Optimal Video Compression using Pixel Shift Tracking

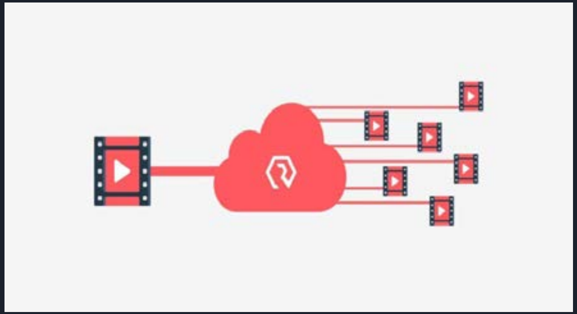
by Hitesh Saai Mananchery Panneerselvam

About Me | [LinkedIn Profile]

- Senior Machine Learning Engineer at Expedia Group
- 7+ Years of Experience in ML/AI
- Experience Working in InsurTech, FinTech & Travel Industries

Introduction

- Today Video comprises approximately 85% of all the internet traffic.
- On daily basis from social media alone we get 10 Pb’s of data per day.
- There are around 20 video compression formats available.
- Most of compression methods are using rigid, rule-based algorithm.
- In Recent time, there is been a surge in video compression algorithms using ML-based models in the last few years and many of them have outperformed several legacy codecs.
- The advantage of ML based approach, that it is adaptable to diverse video formats and ML Framework.



Current Traditional Compression Algorithms

01 H.264(AVC)

02 H.265(HEVC)

03 AV1

04 VP8

Current AI Research Algorithm Methods

01 Residual Encoder

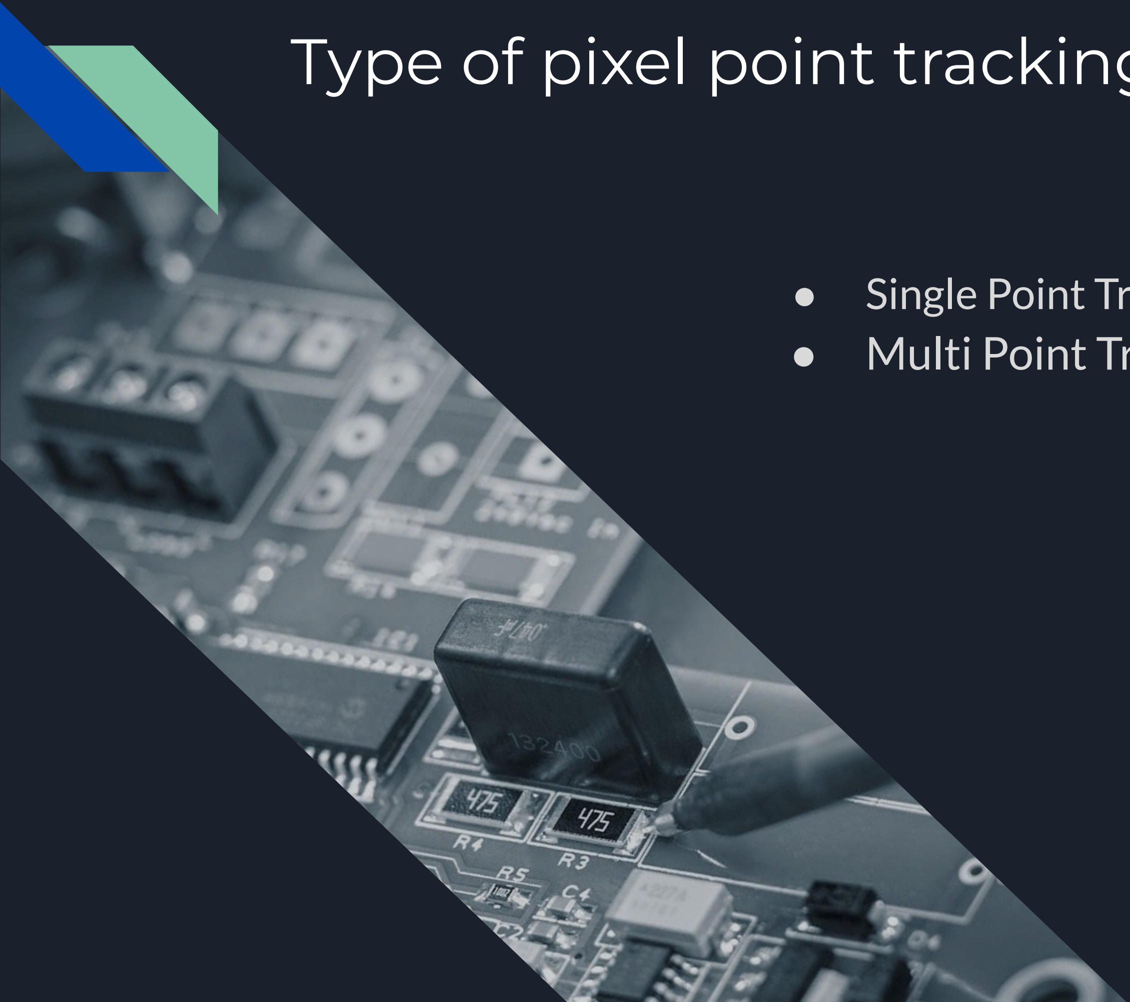
02 Variational Auto Encoder (VAE)

03 Deep Contextual Network (DconvN)

Our Proposed Method

- Reduces redundant pixel data transferred between video frames, unlike a traditional full video compression.
- Identifies and labels similar pixels in consecutive frames as redundant, and storing only shift positions.
- We use pixel tracking trajectory method to find and avoid storing redundant pixel at storage level

Type of pixel point tracking



- Single Point Trajectory
- Multi Point Trajectory

Single Point Trajectory

- **Reference Point Tracking:** Selects a reference point ($\lambda = (x1, y1)$) in the initial frame to track long-term pixel movement, enabling precise analysis of camera motion shifts in video sequences.
- **Persistent Independent Particles (PIPs):** Utilizes PIPs and PIPs++ to process X-fps RGB video, taking a target point coordinate as input and outputting a 1x2 (x,y) coordinate shift per frame to optimize pixel data.
- **Frame-by-Frame Motion Analysis:** Tracks point trajectory across 8-frame sequences, producing coordinate shifts for each frame to quantify pixel movement relative to the previous frame, enhancing video compression efficiency.

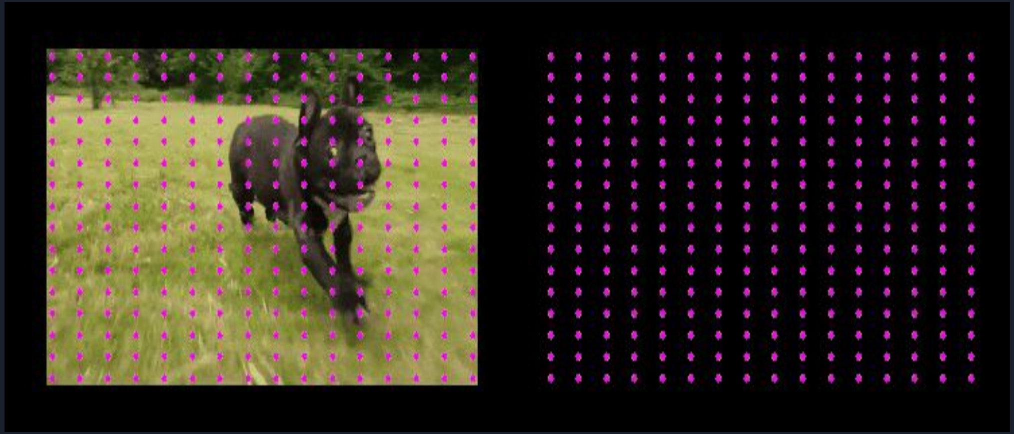
Single Point Trajectory

Source from pips2

Multi Point Trajectory

- Single-Point Tracking Limitations: Effective for static objects with a moving camera, but inadequate for tracking complex or multi-directional motion of objects when the camera is stationary, potentially missing critical movement data.
- Multi-Point Tracking Advantage: Tracks multiple Points of Interest (POIs) simultaneously to capture comprehensive motion dynamics, ensuring accurate detection of movement across the frame, even with a stationary camera.
- Optimized Compression via Grid-Based Tracking: Divides frames into 2D grid blocks, tracking central pixels to calculate shifts applied to all pixels in moving grids, enhancing pixel compression efficiency by focusing on significant motion areas.

Multi Point Trajectory



Source from pips2

Implementation: Compression

- **Step 1:** Select Tracking Point - Choose an arbitrary coordinate point in the initial frame to track camera motion shifts in videos with dynamic camera movement and static elements.
- **Step 2:** Track Motion with PIPs - Use Persistent Independent Particles (PIPs) to process 8-frame RGB video sequences, calculating pixel shifts ($\lambda k = [xn, yn]$) for each frame.
- **Step 3:** Identify Redundant Pixels - Analyze displacement data to distinguish redundant from non-redundant pixels within each frame, leveraging motion shift coordinates.
- **Step 4:** Compress Frames - Apply compression function $\gamma(k)$ to store only non-redundant pixels, nullifying redundant pixels as black (0 value), achieving 80-95% storage reduction.
- **Step 5:** Store Compressed Data - Sequentially compress frames, saving the compressed video and tracking data on disk, ensuring 4-7% data loss and efficient storage.

Implementation: Compression

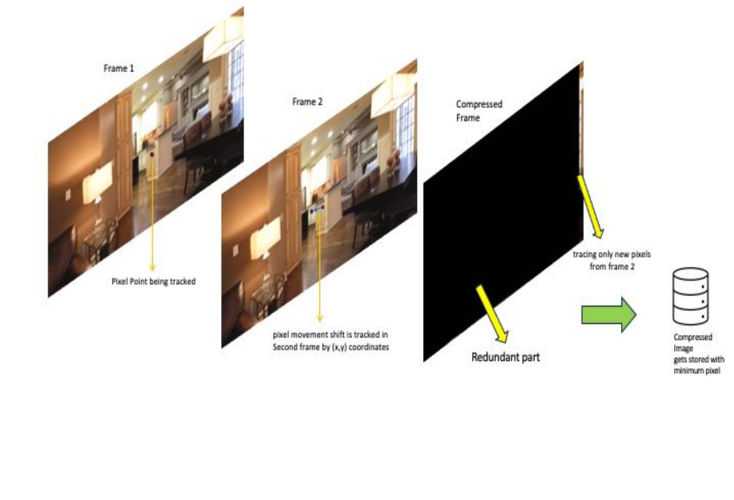
Compression Per Frame,

$\lambda k = [xn, yn]$

$\gamma(k) = (W - (W - xn)) + (H - (H - yn))$

Compression of total video,

$\zeta compress\text{-}Xn\ f{=}1 = \gamma(2) + \gamma(3) + \gamma(4) + .. + \gamma(n)$



Implementation: Decompression

- **Preserve Initial Frame:** Keep the first frame uncompressed as a reference, free from artifacts, to guide decompression of subsequent frames containing only non-redundant data.
- **Retrieve Movement Data:** Use stored point track movement records ($\lambda = [xn, yn]$) to identify pixel shifts for each compressed frame, providing motion dynamics for reconstruction.
- **Reverse Pixel Shifts:** Apply inverse shift values ($\bar{\lambda}k$) to determine original pixel positions, using the previous frame's boundaries (height, width) to extract pixel values.
- **Reconstruct Frames:** Fill missing pixels in the current frame by referencing the previous frame, restoring detail iteratively for each frame using decompression function $\Gamma(k)$.
- **Rebuild Video Sequence:** Sum all decompressed frames (Ψdecompress) to reconstruct the original video, minimizing compression artifacts.

Implementation: Decompression

Decompression Per Frame,

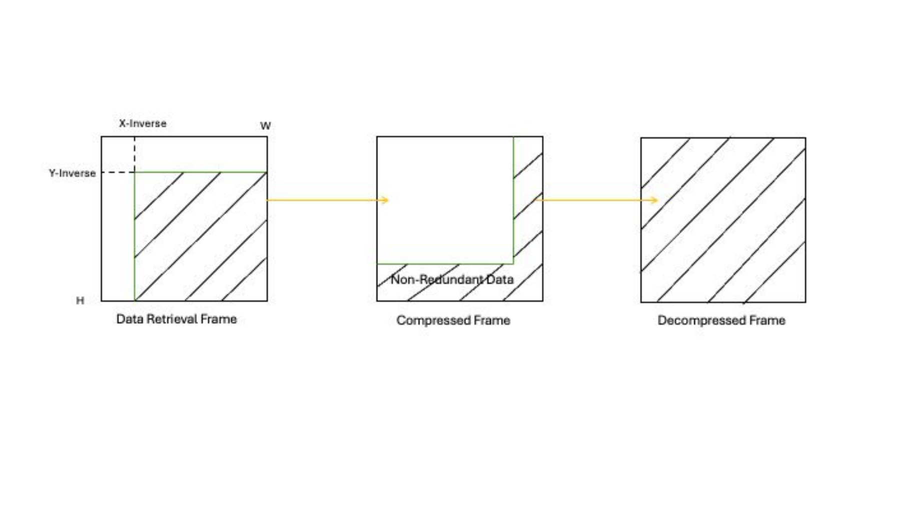
λˆ k = [xn, yn] (4) P F = IF[CFf = CF2, OFf-1, DFf-1]

ψf-1 = P F{xn, W} + P F{yn, H}

Γ(k) = ψf-1 + ξf

Decompression of total video,

ΨdecompressXn f=1 = Γ(1) + Γ(2) + ... + Γ(n) (8)-



Experimentation & Result

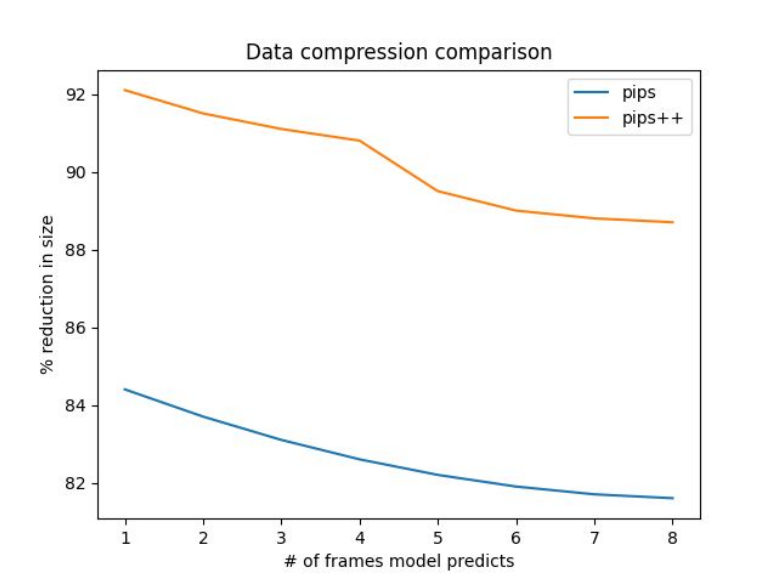
- Single-Point Tracking: Tested single-point tracking (e.g., pixel (100, 450)) using PIPs and PIPs++ on video, predicting coordinates frame-by-frame for high accuracy.
- Prediction Approach: Employed single-frame prediction, calling the model per frame for maximum accuracy, though it's the slowest method.
- Alternative Method: Noted option to predict trajectories for 'n' frames, skipping iterations for faster processing but with slightly reduced accuracy.
- Compression Performance: Achieved compression at 15 time per frame (tpfa), resulting in a compact file size of 36.82 KB, as shown in Table I.
- Decompression Performance: Recorded decompression at 50 tpfa, producing a file size of 238.57 KB, demonstrating efficient storage optimization (Table I).

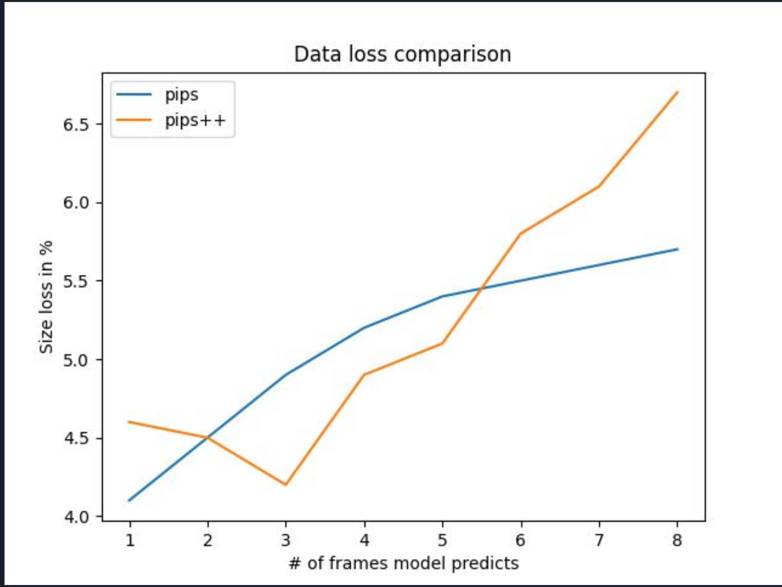
Method	**Speed**	
	tpf [a]	***size in KB***
Compression	15	36.82
Decompression	50	238.57

[a] time per frame in ms

Experimentation & Result

Performance Graph





Future Approaches Directions

- **Multi Point Trajectory**
- **Object Detection and Masking**
- **Similarity Search Matrix**

References

- Optimal Video Compression Using Pixel Shift Tracking | Github Repo
- Video Streaming Enhancements using Deep NN
- PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking
- End-to-end Optimized Image Compression
- Overview of Research in the field of Video Compression using Deep Neural Networks

Thank you!