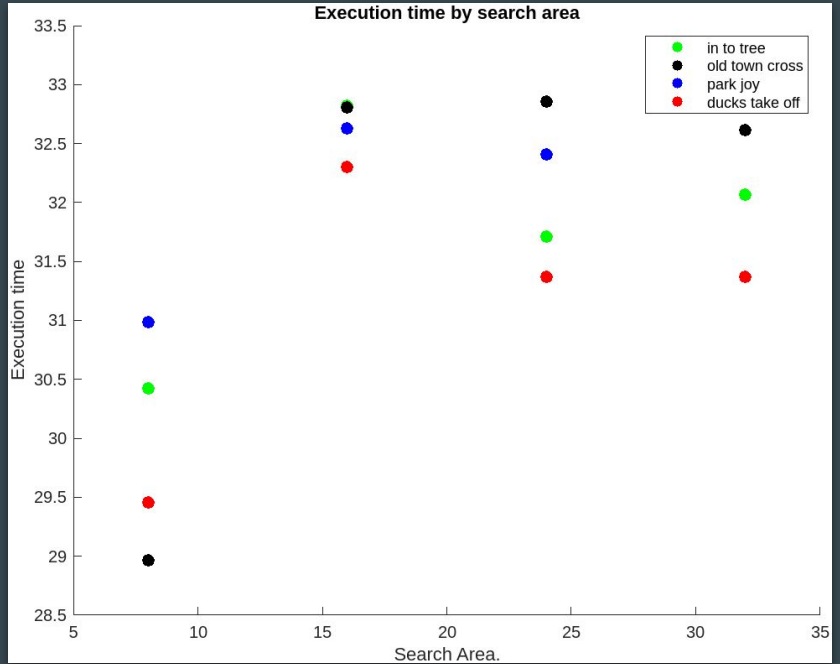


Compression Ratio by search area (no quantization)

Optimal parameters used - Search Area (2 of 2)

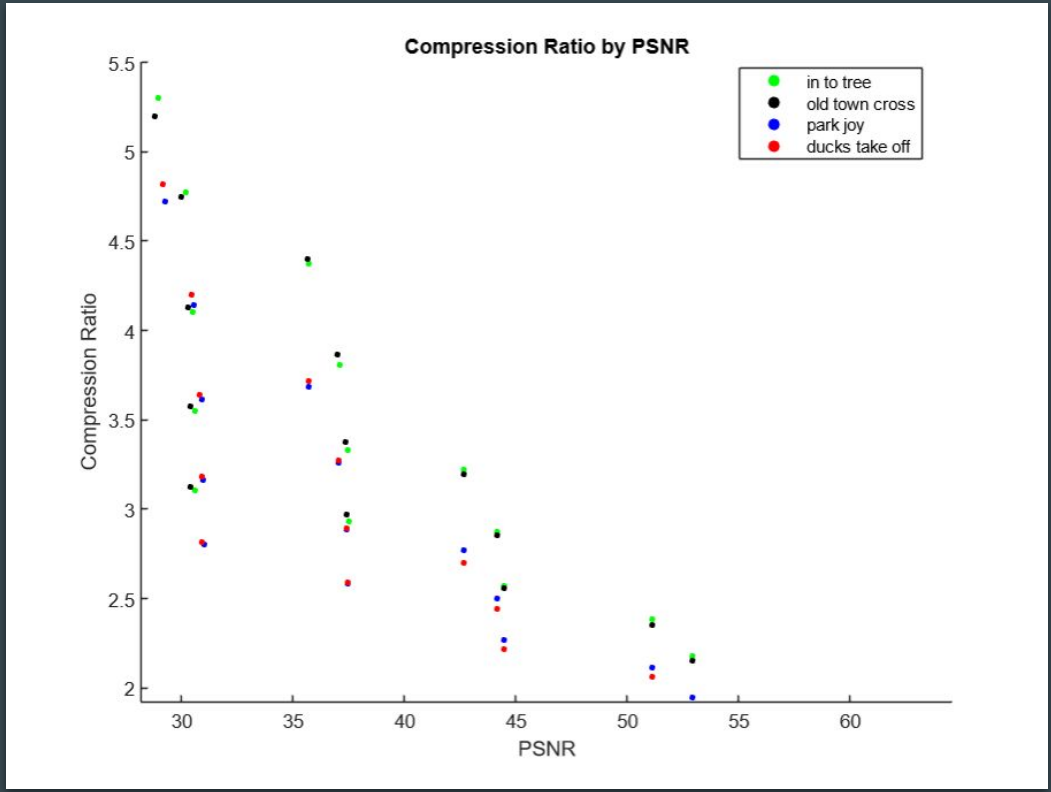


Compression Ratio by Search Area (4 quantization bits per channel)



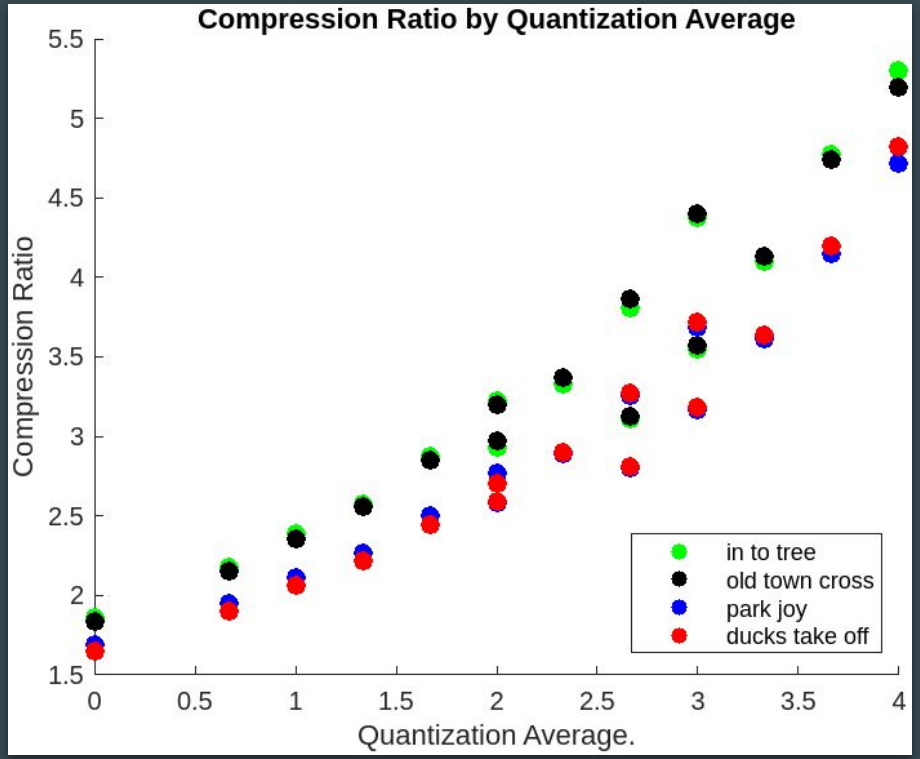
Execution Time by Search Area (no quantization)

Results - Compression Ratio by PSNR



block_size = 16; keyframe_period = 10;
search_area = 16

Results - Compression Ratio by Quantization (average of all channels)

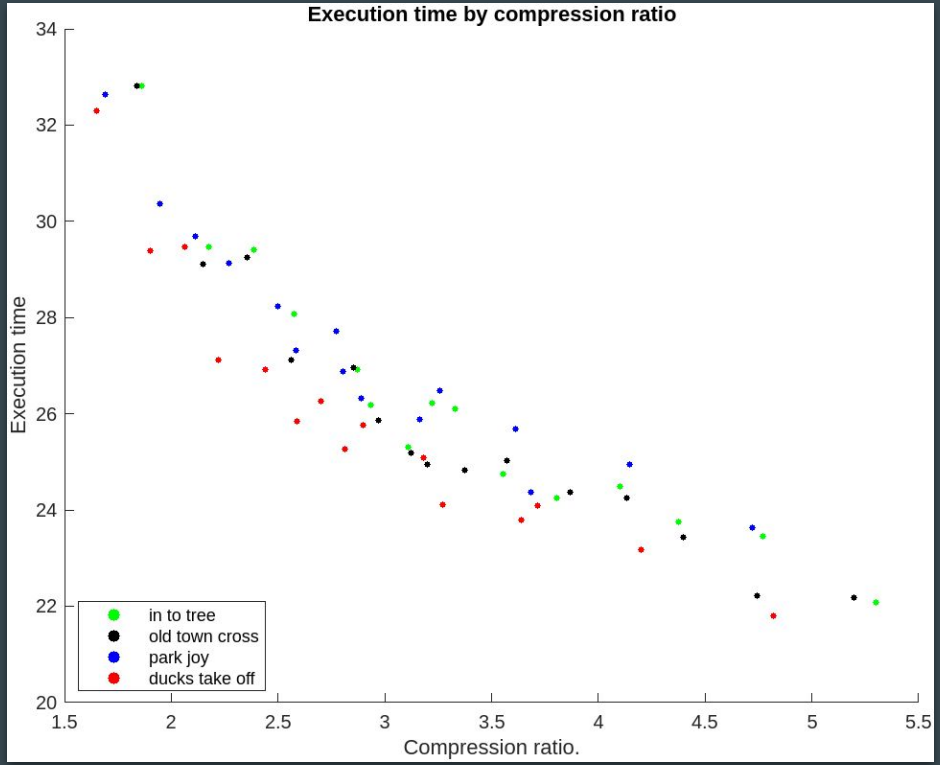


block_size = 16; keyframe_period = 10;
search_area = 16

Results - Execution Time by Compression Ratio

Encoding time:

- no quantization: **about 32s**
- 4 quantization bits/channel**: about 22s**



block_size = 16; keyframe_period = 10; search_area = 16

Optimizations - Block-matching algorithm

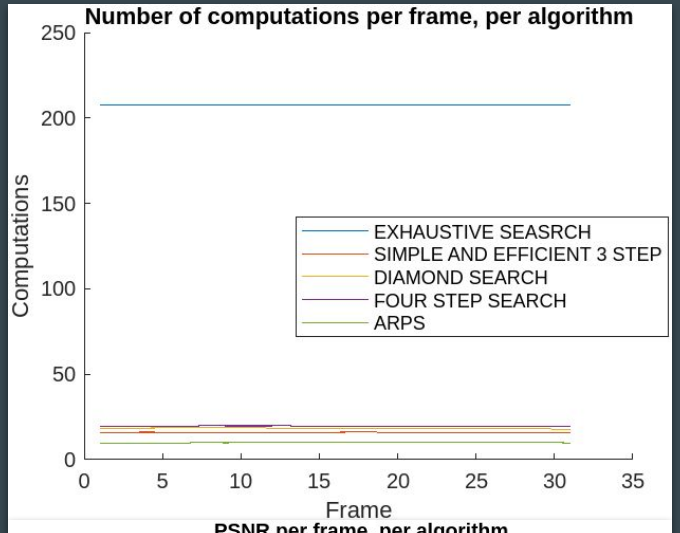
First implementation - ***Exhaustive Search***:

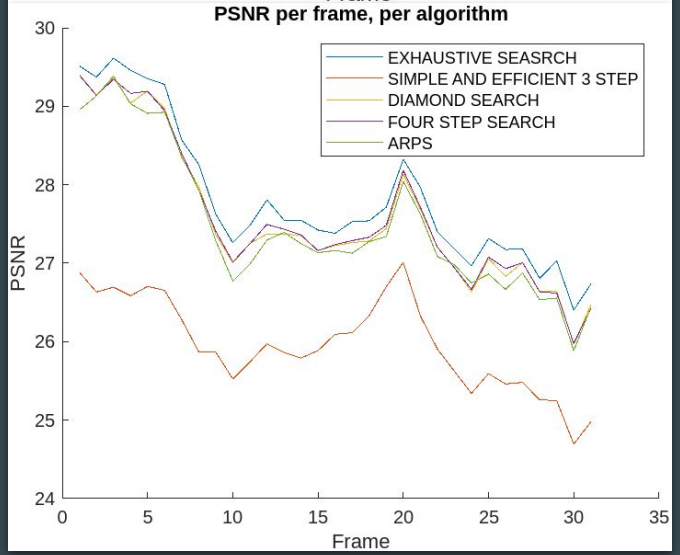
- **Pros**: very efficient
- **Cons:** very (very) slow

Final implementation - ***Four Step Search***:

- **Much faster and only slightly worse PSNR.**

Source





Optimizations - perf - BitStreamWrite/Read efficiency

- Refactored our old implementation of BitStreamWrite/BitStreamRead

- Added IO buffering for less read/write syscalls.

- Performance didn't improve as much as we thought.

```
Samples: 416K of event 'cycles:u', Event count (approx.): 764696124611
Overhead  Command  Shared Object                Symbol
  32.38%  GTD-VC   GTD-VC                       [.] BlockEncoding::calculateMAD
  26.78%  GTD-VC   GTD-VC                       [.] cv::Mat::at<unsigned char>
   9.49%  GTD-VC   GTD-VC                       [.] BlockEncoding::encodeInterframeChannel
   8.97%  GTD-VC   GTD-VC                       [.] GolombCode::encode
   6.36%  GTD-VC   GTD-VC                       [.] BitStreamWrite::write
   5.77%  GTD-VC   GTD-VC                       [.] cv::Mat::at<unsigned char>
   1.76%  GTD-VC   GTD-VC                       [.] BitStreamWrite::should_refresh_small_buffer
   1.55%  GTD-VC   GTD-VC                       [.] BlockEncoding::encodeValue
   1.50%  GTD-VC   GTD-VC                       [.] GolombCode::estimate
   1.10%  GTD-VC   GTD-VC                       [.] GolombCode::mapIntToUInt
```

Before changes

```
Samples: 408K of event 'cycles:u', Event count (approx.): 746088163773
Overhead  Command  Shared Object                Symbol
  32.09%  GTD-VC   GTD-VC                       [.] BlockEncoding::calculateMAD
  26.48%  GTD-VC   GTD-VC                       [.] cv::Mat::at<unsigned char>
   9.67%  GTD-VC   GTD-VC                       [.] BlockEncoding::encodeInterframeChannel
   9.20%  GTD-VC   GTD-VC                       [.] GolombCode::encode
   6.44%  GTD-VC   GTD-VC                       [.] BitStreamWrite::write
   5.81%  GTD-VC   GTD-VC                       [.] cv::Mat::at<unsigned char>
   1.77%  GTD-VC   GTD-VC                       [.] BitStreamWrite::should_refresh_small_buffer
   1.54%  GTD-VC   GTD-VC                       [.] BlockEncoding::encodeValue
   1.52%  GTD-VC   GTD-VC                       [.] GolombCode::estimate
   1.09%  GTD-VC   GTD-VC                       [.] GolombCode::mapIntToUInt
```

After changes

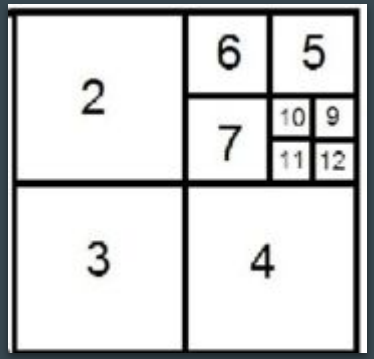
Future improvements/ideas

- DCT
- Subsampling
- Scene detection

- Concurrência
- Arps
- Classe própria para representar imagem em vez opencv mat.

- Run-Length Encoding on Golomb Coding

- Quad tree/other data structure for Golomb ‘m’ parameter & motion vectors
- Predictive methods for ‘m’ parameter & motion vectors



References

- Block matching algorithm @ Wikipedia
- Motion compensation @ Wikipedia
- Matlab implementation of block matching algorithms
- StackOverflow about IO efficiency

🤓 Thanks for listening 🤓