Object Tile Tracking in 360 video using bitrates change

December 12, 2024

1 Introduction

Github Link

With the rapid advancements in multimedia technologies, panoramic or 360-degree video has emerged as a transformative medium, offering immersive viewing experiences. Unlike traditional video formats, panoramic videos capture the entire field of view, allowing users to explore scenes interactively by panning, tilting, or rotating their devices. This format is increasingly utilized in applications such as virtual reality (VR), augmented reality (AR), tourism, real estate, and education, providing viewers with a more engaging and dynamic experience.

The core challenges associated with panoramic video projects include efficient video stitching, compression, and real-time rendering, particularly given the high resolution and data intensity of 360-degree content. Ensuring seamless playback across platforms and devices also demands robust encoding solutions. Popular codecs like HEVC, VP9, and AV1 play a pivotal role in optimizing performance.

This project focuses on the leveraging the bitrate spike to track an object from a tile to another allowing for reducing the load of reducing the detection load when an object jumps from a single tile to another which in a non-linear manner.

2 Literature Review

Several studies have explored the challenges and innovations in panoramic and 360-degree video streaming, object recognition, and image rectification for immersive technologies.

2.1 TileClipper: Lightweight Selection of Regions of Interest from Videos for Traffic Surveillance[1]

This research introduces TileClipper, a method for efficiently identifying and extracting regions of interest (ROIs) in traffic surveillance videos. The proposed

approach enables real-time selection of specific video segments for further analysis, optimizing resource usage by focusing on critical areas of the footage. By employing a lightweight algorithm for video tile extraction, TileClipper reduces computational overhead while maintaining high accuracy in ROI detection, making it suitable for large-scale traffic monitoring systems. This method not only enhances the efficiency of video processing but also supports better scalability and adaptability in diverse surveillance scenarios.

2.2 Oculus 360-Degree Video Streaming

The research paper *"A Measurement Study of Oculus 360 Degree Video Streaming"*[6] investigates the complexities and innovative solutions involved in streaming 360-degree videos on the Oculus platform. The study emphasizes the impact of projection techniques, particularly the offset cubic projection, on video quality and bandwidth efficiency. This projection maps spherical videos onto cube faces aligned with the user's view direction, optimizing resource allocation and enhancing playback quality. Notably, the offset cubic projection reduces data consumption by up to 16.4% compared to traditional equirectangular projections without compromising the visual experience.

Additionally, the study highlights adaptive streaming strategies employed by Oculus, which adjust video resolution and bitrate based on network conditions and the viewer's head orientation. These strategies assess 22 different cube orientations to select optimal video quality for each segment. Despite these advancements, inefficiencies persist, such as wasted segments caused by sudden head movements or inaccurate predictions. The paper concludes with recommendations for further optimization in bandwidth utilization and prediction accuracy.

2.3 Object Recognition in Panoramic Images

The paper *"What's in my Room? Object Recognition on Indoor Panoramic Images"*[5] introduces an advanced object recognition system for indoor 360-degree panoramic images. Building on the BlitzNet deep learning model, this system integrates object detection and semantic segmentation for equirectangular projections, recognizing 14 categories of indoor objects. The method refines detection and segmentation by leveraging spatial context, converting segmentation masks into 3D bounding boxes that accurately represent object positions in 3D space.

A significant contribution of this study is the fusion of 2D semantic segmentation with 3D room layout estimation. The authors argue that understanding a room's layout enhances detection accuracy, as spatial context helps predict object positions more reliably. This approach improves detection robustness, with applications in virtual reality (VR), robotics, and augmented reality (AR).

2.4 Fisheye Image Rectification

The paper *"SimFIR: A Simple Framework for Fisheye Image Rectification with Self-supervised Representation Learning"*[3] presents a novel solution for correcting fisheye image distortions. SimFIR employs self-supervised learning techniques, leveraging a Vision Transformer (ViT) to extract features focused on distortion patterns rather than visual content. This allows the framework to correct distortions effectively without labeled data.

SimFIR's self-supervised approach enhances performance in downstream rectification tasks by learning fine-grained distortion details. Through extensive experiments, the authors demonstrate that SimFIR outperforms existing methods in both synthetic and real-world scenarios, making it valuable for applications in panoramic photography and virtual reality.

2.5 Unified Framework for Panoramic Image Distortion Correction[2]

A unified framework for correcting panoramic image distortions addresses issues associated with wide-angle and panoramic images, particularly those captured with fisheye or ultra-wide-angle lenses. By using projection techniques such as cubic and spherical projections, the framework re-projects distorted images onto geometric surfaces, reducing stretching and compression while preserving image quality.

The framework incorporates advanced feature extraction and matching techniques, including SuperPoint and SuperGlue algorithms, to detect key points and seamlessly stitch images. These innovations improve the usability of panoramic images in virtual reality, surveillance, and autonomous systems, enhancing visual accuracy and efficiency in distortion correction.

3 Our Research

Through our detailed study of the tile clipper, we have gained an understanding of how the bitrate of a video frame is determined. The process for calculating the bitrate of a video file and segmenting it for detailed analysis involves using powerful multimedia tools like FFprobe and FFmpeg. These tools are essential for precise video processing, particularly in handling large video files, especially in the context of panoramic or tiled video analysis.

3.1 Utility functions

3.1.1 Get Video Duration

The function *get_video_duration* uses FFprobe to determine the duration of the input video. This is achieved by extracting the duration metadata of the video stream.

```
ffprobe -v error
-select_streams v:0
-show_entries stream=duration
-of csv=p=0
"file_path"
```

3.1.2 Video Segmentation

The *get_video_segments* function divides the input video into smaller segments of a specified duration (default: 0.5 seconds) and saves them in a designated directory. Each segment is encoded using H.264 for video and AAC for audio, ensuring compatibility and efficiency.

```
ffmpeg -i "/path/to/video.mp4"
-c:v libx264
-c:a aac
-strict experimental
-b:a 192k
-force_key_frames "expr:gte(t,n_forced*0.5)"
-f segment -segment_time "0.5"
-reset_timestamps 1
-map 0
"/path/to/segments/output_part_%d.mp4"
```

3.1.3 Cubemap Conversion

The *convert_video_to_cubemap* function converts the input equirectangular video into a cubemap format. This process rearranges the video frames into a cubic projection using FFmpeg's v360 filter[4]. This transformation is essential for processing 360-degree videos in tile-based formats.

```
ffmpeg -i "/path/to/video.mp4"
-vf "v360=input=equirect:output=c3x2"
-q:v 2
"/path/to/output_cube.mp4"
```

3.1.4 Bitrate Calculation

The *calculate_bitrate* function retrieves the video bitrate for a given file using FFprobe. It is essential for analyzing the data rate requirements of video tiles.

```
ffprobe -v error
-select_streams v
-show_entries stream=bit_rate
-of default=noprint_wrappers=1
"/path/to/video.mp4"
```

3.1.5 Get Video Resolution

The *get_video_resolution* function fetches the video's resolution and scales it appropriately to fit the cubemap format. The resulting dimensions are critical for dividing the video into tiles.

```
ffprobe -v error
-select_streams v:0
-show_entries stream=width,height
-of csv=s=x:p=0
"/path/to/video.mp4"
```

3.2 Cubemap Processing and Tile Extraction

The script performs several steps to process the cubemap video:

3.2.1 Resolution and Cubemap Conversion:

The equirectangular video is converted into a cubemap format ($c3x2$), splitting the video into six faces of a cube. Dimensions for tiles are calculated based on the cubemap format, dividing the video into 3×2 grid cells.

3.2.2 Tile Extraction:

Using FFmpeg's crop filter, the video is divided into individual tiles (12 total: 6 for the left eye and 6 for the right eye, assuming stereo output).

3.3 Segmentation and Analysis

3.3.1 Video Segmentation:

The cubemap video is segmented into 0.5-second parts. Each segment is processed individually.

3.3.2 Tile Bitrate Calculation:

Each segment's tiles are cropped and saved. Their bitrates are calculated and stored in a dictionary *(bitrates)*.

3.3.3 Bitrate Differences:

The script computes the difference in bitrate *(diff_bitrates)* for each tile across segments, enabling analysis of bitrate variation over time.

3.4 Omission of using normalization on difference of bitrates

In this analysis, normalization was not applied as it would disrupt the ratio-based comparisons and threshold-driven logic central to identifying similar patterns in the bitrate differences. The approach depends on the original scale of the bitrate differences to calculate ratios between consecutive segments (i.e., `last_diff / value`) and to apply a threshold of 0.15 for identifying similarity. Normalizing the data would alter its distribution, misaligning the results with the defined threshold and complicating the interpretability of the findings. Moreover, normalization could distort the correlation measures used to refine similarity detection and interfere with the masking logic designed for the raw data. Therefore, maintaining the original scale ensures the integrity, consistency, and accuracy of the analysis.

3.5 Corelation between different tiles

This script analyzes temporal variations in tile-wise bitrate differences across segmented 360-degree video content to identify tiles with similar patterns. It begins by processing a list of bitrate differences (`diff_bitrates`) between consecutive segments, stored in a Pandas DataFrame (`df_diff_bitrates`) for efficient analysis. For each tile in the current segment, the script compares its bitrate difference (`curr_diff`) with the previous segment's differences (`last_diff`) using a ratio (`last_diff / value`). Tiles with ratios close to 1, indicating similarity, are identified using a threshold of 0.15. Certain tiles are excluded from the analysis using a masking mechanism to filter irrelevant or noisy data.

When multiple tiles meet the similarity criterion, the script further evaluates their correlation over a sliding window of up to five segments. The tile with the highest correlation coefficient is selected as the most similar. The results include the indices of the most similar tiles, their respective similarity scores, and the segments in which they were identified. This approach highlights temporal and spatial patterns in bitrate variations, offering insights into consistent behaviors across tiles.

The analysis is useful for optimizing adaptive streaming strategies by identifying patterns in bitrate allocation and for benchmarking performance in tiled encoding systems. By tracking temporal consistency, it can also help detect anomalies or inefficiencies in encoding. However, the results are sensitive to the chosen threshold, masking indices, and the size of the correlation window, which may require tuning based on the video's characteristics. Future improvements could include dynamic thresholding, prioritizing tiles based on perceptual importance, and developing visualization tools for better interpretability. This framework provides a valuable tool for improving 360-degree video encoding and streaming efficiency.

3.6 Use Case in Video Tile Analysis

This methodology is particularly useful for panoramic videos divided into multiple tiles for individual analysis. By segmenting the video and calculating the bitrate for each tile or segment, it is possible to optimize content delivery based on real-time conditions, such as network bandwidth or viewer orientation.

This approach enables a flexible and adaptive streaming experience, ensuring each segment is delivered at the optimal data rate while maintaining overall video quality.

3.7 Yolo detection baseline

This code uses the YOLOv8 model from the ultralytics library to perform object detection on a video. It loads a pre-trained YOLO model, reads a video file, and processes each frame to detect objects. The code draws bounding boxes around detected objects and labels them with the class name and confidence score, provided the detection confidence is greater than 40%. It also assigns different colors to the bounding boxes based on the object class. The processed frames are saved to a new video file. The script uses OpenCV for video capture, frame processing, and saving the output.

4 Conclusion

In this report, we explored the use of bitrate variations in 360-degree video tiles as a means of optimizing object tracking and reducing detection load when an object shifts between tiles. By segmenting panoramic video into cubemap tiles and calculating the bitrate for each segment, we developed a method to track temporal variations in bitrate and identify tiles with similar characteristics. This approach is particularly valuable for improving adaptive streaming and video encoding efficiency in immersive technologies like virtual reality and augmented reality.

Our findings suggest that tracking bitrate patterns between consecutive video segments can enhance the performance of video streaming systems by predicting and managing tile data more effectively. By focusing on the bitrate spikes and their correlation across tiles, we can significantly reduce computational load, thereby enabling smoother playback, especially in scenarios where objects move non-linearly between video tiles.

Overall, this research provides valuable insights into improving the efficiency and quality of 360-degree video streaming and object tracking in immersive environments.

References

[1] Shubham Chaudhary et al. "{TileClipper}: Lightweight Selection of Regions of Interest from Videos for Traffic Surveillance". en. In: 2024, pp. 967–984. ISBN: 978-1-939133-41-0. URL: https://www.usenix.org/conference/atc24/presentation/chaudhary (visited on 12/12/2024).

[2] Frank Ekpar, Hiroyuki Hase, and Masaaki Yoneda. "A unified framework for correcting panoramic image distortions". In: *Proceedings of the 5th WSEAS international conference on Applied computer science*. ACOS'06. Stevens Point, Wisconsin, USA: World Scientific, Engineering Academy, and Society (WSEAS), Apr. 2006, pp. 873–878. ISBN: 978-960-8457-43-0. (Visited on 12/12/2024).

[3] Hao Feng et al. "SimFIR: A Simple Framework for Fisheye Image Rectification with Self-supervised Representation Learning". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. ISSN: 2380-7504. Oct. 2023, pp. 12384–12393. DOI: 10.1109/ICCV51070.2023.01141. URL: https://ieeexplore.ieee.org/document/10378278 (visited on 12/12/2024).

[4] *FFmpeg Filters Documentation*. URL: https://ffmpeg.org/ffmpeg-filters.html#v360 (visited on 12/12/2024).

[5] Julia Guerrero-Viu et al. "What's in my Room? Object Recognition on Indoor Panoramic Images". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2020, pp. 567–573. DOI: 10.1109/ICRA40945.2020.9197335. URL: https://ieeexplore.ieee.org/document/9197335 (visited on 12/12/2024).

[6] Chao Zhou, Zhenhua Li, and Yao Liu. "A Measurement Study of Oculus 360 Degree Video Streaming". In: *Proceedings of the 8th ACM on Multimedia Systems Conference*. MMSys'17. New York, NY, USA: Association for Computing Machinery, June 2017, pp. 27–37. ISBN: 978-1-4503-5002-0. DOI: 10.1145/3083187.3083190. URL: https://dl.acm.org/doi/10.1145/3083187.3083190 (visited on 12/12/2024).