A Mid-Level Representation of Visual Structures for Video Compression

Georgios Georgiadis
Dolby Laboratories
4000 West Alameda Ave.
Burbank, CA 91505
georgios.georgiadis@dolby.com

Stefano Soatto
UCLA Vision Lab
University of California
Los Angeles, CA 90095
soatto@ucla.edu

Abstract

A video coding system is presented that partitions the scene into "visual structures" and a residual "background" layer. A low-level representation ("track-template") of visual structures is proposed that exploits their temporal redundancy. A dictionary of track-templates is constructed that is used to encode video frames. We make optimal use of the dictionary in terms of rate-distortion by choosing a subset of the dictionary's elements for encoding using a Markov Random Field (MRF) formulation that places the track-templates in "depth" layers. The selected "track-templates" form the mid-level representation of the "visual structure" regions of the video. Our video coding system offers improvements over H.265/H.264 and other methods in a rate-distortion comparison.

1. Introduction

With an ever-increasing video consumption rate on the Internet, we are faced with a continuously increasing pressure on available bandwidth[1]. While the new H.265 [8] has improved performance over existing standards, the majority of video compression techniques have traditionally been confined in modeling and predicting pixel values of video frames. We consider an alternative to traditional coding schemes, where we assume that a video has been generated by an underlying scene. Our aim is to model and compress the source (scene) rather than the output (pixel values).

Motivated by video compression, we partition the scene into two types of regions, "visual structures" and a background layer. "Visual structures" are regions of images that trigger isolated responses of a co-variant (feature) detector. These include blobs, corners, edges, junctions and other sparse features generally assumed to correspond to properties of the scene. Structure regions that can be put into correspondence across frames are called "trackable regions". Trackable regions can persist over a large number of frames. We leverage on their temporal redundancy to compress them, by storing their compact representations *once* in the first frame they appear and predicting them in all subsequent ones. This allows compressing any structures that persist in more than one frame. The background layer generally exhibits spatial regularity and can be compressed by standard coding techniques. The visual structures' representations along with the background layer are overlaid together on video frames, which are then further compressed by a standard video encoder.

It has been previously argued that an image can be partitioned into structures and textures ([7, 12, 2]) based on statistics computed in that *one* image. We test whether image structures arise from properties of the scene, by leveraging on the notion of *proper sampling* [13]. Proper sampling requires multiple images of the same scene to determine whether a structure is "real" in the sense of corresponding to something in the scene or an "alias", an artifact of nuisance factors in the image formation process. We model and compress those that satisfy this test and allow a standard video encoder to compress the rest. Finally, partitioning the scene into various types of regions for video coding has also been previously proposed [5, 11, 6, 15]. However these methods do not model "visual structures" to take advantage of their temporal redundancy.

In this work, we introduce the notions of "visual structures" and "trackable regions". We compute a dictionary of track-templates (a low-level representation of visual structures) and then choose a subset of its elements to encode a video sequence using a Markov Random Field (MRF) formulation that places the track-templates in layers (mid-level representation). This allows an optimal use of the dictionary by minimizing the reconstruction error of the predicted frames. We show how this system improves the H.265/H.264 performance in a rate-distortion sense.

[1] Cisco projects that by 2019 there will be 5 million years of video content traveling the Internet every month and that by 2019, video traffic will be 77% of all Internet traffic [4].

2. Visual Structures

Digital images $\{I_{xy}\}_{(x,y)\in\Delta=(1,1):(X,Y)} \in \mathbb{R}^{X\times Y}$ are obtained by averaging a function $I : D \subset \mathbb{R}^2 \to \mathbb{R};\ p \mapsto I(p)$ on a neighborhood $\mathcal{B}$ of the point $p_{xy} \in D$ of size $\sigma > 0$. In general, $I_{xy} = I(p_{xy}) + n_{xy}$ where $n_{xy} = n_{xy}(I)$ is the quantization error. We consider groups of transformations of the sensor plane, $g : D \subset \mathbb{R}^2 \to \mathbb{R}^2; p \mapsto g(p)$, and denote their induced action on the image by $I \circ g \doteq I(g(p))$. For example, the translation group is represented by a translation vector $T \in \mathbb{R}^2$, via $g(p) \doteq p + T$, so that $I \circ g(p) \doteq I(p + T)$. Each group element $g \in G$ determines a "frame". For instance, in the Euclidean plane, the translation group determines a reference frame with origin at the point $T \in \mathbb{R}^2$. The discussion below applies to other finite-dimensional Lie groups of the plane such as Euclidean, similarity, affine, and projective.

Canonization is a constructive process to eliminate the effects of a group G acting on the data (the set of images $\mathcal{I}$). The group organizes the data into orbits. A covariant detector identifies canonical elements of each orbit that covary with the group. Hence, in the corresponding (moving) frame, the data is independent of the group. Formally, a differentiable functional $\psi : \mathcal{I} \times G \to \mathbb{R}; (I, g) \mapsto \psi(I, g)$ is said to be *local*, with *effective support* σ if its value at g only depends on a neighborhood of the image of size $\sigma > 0$, up to a residual that is smaller than the mean quantization error. For instance, for a translational frame g, if we call $I_{|_{\omega_\sigma(g)}}$ an image that is identical to I in a neighborhood ω of size σ centered at position $g \equiv T$, and zero otherwise, then $\psi(I_{|_{\omega_\sigma(g)}}, g) = \psi(I, g) + \tilde{n}$, with $|\tilde{n}| \leq \frac{1}{XY}\sum_{x,y}|n_{xy}|$. For other groups, we consider the image in the reference frame determined by g, or the "transformed image" $I \circ g^{-1}$.

If we call $\nabla\psi \doteq \frac{\partial\psi}{\partial g}$ the gradient of the functional ψ with respect to (any) parametrization of the group, then under the "transversality" conditions $\det(\nabla\nabla\psi) \neq 0$, the equation $\nabla\psi = 0$ locally determines a unique function g (a *canonical representative*) of I, $g = \hat{g}(I)$, via the Implicit Function Theorem. If the canonical representative co-varies with the group, in the sense that $\hat{g}(I \circ g) = (\hat{g} \circ g)(I)$, then the functional ψ is called a *co-variant detector* (e.g. Laplacian-of-Gaussian (LoG) and the difference-of-Gaussians (DoG)). Varying σ produces a *scale-space*, whereby the locus of extrema of ψ describes a *graph* in $\mathbb{R}^3$, via $(p, \sigma) \mapsto \hat{p} = \hat{g}(I; \sigma)$. Starting from the smallest σ, one would have a large number of extrema; as σ increases, extrema will merge or disappear. Although in a two-dimensional scale space, extrema can also appear as well as split, they are increasingly rare as scale increases, so the locus of extrema as a function of scale is well approximated by a tree, called the *co-variant detection tree* [10]. A region $\omega \subset D$ is *canonizable* at scale σ if there exists a co-variant detector ψ that has one and only one isolated extremum in ω at that scale. We call this region a "visual structure". The region may be canonizable at multiple scales.

Canonization yields a number of regions each containing exactly one "structure". An image is properly sampled if any co-variant detector functional operating on the sampled image $\{I_{xy}\} \in \mathbb{R}^{X\times Y}$ yields the "same answer" (topology) that it would if ran on the "original" (continuous) image $I : D \to \mathbb{R}$.

Assuming co-visibility, Lambertian reflection and constant illumination, topological equivalence of co-variant detector responses between the scene and the image can be replaced by that between *different images of the same scene*. Thus, two temporally adjacent images are *properly sampled* at scale σ_0 in a region ω if, for all scales $\sigma \geq \sigma_0$, there exists a one-to-one correspondence between covariant detection trees in ω [13].

Proper sampling yields as a byproduct a partition of the image(s) into two regions: those for which unique correspondence across frames can be established and the rest. We call the former ones *trackable* regions. Trackable regions are both *canonizable* and *properly sampled*. Trackable regions are characterized by the "signature" of each region at the coarsest scale at which it is tracked, for instance the actual pixel values in a neighborhood of the origin of the tracked frame, as well as the frame itself, for example position, orientation and scale for the case of a similarity reference frame.

In practice, to determine the trackable regions, we use a feature point tracker such as [10]. Which regions are classified as trackable regions, depends on the detection threshold of each method. The effects of the threshold are visible in Fig. 1, where the number of tracks decreases by increasing the threshold. The ones that persist are usually the longest and most stable, a fact which we exploit for video compression.

Co-variant detector functionals can be chosen to canonize a variety of groups, from the simplest (translation) to the most complex (homeomorphisms). The larger the group, the more costly it is to encode, the larger the region that can be encoded. The optimal choice of group depends on the statistics of the images being compressed. For the purpose of illustration, in what follows we focus on the similarity group of translations, rotations and isotropic scaling. In many cases one can assume that (planar) rotation is negligible and focus on the location-scale group. Tracking then provides a (moving) reference frame, relative to which one can encode a portion of the region of the image. If the image is undergoing a similarity transformation, typically no change will be observed in the moving frame, which however is sometimes violated.

2.1. Low-Level Structure Representation

A trackable region, with index k, that appears in frames t_1 to t_2, can be represented losslessly by $\hat{F}_k =$

Figure 1. Varying the co-variant detection threshold produces different densities of trackable regions. There are typically three regions of interest, shown in the images. The tracks that persist through a wide a range of thresholds are typically the longest and most accurate.

Figure 2. Examples of track-templates, $H_k^{(avg)}$ (left) and $H_k^{(fst)}$ (right). Each row shows track-templates at different scales (29×29, 15×15 and 7×7). $H_k^{(avg)}$ is smoother since the representation involves averaging, whereas $H_k^{(fst)}$ preserves image discontinuities better.

$\{F_k(t_1), \ldots, F_k(t_2)\}$, where $F_k(t) \doteq \{I_{xy}(t), \forall (x,y) \in \omega_\sigma^k(t)\}$. $F_k(t)$ corresponds to the intensity value at pixel locations (x, y) in a neighborhood ω^k at scale (area) σ at time t. The feature point tracker [10] provides a set of regions $\mathcal{F} = \{\hat{F}_1, \ldots, \hat{F}_K\}$, where K is the number of trackable regions. We model the trackable regions (and structures) in a video, using a time-invariant dictionary element for each region that is of the same size as the region itself. We consider two alternative time-invariant representations, which we call the "track-template":

$$(a) \qquad H_k^{(avg)}(\hat{F}_k) \doteq \frac{1}{\mathcal{T}} \sum_{t=t_1}^{t_2} F_k(t) \,, \tag{1}$$

$$(b) \qquad H_k^{(fst)}(\hat{F}_k) \doteq F_k(1) \,, \tag{2}$$

where $\mathcal{T} = t_2 - t_1 + 1$. $H_k^{(fst)}$ is simply the intensity values of the track in the first frame it appears. In the mean-squared-error sense, the best one in minimizing the reconstruction error is $H_k^{(avg)}$. In Sec. 3.1, we show that by incorporating our method in H.265 we outperform H.265 in a rate-distortion comparison. For practical reasons (explained in Sec. 3.1), we are constrained to use $H_k^{(fst)}$.

One track-template is stored for each trackable region. The collection of all the track-templates, $\{H_1, \ldots, H_K\}$, from a video forms a dictionary, where the scale of each dictionary element is naturally selected to be the coarsest scale at which the track was detected. In Fig. 2, we show elements of the dictionary for a particular video. The track-templates introduce a compression gain at the expense of fidelity. For comparison, if we were to use $\hat{F}_k$ to represent the trackable regions, the distortion would have been 0, but the cost of encoding a track would have been $\beta \times (\sigma+4) \times \mathcal{T}$, where β is a constant representing the cost of storing a double (i.e. $\beta = 8$ bytes), σ is the scale of the track in space and 4 is the number of parameters of the track (x_k, y_k, t_k, k). The track-template instead only requires $\beta \times (\sigma + 4\mathcal{T})$. Hence the compression ratio is $\xi = \frac{(\sigma+4)\mathcal{T}}{\sigma+4\mathcal{T}}$. Note that a compression gain is achieved (i.e. $\xi \geq 1$) for $\sigma \geq 0$ and for $\mathcal{T} \geq 1$. We measure the distortion introduced by computing the dissimilarity of the representation $H_k(\hat{F}_k)$ from each instance of the track $F_k(t)$:

$$q(H_k(\hat{F}_k), \hat{F}_k) = \frac{1}{\mathcal{T}} \frac{1}{\sigma} \sum_{t=t_1}^{t_2} \|H_k(\hat{F}_k) - F_k(t)\|^2 \,, \tag{3}$$

where $\|.\|$ denotes the Euclidean norm. This expression computes the average squared distance per pixel from the representation to the instances of the track. If we are aiming for a specific fidelity, this function can be used to test whether the representation achieves it. In case it does not, we use a simple mechanism that would allow us to achieve that accuracy: We take each track and recursively break it in the middle, treating each half as an independent track. We stop the recursion when the desired fidelity is achieved

for every track. The downside is that each split adds an additional element to the dictionary, which reduces the compression achieved.

3. Mid-Level Structure Representation

The dictionary of track-templates, i.e. $\{H_k(\hat{F}_k)\}_{k=1,\dots,K}$, and the track parameters, i.e. $\{(x_k, y_k, t_k, k)\}_{k=1,\dots,K}\}$, need to be transmitted/stored in order to reconstruct the frames. As is, when the tracks are projected back to the image domain, there will be certain subsets of the domain where tracks would be overlapping. Since the track-templates, $H_k(\hat{F}_k)$, are an approximation of the instantaneous intensity values in each frame, $\{F_k(t_1), \dots, F_k(t_2)\}$, it typically occurs that the intensity value on each pixel in each frame is best reconstructed by one track-template among all that occupy it. To utilize the dictionary as well as possible, we need to choose for each pixel location, the track-template that minimizes the reconstruction error. In terms of coding cost though, this approach is inefficient, since we would need to transmit the index of the track-template for each pixel location.

To reduce the coding cost, we could instead consider each region where two or more tracks intersect and choose the track-template that minimizes the error on each intersection. In this way, we would be assigning one "minimizer" for each intersection, rather than for each pixel. We would then transmit the index of the minimizer and the boundaries of the intersection. However, the number of intersections per frame is large, so the number of parameters would still be prohibitively too many.

Instead, we follow an alternative approach. We choose to assign an ordering of the track-templates by placing them on depth layers. A track-template placed on a layer with a smaller "depth" would be overlaid on top of another one placed on a layer with a larger "depth". Hence, by selecting an ordering of track-templates, we implicitly choose which of them to use to reconstruct the video frames. By following this scheme, we would simply append a scalar to the track parameters, hence characterizing the track-templates with the parameter set (x_k, y_k, t_k, k, d_k), where d_k corresponds to the "depth" or ordering of that particular track-template (in each frame). The proposed solution performs a global optimization over all track-templates, which we propose to solve in one step. Note that a global optimization scheme is necessary since the depth ordering of a track-template can influence any other. An example of the proposed solution is shown in Fig. 3.

To determine the ordering of track-templates our solution draws inspiration from [14], where their model was used for segmentation, ordering and multi-object tracking. Specifically, let $V_T = \{1, 2, \dots, K\}$ be the index set of track-templates. Let $V_I = \{K+1, \dots, K+N\}$ be the index set of intersections. Each intersection is defined to be a unique combination of track-templates overlapping. Let H_k be the appearance of a track-template for $k \in V_T$ (i.e. either $H_k^{(fst)}$ or $H_k^{(avg)}$). Let M_k be the index set of intersections which are occupied by track-template $k \in V_T$. Let d_k for $k \in V_T$ be the relative depth index (ordering) of the track-template. Assume that there are at most L layers, where $L \leq K$ and that $\mathcal{L} = \{0, 1, \dots L-1\}$. Let $l_i \in V_T$ denote the index of the track-template assigned to intersection $i \in V_I$ (i.e. it is the "minimizer"). Fig. 4 illustrates these quantities. In addition, we introduce constraints A_1 and A_2. A_1 couples the track-template with the smallest depth with intersection i, by assigning its index to l_i. A_2 requires that the depth of one track-template is smaller than all others ("unique minimizer"):

$$A_1 : l_i = \underset{\{k|i \in M_k, k \in V_T\}}{\arg\min} d_k, \forall i \in V_I \,, \tag{4}$$

$$A_2 : \forall i \in V_I, \exists \tilde{k} \in \{k|i \in M_k, k \in V_T\} \tag{5}$$

$$s.t.\ \forall k' \in \{k|i \in M_k, k \in V_T\} \backslash \{\tilde{k}\}, d_{\tilde{k}} < d_{k'} \tag{6}$$

These constraints couple several templates together making the optimization complex. The same problem can be solved by considering pairwise relationships of templates and intersections only, but an additional layer of modeling needs to be introduced. Towards that end, we let $z_i = d_{l_i}$ be the depth of each intersection i. We then have the following constraint:

$$A_3 : z_i = \min_{\{k|i \in M_k, k \in V_T\}} d_k, \forall i \in V_I \,. \tag{7}$$

A_3 requires that the depth of an intersection is the same as the depth of the "minimizer" of that intersection. It was shown by [14] that the following equivalence holds: $\forall i \in V_I, A_1 \wedge A_2 \wedge A_3 \Leftrightarrow \wedge_{k \in V_T}(C_{1k} \wedge C_{2k} \wedge C_{3k})$ (for proof refer to [14]), where:

$$C_{1k} : \neg((l_i = k) \wedge (z_i \neq d_k)) \,, \tag{8}$$

$$C_{2k} : \neg((l_i = k) \wedge (i \notin M_k)) \,, \tag{9}$$

$$C_{3k} : \neg((l_i \neq k) \wedge (z_i \geq d_k) \wedge (i \in M_k)) \,. \tag{10}$$

The constraints are only pairwise relationships between template k and intersection i, which can be solved by a pairwise MRF. Specifically, the index set of nodes in the MRF is denoted by $V = V_T \cup V_I$. At each node we have a random variable $\Gamma_i \ \forall i \in V$. Γ_i takes a value γ_i from its label set $\mathcal{G}_i$. The whole MRF comprises of a discrete random vector $\Gamma = (\Gamma_i)_{i \in V}$, which takes a value γ in $\mathcal{G} = \mathcal{G}_1 \times \dots \times \mathcal{G}_{|V|}$. The edges of the MRF connect the templates with the intersections denoted by $\mathcal{E} = \{(k, i)|k \in V_T, i \in V_I\}$. Hence, we have the following energy with configuration γ:

$$E(\gamma) = \sum_{i \in V} \phi_i(\gamma_i) + \sum_{(k,i) \in \mathcal{E}} \psi_{k,i}(\gamma_k, \gamma_i) \,. \tag{11}$$

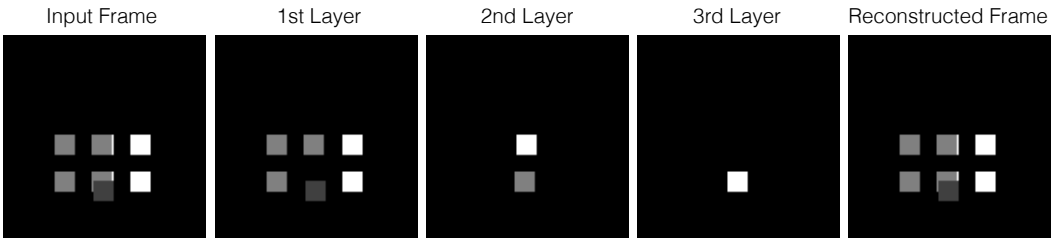

Figure 3. Encoding structures in a frame. Problem illustration. For this instance of the problem the dictionary is composed of 9 track-templates: 4 white, 4 light gray and 1 dark gray square. The original frame is decomposed into 3 layers. Occluded track-templates are pushed to the back layers. Our proposed solution retrieves the 3 middle frames, which along with the track-template parameters are used to reconstruct the input frame (right).

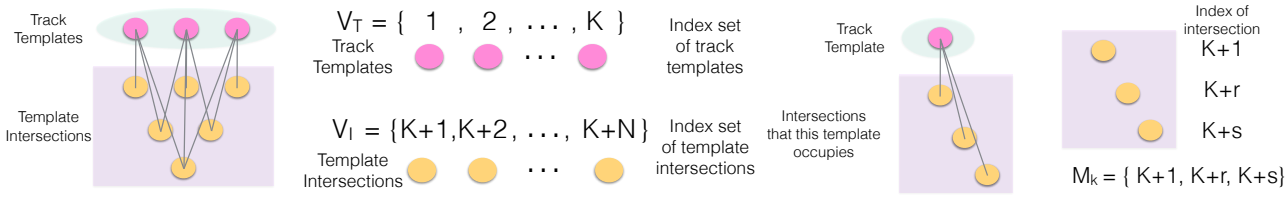

Figure 4. Left to right: (1) Model illustration. Top nodes represents 3 track-templates. Bottom nodes show the intersections of the 3 track-templates. Edges are drawn between every template and intersections occupied by them. (2) Index sets V_T and V_I. (3) M_k.

The nodes have the following potentials:

$$\forall i \in V_I, \gamma_i = (l_i, z_i), \quad \phi_i(\gamma_i) = ||I_i - H_{l_i}|| \,, \tag{12}$$

$$\forall i \in V_T, \gamma_i = d_i, \quad \phi_i(\gamma_i) = \alpha |d_i| \,, \tag{13}$$

where the first expression measures the reconstruction error for a particular intersection and the second one gives a higher preference to smaller depth values. The pairwise potentials are given by:

$$\psi_{k,i}(\gamma_k, \gamma_i) = \psi^1_{k,i}(\gamma_k, \gamma_i) + \psi^2_{k,i}(\gamma_k, \gamma_i) + \psi^3_{k,i}(\gamma_k, \gamma_i) \,, \tag{14}$$

$$\psi^1_{k,i}(\gamma_k, \gamma_i) = \lambda_1 \mathbb{1}((l_i = k) \wedge (z_i \neq d_k)) \,, \tag{15}$$

$$\psi^2_{k,i}(\gamma_k, \gamma_i) = \lambda_2 \mathbb{1}((l_i = k) \wedge (i \notin M_k)) \,, \tag{16}$$

$$\psi^3_{k,i}(\gamma_k, \gamma_i) = \lambda_3 \mathbb{1}((l_i \neq k) \wedge (z_i \geq d_k) \wedge (i \in M_k)) \,. \tag{17}$$

We solve $\gamma^{opt} = \arg\min_\gamma E(\gamma)$ using any standard inference method e.g. TRW-S [9]. The optimization is performed for each frame independently. We fix $\alpha = 1$, $\lambda_1 = 50$, $\lambda_2 = 1$ and $\lambda_3 = 10$. In addition note that the number of layers, L, used to represent the "visual structures" of the video is a parameter that is automatically inferred during optimization and does not require prior knowledge. Finally, unlike [14] where V_T corresponds to objects and V_I corresponds to pixels, in our work these quantities correspond to track-templates and intersections respectively. Intersections were introduced in our problem to tackle the problem of video compression. Note that without this change in the model, [14] would have performed poorly in the context of video compression in terms of coding cost since a minimizer would have been assigned for each pixel.

3.1. Integration With Standard Video Encoders

Once we retrieve the ordering of the track-templates using the previous step we can reconstruct the frames by transmitting the parameter set for each track-template: (x_k, y_k, t_k, k, d_k). The compression ratio of the representation to the uncompressed lossless one is $\xi = \frac{(\sigma+4)\mathcal{T}}{\sigma+5\mathcal{T}}$. For integer-valued σ and $\mathcal{T}$, $\xi \geq 1$ for $(\sigma > 1, \mathcal{T} > 1)$. We are therefore able to compress any track of length $\mathcal{T} \geq 2$ (i.e. the "trackable regions"). Using the depth d_k, we can reconstruct each frame by overlaying the structured regions with a smaller depth on top of others. In Fig. 5, we show how a frame from a video is decomposed into layers and then reconstructed to recover all trackable regions.

To store the track-templates we use the following procedure: At encoding, each track-template is stored *once* in the first video frame it appears and in the remaining frames we store a constant intensity value e.g. 0 or the mean of the local neighborhood. In addition, we also store the track-templates' parameters i.e. $\{(x_k, y_k, t_k, k, d_k)\}_{k=1,\ldots,K}$. At decoding, we are able to recover each track-template by simply selecting the appropriate image region that corresponds to that track from the frame that it was stored during encoding. We then propagate the track-template to other frames using its stored parameters. Note that with this approach, it is impossible to recover $H^{(avg)}_k$. This is due to the fact that a track-template that is not on the top layer (i.e $d_k = 0$) cannot be recovered, since another track at a "higher" layer has been overlaid on top of it. When using $H^{(fst)}$ though, track-templates are always put on the top layer in the first frame they appear. This allows us to

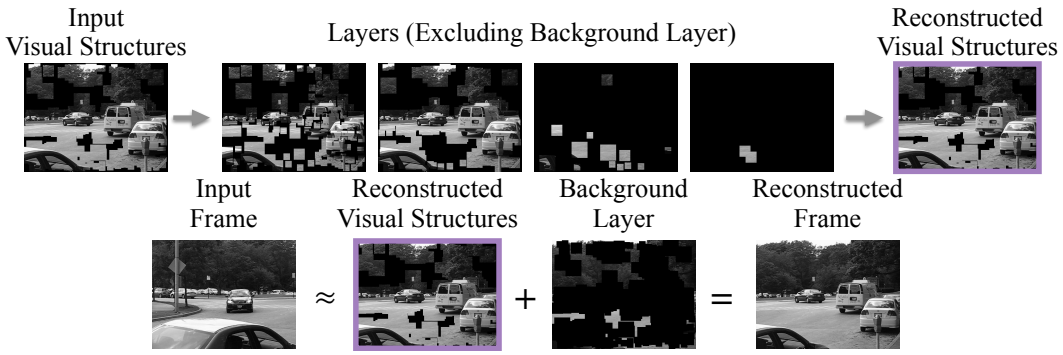

Figure 5. Reconstructing a frame. Visual structures are decomposed into depth layers and reconciled by overlaying them. The input frame is reconstructed by adding back to the visual structures the background layer.

recover the exact track-templates at the decoder and hence reconstruct the structured regions.

Regions of the image not occupied by track-templates are encoded as a background layer. Each frame sent to a standard video encoder (e.g. H.265) is composed by the track-templates that first appear in that frame and the background layer (Fig. 5).

4. Experiments

We investigated how well the structure representation reconstructs the individual instances of the track, without applying the recursive, splitting method described in Sec. 2.1. Towards this end, we used the 10 car and 2 people sequences from the MOSEG dataset [3]. The sequences range from 19-60 frames, but for this experiment, we only used the first 19 frames for all videos to achieve uniformity in the results. We computed the structure representations and reconstructed the trackable regions of the videos using our proposed solution. For each track-template, we computed its average reconstruction error per pixel for each instance of the track according to Eq. 3 for both $H_k^{(avg)}$ and $H_k^{(fst)}$. We used 5 different scales for tracking with the smallest one being 7×7 and the largest being 35×35. In addition, we have varied the detection threshold of tracks and selected 3 representative levels. Typical distributions of tracks on the image domain, for the three thresholds, are shown in Fig. 1.

In Fig. 6, we show how $q(H_k(\hat{F}_k), \hat{F}_k)$ varies for both representations as a function of the length of the track and the scale of the track. We also show histograms of the distribution of tracks according to scale and length. For $H_k^{(avg)}$, the reconstruction error per pixel increases with increasing lengths and scales, but the increase is small and hence shows that the average can reliably represent tracks, even if they are long. For $H_k^{(fst)}$, the error increases only slightly faster.

We used our proposed system to encode the first 5 frames of the 12 video sequences from MOSEG. We used H.265/H.264 to encode the frames with the structure representations and background layers placed on them. We have also encoded the videos using HEVC/H.265 (HM 16.2)[2], H.264 (JM 18.6 Reference Software [1]) and JPEG. Note that our method can be used along any other video encoding system, replacing H.265/H.264. In Fig. 7 we plot PSNR (dB) against bit rate (kbps) for our approach ("VS+H.265", "VS+H.264"), H.265, H.264 and JPEG. For better coverage of the image domain, we expanded the domain of each track-template by a factor of 3^2. To achieve varying fidelity for all methods, we varied the quantization levels.

We consistently outperform all other methods in all sequences. In these experiments, the representations achieve at least 25 dB in PSNR for each of the instances of the track they are representing (using our recursive, splitting algorithm), before they are passed to H.265/H.264. At lower fidelity, the performance gain of our method diminishes due to the parameter overhead that needs to be transmitted. At higher fidelity our approach benefits from taking advantage of the temporal redundancy of the tracks and it is much more efficient than competitive approaches.

Fig. 8 illustrates where the gain is achieved in our methods. For the last frame of the video sequences, we show which regions were predicted from previous frames (non-transparent regions) and which first appeared in this frame (semi-transparent). Generally, the larger the percentage of tracks that are temporally predicted, the larger the improvement is over other methods. While H.265/H.264 encodes the temporally predicted tracks, our encoder predicts them from previous frames. Our algorithm takes on average 96 seconds on the encoder side and 0.5 seconds on the decoder side per frame (excluding the computational time required by H.265/H.264), for a MATLAB/C++ implementation on an Intel 2.4 GHz dual core processor machine.

Future challenges. Tracks on fast moving objects such as cars are split in some sequences, hence reducing the temporal exploitation that could have been achieved. To overcome this, a richer representation is required, possibly one that is

[2] https://hevc.hhi.fraunhofer.de/

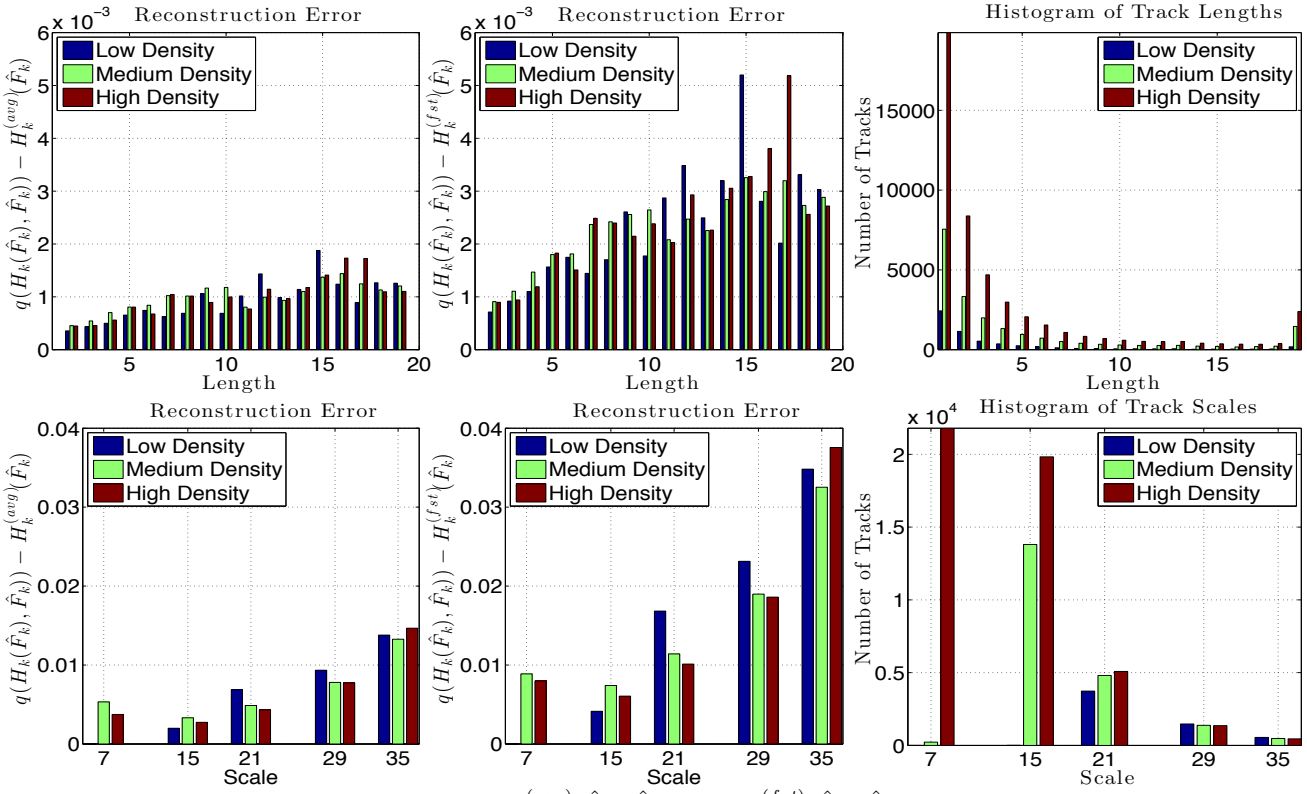

Figure 6. Results for tracks in MOSEG [3]. Top: $q(H_k^{(avg)}(\hat{F}_k), \hat{F}_k)$ and $q(H_k^{(fst)}(\hat{F}_k), \hat{F}_k)$ as a function of length and a histogram of track lengths. Bottom: $q(H_k^{(avg)}(\hat{F}_k), \hat{F}_k)$ and $q(H_k^{(fst)}(\hat{F}_k), \hat{F}_k)$ as a function of scale and the distribution of track scales.

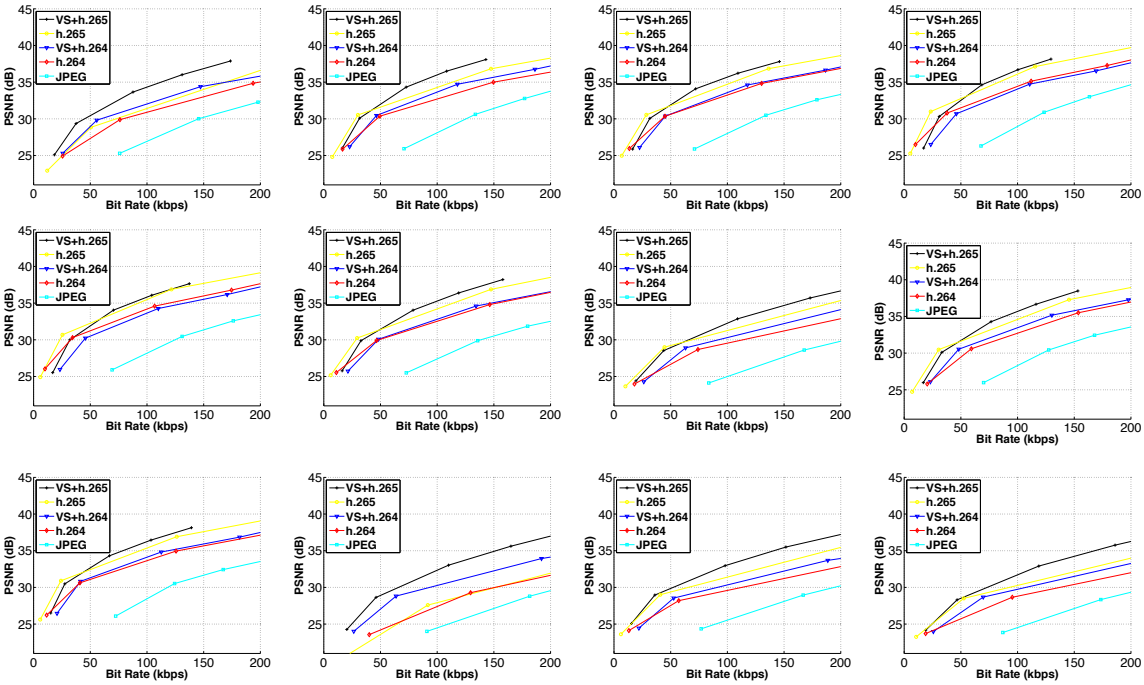

Figure 7. PSNR against bit rate. "VS+H.265"(black) and "VS+H.264"(blue) outperform respectively H.265(yellow) and H.264(red). Figures correspond to the sequences in Fig. 8.

time-varying, but still more compact than simply encoding each of the instances of the track in each frame. The mid-

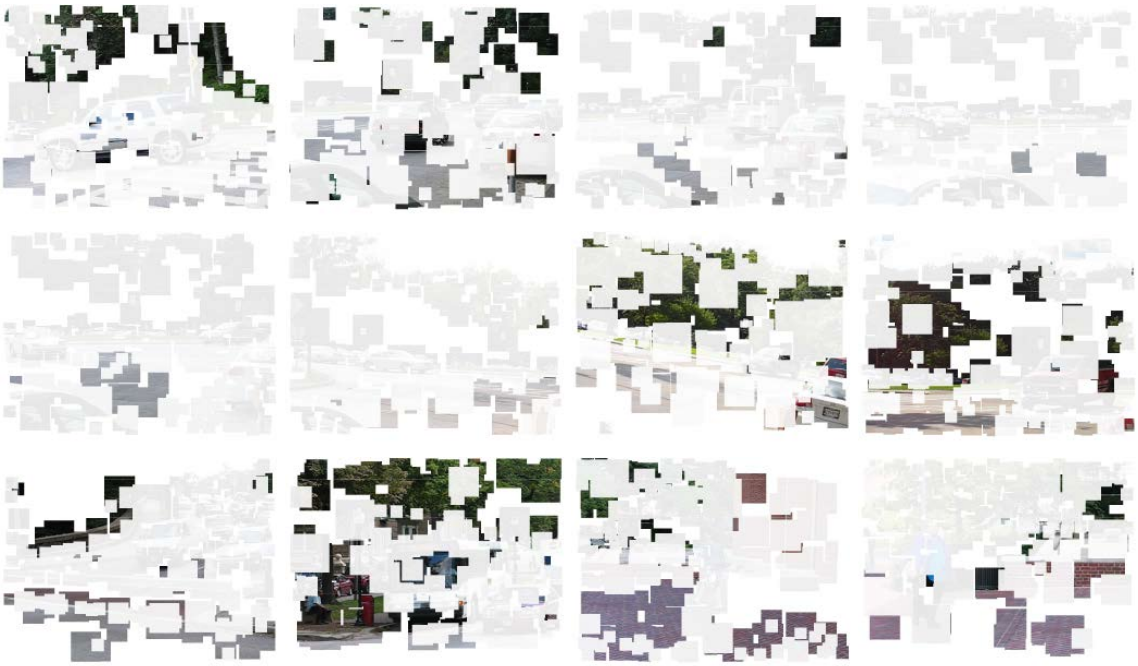

Figure 8. Propagated and newly-created tracks. Non-transparent tracks correspond to tracks that are motion-predicted from previous frames. Semi-transparent tracks are tracks that start in this frame. All results shown are correspond to the fifth frame of each video.

level representation of tracks is independent of such choice, hence the overall approach allows flexibility in what low-level representation we could use.

Acknowledgments. We would like to thank Avinash Ravichandran and Chaohui Wang for valuable discussions. Research supported by AFOSR - FA9550-15-1-0229 and ONR - N00014-15-1-2261.

5. Conclusion

We presented an alternative system to traditional video encoders, which was shown to exploit the temporal redundancy of visual structures. The frames were partitioned into structures and background layers. Structures are compressed using a time-invariant representation (low-level representation). They are then ordered in terms of reconstruction error and are used to reconstruct the video along with the background layers (mid-level representation). Our method can be wrapped around standard encoders such as H.265 and H.264 and it outperforms both of them in a rate-distortion criterion. Finally, the mid-level representation proposed could potentially have other uses beyond compression such as action recognition and other high level applications.

References

[1] H.264/AVC JM Reference Software, Aug. 2008.

[2] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *TIP*, 12(8):882–889, 2003.

[3] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. *ECCV*, 2010.

[4] CISCO. Entering the zettabyte era, visual networking index. CISCO VNI, 2014.

[5] G. Georgadis and S. Soatto. Exploiting temporal redundancy of visual structures for video compression. *DCC*, 2015.

[6] G. Georgiadis and S. Soatto. Scene-aware video modeling and compression. In *Data Compression Conference*. April 2012.

[7] C. Guo, S. Zhu, and Y. N. Wu. Toward a mathematical theory of primal sketch and sketchability. *ICCV*, 2003.

[8] ITUT-Recommendations. http://www.itu.int/itu-t/recommendations/.

[9] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 2006.

[10] T. Lee and S. Soatto. Video-based descriptors for object recognition. *Image and Vision Computing*, 2011.

[11] P. Ndjiki-Nya, D. Bull, and T. Wiegand. Perception-oriented video coding based on texture analysis and synthesis. In *ICIP*, 2009.

[12] B. Sandberg, T. Chan, and L. Vese. A level-set and Gabor-based active contour algorithm for segmenting textured images. *UCLA CAM report*, 2002.

[13] S. Soatto. *Steps Toward a Theory of Visual Information*. ArXiv http://arxiv.org/abs/1110.2053, 2010.

[14] C. Wang, M. de La Gorce, and N. Paragios. Segmentation, ordering and multi-object tracking using graphical models. In *ICCV*, 2009.

[15] J. Wang and E. Adelson. Layered representation for image sequence coding. In *ICASSP*, 1993.