March 25, 2019

Entropy-Adaptive Filtering

Jana Ehmann

Chia-Kai Liang

Hangyu Kuang

Ching Yin Derek Pang

Entropy-Adaptive Filtering

Abstract:

This publication describes an entropy-adaptive filtering, which reduces compression artifacts for videos of any given complexity and at any given video-encoding bit-rate. Unlike other video filtering designs, the entropy-adaptive filtering minimizes the likelihood of compression artifacts by reducing the entropy level of the input video.

Keywords:

Entropy-adaptive filtering, Gaussian pyramid, Laplacian pyramid, temporal filtering, compression artifact, artifacts, blockiness, distortion, spatial filter, video, motion image, image, moving image, moving picture, mosquito effect, ringing, color banding, the Gibbs phenomenon.

Background:

Many mobile videographies generate entropy videos with visible compression artifacts. These compression artifacts are a common occurrence in social and media software applications. A lack of field depth, a user's unsteady handling of a user equipment (UE), a user's fast camera motion, high camera noise, the visual complexities of natural scenes, and other factors, often result in videos with unnatural sharp details, high degrees of motion, and complex textures. These characteristics help increase video entropy and put a strain on the compression efficiency of modern video codecs, which cause many visible artifacts, such as blockiness, pixelation, mosquito noise, ringing, color banding, and the like.

One common unwanted video artifact is blocking, which often occurs at low bit-rates when high-frequency components are heavily quantized without considering the correlation to the adjacent blocks and results in discontinuities at block borders. In contrast, at high bit-rates, ringing, also known as the Gibbs phenomenon, is the more-prevalent undesired artifact. Whereas the mosquito noise, which is local flickering near the edges of video images, is associated with movement. These are just a few of the many undesired artifacts that software camera developers have tried to lower over the years.

To decrease undesired artifacts, decreasing the entropy level of the input video is beneficial in image processing.

Description:

A pre-processing entropy-adaptive filtering helps suppress undesired compression video artifacts during encoding, while preserving image details. The entropy-adaptive filtering meets one or more of the following criteria:

1. The ability to reduce image noise and image entropy.
2. Low latency and low power to support the capture of high-resolution images.
3. Adaptivity to scene complexity to preserve image details and minimize artifacts.
4. Adaptivity to different software application bit-rates.
5. Adaptivity to support different frame rates and spatial resolutions.
6. Encoder-adaptivity to support different video encoder implementations with different sets of compression tools and features.
7. Manual tuning of system parameters for custom quality control tuning.

The entropy-adaptive filtering reduces compression artifacts for videos of any given complexity and at any given video encoding bit-rate. Unlike other video filtering designs, the entropy-adaptive filtering minimizes the likelihood of compression artifacts by reducing the entropy level of the input video.

Leveraging non-linear spatial-temporal filtering[1], the entropy-adaptive filtering offers both noise reduction and entropy reduction properties, while preserving important image details. As a result, the entropy-adaptive filtering may also function as a video denoising filtering.

Figure 1 illustrates a system design overview of the entropy-adaptive filtering.

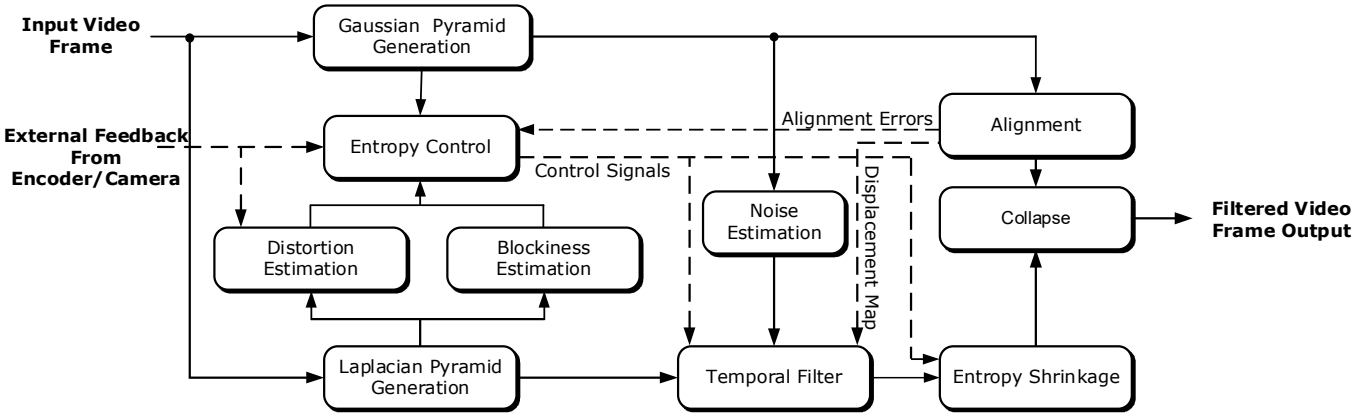

Figure 1

As shown in Figure 1, the system design of the entropy-adaptive filtering may include the following main components:

1. Gaussian pyramid generation, which generates multi-scale images.
2. Laplacian pyramid generation, which also generates multi-scale images.
3. Alignment (or motion compensation) to perform motion-compensated temporal filtering.
4. Distortion estimation to predict the input video frame distortion level.
5. Blockiness estimation to predict the likelihood of blockiness of the input video frame.
6. Entropy control to relay proper feedback information to enable entropy-adaptive filtering.
7. Temporal filtering for each Laplacian pyramid scale level.
8. Entropy shrinkage to apply spatial filtering on Laplacian images.
9. Pyramid collapse to merge each filtered Laplacian level coefficient.
10. Parameter training and tuning control (not explicitly shown).

As shown in Figure 1, for each incoming (or input) video frame, a Gaussian pyramid and a Laplacian pyramid is applied to decompose the frame in multi-scale levels. Then, the Gaussian pyramid is used for temporal alignment with previously buffered video frames. In addition, the

Gaussian pyramid may also be used for temporal noise estimation for each pyramid level, which is usually performed offline. To analyze the image for proper entropy filtering, the system design performs a distortion estimation and a blockiness estimation on the Laplacian pyramid. Moreover, entropy controls combine this analysis with an external feedback from video encoders and cameras with parameters, such as average frame quantization parameters (QP), bit-rate, lux index, and the like to create proper entropy-filtering parameters. Based on the computed filtering parameters, temporal noise estimation, and the alignment results, the system design applies temporal filtering on the Laplacian pyramid. To further remove non-perceptual details and lower compression artifacts, the system design applies added spatial filtering on the Laplacian pyramid using entropy shrinkage. Finally, as shown in Figure 1, the system design collapses back together the Laplacian and the Gaussian pyramids to create the filtered video frame output.

In a more generalized implementation, one may view the entropy-adaptive filtering system as a filter that adjusts according to the input video frame and the encoder feedback, as illustrated in Figure 2.

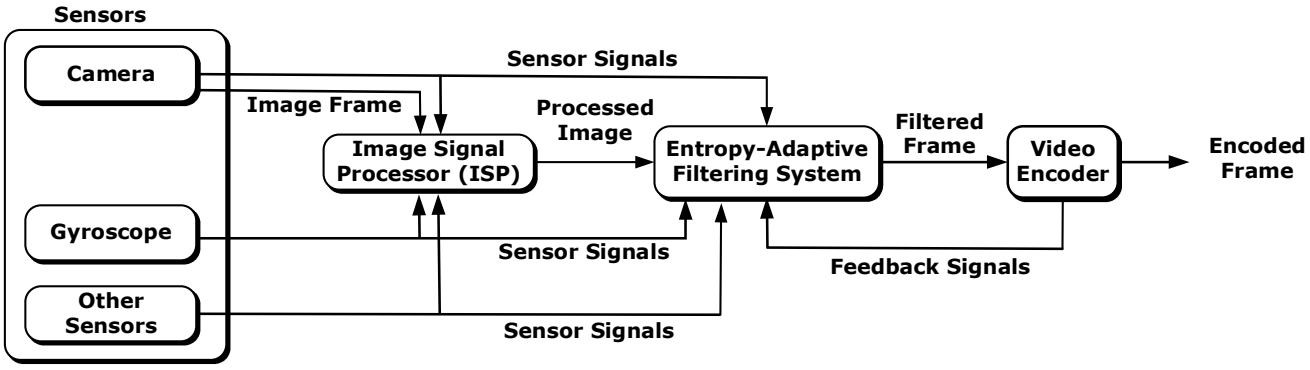

Figure 2

Below is a high-level overview of the main components of the system design of the entropy-adaptive filtering, as shown in Figure 1.

Gaussian and Laplacian Pyramid Generation

Like in a temporal filter[1], at time *t*, the system design decomposes each frame into a Gaussian pyramid and a Laplacian pyramid. The system design's alignment uses the Gaussian pyramid to compute a displacement map used for motion-compensated temporal filtering, while the temporal filtering and the entropy filtering process the Laplacian pyramid. Instead of the Gaussian and the Laplacian pyramid generation, in an alternative implementation, the system design may use other multi-scale analyses, such as a steerable pyramid, a wavelet, a curvelet, or a ridgelet transformation.

Alignment (or Motion Compensation)

The alignment stage aligns each Laplacian level from the earlier video frame to the current video frame to perform temporal filtering. Using the Gaussian pyramid in a hierarchical manner, the system design computes a displacement map. One implementation of motion estimation is to use a fast-optical flow method or an optimization of such method[1, 2].

Distortion Estimation

The goal of entropy filtering is to reduce distortion below a specified threshold to decrease the likelihood of compression artifacts. The distortion for a given Laplacian image at a level *l* is approximated by using Eq. 1.

$$d_l = g_l(R_l)\sigma_l^2 2^{-2R_l} \qquad \text{(Eq. 1)}$$

where σ_l^2 is the average energy per sample of the Laplacian image or patch, R is the total bit-rate allocated to the frame, R_l is the estimated number of bits per sample that are applied at each level, and $g(R)$ is a weak function of R, which approaches one (1) as R goes to zero (0), and fast-

approaches a constant ϵ_l^2 as the bit-rate R increases. The values of $g(R)$ and ϵ_l^2 depend on the encoder's coding method. The total estimated distortion of the video frame D_p is calculated using Eq. 2.

$$D_p = \sum_{l=0}^{N-1} d_l \qquad \text{(Eq. 2)}$$

where d_l represents a distortion at a Laplacian pyramid level l.

Eq. 2 shows that decreasing the average energy at each Laplacian level achieves a reduction in the distortion D_p.

One drawback of Eq. 1 is that it does not account for motion-compensated coding. To this end, the system design may use Eq. 3 to estimate the distortion of a motion-compensated coded video signal using the residual errors D_r obtained during the alignment stage.

$$D_r = g_r(R)\sigma_r^2 2^{-2R} \qquad \text{(Eq. 3)}$$

where σ_r^2 is the variance of residual errors from motion-compensated video coding. Using Eq. 2 and Eq. 3, the system design may estimate the distortion of the input video frame by using Eq. 4.

$$\widehat{D} = min\left(D_p, D_r\right) \qquad \text{(Eq. 4)}$$

When the target video encoder supplies the quantization parameter (QP) information, the system design may use it to estimate the distortion of an 8-bit video frame using Eq. 5.

$$D_{QP} = \frac{255^2}{10^{\frac{a_{QP} \cdot QP + b_{QP}}{10}}} \qquad \text{(Eq. 5)}$$

where a_{QP} and b_{QP} represent model parameters that help estimate the distortion using QP.

Nevertheless, QP information feedback often suffers from one or more frame delays depending on the system design constraints. In the cases when the exact QP information value for the current frame is not available, the system design estimates the current QP information using past QP information feedback, as shown in Eq. 6.

$$\widehat{QP}(t) = \sum_{k=d}^{K} \alpha_{QP}(k) QP(t-k) \qquad \text{(Eq. 6)}$$

where *K* represents the windowed sample size, *t* represents the current time, and *d* represents the expected feedback delay.

The system design may also estimate the QP information using an autoregressive-moving-average (ARMA) model.

Blockiness Estimation

For low bit-rate videos, blockiness is often a more-dominant artifact compared to blurring artifacts. Therefore, the system design places many constraints to minimize blockiness. To determine the likelihood of blockiness, the system design uses feedback signals. Particularly, the system design uses the spatial nonuniformity in the background luminance (or texture masking), and the average background luminance surrounding the artifact (or luminance masking).

Blockiness most-often occurs on a flat region or a slow-varying gradient region of a video. To measure the flatness of the texture, the system design may measure the high-frequency response of an input image, which is already available in the lower Laplacian levels. To determine whether a patch or a frame has a visible flat region $A_{flatness}$, the system design calculates the ratio between the number of Laplacian samples whose response value falls below a set threshold number of samples and the total number of samples. Alternatively, the system design may divide the Laplacian level into patches. For each patch, it calculates the median magnitude (absolute value) of the responses. In this alternative approach, $A_{flatness}$ is defined as the ratio between the number of the patches whose median value falls below a set threshold value and the total number of patches. In another implementation, instead of using Laplacian images, the system design may apply an edge filter or a high-pass filter to calculate $A_{flatness}$.

Regardless of how the system design calculates $A_{flatness}$, it calculates the probability of blockiness due to texture masking $P_{texture}(blockiness)$ by using Eq. 7.

$$P_{texture}(blockiness) = C_{texture}e^{-b_{texture}A_{flatness}} \quad \text{(Eq. 7)}$$

where $b_{texture}$ and $C_{texture}$ are empirical parameters that relate $A_{flatness}$ to $P_{texture}(blockiness)$.

For luminance masking, blockiness is often more visible in a dark region than a bright region. The system design calculates the average luminance value at any of the Gaussian pyramid levels or directly from the highest Gaussian pyramid level. Using the property of luminance masking, the system design calculates the probability of blockiness $P_{luminance}(blockiness)$ using a polynomial or exponential model, such as the one in Eq. 8.

$$P_{luminance}(blockiness) = \begin{cases} \left(\frac{\bar{I}_{mean}}{\beta_l}\right)^{\alpha_{lum}}, & if\ \ 0 \leq \bar{I}_{mean} \leq \beta_{lum} \\ (\gamma_{lum})\left(\bar{I}_{mean} - \beta_{lum}\right)^{\upsilon_{lum}}, & otherwise \end{cases} \quad \text{(Eq. 8)}$$

where $\bar{I}_{mean}$is the average luminance for a given patch or image frame, $\beta_{lum}$is the average luminance value with peak response, and γ_{lum}, β_l, α_{lum}, and υ_{lum} are empirical parameters that control the degree of luminance masking.

Alternatively, the system design may consider only dark scenes to lower the likelihood by evaluating a camera's lux index. In this alternative, the system design combines both the texture and the luminance masking effects and supplies the likelihood of blockiness $P(blockiness)$, as detailed in Eq. 9.

$$P(blockiness) = P_{texture}(blockiness) \cdot P_{luminance}\ (blockiness) \quad \text{(Eq. 9)}$$

Also, the system design may consider either the texture effect or the luminance masking effect, separately.

Entropy Feedback Signal and Control

The system design includes an entropy-adaptive filter, which relies on feedback signals to determine the degree of filtering applied to the filtered video frame output. The ability to react according to the feedback enables a rate-adaptivity and a content-adaptivity for the system design. There are two types of feedback signals — intrinsic and extrinsic. Intrinsic feedback signals are the alignment errors, a distortion estimation, and the Laplacian and the Gaussian images. In contrast, extrinsic feedback signals originate from the outside of this system design from sources, such as video encoders, camera capture systems, gyroscope sensors, and other parts of the camera system.

Temporal Filtering

Temporal filtering[1] is one way to reduce the entropy level of a video for better compression efficiency. Not only can it remove the inherent noises generated by digital sensors during capture, but it also allows the removal of any temporal deviation between successive frames that may not be visible by a viewer. One way to implement a temporal filter is the use of an infinite impulse response (IIR) filter, which is robust to alignment error artifacts, such as ghosting, unwanted blending, or excessive blurring[1].

The IIR filter can boost high-frequency components to restore high-frequency features, while keeping the denoising strength. To perform IIR filtering, the system design may divide each video frame into patches. In addition, among other strategies, the system design may separately calculate the interpolation weight between the current frame and the earlier filtered frames for each patch in each Laplacian pyramid level.

One strategy is to define the temporal smoothing, as outlined in Eq. 10.

$$L_r^l(p) = \omega_c^l L_c^l(p) + \omega_f^l (L_a^l(p) + I \cdot L_\Delta^l(p)) \qquad \text{(Eq. 10)}$$

where $L_r^l(p)$ is the temporal filtered output of the Laplacian output at level l and at pixel location p, $L_c^l(p)$ is the Laplacian output of the current video frame, $L_a^l(p)$ is the previous filtered output after alignment application, $L_\Delta^l(p)$ is the difference between $L_c^l(p)$ and $L_a^l(p)$, ω_c^l is the filtering weight of the current frame, ω_f^l is the filtering weight of the previous aligned frame, and I is the interpolation factor between the current frame and the previous aligned frame.

The interpolation factor I lies in the range of [0, 1], where a higher value of I indicates a lower contribution from the previously aligned frame — differently said, a lower smoothing effect. To facilitate entropy-adaptive filtering, I is selected by choosing interpolation weights, such as I_a based on the alignment error, I_Δ based on the expected camera noise and the observed pixel differences, and I_e based on the current energy or the distortion level of the Laplacian pyramid encoded bit-rate. Eq. 11 shows the relation between the interpolation factor I and the various interpolation weights I_a, I_Δ, and I_e.

$$I = \max(I_a, \min(I_\Delta, I_e)) \qquad \text{(Eq. 11)}$$

If the alignment confidence is low, the system design applies a high interpolation factor or a minimal interpolation weight I_a to avoid an interpolation artifact propagation. Otherwise, the system design adjusts the interpolation factor according to the camera noise or the compression distortion. The interpolation weight I_Δ depends on the observed pixel differences and the expected camera noise level. The sensor noise calibration may estimate the expected camera noise level and accounts for the noise impact from each image processing stage (e.g., color space conversion, tone mapping, gamma correction). In addition, the system design may express the interpolation weight I_Δ using a Wiener Filter.

For faster computation and reduced sensitivity to over-smoothing, however, the system design may express the interpolation weight I_Δ using a sigmoid filter, as shown in Eq. 12[1].

$$I_e(p) = \frac{1}{1+e^{-(\left|L_p^l(p)\right|-m(p))}} \qquad \text{and} \qquad m(p) = 1 + \alpha_t\left(1 - e^{-D_P \cdot \beta_t}\right) \qquad \text{(Eq. 12)}$$

where $m(p)$, in this case, depends on D_P instead of the camera noise, and α_t and β_t determine the sensitivity of the interpolation factor relative to the distortion level and the level of pixel differences.

Entropy Shrinkage

The entropy-adaptive filtering uses a technique called shrinkage or soft-coring. The soft-coring technique suppresses the small-value coefficients of each Laplacian level to near zero (0), while keeping the large-value coefficients by using a scale factor, as detailed in Eq. 13.

$$L_s^l(p) = m_l^s \cdot L_r^l(p) \cdot \left(1 - e^{\left[\frac{-|L_r^l(p)|}{\tau_l^s}\right]^{\gamma_l^s}}\right) \qquad \text{(Eq. 13)}$$

where $L_r^l(p)$ is the temporal filtered response value at a Laplacian level l, m_l^s controls the slope at the high-value coefficient region at level l, τ_l^s controls the range of small-value coefficient that will be suppressed to near zero (0), and γ_l^s controls the tightness of small-value coefficients relative to zero (0) so that the relationship $|L_s^l(p)| \leq \tau_s^l$ is satisfied.

Assuming d_{max}^l is the target maximum distortion that the system design aims to keep at a Laplacian level l to prevent compression artifacts, then, the average reduction in the Laplacian energy is calculated using Eq. 14 and Eq. 15.

$$\Delta\sigma^2 = max\left(\sigma^2 - \frac{d_{max}^l}{g_l(R)2^{-2R}}, 0\right), \qquad \text{for } D_p \leq D_r \qquad \text{(Eq. 14)}$$

$$\Delta\sigma^2 = max\left(\frac{\zeta_l \cdot 4^l}{\sum_{k=0}^{N-1} 4^k}(\sigma_r^2 - \frac{D_{max}}{g_r(R)2^{-2R}}), 0\right), \quad \text{for otherwise} \qquad \text{(Eq. 15)}$$

where ζ_l adjusts for overestimation or underestimation of the perceptual quality relative to distortion.

When the target video recorder supplies the QP information, the system design uses Eq. 4 to estimate the distortion of the input video frame. Subsequently, Eq. 14 and Eq. 15 give way to Eq. 16.

$$\Delta\sigma_l^2 = max\left(\frac{\zeta_l \cdot 4^l}{g_l(R)2^{-2R}\sum_{k=0}^{N-1} 4^k}(D_{qp} - D_{max}), 0\right), \quad \text{for } D_p > D_{qp} \qquad \text{(Eq. 16)}$$

In one implementation, instead of using the separated distortion estimations D_p, D_r, or D_{qp}, the system design computes the energy reduction $\Delta\sigma_l^2$ as a weighted prediction between Eq. 14, Eq. 15, and/or Eq.16. Furthermore, the system design may weigh differently the respective distortions at different regions of the image according to the region of interest.

Using the estimated energy reduction $\Delta\sigma_l^2$, the system design calculates the corresponding soft-coring parameters m_l^s and τ_l^s. One strategy for energy reduction is to first suppress small-value coefficient to zero (0) up to a perceptual threshold τ_l^{max}, which is a manual tunable threshold for determining the maximum τ_l^s. Then, if excessive energy remains, the system design adjusts m_l^s to scale the variation magnitude in each Laplacian pyramid level, up to a perceptual reduction threshold τ_l^{min}. Eq. 17 and Eq. 18 show how the system design calculates τ_l^s.

$$\arg min_{\tau_l^s \le \tau_l^{max}} \tau_l^s. \qquad \text{(Eq. 17)}$$

$$such\ that \quad \sum_{p \epsilon P} \prod\left(L_r^l(p) \le \tau_l^s\right) L_r^{l2}(p) \ge \Delta\sigma_l^2 \qquad \text{(Eq. 18)}$$

where $\prod(\cdot)$ is an indicator function, and p is the set of all possible pixel locations at a Laplacian pyramid level l.

If the system design determines that the above constraint is achievable, it sets τ_l^s=τ_l^{s*} and $m_l^s = 1$. In contrast, if the system design determines that the above constraint is unachievable, it sets τ_l^s=τ_l^{max} and m_l^s is, then, calculated by the remaining energy, as shown in Eq. 19 and Eq. 20.

$$\Delta\sigma_l^{2\prime} = \Delta\sigma_l^2 - \sum_{p\epsilon P} \prod\left(L_r^l(p) \leq \tau_l^s\right) L_r^{l2}(p) \qquad \text{(Eq. 19)}$$

$$m_l^s = \max\left(\frac{\Delta\sigma_l^{2\prime}}{\sum_{p\epsilon P} \prod(L_r^l(p) \leq \tau_l^s) L_r^{l2}(p)}, m_l^{min}\right) \qquad \text{(Eq. 20)}$$

In one implementation, m_l^{min} may consider the texture and the luminance masking effect to avoid unnecessary smoothing since blockiness and other compression artifacts may not be visible in a highly-texturized scene. In such case, the system design uses Eq. 21.

$$m_l^{min} = \max(m_l^{smooth}, m_l^{blk}) \qquad \text{(Eq. 21)}$$

where m_l^{smooth} defines the lowest reduction ratio that can be applied before over-smoothing becomes visible, and m_l^{blk} defines the lowest reduction threshold that is sufficient for reducing blockiness artifacts when accounting for texture and luminance masking effects.

The system design determines m_l^{blk} using Eq. 22.

$$m_l^{blk} = \max(2 - e^{s_{blk} \cdot p(blockiness)}, 0) \qquad \text{(Eq. 22)}$$

When *p(blockiness)* is zero (0) then, m_l^{blk} will be one (1) — implying no high-value coefficient reduction. When $p(blockiness) \geq \frac{\ln(2)}{s_{blk}}$ then, m_l^{blk} will be zero (0) — implying maximum high-value coefficient reduction.

In another implementation, when motion compensation predicts the video without artifacts, the system design adds m_l^{align} to prevent excessive smoothing, as shown in Eq. 23.

$$m_l^{min} = \max(m_l^{smooth}, m_l^{blk}, m_l^{align}) \qquad \text{(Eq. 23)}$$

The system design may apply the factor m_l^{align} in lieu of applying distortion estimation for motion compensation. One may express m_l^{align} as a function of alignment errors using the Gaussian pyramid, as shown in Eq. 24.

$$m_l^{align} = \frac{1}{a_{align} + e^{b_{align} \cdot \delta_T^2 + c_{align}}} \qquad \text{(Eq. 24)}$$

The system design may also include other visual and/or artistic factors to impose constraints on m_l^{min} to avoid unnecessary energy reduction or to boost energy in certain sub-band levels in the Laplacian pyramid. These factors may include things like image sharpening, background / region-of-interest blurring, etc.

Pyramid Collapse

Once the system design applies proper temporal filtering and entropy shrinkage to the Laplacian pyramid levels, the system design collapses all levels in the filtered video output frame. It is worth noting that the system design, to maintain temporal denoising stability, only stores temporally-filtered frames before entropy shrinkage is applied to the temporal infinite input response (IIR) filtering

Parameter Training and Tuning Control

The system design may make available to the user many system and model parameters for manual tuning. In addition, some of the model parameters, such as parameters from Eq. 1 to Eq. 4, can also be trained using a video dataset. Data training enables higher autonomy and adaptivity in the system design. For better performance, the video dataset is collected and encoded using the same capture system settings and target video encoder during runtime.

In summary, unlike other video filtering designs, this entropy-adaptive filtering minimizes the likelihood of compression artifacts by reducing the entropy level of the input video.

References:

1. Ehmann, J., Chu, L., Tsai, S. and Liang, C. (2018). Real-Time Video Denoising on Mobile Phones. In: *25th IEEE International Conference on Image Processing (ICIP)*. pp.505-509.
2. Kroeger, T., Timofte, R., Dai, D. and Van Gool, L. (2016). Fast optical flow using dense inverse search. In: *14th European Conference on Computer Vision*. pp.471-488.