Inference of scene representations for video analysis

Georgios Georgiadis
Advised by: Stefano Soatto
Dissertation Prospectus
University of California, Los Angeles
Computer Science Department
giorgos@cs.ucla.edu

Abstract

The growth of video consumption over the Internet necessitates its analysis and understanding. Video analysis is necessary to improve video retrieval, interpretation, recognition and other applications. Whereas for these tasks, representations that capture the main properties of video have been investigated, this is still not done for the purpose of video compression. Video compression algorithms need to raise the bar in terms of compression performance in order to bring down the pressure on bandwidth. Current designs [31], largely ignore the scene and instead encode pixel values. We believe that in order to break the "compression wall", it is necessary to instead model the scene and its properties and encode them accordingly.

We investigate possible representations for different types of regions such as "structures" and "textures". We provide definitions and develop algorithms that stem out directly from them for texture compression, synthesis, segmentation and structure / texture partition. We develop a system that finds pixel-wise accurate boundaries of textures and structures. We show that our representations can be used for video compression with huge improvements in terms of compression ratios. Finally, we demonstrate the importance of video representation in a different application, that of detection of independent motions in a video. Our approach outperforms current state of the art methods.

Contents

1 Introduction

Video understanding is a key area of research in the computer vision community. The need to analyze videos, represent them appropriately and interpret them has given rise to a plethora of works by researchers. Applications drive the research to directions such as action recognition, object recognition from video, event detection and others. At the same time, pressure from the applications[1] calls for new approaches to break the "video compression wall". To do so, encoders ought to explicitly model and exploit the phenomenology of the data-formation process, rather than treating video as a generic bit-stream. In other words, rather than encoding the video, one should encode some "representation" of the *scene* that generated it.

The representation (encoding) depends on the task: The best encoding for human fruition (e.g., compression of movies) is different from the best for surveillance, robotic navigation, or content-based video retrieval [58]. But computer vision techniques can be used for both video compression and analysis or interpretation of content. In this work we first focus on human fruition and show how our video representation can be used for compression. Future work involves exploring representations that could be used for high level tasks such as action recognition.

Two phenomena of the optical data-formation process are critical to compression: *occlusion* and *scaling*. Occlusion implies that there are portions of the scene, or the light source, that are visible in one image but not another, and therefore must be encoded anew. *Scaling* implies that there is no "Nyquist frequency" in the scene, as moving closer to objects causes more and more structure to appear. This causes phase transitions (singular perturbations) in the representation, and forces the continuous limit to be part of the analysis.

Our goal is to develop algorithms that achieve high compression performance relative to a perceptual metric by encoding *not* the video stream directly, but instead the *scene* that generated it, albeit without explicitly reconstructing it. To this end, in Chapter 2 we propose a formal definition of "visual texture" and characterize it by the approximate sufficient statistics of the underlying process. These are inferred from data and used for compression, extrapolation and segmentation. The formalization highlights relations between textures and other early vision operations, such as co-variant feature selection and correspondence. We show that co-variant frames ("structures") are the complement of textures in an image. Unlike prior work on texture/structure partitioning, however, we show that such a decomposition requires multiple images: to attribute structures in the image to properties of the scene, a proper sampling condition has to be satisfied, which requires multiple realizations. In Chapter 3 we propose a method to improve the spatial localization of such a partition. In Chapter 4 we show how video compression can benefit from our approach.

To further demonstrate the importance of scene representations we present another task: detection of salient regions of the image (Chapter 5). In particular, we are interested in capturing small objects that move independently in the scene, such as vehicles and people as seen from aerial or ground vehicles. Many of the scenarios of interest challenge

[1] CISCO projects that, by 2014, 90% of Internet traffic, and 64% of wireless traffic will be video. In 2010, video surpassed peer-to-peer traffic on the Internet [11].

existing schemes based on background subtraction (background motion too complex), multi-body motion estimation (insufficient parallax), and occlusion detection (uniformly textured background regions). We adopt a robust statistical inference approach to simultaneously estimate a maximally reduced regressor, and select regions that violate the null hypothesis (co-visibility under an epipolar domain deformation) as "salient". We show that our algorithm can perform even in the absence of camera calibration information: while the resulting motion estimates would be incorrect, the partition of the domain into salient vs. non-salient is unaffected. We demonstrate our algorithm on video footage from helicopters, airplanes, and ground vehicles. Finally, in Chapter 6 we outline our plan for future work.

1.1 Related Work

Video understanding is a diverse area of research with several different goals and applications. Typical areas of research include among others, tracking e.g. KLT and particle filter [61, 66, 50], optical flow [26, 8, 41], occlusion detection [3, 6], video compression and action recognition. These areas find applicability in surveillance (where the goal would be to detect unusual or suspicious behavior), video content filtering (removal or flagging of inappropriate video material), the movie industry (storage and transmission of movies), autonomous driving, eye tracking, sports analysis, games and gesture recognition, medical industry and many others.

In this work we concentrate in understanding how the scene interacting with the image formation process influences what we observe in video. We concentrate on two applications: video compression and independent motion detection. Our work is obviously related to all video compression standards that offer end-to-end solutions such as H.262, H.263, H.264 [31]. Unlike these methods though, we do not subdivide the image frames into macroblocks but instead into regions that exhibit some type of redundancy. That could be spatial stationarity of pixel values or of the motion field. Hence an integral part of our work is clustering of the image domain that is usually done by segmentation algorithms e.g. gPb, mean shift [2, 12, 16, 33].

Texture analysis and modeling [45, 14, 15, 40, 36] including texture/structure partitioning [5, 21, 76, 72], texture segmentation [9] is also related to this manuscript. We do not address phenomenology (*e.g.,* "3-D textures" generated by the interplay of shape and illumination [60] vs. "decal textures" due to radiance) and the algorithms used to pool the statistics (*e.g.,* patch-based [68] vs. statistical vs. geometric methods). We refer the reader to more extensive treatises, surveys and edited volumes such as [22] for a broader view of the texture literature.

Traditional texture compression schemes involve a transformation stage (e.g. by applying the Discrete Cosine Transform (DCT) or the Discrete Wavelet Transform (DWT) to the image) which aims to compact the spectral energy into a few coefficients. These coefficients are then quantized and typically the quantization steps are set by taking into account some perceptual metric [4, 17, 46].

Evaluating the performance of our approach for the purpose of compression is difficult since there has not been a universally accepted perceptual metric. Current approaches evaluate their performance using the PSNR metric. This is an unsuitable metric for video compression since it does not capture the visual perception of humans

([44] and refs.). A recent paper by Moorthy et al. [47] gives an overview of current algorithms used for image and video quality assessment. For 2D image quality assessment (IQA) they propose the usage of the sturctural similarity index (SSIM) [71] and the information-theoretic visual information fidelity (VIF) [53]. They argue that these two indices correlate better with human perception compared to other IQA approaches. VIF [53] is an information theoretic based metric that essentially compares the mutual information between the original image and the output of an HVS model with the mutual information between the reconstructed image and the output of the HVS model.

Although some argue (e.g. [47]) that in order to further improve the QA algorithms we will need to incorporate advanced human visual system (HVS) models, others (e.g. [52]) argue that this could also be a limitation: these schemes will be as good as our understanding of the human visual system (which for now is far from perfect). In addition, HVS models fit their parameters to psychophysical data that were developed using stimuli such as sine waves and Gabor patches [52]. Hence their generalizability to images and videos is under question. In addition, in order to apply approaches such as [64] we need to also have access to the data derived from experiments performed on human subjects.

Also relevant to our work is texture synthesis. The most successful algorithms are based on the Markov Random Field (MRF) model. The assumption is that texture is a realization of a local and stationary random process. Hence a pixel value depends only on its neighbors and is independent of its absolute position. The two main classes of texture synthesis algorithms are pixel based and patch based synthesis algorithms.

In pixel based algorithms we progressively synthesize the image by estimating one pixel at a time. We usually initialize the algorithm with a small seed region from the input. We repeat the process until the whole output image is generated. Pixel based algorithms can generally be slow. According to [39] k-coherence [73] is among the top performers in both speed and quality.

Patch based algorithms can be faster and also yield better results. Patches instead of pixels are synthesized at a time. The issue comes in handling adjacent patches: some algorithms simply overwrite older patches [48], others blend them by averaging [40], and other optimize by dynamic programming [15] or graph cuts [35].

An alternative to the above two classes is a method that lies in between. Authors of [36] use an EM algorithm to optimize an energy function (which could in principle be used as a metric in evaluating texture synthesis) over the synthesized (i.e. output) pixels and the input neighborhoods. At the E-step neighborhoods are fixed and the pixel values are estimated by minimizing the energy function. At the M-step the output pixel values are fixed and a search for the best possible neighborhoods from the input image is performed. These two steps are repeated until convergence. The algorithm is done in a multi-resolution, multi-scale fashion.

In this manuscript we also discuss another application of video understanding: independent motion detection. To do that we pose the problem as finding salient regions of the video. Several algorithms [24, 28, 49, 51, 74] have been proposed to detect salient regions with background subtraction techniques by removing the stationary camera limitation. However, these methods largely rely on estimation of a homography or a 2D affine transform to compensate for the camera motion for the scenes that can be approximated well by a plane and do not work in the case of a complex scene.

Most relevant to our work is Sheikh et al. [54] which also detects independently moving objects by exploiting a geometrical constraint. In their case, they exploit the rank-constraint on trajectories on the background and hence they overcome the restrictions on the scene models shared by others. However, this method requires a large number of frames to operate.

2 Texture, Structure and Visual Matching

"Visual textures" are regions of images that exhibit some form of spatial regularity. They include so-called "regular" or "quasi-regular" textures (Fig. 1 left), "stochastic" textures (center-left), possibly deformed either in the domain (center-right) or range (right). The notion has a long history in visual perception and computational vision, but also in computer graphics, computational geometry, content-based image retrieval, with slightly different characterizations. Thus our first aim is to define textures in the context of vision-based decision tasks, such as detection, recognition, localization and categorization. It is generally assumed that knowing that a portion of the image is a "texture" is *useful* for these tasks. However, the data processing inequality ([13] Thm. 2.8.1) suggests otherwise, for any decision not tied to the task in general increases the expected risk. Naturally, therefore, textures raise the issue of approximating the sufficient statistics of the underlying process.

On the other hand, "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. Co-variant frames enable correspondence among different images of the same scene, which in turn enables reconstruction of its shape.

It has been argued that an image can be partitioned into structures and textures [21] based on statistics computed in that *one* image. However, consider a random-dot stereogram. It fits most definitions of textures, but point-to-point correspondence can be established [34], so it should be treated as a collection of structures. This conundrum is resolved in our framework.

For image analysis to respect the data processing inequality, it should yield properties of the underlying scene, object, or category, and reduce variability due to nuisance factors such as vantage point, illumination etc. Unfortunately, what is usually assumed to be a "texture" is not a property of the scene, as it depends on the vantage point, optical characteristics of the sensor and its spatial quantization, etc. We therefore introduce a notion of *proper sampling* to test whether image structures arise from properties of the scene. Such sampling is different than traditional Shannon-Nyquist theory, since changes of viewpoint yield structural changes in the image, and therefore no "Nyquist frequency" exists. Instead, 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. Therefore, in our framework texture analysis and texture/structure partitioning requires *multiple images* and yields temporal correspondence (where possible) as a byproduct. This is unlike [72] where such "perceptual transitions" are decided in a single image, and therefore cannot be expected to be insensitive to nuisance factors.

We provide background material (Sect. 2.1) necessary to make this manuscript self-contained, a definition of texture (Sect. 2.3) that encompasses all existing informal definitions, a description of inference algorithms that stem naturally from the definition (Sect. 2.4), including a novel texture compression (summarization) algorithm based on an approximate sufficient statistic compatible with the Information Bottleneck principle [65]; a novel algorithm for texture extrapolation or inpainting that is a straightforward

Figure 1: *Left to right: regular textures, stochastic texture, domain-deformed and range-deformed textures*

modification of [36] that captures the correct Markov structure; a novel algorithm for texture segmentation and a characterization of the relation between textures and structures (Thm. 1).

2.1 Background

Digital images $\{I_{ij}\}_{(i,j)=1:(N,M)} \in \mathbb{R}^{M\times N}$ are obtained by averaging a function $I : D \subset \mathbb{R}^2 \to \mathbb{R};\ x \mapsto I(x)$ on a neighborhood $\mathcal{B}$ of the point $x_{ij} \in D$ of size $\epsilon > 0$. In general, $I_{ij} = I(x_{ij}) + n_{ij}$ where $n_{ij} = n_{ij}(I)$ is the quantization error. We consider groups of transformations of the sensor plane, $g : D \subset \mathbb{R}^2 \to \mathbb{R}^2; x \mapsto g(x)$, and denote their induced action on the image by $I \circ g \doteq I(g(x))$. The simplest instance is the translation group, represented by a translation vector $T \in \mathbb{R}^2$, via $g(x) \doteq x + T$, so that $I \circ g(x) \doteq I(x + T)$. 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 $T \in \mathbb{R}^2$. Similarly, an element of the Euclidean group $g = (R, T) \in SE(2)$, where R is a rotation matrix ($R^T R = I$ and $\det(R) = 1$), determines a reference frame with origin at T and coordinate axes aligned with the columns of R. All the considerations below apply to other finite-dimensional Lie groups of the plane including Euclidean, similarity, affine, and projective. Some considerations also apply to (infinite-dimensional) planar diffeomorphisms, but with some technical complications.

2.1.1 Structures and Proper Sampling

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 a canonical element of each orbit that co-varies with the group; in the corresponding (moving) frame, the data is then 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* ϵ if its value at g only depends on a neighborhood of the image of size $\epsilon > 0$, up to a residual that is smaller than the mean quantization error. For instance, for a translational frame g, if we call $I_{|_{\mathcal{B}_\epsilon(g)}}$ an image that is identical to I in a neighborhood of size ϵ centered at position $g \equiv T$, and zero otherwise, then $\psi(I_{|_{\mathcal{B}_\epsilon(g)}}, g) = \psi(I, g) + \tilde{n}$, with $|\tilde{n}| \leq \frac{1}{NM} \sum_{i,j} |n_{ij}|$. For instance, a functional that evaluates the image at a pixel $g \equiv T = x \in \mathcal{B}_\epsilon(x_{ij})$, is local with effective support

ϵ. For groups other than translation, we consider the image in the reference frame determined by g, or equivalently consider 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*. Examples of (translation-) co-variant detectors are Laplacian-of-Gaussian (LoG), the difference-of-Gaussians (DoG) the Hessian-of-Gaussian (HoG), Harris' corner and its variants. Although these detectors are not local, they are often treated as such. Varying the scale parameter σ produces a *scale-space*, whereby the locus of extrema of ψ describes a *graph* in $\mathbb{R}^3$, via $(x, \sigma) \mapsto \hat{x} = \hat{g}(I; \sigma)$. Starting from the finest scale (smallest σ), one will have a large number of extrema; as σ increases, extrema will merge or disappear. Although in two-dimensional scale space extrema can also appear as well as split, such *genetic* effects (births and bifurcation) are increasingly rare as scale increases , so the locus of extrema as a function of scale is well approximated by a tree, which is called *co-variant detection tree* [37].

We say that 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. Note that the same region may be canonizable at multiple scales. The canonization process yields a number of regions each containing exactly one "structure," that is an extremum with respect to the chosen group at the given scale. Such structures may be "real," in the sense of corresponding to some property of the *scene*, or they may be "aliases" due to sampling artifacts, noise, and other phenomena that do not depend on the scene. To distinguish between these two, we introduce the notion of *proper sampling*. Roughly speaking, an image is properly sampled if any co-variant detector functional operating on the sampled image $\{I_{ij}\} \in \mathbb{R}^{N \times M}$ yields the "same answer" (topology) that it would if run on the "original" (continuous) image $I : D \to \mathbb{R}^+$. Under the assumptions of co-visibility, Lambertian reflection and constant illumination, topological equivalence of co-variant detector responses between the scene and the image can be replaced by topological equivalence of co-variant detector responses between *different images of the same scene*. These could be just two samples of the same process, so the only difference between the two is the noise realization, or two images taken from different vantage points, different resolution, etc. [58]. 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 exist a one-to-one correspondence between covariant detection trees in Ω. In other words, for each scale $\sigma \geq \sigma_0$, for each local canonical representative $\hat{g}(I; t, \sigma)$ in the first image, there exists one and only one canonical representative in the second one, $\hat{g}(I; t+1, \tilde{\sigma})$, for some $\tilde{\sigma}$. Testing for proper sampling entails establishing correspondence of covariant selection trees [37].

2.1.2 Stationarity and Ergodicity

Consider a subset $\omega \subset \mathbb{Z}^2$, with cardinality $|\omega|$, and functions ϕ (*"features"*) that map image values onto a vector space $\mathbb{R}^W$. A *"local"* feature $\phi_\omega \doteq \phi(I(\omega))$ operates on a restriction of the image to a subset $\omega \subset \mathbb{Z}^2$, $I(\omega) \doteq \{I(x),\ x \in \omega\}$. A probability

distribution $dP(I)$ on the set of images induces a distribution on the feature via the map ϕ_ω, $dP(\phi_\omega) = dP(\phi(I(\omega))$. Given a group G, a set ω and a function ϕ_ω, we say that the distribution $dP(I)$ is *G-stationary* in ϕ_ω if there exists $g \in G$ such that $\mathbb{E}(\phi_{g(\omega)})$ is translation-invariant, that is

$$\mathbb{E}(\phi_{g(\omega)}) = \mathbb{E}(\phi_{g(\omega)+T}), \quad T \in \mathbb{R}^2 \tag{1}$$

where $g(\omega) = \{g(x) \mid x \in \omega\} \cap \mathbb{Z}^2$ and $g(\omega) + T = \{g(x) + T \mid x \in \omega\} \cap \mathbb{Z}^2$. With an abuse of nomenclature, we say that the image I is stationary, rather than the distribution $dP(I)$.

If (1) is satisfied only for T that belongs to a discrete subgroup of planar translations (Frieze symmetries, see [42]), the process is *cyclo-stationary*. In practice, the image is only defined on a bounded domain, so we introduce the notion of *local stationarity*. Given ω and a superset $\bar{\omega} \supset \omega$, we say that I is locally stationary in $\bar{\omega}$ if (1) is satisfied *not* for all $T \in \mathbb{R}^2$, but only for those such that $g(\omega) + T \subset \bar{\omega}$. We call such T's *admissible*, and $\sigma = |\bar{\omega}|$ the stationarity scale. Note that the value of the statistic ϕ_ω remains unchanged if we consider any superset of ω; in particular, we have $\phi_\omega = \phi_{\bar{\omega}}$. The *largest* admissible region where the stationarity assumption is satisfied will be called Ω ("big-omega"). Note that $\omega \subset \bar{\omega} \subset \Omega$.

Stationarity implies that there is an underlying statistical model which describes the image in the region Ω. However, when it comes to performing *statistical inference*, one has to ensure that this model can be consistently inferred from data. This requires linking the sample properties to the ensemble (probabilistic) properties, and is captured by the notion of *ergodicity*. A stationary process is ergodic if sample averages converge to ensemble averages (expectations):

$$\frac{1}{N}\sum_{i=1}^{N} \phi_{g(\omega)}(I(g(\omega) + T_i) \xrightarrow{\text{a. s.}} \mathbb{E}(\phi_{g(\omega)}(I(g(\omega) + T_i)) \tag{2}$$

for all $T_i \in \mathbb{R}^2$. Stationarity can then be tested by comparing samples of I in $g(\omega)$ to admissible samples in the transformed domain $g(\omega) + T_i$. The maximum number of different samples N is bounded by the area of $\bar{\omega}$, so for any finite $|\bar{\omega}|$ there will be a threshold, $\theta = \theta(|\bar{\omega}|)$ to decide whether the process is stationary, yielding an *empirical stationarity test*. Note that the sole fact of performing an empirical stationarity test from *one* image, implicitly requires that the underlying process is ergodic.

The fact that a statistic is stationary does not imply that it is *sufficiently informative* in the sense of enabling the statistical characterization of the process. To that end, we introduce a notion of Markovianity below.

2.2 Markovianity and sufficient reduction

Once established that a process is stationary, hence spatially predictable, we can inquire on the existence of a statistic that is *sufficient* to perform the prediction. We say that a process is Markovian if every set $A \subset \Omega$ admits a neighborhood $\mathcal{N}(A)$, such that a statistic $\phi_{\mathcal{N}(A)}$ computed in $\mathcal{N}(A)$ makes $I(A)$ independent of the "outside" $I(A^c)$, where A^c is the complement of A in Ω:

$$I(A) \perp I(A^c) \mid \phi_{\mathcal{N}(A)}. \tag{3}$$

This makes the process I with measure $dP(I)$ a Markov Random Field (MRF). Of particular interest is the case when the neighborhood structure $\mathcal{N}(A)$ is induced by a set $\omega_x \doteq \mathcal{N}(x)$ which satisfies the property $\omega_{x+T} \doteq \mathcal{N}(x+T) = \omega_x + T = \mathcal{N}(x) + T$, $\forall T \in \mathbb{Z}^2$, so that the neighborhood structure is spatially homogeneous. Of course any *stationary* Markov random field satisfies this property. We shall denote by $\mathcal{N}_\omega(A)$ the neighborhood structure induced by ω where the subscript x has been dropped for obvious reasons. Note that the region ω and the statistic ϕ_ω that we use to define Markovianity are not the same we used to define stationarity in the previous section. We are overloading the notation to avoid introducing too many new symbols.

Remark 1 (Markov neighborhoods) *The "neighborhood" $\omega \backslash x$ of x consists of all pixels that are connected to x according to the Markov structure of the underlying process, and should not be confused with the set of pixels that are connected to x according to the lattice structure of the image (e.g the 4-connected or 8-connected neighbors). While it may be possible to predict the value of a pixel x given its lattice neighbors, this does not imply that such a neighborhood captures the Markov structure. For instance, consider a checkerboard image: The value of a pixel (black or white) can be predicted given its lattice neighbors, but this does not mean that $|\omega| = 8$ pixels, as this neighborhood does not allow predicting the value of pixels outside ω. In this case, the correct ω must include at least one period of the underlying signal. In fact condition (3) has a global nature and it is equivalent to $I(x) \perp I(\omega^c) \mid \phi_{\omega - x}$ once the neighborhood structure has been fixed.*

Equation (3) establishes $I(\mathcal{N}_\omega(\mathcal{A}))$ as a (Bayesian) *sufficient statistic* for $I(A)$. In general, there will be many regions ω that satisfy this condition; the one with the smallest area $|\omega| = r$, is a *minimal sufficient statistic*. From now on, we will refer to ϕ_ω as the minimal *Markov sufficient statistic.*

2.3 Textures

A texture is a region of an image that can be rectified into a sample of a stochastic process of a planar lattice that is locally stationary, ergodic and Markovian. More precisely, assuming for simplicity the trivial (translation) group $g(x) = x + T$, a region $\Omega \subset D \subset \mathbb{R}^2$ of an image is a texture at scale $\sigma > 0$ if there exist regions $\omega \subset \bar{\omega} \subset \Omega$ such that I is a realization of a stationary (Eq. 1), ergodic (Eq. 2), Markovian process (Eq. 3) locally within Ω, with $I(\omega)$ a Markov sufficient statistic and $\sigma = |\bar{\omega}|$ the stationarity scale.

If the group G is non-trivial, we say that a region Ω is a texture relative to the group G at scale $\sigma > 0$ if there exists a group element $g \in G$ such that $I \circ g^{-1}$ is a texture relative to the translation group. Then, the Markov sufficient statistic is $I \circ g^{-1}(\omega)$ and the stationarity scale $\sigma / |J_g|$, where the denominator is the determinant of the Jacobian of G computed at g. The group element g can be found by *canonization* [58]; for the specific case of the projective group, [75] provides a simple rank-minimization-based procedure (Fig. 2).

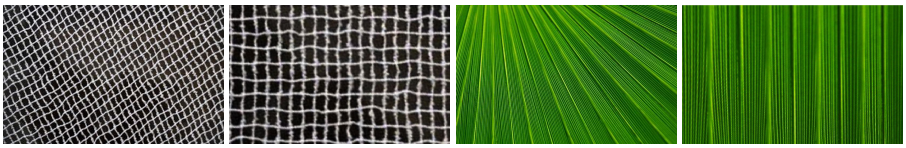

Figure 2: *Affine and projective textures and their rectified versions. The transformation g can be determined in pre-processing via canonization [75, 58], or can be described to parametrize the statistic ϕ_ω and inferred as part of the compression process (i.e. in the search for ω).*

2.3.1 Characterization

Let us assume, for the moment, that the group G is trivial (planar translations). Recall that the definition of the Markov sufficient statistic implies that ω is such that, $\forall A \subset \Omega$,

$$I(A) \perp \underbrace{I(\Omega - A)}_{\text{“outside''}} \mid \underbrace{I(\mathcal{N}_\omega(A))}_{\text{“inside''}} \tag{4}$$

or, in terms of Kullback-Liebler divergence between conditional distributions:

$$\mathbb{KL}\left(p(I(A)|\mathcal{N}_\omega(A)); p(I(A)|\Omega - A)\right) = 0 \tag{5}$$

and yet again in terms of conditional entropy, $H(I(A)|\mathcal{N}_\omega(A)) = H(I(A)|\Omega - A)$. Without a complexity constraint, there are many regions ω that do so; we therefore seek for the *smallest* one, by solving

$$\hat{\omega}(\beta) = \arg\min_\omega H(I(A)|\mathcal{N}_\omega(A)) + \frac{1}{\beta}|\omega|. \tag{6}$$

Note that this is a consequence of the Markovian assumption; it can be shown that the solution $\hat{\omega}(\beta)$ to (6), which can be seen as a version of the Information Bottleneck principle [65], converges to the sufficient statistics ω with β “large enough” (say $\beta \to \infty$). As a special case, we can choose ω to belong to a parametric class of functions, for instance square neighborhoods of x, excluding x itself, of a certain size σ, $\mathcal{B}_\sigma(x)$, so the optimization above is only with respect to the positive scalar σ. In practice, we do not know the probabilistic description of the random field, so the best we can do is to approximate the entropy in (6) from sample data $H(I(A)|\mathcal{N}_\omega(A)) \simeq -\frac{1}{M}\sum_{i=1}^M \log p(I(A_i)|\mathcal{N}_\omega(A_i))$, where $\mathcal{N}(A_i)$ is a neighborhood of $A_i \doteq A + T_i \subset \Omega$. Here p can either be finitely parameterized or specified in a non-parametric fashion by samples in a region $\bar{\omega}$, with $\omega \subset \bar{\omega} \subset \Omega$. For instance, given $\bar{\omega}$ we can draw K regions of size $r = |\omega|$. The larger the r, the smaller the K, so we can write $K = K(r, \sigma)$ with $\sigma = |\bar{\omega}|$. For instance, if $\bar{\omega}$ is a square neighborhood of side σ, then $K = 4r\sigma - 4r^2 + 1$. To compute $\log p(I(A)|\mathcal{N}_\omega(A))$, one can “synthesize” the image in the set A, given fixed values of $I(\mathcal{N}_\omega(A))$ which can be done by a nonparametric texture synthesis algorithm (see e.g. [14] and section 2.4.2) for fixed ω and $\bar{\omega}$. If we call $\hat{I}(A)$ the “synthesized” texture in A, this yields an estimator of the entropy $\hat{H}(I(A)|\mathcal{N}_\omega(A_i)) \doteq \log\left(\frac{1}{M}\sum_{i=1}^M d(I(A_i), \hat{I}(A_i))\right)$. Note that this function depends

on both $r = |\omega|$ as well as $\sigma = |\bar{\omega}|$. The larger $\bar{\omega}$ the better the estimate, so we must trade off σ. However, the size of ω is automatically traded off in $K(r, \sigma)$: Choosing $r = \sigma$ will yield only one sample $K = 1$, and therefore the prediction error $d(I(A), \hat{I}(A))$ will be large. Similarly, too small r will cause many false matches of $I(\omega - \{x\}))$ with poor predictive power for $I(x)$. The tradeoff will naturally settle for $1 < r < \sigma$. Therefore, we can simultaneously infer both σ and r by minimizing the sample version of (6) with a complexity cost on $\sigma = |\bar{\omega}|$:

$$\hat{\omega}, \sigma = \arg \min_{\omega, \sigma = |\bar{\omega}|} \hat{H}(I(A)|\mathcal{N}_\omega(A)) + \frac{1}{\beta}|\bar{\omega}|. \tag{7}$$

Note that both ω and $\bar{\omega}$ will be necessary for extrapolation: ω defines the Markov neighborhood used for comparing samples, and $\bar{\omega}$ defines the region where such samples are sought to approximate the probability distribution $p(I(A)|\mathcal{N}_\omega(A))$.

2.4 Inference

Eq. (7) provides means to infer both the Markov sufficient statistic $I(\hat{\omega})$ as well as the scale $\sigma = |\bar{\omega}|$ of a texture, assuming that the stationarity and ergodicity assumptions are satisfied. Testing for stationarity (ergodicity must be assumed and cannot be validated) amounts to inferring Ω, a *texture segmentation* problem. To do so, we must first provide a procedure for extrapolation or "texture inpainting". An extrapolation algorithm is also described. Below we provide the algorithms developed for these procedures.

2.4.1 Compression

The definition of texture in terms of MRF presents a challenge for compression, since the inference of ω requires a search over all possible subsets of Ω and its separators (Remark 1). To infer ω in a computationally viable way, we propose an alternative which is based on [7]. Because of the stationarity assumption, what matters about ω is not its shape, but just its size $|\omega|$. Consider therefore a region of growing size $|\omega_k| = 1, 2, \ldots, n$ and any statistic ϕ computed in ω_k, $\phi(I(\omega_k))$. Its entropy as a function of k will follow a "staircase" behavior [7], where the lower edge of each plateau is $k_i = |\omega|$. This is consistent with the fact that textures exist at multiple scales (see Fig. 3). For the purpose of compression, we are interested in the smallest k_i. This algorithm is just an approximation, as the staircase behavior is implied by stationarity and markovianity, but there may be pathological cases where the entropy of certain statistics can exhibit staircase behavior and yet the region does not satisfy the definition of texture. Finally, given ω we can then infer $\bar{\omega}$ using Alg. 1. This greedy algorithm selects patches of size $|\omega|$ to capture the global variability of the texture.

2.4.2 Extrapolation

Given a compressed representation $I(\bar{\omega})$, we can in principle synthesize novel instances of the texture by sampling from $dP(I(\omega))$ within $\bar{\omega}$. In a non-parametric setting this is done by sampling directly neighborhoods $I(\omega)$ within $\bar{\omega}$. To extrapolate the texture from a given sample $I(\bar{\omega})$ compatibility conditions have to be ensured at the boundaries

Algorithm 1: Algorithm for inferring $\bar{\omega}$

Initialize a set $R = \emptyset$, ϵ
Sample N patches $\{x_i : i = 1, \ldots N\}$ of size $|\omega|$ from Ω
foreach x_i **do**
  Compute $D = d(I(x_i), \hat{I}(x_i))$ where $\hat{I}(x_i)$ is the Nearest Neighbor of $I(x_i)$ among the rest $N - 1$ patches
  if $D < \epsilon$ **then**
    $R := R \cup x_i$
Let $\bar{\omega}$ be a sampled region from Ω of size $|R|$

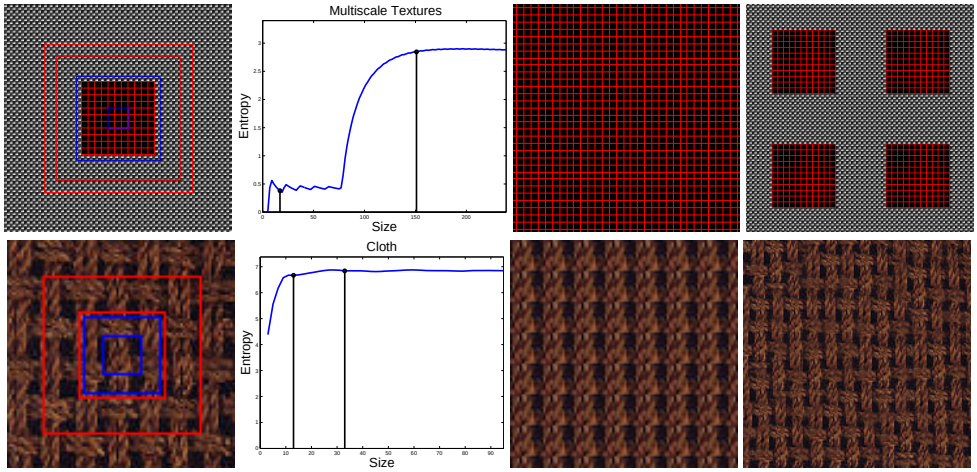

Figure 3: *Multiscale analysis of textures. Top row, left to right: Texture "within" texture. Entropy plot. Synthesized texture at small scale, synthesized texture at a higher scale. Bottom row: Different textures appearing at different scales. The regions surrounded by the blue rectangles are the textures at the smaller scale (shown in the entropy plot) and the regions surround by the red rectangles are the textures at the larger scale. For each scale we show both ω and $\bar{\omega}$ (with the bigger rectangle of each color corresponding to the respective $\bar{\omega}$). It can be seen that the smaller scale legitimately captures the texture of a single rope thread, but fails to capture the texture of the rug that consists of woven threads. That is captured by the larger region (right).*

of $\bar{\omega}$. Hence, to satisfy both appearance and compatibility conditions, we minimize the following energy function [36]:

$$E(\hat{I}_s; I(\bar{\omega})) = \sum_{\tilde{\omega} \in \hat{\Omega}} \|\hat{I}_{\tilde{\omega}} - I_{\tilde{\omega}}\|^2 \tag{8}$$

where $\hat{I}_s$ is the textured region to be synthesized and $I(\bar{\omega})$ is the input texture. The vectors $I_{\tilde{\omega}}$ and $\hat{I}_{\tilde{\omega}}$ are neighborhoods of the input and synthesized textures respectively, centered at the central pixel of $\tilde{\omega}$'s neighborhood. We sample $\tilde{\omega}$ on a grid every $s = \frac{n_i}{4}$, where n_i is the cardinality of $I_{\tilde{\omega}}, \hat{I}_{\tilde{\omega}}$ (more on this later). We let $\hat{\Omega}$ denote the collection

of these $\tilde{\omega}$. The Algorithm used to minimize the above energy function is given in Alg. 2. Additionally, the process is performed in a multi scale fashion. Note that in [36] the scales were set manually. In our scheme we select them automatically based on the size of ω inferred by our algorithm. We repeat the procedure over 3 neighborhood sizes: $\{|n_i| : i = 1, 2, 3\} = [\omega, \frac{\omega}{2}, \frac{\omega}{4}]$. Finally, the minimization procedure is repeated over a number of different output image sizes.

Algorithm 2: Texture Extrapolation

Initialize $\hat{I}_s^{(1)}$ to a random texture and sample $\hat{I}_\omega^{(1)}, \forall \omega \in \hat{\Omega}$
for $i = 1, \ldots, N$ **do**
  Let $I_\omega^{(i+1)}$ be the Nearest Neighbor of $\hat{I}_\omega^{(i)}$
  Update $\hat{I}_s^{(i+1)} = argmin_{\hat{I}_s} E(\hat{I}_s; I(\bar{\omega}))$
  Resample $\hat{I}_\omega^{(i+1)}, \forall \omega \in \hat{\Omega}$
  if $\hat{I}_\omega^{(i+1)} = \hat{I}_\omega^{(i)}, \forall \omega \in \hat{\Omega}$ **then**
    break;

2.4.3 Segmentation

Given an image I we want to find Ω_i for the different textures in the image. The algorithm (Alg. 3) requires knowledge of the number of regions to be segmented (model selection) that can be determined using Agglomerative Information Bottleneck (AIB) [56]. Note that it is possible for a texture region to not have a well-defined boundary. In that case, boundary compatibility conditions have to be imposed with other structures in the image, as we discuss in Sect. 2.5.

Algorithm 3: Segmentation algorithm.

Initialize $N = 12, M = 30, K$
Sample (overlapping) square patches ω of sides N on a dense grid from the image
Let $\{\omega\}$ be the set of sampled ω's from previous step
foreach ω **do**
  Calculate a histogram of intensity values with M bins for each patch
Calculate an empirical probability $p(M|\{\omega\})$
Use AIB [56] to sequentially merge ω's that that decrease $I(\{\omega\}, M)$ the least
Cut the tree built by AIB in K clusters to obtain K-clustering of the ω's
foreach *pixel* x_i **do**
  Calculate in which ω's, x_i falls in
  Find which cluster K, x_i belongs to, using a max vote of the memberships of the ω's it falls in
Form regions Ω_i based on clustering of pixels

Note that in some cases a texture may not have a well-defined boundary, as the statistics can vary spatially in a smooth fashion that violates the stationarity assumption.

2.5 Textures and Structures

In this section we establish the relation between textures, defined in Sect. 2.3 and structures, defined in Sect. 2.1.1. Consider a point $x \in D$ and its neighborhood. If it is canonizable at a scale ϵ, there is a co-variant detector with support ϵ (a statistic) that has an isolated extremum. This implies that the underlying process is not stationary at the scale $|\bar{\omega}| = \epsilon$. Therefore, it is not a texture. It also implies that any region ω of size $\epsilon = |\omega|$ is not sufficient to predict the image outside that region. This of course does not prevent a region that is canonizable at ϵ to be a texture at a scale $\sigma >> \epsilon$. Within a region σ there may be multiple frames of size ϵ, spatially distributed in a way that is stationary/Markovian. Vice-versa, if a region of an image is a texture with $\sigma = \bar{\omega}$, it cannot have a unique (isolated) extremum within $\bar{\omega}$, lest it would not be a sample of a stationary process. Of course, it could have multiple extrema, each isolated within a region of size $\epsilon << \sigma$. The above argument is a sketch of a proof of the following:

Theorem 1 (Structure-Texture) *For any given scale of observation σ, a region $\bar{\omega}$ with $|\bar{\omega}| = \sigma$ is either a structure or a texture.*

Hence one can detect textures for each scale, as the residual of the canonization process described in Sect. 2.1.1. One may have to impose boundary conditions so that the texture regions fill around structure regions seamlessly. This can be accomplished by an explicit generative model that enforces boundary conditions and matches marginal statistics in the texture regions, following , as illustrated by [77]. However, this has to be done across all scales, unlike [77], since whether a region is classified as texture or structure depends critically on the scale σ. Since the decision of what is a structure depends on time (or, more in general, on the presence of multiple images), consequently, what is a texture is also a decision that requires multiple images. For instance, the random dot stereogram is everywhere canonizable and properly sampled at some scale σ, since one can establish correspondence. It is, however, a texture at all coarser scales, where any statistic is translation invariant.

2.6 Experiments

Compression and extrapolation. We show results of our texture compression algorithm in Fig. 5. In the odd columns we show the input textures (Ω is the entire image domain). Within Ω we show ω and $\bar{\omega}$ inferred by our algorithm by indicating their boundaries with red boxes. On the even columns we show the synthesized textures from $\bar{\omega}$ using our extrapolation algorithm. Qualitatively, the original textures are successfully re-synthesized, which shows that $\bar{\omega}$ is sufficient to capture the characteristics of that texture within the threshold used for inference. To determine $|\omega|$ we calculate the sampled entropy at each scale and we use an automatic scale selection algorithm that determines the lower edge of the plateau. We calculate, μ_e, the mean value of entropy in the last k scales (in these experiments $k = 10$) and the standard deviation, s_e, of the entropies calculated at all scales. We set a threshold $\lambda = \mu_e - \frac{s_e}{6}$ and look for the smallest scale at which the entropy exceeds λ.

To determine $\bar{\omega}$, according to Sec 2.4, we need to accept "representatives" that are "close" to each patch sampled from Ω. In these experiments, we sample 3000 patches

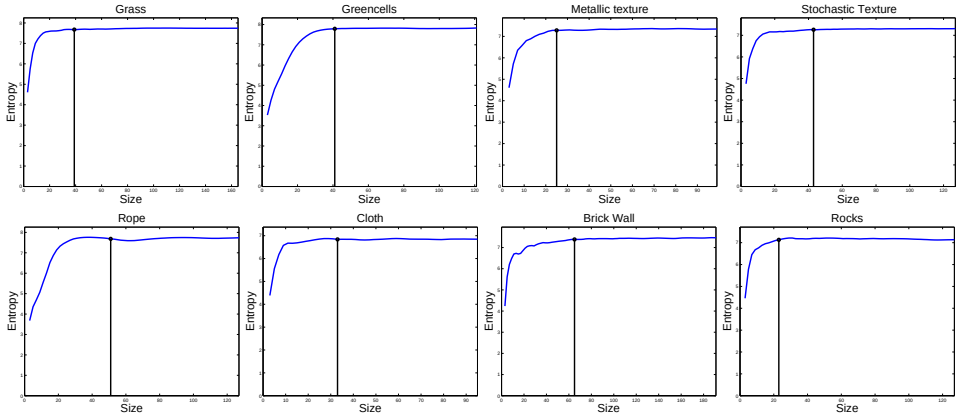

Figure 4: *Entropy plots for the* 8 *textures shown in Fig 5. The black line indicates the scale (specifically the size of the side) of* ω *selected by our algorithm.*

from Ω (which is related to the parameter $\theta = \theta(|\bar{\omega}|)$ mentioned in Sec. 2.1.2) and accept as a representative any patch that is less than 3×10^{-3} (per pixel distance) away from its nearest neighbor.

In Fig. 4 we show the entropy plots of the histograms of pixel values for the same textures. In black we show the location detected by our algorithm as the lower edge of the plateau. These sizes correspond to the small red boxes in Fig 5. Although the histogram is a vey crude first-order statistic, it exhibits the anticipated behavior, and is sufficient to capture the scaling properties of the texture.

To further illustrate the multi scale nature of textures, we calculate the entropy of the intensity values as a function of increasing size of ω for a synthetic texture (Fig. 3). A synthetic configuration of red lines on black background is surrounded by another texture exhibiting different spatial characteristics. The entropy of this image is shown in Fig. 3. It can be observed that two plateaus are formed, one corresponding to the first texture and the second corresponding to the combination of the two textures. Synthesizing at these two scales, one can generate different types of textures. Another example is shown in the bottom row of the same figure. Here, the same texture is exhibiting different repetitive patterns at different scales. The synthesized textures at these two different scales (indicated in the entropy plot) are shown on the bottom right.

As expected, the quality of the synthesized texture depends critically on the size of $\bar{\omega}$, and so does storage cost. In the next experiment, we attempt to characterize such a "rate-distortion" tradeoff. Distortion is not easy to measure since the goal is to create samples that are perceptually indistinguishable, but could differ significantly at the pixel level. Therefore, standard distortion figures such as PSNR are of no use. Standard perceptual similarity scores, such as SSIM [71], similarly fall short of capturing the perceived quality of the synthesized textures (see Fig. 6). We have therefore filter the synthesized and original textures by a filter bank [38]; then, we calculate the histograms of the filter responses and used the χ^2 distance to measure the distance between each filter response of the two textures. We take the texture (dis-)similarity to be the averaged

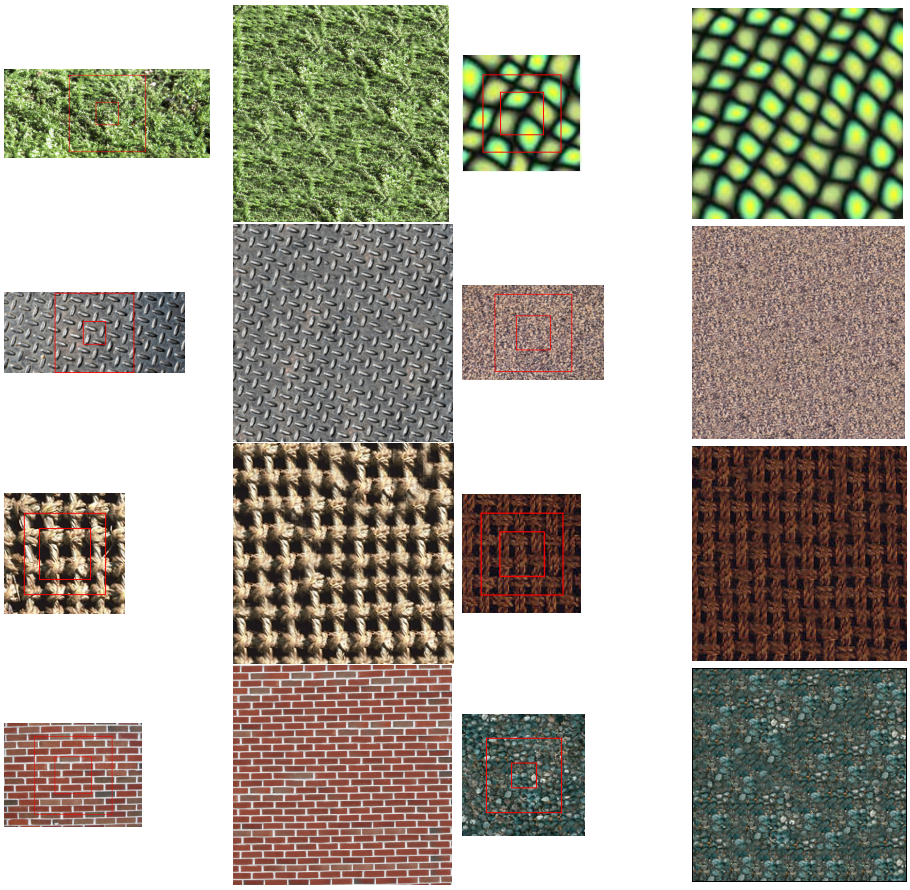

Figure 5: *Odd Columns: Input texture. The large red box indicates the inferred scale of $\bar{\omega}$. The smaller red box indicates the inferred scale of ω. Even Columns: Synthesized textures from $\bar{\omega}$. The perceptual characteristics of the textures have been captured, indicating that $I(\bar{\omega})$ is indeed a Markov sufficient statistic, at least sufficient for the purpose of perceptual comparison.*

result over all distances for each histogram pair (see Fig. 6). The distance decreases as the size of $\bar{\omega}$ increases indicating that the input and synthesized textures are becoming increasingly similar.

Fig. 7 illustrates the role of the group G in compression. The texture is compressed and re-synthesized without canonization of the rectifying homography. This shows that the Markov sufficient statistic is fairly large. Compressing the rectified texture, and then transforming the synthesized texture by the inverse of the canonizing transformation, yields a perceptually similar reconstruction at a smaller coding cost, even after accounting for the 8 numbers necessary to encode the homography.

In terms of computational complexity, for the experimental setup discussed here, inferring ω takes around 1.15 seconds; inferring $\bar{\omega}$ takes around 89 seconds, and synthe-

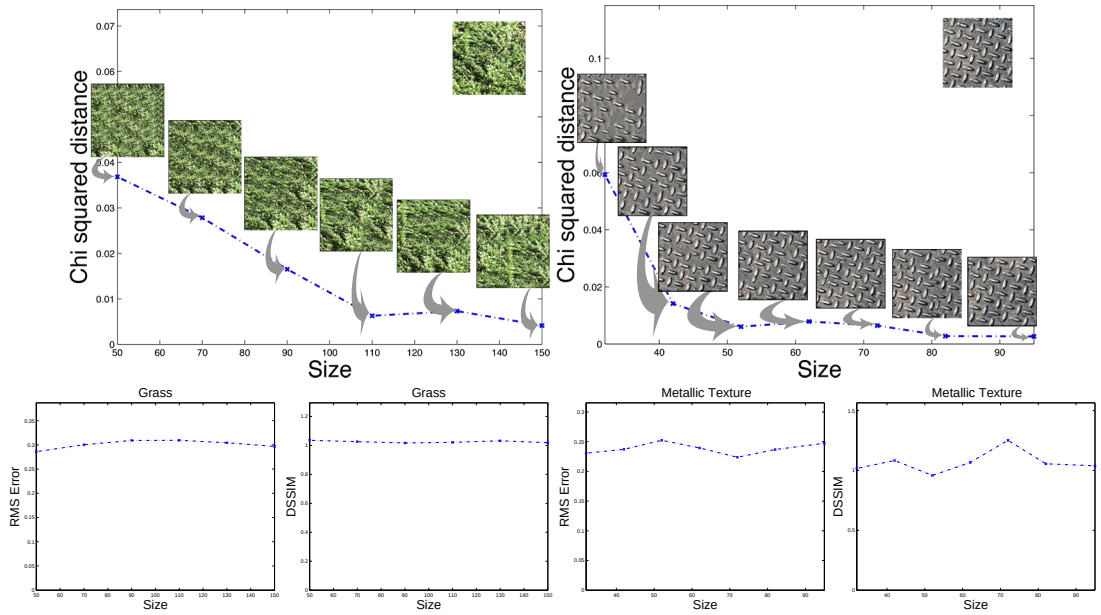

Figure 6: *Rate-Distortion curves. Top: Mean distance of filter response histograms against $\bar{\omega}$ size. At each point, we show the synthesized texture given that scale of $\bar{\omega}$. At the top right we show the original texture. The qualitative behavior, as expected, indicates that larger size of $\bar{\omega}$ yields synthesized textures that are increasingly similar to the original sample. Bottom: Plots of RMS Error and DSSIM [71] a function of the size of $\bar{\omega}$. Standard metrics used for measuring the fidelity of a reconstructed image fail to capture the perceptual quality.*

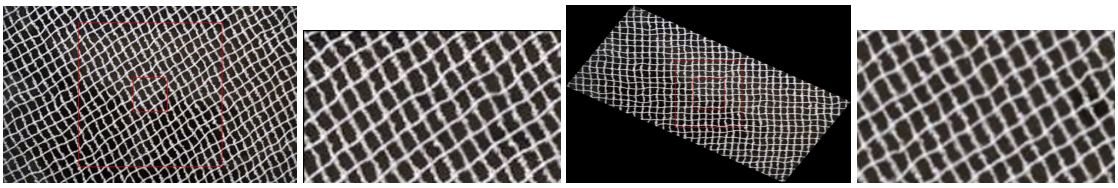

Figure 7: *The texture in Fig. 2 is compressed and re-synthesized without prior rectification. (first and second figures). The texture is then rectified, compressed, re-synthesized and retransformed back with the inverse of the canonizing transformation (third and fourth figures). The two approaches achieve approximately the same perceptual quality but the rectified texture does so at a lower complexity cost ($|\bar{\omega}_{rectified}| \simeq |\frac{\bar{\omega}_{original}}{4}|$).*

sizing the texture at a size of 256×256 takes around $2 - 3$ minutes. The computational time to infer $\bar{\omega}$ depends on θ. The more patches sampled, the slower it is, but given that there are fast methods for finding Nearest Neighbors, this can be done efficiently.

Segmentation. We test Alg. 3 on six images shown in Fig. 8. We manually annotated the ground truth. We provided the number of regions to segment, although automatic selection would have been possible (Sec. 2.4). We evaluate our algorithm using the

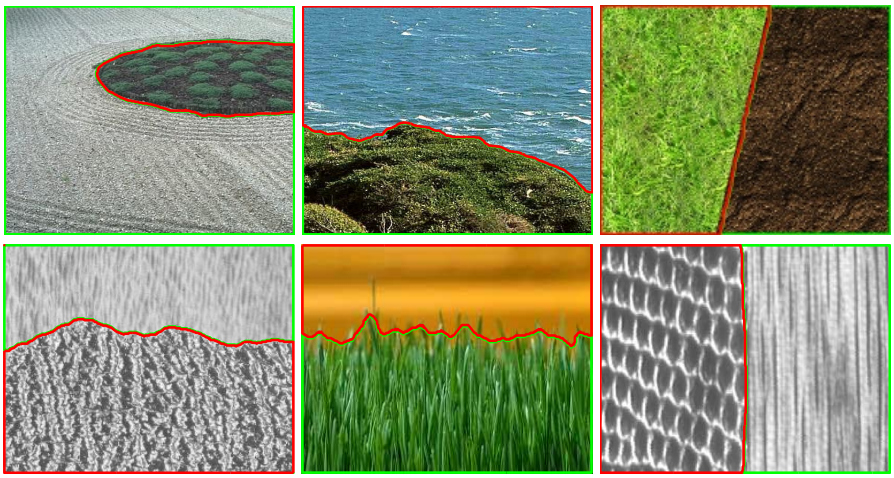

Figure 8: *Texture segmentation. The green and red colors show the boundaries of the regions detected. Our algorithm successfully obtains pixel-precision segmentation results.*

	Image1	Image2	Image3	Image4	Image 5	Image 6
Ours	0.8152	0.9844	0.9789	0.9612	0.9275	0.9776
[2]	0.8063	0.9667	0.9932	0.9834	0.7545	0.4975

Table 1: *Texture segmentation. Quantitative comparison of our method against [2] based on the Segmentation Covering metric. We outperform [2] in 4 out of 6 images and perform comparably in the other two.*

segmentation covering metric [2]: to compare a segmentation S with S' by $\mathcal{C}(S,S') = \frac{1}{N}\sum_{R\in S}|R|max_{R'\in S'}\mathcal{O}(R,R')$ [1], where N denotes the total number of pixels in the image. $\mathcal{O}(R,R')$ is the overlap between two regions R, R' and is given by $\mathcal{O}(R,R') = |R\bigcap R'|/|R\bigcup R'|$. The results are shown in Table 1. The reported results for [2] were obtained by varying the scale to obtain the highest possible score per image. It can be seen that even though our algorithm is significantly simpler and faster it outperforms [2]. The average runtime for images of size 256×256 was 127 seconds.

Structure/Texture partition. We show examples of structure/texture partition in Fig. 9. Isolated extremums are detected using [37]. It can be observed that structures and textures, at some scale σ, are indeed complimentary within an image.

The runtimes reported for all experiments refer to our non-optimized implementations in MATLAB for an INTEL 2.4 GHz dual core processor machine.

2.7 Discussion

We have presented a novel definition of texture and characterized its relation to other low-level visual processing operations. The definition encompasses most of the existing intuitive notions of texture. We have left out so-called "quasi-regular" textures.

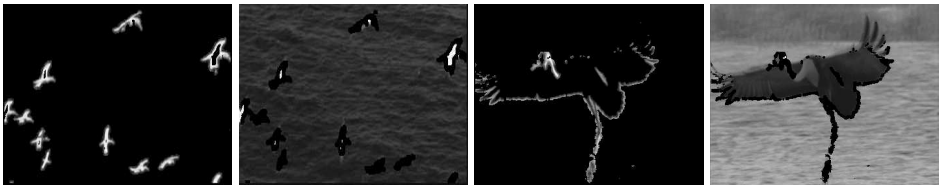

Figure 9: *Structure/Texture partition: (1st, 3rd) structured regions, (2nd, 4th) textured regions.*

However, this should just be a special case of the definitions above, where the conditional entropy implied in the definition of Markov (approximate) sufficient statistic is small, so qualitatively the sample looks almost periodic. The case of "deterministic" or "regular" textures is a degenerate case where all the spatial distributions are deltas, and one is left with determining the minimal sufficient statistic (now $\omega = \bar{\omega}$), and the group G, that now could include various symmetry groups of the plane (the "periodicity") [42].We have shown how the definition lends itself to the specification of a number of algorithms for texture analysis, including compression, extrapolation and segmentation.

3 Encoding Scene Structures for Video Compression

We describe an approach to partition a video stream into *structure* regions that are temporally encoded and disjoint from *texture* regions, that are synthesized so as to preserve the statistical properties of the original data stream. Structures encode regions of an image that can be put into correspondence in different images of the same scene, and are encoded via a dictionary that takes into account spatial and temporal regularities. Textures are synthesized in a manner that preserves perceptual similarity.

Our work is naturally related to video compression schemes as reflected in standards such as H.262, H.263, H.264 [31]. While standard methods perform both spatial and temporal prediction of the measured signal, they rarely model the data formation process explicitly in terms of the underlying *scene.* We do not advocate explicit reconstruction of the underlying scene; however, we describe "structures" in images as a function of corresponding structures in space, that under suitable conditions can be inferred from temporal correspondence [19]. Thus, "trackable" regions can be temporally encoded making use of optical flow whereas non-trackable regions can be encoded spatially. The residual of the encoding process can then be encoded as standard. Another approach is followed by [76] where the frames are divided into four types of regions: sketchable and trackable, non-sketchable and trackable, sketchable and non-trackable and non-sketchable and non-trackable. Each type of region is encoded individually taking advantage of its properties.

In this paper, we propose a sequential approach whereby we first detect regions of the image that contain *structures* and are *properly sampled* (Section 3.1). We then exploit the complementarity of *structures* and *textures* [19] to encode the remainder of the image by performing texture segmentation (Section 3.3.3). We then infer a compressed representation of the textures (Section 3.3.1) that enables us to synthesize a perceptually similar realization at decoding (Section 3.3.2). This approach will suffer in traditional evaluation metrics, but is designed to yield perceptually equivalent encoding/decoding at smaller complexity. To encode the structure regions, we create a dictionary by taking into account the spatial and temporal components of such regions. For textured regions we store the compressed representation of each texture (Section 3.4).

3.1 Preliminaries

To make the paper self-contained, this section summarizes the notation and definitions of [19]. Please refer to [19] for further information. Images $\{I_{ij}\}_{(i,j)=1:(N,M)} \in \mathbb{R}^{M\times N}$ are quantized versions of functions $I : D \subset \mathbb{R}^2 \to \mathbb{R};\ x \mapsto I(x)$ where $\mathcal{B}$ is a neighborhood of $x_{ij} \in D$ of size $\epsilon > 0$. Groups $g : D \subset \mathbb{R}^2 \to \mathbb{R}^2; x \mapsto g(x)$ induce actions $I \circ g \doteq I(g(x))$, for instance translation $T \in \mathbb{R}^2$, via $g(x) \doteq x + T$. Some transformations undergone by (subsets of the domain and range of) an image over time can be represented as groups, for instance translation, Euclidean, similarity, affine, projective, all the way to the diffeomorphic group. This is relevant to compression because the entire orbit of a group transformation can be encoded by a single element of the group. The process of choosing such a "canonical" element is called *canonization* and described in detail in [58]. Succinctly, this is done through the use of a *co-variant detector*, a functional $\psi : \mathcal{I} \times G \to \mathbb{R}; (I, g) \mapsto \psi(I, g)$ that

has isolated extrema (loci where $\nabla\psi = 0$) that co-vary with the group, so if $g = \hat{g}(I)$ is such that $\nabla\psi(\hat{g}(I)) = 0$, then $\hat{g}(I \circ g) = (\hat{g} \circ g)(I)$. Examples of (translation-) co-variant detectors are the Laplacian-of-Gaussian (LoG), the difference-of-Gaussians (DoG), the Hessian-of-Gaussian (HoG), the Harris' corner and its variants. Because images are not differentiable, co-variant detectors are typically regularized operators with a scale parameter σ. 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. Note that the same region may be canonizable at multiple scales. The canonization process yields a number of regions each containing exactly one "structure," that is an extremum with respect to the chosen group at the given scale. Such structures may be "real," in the sense of corresponding to some property of the *scene*, or they may be "aliases" due to sampling artifacts, noise, and other phenomena that do not depend on the scene. An image is *properly sampled* at a scale σ if any co-variant detector functional operating on the sampled image $\{I_{ij}\} \in \mathbb{R}^{N \times M}$ at scale σ yields the "same answer" (topology) that it would if run on the "original" (continuous) image $I : D \to \mathbb{R}^+$.

In other words, proper sampling indicates *topological equivalence* of the response of co-variant detectors between the "original" and the sampled image. Unfortunately, we do not have the "original" image. However, under three assumptions (co-visibility, Lambertian reflection and constant illumination), topological equivalence of co-variant detectors responses between the scene and the image can be replaced by topological equivalence of co-variant detector responses between *different images of the same scene* [58]. Testing for proper sampling entails establishing correspondence of covariant selection trees [37].

3.2 Temporal redundancy

Proper sampling yields as a byproduct a partition of the image(s) into two regions. Those for which unique correspondence can be established, and the rest. We call the former ones *trackable* regions. They are both canonizable *and* properly sampled. Trackable regions are characterized by the "signature" of each region at the finest 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 instance position, orientation and scale for the case a similarity reference frame.

To determine trackable regions, we use [37], that requires a detection threshold. The effects of such a threshold are visible in Figure 10, where the number of tracks decreases by increasing the threshold. The ones that persist are usually the most accurate. One can see that almost all trajectories on the sea are eliminated. In later sections we will argue that those regions are spatially stationary and will exploit this property for compression. Trackable regions exhibit temporal redundancy and we would like to exploit that instead.

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, and there is no choice that is best for any sequence. For the purpose of illustration, in what follows we will 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

Figure 10: *Varying the co-variant detection threshold produces different numbers of trackable regions. By looking at the number of tracks produced as a function of the threshold it can be observed that there are essentially three regions of interest. The first region corresponds to the tracked points shown in the leftmost image. These tracks contain highly unstable trajectories that provide limited compression benefit. The second region corresponds to the central image Here the quality of the tracks is higher as extended tracks provide improved temporal redundancy. Finally, we have the third case where the number of tracks is significantly lower, but the tracks are much more stable and of higher quality.*

location-scale group.

Tracking then provide 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, no change will be observed in the moving frame. Most typically, however, similarities are not sufficient to explain the complexity of the image even in a small neighborhood and therefore the region will progressively change over time.

A simple approach to encode such changes is to create a dictionary element for each region of the same size as its neighborhood that will be time invariant i.e. constant. The best possible representation would be the average of the pixel values over time. In mathematical notation, if $F(t) \doteq \{I(x,t), x \in \mathcal{B}_\sigma(t)\}$ are the pixel values in an neighborhood $\mathcal{B}$ at scale σ at time t, we represent a trackable region in a dictionary as $T = \frac{1}{N_2 - N_1} \sum_{t=N_1}^{N_2} F(t)$ where N_1, N_2 are the first and last frames that track appears. Hence the scale of the dictionary element is naturally selected to be the finest scale at which the track was detected and its temporal appearance is averaged out to further improve compression rates. Hence the dictionary is composed of elements of different spatial resolutions resulting in a multi scale representation of the trackable regions.

3.3 Spatial redundancy

We exploit spatial redundancy by encoding stationary regions through their sample statistics. We consider three regions: $\omega, \bar{\omega}$ and Ω. The generator $\omega \subset \mathbb{Z}^2$, with cardinality $|\omega|$, is the smallest region where some statistic ϕ_ω (*"features"*) is defined. A probability distribution $dP(I)$ on the set of images induces a distribution on the feature via the map ϕ_ω: $dP(\phi_\omega) = dP(\phi(I_{|_\omega}))$, that is *G-stationary* if there exists $g \in G$ such that $\mathbb{E}(\phi_{g(\omega)})$ is translation-invariant:

$$\mathbb{E}(\phi_{g(\omega)}) = \mathbb{E}(\phi_{g(\omega)+T}), \quad T \in \mathbb{R}^2 \tag{9}$$

where $g(\omega) = \{g(x) \mid x \in \omega\} \cap \mathbb{Z}^2$ and $g(\omega) + T = \{g(x) + T \mid x \in \omega\} \cap \mathbb{Z}^2$. In order to perform an empirical test, we need a larger region $\bar{\omega} \supset \omega$ where to aggregate the statistics. I is locally stationary in $\bar{\omega}$ if (9) is satisfied *not* for all $T \in \mathbb{R}^2$, but only for

those such that $g(\omega)+T \subset \bar{\omega}$. We call such T's *admissible*, and $\sigma = |\bar{\omega}|$ the stationarity scale. The region Ω is the *largest* admissible region where the stationarity assumption is satisfied. In order to perform a stationarity test based on empirical data we rely on the assumption that the underlying process is ergodic, that is

$$\frac{1}{N}\sum_{i=1}^{N} \phi_{g(\omega)}(I_{|_{g(\omega)+T_i}}) \xrightarrow{\text{a. s.}} \mathbb{E}(\phi_{g(\omega)}(I_{|_{g(\omega)}})) \tag{10}$$

for all $T_i \in \mathbb{R}^2$. Assuming that the conditions above are satisfied, stationarity can be tested by considering samples of the image I in $\bar{\omega}$, and comparing them to admissible samples in a transformed domain $\bar{\omega} + \bar{T}$.

Once established that a process is stationary, hence spatially predictable, we can inquire on the existence of a statistic that is *sufficient* to perform the prediction. We say that a process is Markovian if every position $x \in \Omega$ admits a neighborhood $\omega \ni x$, such that a statistic ϕ_ω computed "inside" ω (after excluding x, lest the statement would be a tautology) makes $I(x)$ independent of the "outside" $I_{|_{\omega^c}}$, where ω^c is the complement of ω in Ω:

$$I(x) \perp I_{|_{\omega^c}} \mid \phi_{\omega \setminus x}. \tag{11}$$

Equation (11) establishes $I_{|_\omega}$ as a *sufficient statistic*. In general, there will be many regions ω that satisfy this condition; the one with the smallest area $|\omega| = r$, is a *minimal sufficient statistic*. From now on, we will refer to ϕ_ω as the minimal *Markov sufficient statistic*.

With all the notions introduces thus far, we can define a texture as *a region of an image that can be rectified into a sample of a stochastic process of a planar lattice that is locally stationary, ergodic and Markovian [19].*

More precisely, assuming for simplicity the trivial (translation) group, a region $\Omega \subset D \subset \mathbb{R}^2$ of an image is a texture at scale $\sigma > 0$ if there exist regions $\omega \subset \bar{\omega} \subset \Omega$ such that I is a realization of a stationary (Eq. 9), ergodic (Eq. 10), Markovian (Eq. 11) process locally within Ω, with $I_{|_\omega}$ a Markov sufficient statistic and $\sigma = |\bar{\omega}|$ the stationarity scale. In this manuscript we will restrict our attention to the case where G is the location-scale group.

In order to infer the description of a texture, we therefore look for Markov sufficient statistics. Without a complexity constraint, there are many regions ω that satisfy the conditions; we therefore seek for the *smallest* one, by solving

$$\hat{\omega} = \arg\min_{\omega} H(I(x)|\omega - \{x\}) + \frac{1}{\beta}|\omega|. \tag{12}$$

Note that this is a consequence of the Markovian assumption and the resulting Markov sufficient statistic satisfies the Information Bottleneck principle [65] with $\beta \to \infty$.

In practice, we do not know the probabilistic description of the random field, so the best we can do is to approximate the entropy in (12) from sample data:

$$H(I(x)|\omega - \{x\}) \simeq -\frac{1}{|\Omega|}\sum_{i=1}^{|\Omega|} \log p(I(x_i)|\omega_i - \{x_i\}) \tag{13}$$

where ω_i is a neighborhood of x_i. Here p can either be finitely parameterized or specified in a non-parametric fashion by samples in a region $\bar{\omega}$, with $\omega \subset \bar{\omega} \subset \Omega$. For instance, given $\bar{\omega}$ we can draw a number K of regions of size $r = |\omega|$. The larger r, the smaller K, so we can write $K = K(r, \sigma)$ with $\sigma = |\bar{\omega}|$. For instance, if $\bar{\omega}$ is a square neighborhood of side σ, then $K = 4r\sigma - 4r^2 + 1$.

To compute $\log p(I(x)|\omega - \{x\})$, we search among $k = 1, \dots, K(r, \sigma)$ for the region ω_k that best matches a neighborhood ω of x:

$$\hat{k} = \arg\min_{k=1:K} d(I(\omega_k - \{x_k\}), I(\omega - \{x\})) \tag{14}$$

where d is some distance among images restricted to their neighborhoods. If we call $\hat{I}(x) = I(x_{\hat{k}})$, we approximate $-\log p(I(x)|\omega - \{x\}) \simeq d(I(x) - \hat{I}(x))$

which yields an estimator of the entropy:

$$\hat{H}(I(x)|\omega - \{x\}) \doteq \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} d(I(x_i), \hat{I}(x_i)). \tag{15}$$

Note that this function depends on both $r = |\omega|$ as well as $\sigma = |\bar{\omega}|$. The larger $\bar{\omega}$ the better the estimate, so we must trade off σ. However, the size of ω is automatically traded off in $K(r, \sigma)$: Choosing $r = \sigma$ will yield only one sample $K = 1$, and therefore the prediction error $d(I(x) - \hat{I}(x))$ will be large. Similarly, too small a σ will cause many false matches of $I(\omega - \{x\}))$ with poor predictive power for $I(x)$. The tradeoff will naturally settle for $1 < r < \sigma$. Therefore, we can simultaneously infer both σ and r by minimizing the sample version of (12) with a complexity cost on $\sigma = |\bar{\omega}|$:

$$\hat{\omega}, \sigma = \arg\min_{\omega, \sigma = |\bar{\omega}|} \hat{H}(I(x)|\omega - \{x\}) + \frac{1}{\beta}|\bar{\omega}|. \tag{16}$$

Note that both ω and $\bar{\omega}$ will be necessary for extrapolation: ω defines the Markov neighborhood used for comparing samples, and $\bar{\omega}$ defines the region where such samples are sought to approximate the probability distribution $p(I(x)|\omega - \{x\})$.

Eq. (16) provides means to infer both the Markov sufficient statistic $I(\hat{\omega})$ as well as the scale $\sigma = |\bar{\omega}|$ of a texture, assuming that the stationarity and ergodicity assumptions are satisfied. Testing for stationarity (ergodicity must be assumed and cannot be validated) amounts to inferring Ω, a *texture segmentation* problem. An extrapolation algorithm is also described. Below we provide the algorithms developed for these procedures.

3.3.1 Compression

Given $\{I(x), x \in \Omega\}$, compression is achieved by inferring the (approximate) minimal sufficient statistic ω and the stationarity scale σ by solving (16). Then $I(\bar{\omega})$, for any $\bar{\omega} \subset \Omega$ with $|\bar{\omega}| = \sigma$ is stored. To infer the unknowns, we parametrize both ω and $\bar{\omega}$ to be square neighborhoods. In addition, we approximate $d(I(x_i), \hat{I}(x_i))$ with the KL-divergence between the distributions of pixel values around the neighborhoods of x_i and $x_{\hat{k}_i}$ (Alg. 4 [19]).

Algorithm 4: Compression algorithm.

Initialize $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_N], r = [r_1, r_2, \ldots, r_N]$
foreach σ **do**
    Sample square patches $\bar{\omega}$ of size σ from Ω
    foreach $\bar{\omega}$ **do**
        foreach r **do**
            Sample square patches ω of size r from $\bar{\omega}$
            Compute $\hat{H}(I(x)|\omega - \{x\}) + \frac{1}{\beta}|\bar{\omega}|$
Let $\hat{\omega}, \sigma = \arg\min_{\omega, \sigma = |\bar{\omega}|} \hat{H}(I(x)|\omega - \{x\}) + \frac{1}{\beta}|\bar{\omega}|$

3.3.2 Extrapolation

Given a compressed representation $I(\bar{\omega})$, we can in principle synthesize novel instances of the texture by sampling from $dP(I_{|_\omega})$ within $\bar{\omega}$. In a non-parametric setting this is done by sampling directly neighborhoods $I(\omega)$ within $\bar{\omega}$. To extrapolate the texture from a given sample $I(\bar{\omega})$ compatibility conditions have to be ensured at the boundaries of $\bar{\omega}$. Hence, to satisfy both appearance and compatibility conditions, we minimize the following energy function [36]:

$$E(I_s; I(\bar{\omega})) = \sum_{\omega \in N(B)} \|I_\omega - \hat{I}_\omega\|^2 + \mu \sum_{\omega \in N(B)} \|DI_\omega - D\hat{I}_\omega\|^2 \qquad (17)$$

where I_s is the textured region to be synthesized and $I(\bar{\omega})$ is the input texture. The vectors I_ω and $\hat{I}_\omega$ are neighborhoods of the input and synthesized textures respectively, centered at the central pixel of ω's neighborhood. D is the differentiation operator. In principle, $N(B)$ can be all possible ω neighborhood's within $\bar{\omega}$. For computational complexity reasons, we only consider a subset of them. In [36] neighborhoods are sampled on a grid. We sample at randomly selected locations similar to [19]. The energy above is minimized alternating direct differentiation with respect to $\hat{I}_\omega$ (that yields a linear system of equations) and using the result to determine the set $\{I_\omega\}$ using nearest-neighbor (NN) search. The process is iterated until the energy decrease is negligible, as customary. In addition, a re-weighting scheme is used to improve outlier rejection (similar to iteratively re-weighted Least Squares (IRLS)). Note that in [36] the neighborhood sizes were set manually. In our scheme we select them automatically based on the size of $\bar{\omega}$ calculated using Alg. 4 [19]. We repeat the procedure over 3 neighborhood sizes: $\{|n_i| \ : \ i = 1, 2, 3\} = [r, 2r, 3r]$. Finally, the minimization procedure is repeated over a number of different output image sizes. The relative weight parameter is set to $\mu = 10$.

3.3.3 Segmentation

Given an image I we want to find Ω_i for the different textures in the image. The algorithm (Alg. 5, [19]) requires knowledge of the number of regions to be segmented (model selection) that can be determined using Agglomerative Information Bottleneck

(AIB) [56]. Note that it is possible for a texture region to not have a well-defined boundary. In that case, boundary compatibility conditions have to be imposed with other structures in the image, as we discuss in Section 3.4.

Algorithm 5: Segmentation algorithm.

Initialize $N = 12, M = 30, K$
Sample (overlapping) square patches ω of sides N on a dense grid from the image
Let $\{\omega\}$ be the set of sampled ω's from previous step
foreach ω **do**
    Calculate a histogram of intensity values with M bins for each patch
Calculate an empirical probability $p(M|\{\omega\})$
Use AIB [56] to sequentially merge ω's that that decrease $I(\{\omega\}, M)$ the least
Cut the tree built by AIB in K clusters to obtain K-clustering of the ω's
foreach *pixel* x_i **do**
    Calculate in which ω's, x_i falls in
    Find which cluster K, x_i belongs to, using a max vote of the memberships of the ω's it falls in
Form regions Ω_i based on clustering of pixels

In order to perfect the encoding, we rely on the partition of the image into texture regions and structure regions, that relies on the analysis in [19]. Consider a point $x \in D$ and its neighborhood. If it is canonizable at a scale ϵ, there is a co-variant detector with support ϵ (a statistic) that has an isolated extremum. This implies that at a scale $|\bar{\omega}| = \epsilon$, it is not possible to capture the variability of the texture and, as such, it is not possible to model it as a texture. It also implies that any region ω of size $\epsilon = |\omega|$ is not sufficient to predict the image outside that region.

This of course does not prevent a region that is canonizable at ϵ to be a texture at a scale $\sigma >> \epsilon$. Within a region σ there may be multiple frames of size ϵ, spatially distributed in a way that is stationary/Markovian. Vice-versa, if a region of an image is a texture with $\sigma = \bar{\omega}$, it cannot have a unique (isolated) extremum within $\bar{\omega}$, lest it would not be a sample of a stationary process. Of course, it could have multiple extrema, each isolated within a region of size $\epsilon << \sigma$.

Thus for any given scale of observation σ, a region $\bar{\omega}$ with $|\bar{\omega}| = \sigma$ is either a structure or a texture. Hence one can detect textures for each scale, as the residual of the canonization process described in Sect. 3.1. One may have to impose boundary conditions so that the texture regions fill around structure regions seamlessly. In the next section we describe how this is done in our framework.

3.4 Evaluation

Once we perform co-variant detection and proper sampling we encode the trackable regions by their corresponding dictionary element as discussed in Section 3.2. A typical result is shown in the first image in Figure 11. The rest of the domain of the frame is a candidate for texture, following the complementarity condition of Theorem 1 in [19].

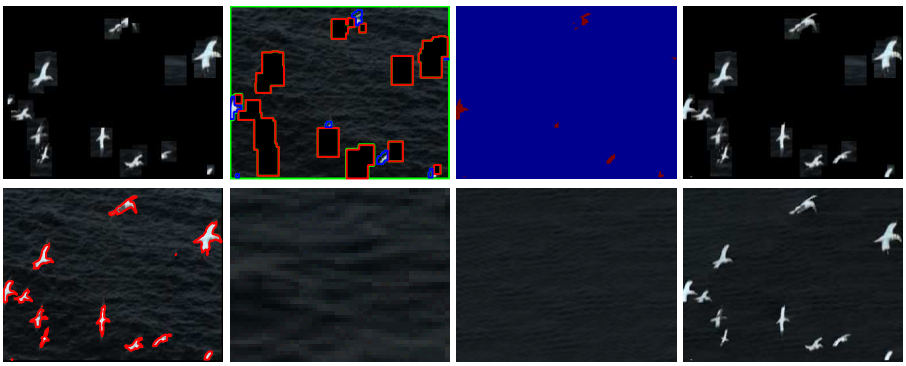

Figure 11: *Pipeline. From left to right, top to bottom: (1) Tracked regions map. (2) Initial segmentation (red indicates boundaries of tracked regions, blue and green indicate boundaries of texture region candidates). (3) Regions of the image that did not satisfy the stationarity assumption underlying the texture hypothesis. (4) Updated tracked region map. (5) Final structure / texture partition. (6) Compressed textured region. (7) Synthesized textured region. (8) Natural blending of textures and structures.*

We perform texture segmentation using Algorithm 5 and we obtain the second image of Figure 11. For AIB we use the implementation from [69]. Structured regions are excluded from the segmentation and the segmentation algorithm is restricted to the rest of the domain.

It can be observed that the tracking mechanism fails to track some of the structure regions. This is unavoidable as the texture/structure partition is an early commitment based on low-level statistics. It is therefore important that the subsequent stages of processing can compensate for such unavoidable errors. In particular, in the example above, trying to synthesize those regions by the algorithm described in Section 3.3.2 will fail. Therefore, a stationarity test needs to be applied to verify the validity of the texture assumption before texture synthesis. This test rejects the regions within the blue boundaries, that are therefore excluded from the synthesis process, and encoded as structures instead. Since the tracking mechanism often fails to detect small structures, but small structures can be perceptually salient, such a *repechage* mechanism is critical to the successfully encode complex scenes. In the third image of Figure 11 we show the domain of the regions that have failed the stationarity test and that were initially not detected by the co-variant detection mechanism.

The collection of trackable regions is then updated, and the final partition is shown in the fourth panel in Figure 11. We then reiterate the texture segmentation routine to obtain the different textured regions in the frame. We compress each texture region using Algorithm 4. A typical result is shown in the sixth panel of Figure 11. Finally we store the dictionary and the locations of the trackable regions and the compressed representations of the textures. When more than one texture is detected in a video frame, we also store the texture boundaries.

At decoding time, we synthesize the textures using Algorithm 5 (seventh image of Figure 11) and overlay the trackable regions using the representation stored in the

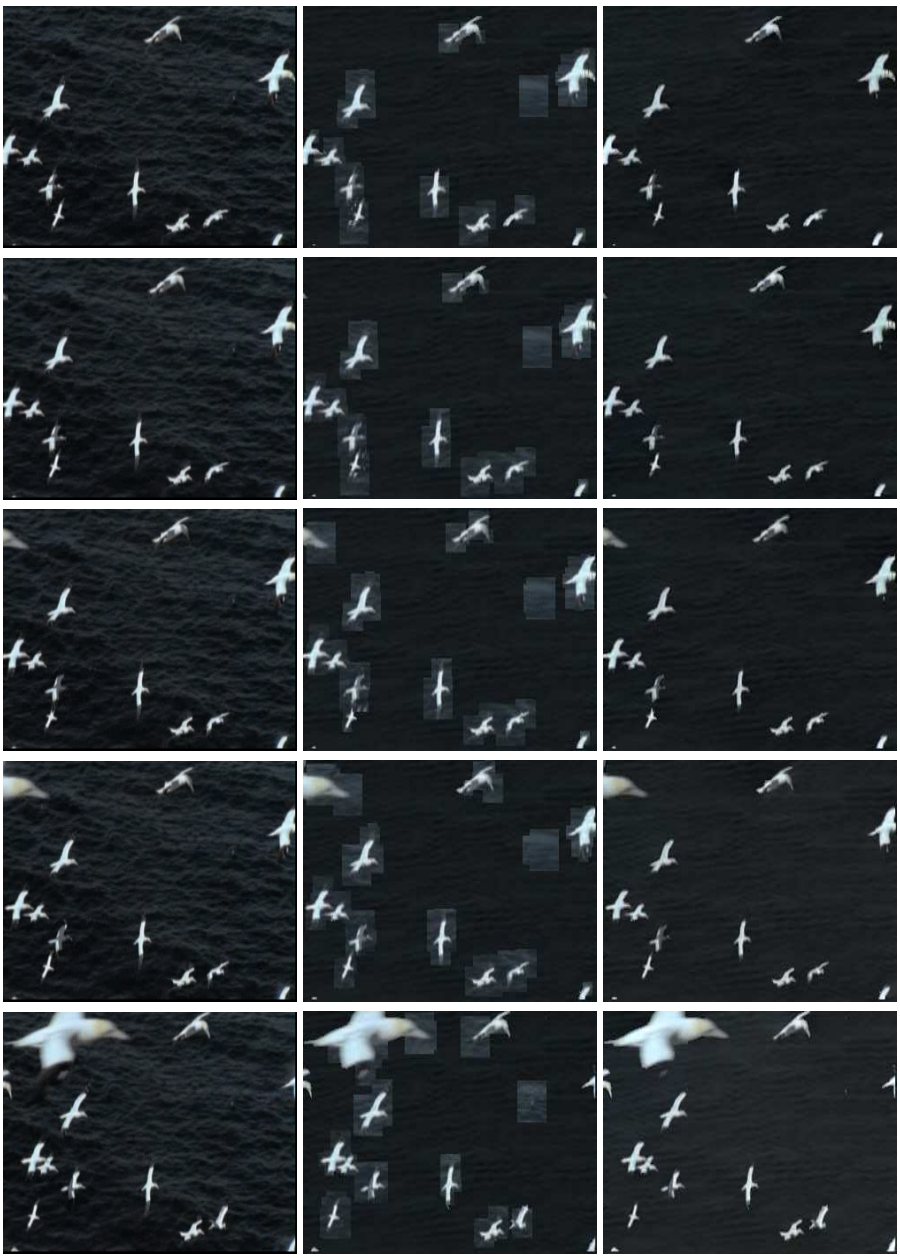

Figure 12: *Samples from the "bird-sea" sequence. Left: Input sequence, Center: Overlaid tracked regions and synthesized textures. Right: Natural blending of synthesized textures and tracked regions.*

Figure 13: *Reconstruction of structured regions. Left: Input frame, Center: Video Primal Sketch, Right: Our method. Our method successfully preserves salient regions.*

dictionary. Representative results of this procedure are shown in the central column of Figure 12. It is immediate to see that boundary conditions are not matched across the texture/structure partition, resulting in salient perceptual artifacts. In order to enforce compatibility at the boundary in a way that is consistent with the statistics of natural images, we determine pixel-level boundaries of the textured regions, and restrict the domain of the texture representation to this region, leaving the texture synthesis algorithm to explain the remainder. To determine the boundary, we perform texture synthesis on the entire domain, and restrict structures to regions that exhibit large residuals. Hence we can build accurate boundaries for the texture as shown in the fifth panel of Figure 11. To enforce prior knowledge we have of the statistics of natural images, we exploit boundary gradients to produce a gradient field that is (approximately) integrable, a typical property of natural images [63]. We use the algorithm proposed by [63] and typical results are shown in the right column of Figure 12.

We also compare our method with [76] in Figure 13. It can be seen that their method misses important structures that are detected by our approach.

3.5 Conclusions

We have described an encoding of structure regions using a dictionary representation, and exploited the partition of the image domain into structures and textures to exploit both spatial and temporal redundancy for video compression. Our approach is an extension of [19], but improves the results therein by considering a tight partition respecting object boundaries and the statistics of natural images.

4 Scene-Aware Video Modeling and Compression

We describe a video compression methodology that exploits the structure of the data formation process, whereby the "source" is the scene, and the "channel" includes scaling and occlusion phenomena that are critical elements of image formation. Thus our scheme involves occlusion detection, optical flow computation, texture/structure partition, and a notion of proper sampling. We show preliminary but promising results that exceed baseline compression performance, albeit at an increased computational cost. We believe that for video compression to move forward it is necessary to take into account the phenomenology of the data formation process. Rather than compressing the images, one should compress the scene; scaling and occlusion phenomena play a critical role in this process, that we explore in this manuscript.

Occlusion phenomena (occluding boundaries), together with material transitions (material boundaries) and illumination boundaries (cast shadows), are indirectly accounted for in existing video compression schemes: They all yield highly kurtotic gradient distributions that are captured by sparsity-inducing priors. However, of the three, only occlusion phenomena yield an "information increment" (innovation), where future data cannot be explained with past data. Therefore, it is important to be able to *detect occlusions*, which we do in Sect. 4.2.1. This also relates to optical flow which addresses temporal redundancy.

Scaling and quantization challenge the traditional sampling paradigm, and calls for a new notion of *"proper sampling"*, that we introduce in Sect. 4.2.2. This relates also to the notion of "texture" and addresses spatial redundancy.

In Sect. 4.3 we report our experimental results. Although our focus is on outlining a modeling framework, we show that even a non-optimized implementation of our pipeline beats baseline performance using the PSNR score. In Sect. 4.4 we point at limitations and challenges in our approach.

4.1 Formalization

A (grayscale) video $I : D \subset \mathbb{Z}^2 \times \mathbb{N} \to \mathbb{N}; (x, t) \mapsto I(x, t)$ is a spatially and temporally quantized version of a hidden signal that we call *radiance* ρ. If we consider, for the sake of example, a planar scene parallel to the image, then a motion orthogonal to it induces a re-scaling of the image domain: $I(x, t) = \int \delta_\epsilon(x - \tilde{x})\rho(s(t)\tilde{x})s(t)d\tilde{x}$, where δ_ϵ is the quantization kernel (for instance an indicator function supported on the pixel). This shows that we cannot just discretize the radiance by defining ρ on a discrete domain, because by going sufficiently far from the scene we can make $s(t)$ sufficiently small, and therefore the sampling of $s(t)\tilde{x}$ sufficiently high-rate. Instead, we must consider it as a (continuous) function from surfaces embedded in three-dimensional space to the positive reals. If the scene is not a fronto-parallel plane, but has arbitrary shape, it induces a deformation of the image domain that is an epipolar transformation [43], which can be shown to admit as closure the entire group of planar diffeomorphisms [62], so $I(x, t) = \int \delta_\epsilon(x - \tilde{x})\rho(w(\tilde{x}, t))|J_w(x, t)|d\tilde{x}$. Therefore, we model the radiance as a function $\rho : \mathbb{R}^2 \to \mathbb{R}^+$, and scene or viewer's motion as a domain diffeomorphism $w : \mathbb{R}^2 \to \mathbb{R}^2$. If we think of the radiance as the "representation" that we wish to

infer, encode, store, and reconstruct, in addition to unknown domain deformations w, we have unknown transformations of the range of the data. The simplest are contrast transformations $\kappa : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+; (I,t) \mapsto \kappa(I,t) \doteq \kappa(t) \circ I$, that are continuous monotonic transformations of the range space of the image. For simplicity, we will include the quantization kernel in the contrast transformation, and therefore consider models of the form:

$$I(x,t) = \kappa(t) \circ \rho \circ w(x,t) + n(x,t) \tag{18}$$

where n is the residual that lumps all unmodeled phenomena (mutual-illumination, inter-reflection, transparency, translucency, vignetting, etc.) as well as sensor noise and other more standard phenomena. The model (18) describes the temporal evolution of the data *in the co-visible portions of the scene*. These are portions of the scene that are visible in more than one image at a time. In the complement, i.e. the (multiply-connected) *occluded region* $\Omega(t) \subset D$, the image at a given time t can in principle take any value compatible with the marginal statistics of natural images. So, the residual $n(x,t)$ can be assumed to be small, spatially and temporally white and homoscedastic in the co-visible region $D \backslash \Omega(t)$, but it could be arbitrarily large in the occluded region $\Omega(t) \subset D$. However, such regions are typically sparse, in the sense that the occluded area $|\Omega(t)|$ is small relative to D.

Given a collection of images, $\{I(x,t)\}_{x \in D, t \in [0,\, T]}$, our goal is to infer a model $\Sigma = \{\hat{\kappa}, \hat{\rho}, \hat{w}\}$, that is as "simple" as possible, and that yields as "small" a residual $\sum_{t,x} |n(x,t)|$. This can be framed as an optimization problem, where we must declare what we mean by "simple", what we mean by "small", and define the class of models Σ. Unlike other tasks, such as decision and control, in reconstruction we do not care about model *identifiability*, so long as we can find *any* model that fits the bill. In the next section we describe these ingredients and the ensuing algorithmic instantiations.

4.2 Encoder

4.2.1 Occlusions, optical flow, and temporal redundancy

Occlusion detection is a binary classification problem where each pixel is labeled as either *co-visible* ($x \in D \backslash \Omega(t)$) or occluded ($x \in \Omega(t)$). Note that this does not depend on how many "objects" there are in the scene, or occlusion layers: A point on the scene is either visible or not. In order to test for occlusion, we must invalidate the co-visibility hypothesis, that entails searching for the domain diffeomorphisms that maps one image onto another. If we restrict our attention to two temporally adjacent images, this problem is known as *optical flow*: $\rho(x) = I(x, t-1)$ and the residual $n(x,t)$ in (18) is the sum of two components: $e_2(x,t)$ that is *dense*, (defined for all $x \in D$) but statistically "simple" (spatially and temporally uncorrelated, isotropic, homoscedastic etc.) with a small variance. The other $e_1(x,t)$ can be arbitrarily large, but it is only defined on the occluded domain $x \in \Omega(t)$. Thus the problem is to simultaneously infer optical flow $w(x,t)$ as well as the occluded region $\Omega(t)$, that is time-varying, multiply-connected, and in general can be rather complex (think the occlusions induced by motion in front of a barren tree). Ayvaci et al. [3] have framed this problem as a convex variational optimization, and solved it with numerically efficient Augmented Lagrangian methods.

We will therefore take this stage as a building block and assume that, at each time t, we have an estimate of the characteristic function of the occluded region, $\hat{\Omega}(t) \subset D$, as well motion field $\hat{w}(x,t), x \in D \backslash \hat{\Omega}(t)$ that solve

$$\hat{w}(x,t), \hat{\Omega}(t) = \arg\min_{w,\Omega} \|e_2\|_{\ell^2(D)} + \lambda \|e_1\|_{\ell^1(D\backslash\Omega)} \tag{19}$$
$$\text{s. t.} \quad I(x,t) = I(w(x,t), t-1) + e_1(x,t) + e_2(x,t)$$

plus some regularization of the motion field w; λ is a tradeoff factor (multiplier). Note that we have not included a range transformation $\kappa(t)$, since we do not expect significant contrast changes between adjacent images, and the small residual can be lumped into e_2. In Figure 14, we show results on a few test sequences based on the algorithm used. While optical flow tells us what *regions* of the images are occluded, and therefore regions

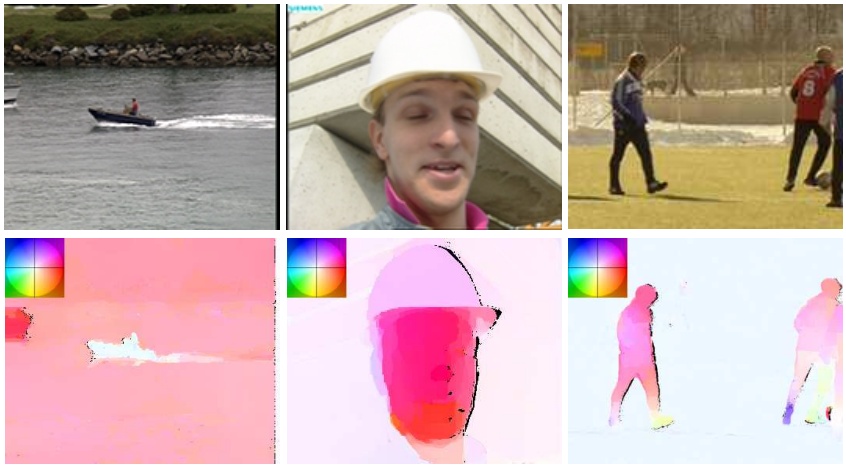

Figure 14: (COLOR) **Occlusion detection and optical flow:** *Original images (top) and optical flow (bottom), visualized according to the color scheme shown in the top left corner of each image. Black regions are occluded, so no motion estimate is available since there is no region in image $I(x,t-1)$ that, transported with $w(x,t)$ yields $I(x,t)$.*

where we cannot exploit temporal consistency, not every *pixel* in the co-visible region has a unique correspondent. Those that do can be encoded once and then "tracked" through subsequent frames. Those that do not can be *sampled* or *filled in* independently in successive frames, as we discuss in the next section.

Encoding the motion field $w(x,t)$ is, in principle, more than twice as costly as encoding the image, since $I(x,t) \in \mathbb{N}$, whereas $w(x,t) \in \mathbb{R}^2$. So to achieve compression one has to exploit the spatial regularity of the motion field, that is assumed to be piecewise smooth everywhere away from occlusions (unlike the image that cannot be realistically assumed to be piecewise smooth). For any given tolerance ϵ, one can devise a partition of the co-visible region such that the cumulative approximation is within ϵ. This corresponds to a *motion segmentation* task. We break this process in two parts. The first involves detecting contours and the second part involves using those contours to

segment the image frame. Breaking this process into these two components allows us to retrieve several different segmentations depending on the strength of the boundary. As we will see, this enables adaptive merging of neighboring regions to trade off complexity and fidelity. We use [2] for contour detection and segmentation. In Figure 15 we show representative samples of this process.

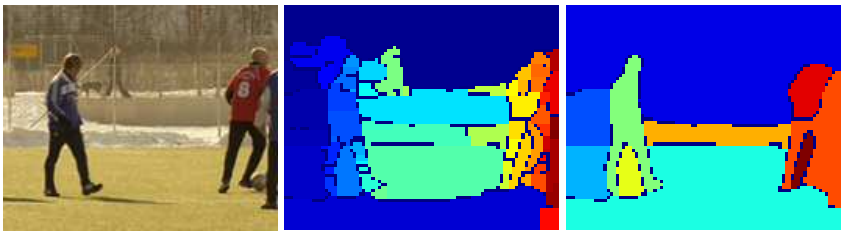

Figure 15: *Segmentation [2]. Varying a threshold yields structurally different segmentations. Original test frame (left). Segmentation using a low threshold (center). Segmentation with a higher threshold (right).*

4.2.2 Proper sampling

Co-visible regions can be put in correspondence, in the sense that there exists a deformation of the co-visible domain that maps one image onto another: $D\backslash\Omega(t) = \{w(x,t) \mid x \in D\backslash\Omega(t-1)\}$. However, individual pixels can have multiple correspondents, for instance when the co-visible region is homogeneous or self-similar. These regions will be spatially encoded as "textures" in the next section. Everywhere else, we can establish a local reference frame via a process known as *canonization* [58]: Co-variant detector functionals [37] operate on a multi-scale representation of the data and are used to detect extrema that correspond to *"canonical"* local frames. However, we do not know whether such extrema are due to the "scene" (e.g. a material transition), or are "aliasing" phenomena (e.g. discontinuities due to spatial quantization or noise. Ideally, for the discretization to be a *proper sampling*, we would like for the response of co-variant detector functional operating on the image $I(x,t)$ to be *topologically equivalent* to the response of the same functionals operating on the scene $\rho \circ w(x,t)^{-1}$. Unfortunately, we do not know the scene, so this hypothesis is not testable. However, under the assumptions of Lambertian reflection and co-visibility, proper sampling is equivalent to topological consistency between *different images of the same scene*, for instance $I(x,t), I(w(x,t), t-1)$. The portions of the regions that are properly sampled are then "trackable" [37], and can be encoded once. The aliased structures are instead lumped in the ensemble description that is treated by the texture module described in Sec. 4.2.5. Additional processing can be performed to aggregate tracks over multiple views [61]. In Figure 16 covering trajectories on sequences are shown.

4.2.3 Structure/texture partition

Once proper sampling is established, and therefore temporal redundancy is accounted for, we consider spatial regularity. This can be of two kinds: Spatial regularity of trackable

Figure 16: *Covering of trackable regions [61]. The rest is encoded by the texture module.*

regions and their description [37], and spatial stationarity of the (colored) noise process that is independently sampled in different temporal frames. This corresponds to the notions of "regular" and "stochastic" textures [72].

As discussed in Section 4.2.1, encoding the motion field point-wise in trackable regions would be costly. Given a segmentation, we can assign the same motion vector to the entire region, or a sub-partition. Unfortunately, segmentation does not usually respect structure/texture partition and can contain areas that are trackable, or not. Since scale composed with quantization is a *semi-group*, rather than a group, we have to consider multiple segmentations at different scales (hierarchical segmentation) and aggregate motion vectors only at the finest scales. We then merge regions in a greedy fashion to satisfy both motion constraints and structure/texture partition. We show representative results in Figure 17.

Figure 17: *Original Test Sequences (top). Texture/Structure partition (bottom). Regions in green are labeled trackable regions and we can assign a motion vector to them. Textured regions are either homogenous regions or stochastic textures and it is preferable to spatially predict them since they cannot be tracked.*

4.2.4 Encoding trackable regions

For a trackable region in frame $I(x, t+1)$, the encoder can select either temporal or spatial prediction. For temporal prediction, we calculate the median optical flow within the region and use it to find the corresponding region in $I(x, t)$, together with the residual error. This is done for each region independently. This allows the encoder to select prediction rules (spatial or temporal) independently for each region. Note that at occluded regions, we do not perform motion estimation.

Spatial prediction can be performed in trackable regions provided that the motion, and the local description of the image around each co-variant frame, is spatially regular, so the image can be considered as a sample from a stationary and ergodic spatial process. Spatial prediction thus yields a form of texture segmentation. However, unlike the case described in the next section, the prediction has to be *temporally consistent*. To this end, we calculate the mean value of each region in an image and propagate it through its motion vectors, to temporally adjacent images. Again, this process is performed independently for each region and yields a (spatial) prediction, together with a residual.

4.2.5 Encoding non-trackable regions

Non-trackable texture regions can be encoded independently in each image, since by definition there is no temporal consistency. One can still exploit spatial redundancy and perform temporal prediction as done in Sect. 4.2.4. Rather than spatially extrapolating a realization we extrapolate statistics and sample independently at each image, thereby eliminating temporal consistency. This is equivalent to texture synthesis and is described next. In addition, the encoder has still the option of using temporal prediction, to enable overcoming errors in the texture/structure partition.

In the texture synthesis mode, we chose to use a variation of the algorithm of [36] because it is relatively efficient and gives results that are perceptually acceptable, even though we are aware that we will pay a price in terms of PSNR. This non-parametric sampling-based approach takes a subset of a texture region and expands it. This is accomplished by minimizing the following energy:

$$E_t(I_{|_B}; \{I_{|_{N(B)}}\}) = \sum_{p \in N(B)} \|I_p - \hat{I}_p\|^2 + \mu \sum_{p \in N(B)} \|DI_p - D\hat{I}_p\|^2 \qquad (20)$$

where $I_{|_B}$ is a vectorized version of the textured region to be synthesized and $\{I_{|_{N(B)}}\}$ is the set of vectorized neighborhoods from the input texture. The vectors $\hat{I}_p$ and I_p are neighborhoods of the input and synthesized textures respectively, centered at pixel p, that are closest in appearance. D is the differentiation operator. In principle, $N(B)$ can be every pixel on the synthesized image grid. For computational complexity reasons, we only consider a subset of them.

The energy above is minimized using an EM-like alternating minimization. The first step minimizes the energy by direct differentiation with respect to I_p that yields a linear system of equations. The second step uses the calculated I_p to find the set $\{\hat{I}_p\}$, using nearest-neighbor (NN) search. The process is repeated until the decrease in energy is negligible, as customary. In addition, a re-weighting scheme is used to improve outlier rejection (similar to iteratively re-weighted Least Squares (IRLS)).

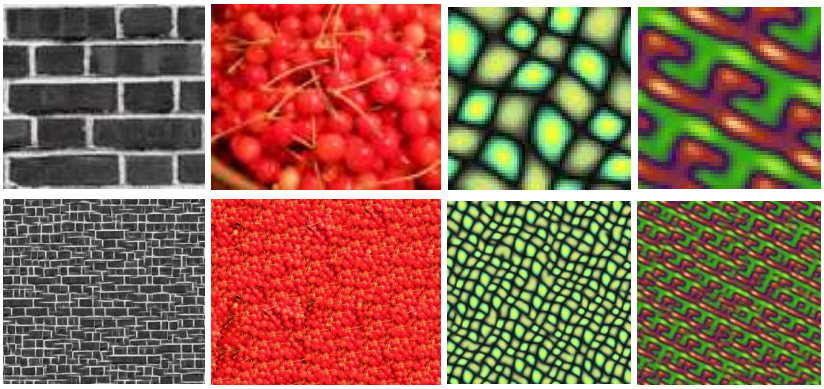

Figure 18: *Low resolution image patches (top). Image Patches of higher resolution synthesized with our algorithm (bottom).*

The scheme is repeated over 3 neighborhood sizes: $w = [32, 16, 8]$. Finally, the minimization procedure is repeated over a number of different resolutions (i.e. image sizes). Therefore, the whole algorithm is performed in a multi-resolution multi-scale fashion. The relative weight parameter is set to $\mu = 10$.

The algorithm works well for fine-scale textures; however, textures that exhibit structure at coarse scale (e.g., the bricks in Figure 18) require large region sizes to be captured. This is particularly severe when the regions are selected with a uniform prior. To minimize this effect, we replace M-Step with the following minimization problem for all I_p, $p \in N(B)$:

$$\hat{\alpha} = \arg\min_{\alpha_p} \|I_p - Y\alpha_p\|_2^2 + \lambda||\alpha_p||_1 \tag{21}$$

where $Y = [I_1\ I_2\ \ldots\ I_N]$ are the concatenated vectorized neighborhoods from the set $\{I_{|_{N(B)}}\}$ and N is the total number of neighborhoods in the original image that we consider. Thus, instead of selecting one NN, we choose a sparse linear combination. This improves the stability of the algorithm in degenerate cases. We show results of our texture synthesis algorithm in Figure 18.

Artifacts are still visible for the brick texture, but they are less severe than without this additional step. To synthesize textures, we always encode a quarter of the size of the region and synthesize the rest.

4.2.6 Encoding video frames

After calculating the predicted images using the spatial, temporal and synthesize modes, the encoder chooses for each region the prediction mode that yields the smallest residual. We maintain an index function for the prediction mode used for each region. The predicted and error images are then formed. If a region is predicted temporally, the predicted region is taken to be the one calculated with the temporal prediction. The same applies for the other two modes. Since each region has been predicted independently, there is no issue of inconsistency. The error image is transformed and quantized in

the same way as in H.264 [31], zig-zag scanned and entropy encoded. We encode the boundary indicator function of the regions using run length encoding (RLE). We also use RLE for the indicator function of the occluded pixels. We use entropy coding on the output of the RLE. Motion vectors, mean values for spatial prediction and the small patches for textured regions are also entropy encoded. The predicted image plus the error image form the new image frame, which is then subsequently used as the previous frame for encoding the following frame. Hence the decoder is embedded in the encoder, similar to modern coding standards. All quantities are quantized at the quantization level specified by the user.

4.3 Experiments

To evaluate our encoder we compared its performance with four different encoding methods. In the baseline method used (blue curve in Figure 20), we encoded the video sequence by quantizing and entropy coding each frame independently. In the second method (yellow curve) we calculated the difference image between the last encoded frame and the next frame. We quantized and entropy coded the difference. The other two methods we compared with was MJPEG (green curve) and H.264 JM ver.18.0 (orange curve)[2]. In the H.264 JM implementation, we used default parameters provided by the authors. We varied the performance of the encoder by changing the quantization parameters. In our encoder we fixed the parameters for all experiments and we also only varied the quantization parameter. For optical flow estimation and tracking we used the default parameters set by the authors of each paper. For the hierarchical segmentation we used 3 levels. In Figure 19 we show the prediction performance of our encoder. The left column corresponds to the predicted image. Small or zero error (middle column) can be achieved in most of the image domain. Where large deformations of the domain occur, the translational model fails (e.g. the face of foreman).

Quantitative results are shown in Figure 20 for four video sequences of QCIF resolution. PSNR scores of the Y-channel are reported. We used 50 frames from each video sequence [3]. It can be seen that our encoder significantly outperforms the other methods at almost all bit rates. The biggest gain comes at high bit rates where the gain in PSNR in using our encoder compared to others becomes significantly large.

Following is a break-down of the contribution of each module to the overall computational cost. For VGA resolution video, the GPU implementations of the optical flow, tracking and segmentation algorithms take 10, 2 and 13 seconds respectively. Segmentation can be ran in parallel with the other two algorithms. Texture synthesis takes approximately 5 seconds per frame. It was found that the average encoding time per frame was 21 seconds. The reported computational time was recorded on a dual core 2.4 GHz desktop with an NVIDIA GeForce GTX 560 Ti GPU.

4.4 Discussion

We have presented a modeling framework for video compression that exploits the basic phenomenology of optical data-formation, focusing on occlusion and scale. Temporal

[2]The H.264 JM implementation can be found at http://iphome.hhi.de/suehring/tml/

[3]The video sequences can be found at http://media.xiph.org/video/derf/

Figure 19: *Qualitative results of our encoder. Predicted image using spatial, temporal and texture synthesis prediction modes (left). Residual error: black corresponds to zero error, white to large (center). Reconstructed video frames (right).*

consistency is exploited by detecting *co-visible regions* (as the complement of occluded regions) and encoding them once, together with a compressed motion model. Spatial consistency is exploited by partitioning the data into structures (that are encoded by sparse bases) and "texture" (samples from a locally stationary distribution), that can be encoded in ensemble form, and sampled at decoding. The decision as to what is "properly sampled", and therefore can be tracked, as well as what is visible, can only be made by considering multiple images.

While the compression performance of our approach is demonstrably better than existing schemes, this is not surprising since the model class we consider is far larger than that implied in traditional approaches. The obvious downside is a significant computational complexity. However, we believe that rather than incrementally refining intrinsically linear superposition models with additive uncertainty, embracing the full non-linearity implied by occlusion and scaling is necessary to break the video compression wall and develop the next generation video compression schemes.

Also, the results we have displayed use standard PSNR as a performance score.

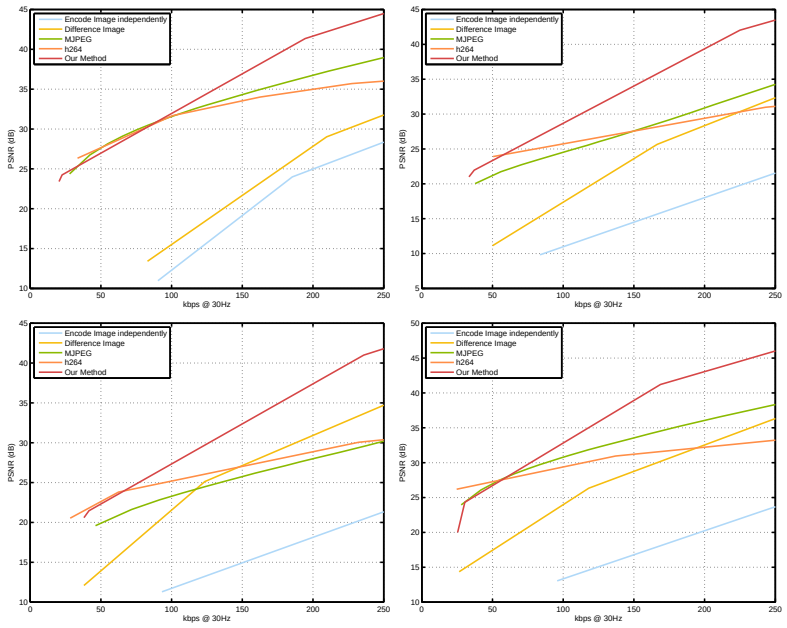

Figure 20: *Quantitative comparison of our encoder with other methods. Our method significantly outperforms all other methods (including H.264) in a wide range of kbps in terms of PSNR. Results shown are for the sequences city (upper left), flower garden (upper right), mobile (bottom left) and coastguard (bottom right). Comparisons were ran for a duration of 50 frames.*

Clearly this is inadequate for the task of compression video for human fruition. A perceptual score is more appropriate, although the lack of a universally accepted model of perceptual performance prompted us to use the more conventional PSNR. It is our intention to further the development of perceptual scores, and integrate them into the spatio-temporal encoding of the scene radiance, so that albeit not observable (unique), at least the model can be *minimal* with respect to the specific task at hand. This is subject of on-going research.

5 Actionable Saliency Detection: Independent Motion Detection Without Independent Motion Estimation

A subset of a sensing field (e.g. visual) is ordinarily deemed "salient" if it is "sufficiently different" from its surroundings. Saliency is therefore a detection and localization task (illustrated in Fig.21), often motivated by resource constraints: if one can process only a subset of the data, which subset is most "valuable" or "informative"?

Traditionally, saliency detection has been agnostic of the underlying task. More recently, however, several authors have attempted framing saliency detection in an information-theoretic context, by looking at the "most informative" subset of the data, where "information" is measured in the traditional sense of Wiener and Shannon. For instance, Itti and Baldi [30] measure the relative entropy between the prior and the posterior of an image, interpreted as a distribution of pixel values, and use it as a measure of saliency or "surprise".

In this paper, we focus on classes of tasks that involve decisions about the *scene*, rather than about the *image*. These include detection, localization, recognition of objects, events, or spatial locations from images, as well as navigation, manipulation and other spatial control tasks. While often "salient" locations in the image correspond to salient geometric or topological characteristics of the scene (e.g. occluding boundaries), this is not always the case (e.g. material or illumination boundaries). Moreover, whether a salient region of the image does indeed correspond to a geometric or topological characteristic of the scene cannot be positively ascertained from one image alone; therefore, we are interested in saliency detection mechanisms that involve *multiple images*. Of course, because part of the motivation for detecting salient regions is to expedite processing (at the expense of a loss in discriminative power), we are interested in temporally adjacent images (*"small baseline"*), such as two or more temporally consecutive frames of a video.

When the camera is static, as in the case of video surveillance, anything that *moves* is salient. There is a considerable amount of literature on *background subtraction*, that can be thought of as a form of saliency detection for the specific case of surveillance tasks (see [32] and references therein). However if the camera is moving, then detecting objects that are moving independently is a notoriously difficult problem, for it amounts to detection of independent rigid motions. This involves model selection and regression to find the independently moving objects and their motion. And yet, even when driving, we can easily spot a moving animal in the distance. When flying we can detect another flying vehicle, or vehicles moving on the ground. Several attempts to perform "background subtraction from moving cameras" [54] have improved efficiency compared to multi-rigid motion estimation, that was using algebraic geometric methods [70] or sampling methods that would clearly not be viable for the task of rapid detection of "informative" regions of a video. Moreover, there is no direct link between any of these algorithms and a notion of what "informative" means.

A definition of "information" in the context of visual decision tasks [57], that draws on ideas from Gibson's Ecological Approach to Visual Perception [20], can shed some light on this issue. While the complexity of the image is not necessarily related to its value in a visual decision task, the complexity of the part of the image that would be

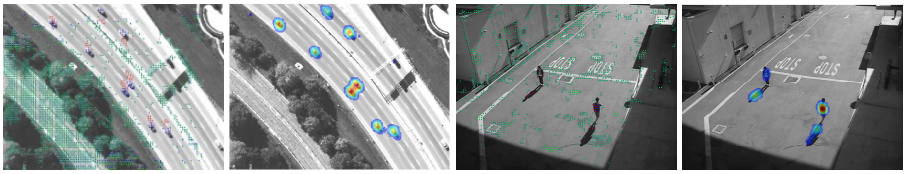

Figure 21: *Detecting salient regions under camera motion: (1st, 3rd): Tracked feature points (blue) are classified as inliers (green) or outliers (red). (2nd, 4th): Estimated salient point density obtained by our algorithm.*

discovered after a finite time interval represents the *"Actionable Information Increment"* provided by the "next image" [57]. It is the decrease in uncertainty about the scene provided by the data. Such a discovery could be due to motion of the viewer, or motion of an object within the scene, or both. In any case, this suggests that *occlusion detection* is a natural form of saliency detection.

Unfortunately, occlusion detection fails to capture important visual phenomena, and indeed even *fails to capture occlusion phenomena* in many cases of practical importance, as we describe next. Therefore, in Sect. 5.2 we propose an alternative scheme for detecting salient regions in videos.

5.1 Occlusion detection fails to detect occlusions

Occlusions are defined as portions of the domain of an image captured, say, at time $t + dt$, that correspond to (are projections of) portions of the *scene* that were occluded from the vantage point where the image at time t was captured. That is, occlusions are something you see in an image but not the other. Unfortunately, such occlusions cannot always be detected in the image: for the examples we mentioned above, if a car is seen from an airplane while traveling on a road that has fairly homogeneous texture, occlusion detection fails. Similarly, a person walking against a white wall can be explained as the person painted on the wall, and deforming with it. This is because occlusion detection from images is based on a hypothesis testing process where the null hypothesis is that portions of two images are *co-visible* when there exists a diffeomorphism ("optical flow") that takes one image onto the other, up to a residual that is statistically simple (white, homoscedastic, and independently distributed) [3].

In formulas, we have that for any given subset Ω of the image domain D, where an image $I : D \to \mathbb{R}$ is measured at each instant of time, the null hypothesis that Ω is *co-visible* between t and $t + dt$ can be written as:

$$H0 = \{\exists \text{ a diffeo } w : \Omega \to D \mid I(x, t + dt) - I(x + w(x), t) \overset{\text{IID}}{\sim} \mathcal{N}\} \tag{22}$$

where the residual $n(x, t) \doteq I(x, t + dt) - I(x + w(x), t)$ is spatially and temporally white, independent and identically distributed according to a simple description, such as a bivariate Normal distribution, $\mathcal{N}$, with diagonal covariance. This means that *co-visible regions are diffeomorphically equivalent up to white noise:* there exists a differentiable and differentially invertible map that takes one image onto the next, except for a white

residual. An occlusion is detected as a violation of the null hypothesis, that is when *no* diffeomorphism can be found that can explain the next image using the previous one and the addition of white noise.

Therefore, a car moving on a road (thus generating an occlusion) can be explained as a car painted on the road (generating no occlusion), and the road-car ensemble stretching and compressing to yield images that are indistinguishable from those actually measured. Yet, we can effortlessly detect moving cars from a moving aerial vehicle (Fig. 23,24).

5.2 Key idea

The problem with occlusion detection is that *equivalence up to a diffeomorphism* is too general, and can explain as ordinary (no violation of the null hypothesis) situations that we want to consider salient. We would indeed prefer to detect as salient, any violations of the *rigidity* assumption, but we do not want to perform independent detection of multiple rigid bodies, because that strides with our goal of computational efficiency.

The key idea of this paper is to still pursue saliency detection as violation of co-visibility, but define co-visibility in terms *not* of diffeomorphic equivalence, but rather *epipolar equivalence*. This means that, of all possible diffeomorphisms $w : D \to D$, we only consider those that are compatible with an overall *rigid motion* of the viewer (ego-motion).

In principle, this could be done by computing the "dominant motion", and then detecting outlier regions as salient. However, we do not actually care to even estimate the motion of the viewer; we just want to compute the discriminant for the null hypothesis (22) in the most efficient way, so that it would depend on the smallest possible number of free parameters. As we show in Sect. 5.3, this number is *two*.

At face value, what we propose looks more complicated than testing for diffeomorphic equivalence $H0$, for we would have to enforce the additional condition that the diffeomorphism is compatible with a rigid motion. In formulas, after testing $H0$ we would have to test for:

$$\begin{aligned} H1 = \{ & \exists\, V \in \mathbb{S}^2, \omega \in \mathbb{R}^3,\ Z : D \to \mathbb{R}^+ \mid \\ & w(x) = \pi(\widehat{\omega} \bar{x} Z(x) + V) \} \end{aligned} \tag{23}$$

where V is the translational velocity direction, ω is the rotational velocity vector, $Z(x)$ is the depth map, $\bar{x} = [x^T\ 1]^T$ is the homogeneous coordinate of x, and π is a canonical central projection [4]. In words, in order to determine whether a region is salient, we would have to search at each instant for all possible translational directions, rotational velocities and depth maps until *none* of them fits the data up to a white residual. The two hypotheses can be tested simultaneously by substituting the expression of w in (23) into (22). The result would be akin to devising a *robust ego-motion estimation* scheme, whereby one simultaneously tries to find the translational direction V, rotational velocity ω, depth map Z, *and* occluded region Ω. This has been indeed done before in

[4] Note that $\hat{\omega} \doteq \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$ belongs to the Lie algebra of the skew-symmetric matrices $so(3) \doteq \{S | S^T = -S\}$.

the literature on "dominant motion estimation" [29] and robust motion estimation [55], and relates to robust statistics [27] and outlier rejection in motion estimation [18].

This would already be an improvement on multi-body motion segmentation. If we have a number, say K, of independently moving objects, and N sensors, then multiple motion estimation requires inferring $N + 5K$ unknown parameters [70]. Dominant motion estimation, on the other hand, only requires inferring $N + 5$ parameters in order to build the discriminant for $H0 \cup H1$. Nevertheless, when N is large, this becomes prohibitive. When the calibration of the camera is unknown, in addition to these parameters one would also have to infer 5 additional parameters (optical center $x_0 \in D$, focal length f, aspect ratio s and skew θ).

As we have already anticipated, our goal is not to estimate ego-motion, but to detect salient regions in the image based on violation of rigidity. Therefore, we seek for ways to reduce the discriminant to its minimal form, which we do in Sect. 5.3.

5.3 Derivation of the discriminant

If we consider the instantaneous motion of the scene relative to the viewer, where the entire scene is moving rigidly, the deformation of the entire domain of the image can be explained as a function of the motion (translational velocity direction V and rotational velocity ω) and the shape of the scene, described by a scalar function from the image domain D to the positive reals, $Z : D \to \mathbb{R}^+$, as described in (23). If we call $y(x) \in \mathbb{R}^2$ the velocity of the projection of the point with coordinates $\bar{x}Z(x) \in \mathbb{R}^3$ onto the image, we have that [59]:

$$y(x) = \mathcal{A}(x)\frac{V}{Z(x)} + \mathcal{B}(x)\omega \tag{24}$$

where:

$$\mathcal{A}(x) \doteq \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix} \tag{25}$$

$$\mathcal{B}(x) \doteq \begin{bmatrix} -x_1x_2 & 1+x_1^2 & -x_2 \\ -1-x_2^2 & x_1x_2 & x_1 \end{bmatrix} \tag{26}$$

Traditional dominant motion estimation and robust statistical approaches search for the unknown motion V, ω and range map $Z(\cdot)$ that solve the following optimization problem:

$$\hat{Z}, \hat{V}, \hat{\omega} = \arg\min \int_D \|y(x) - \mathcal{A}(x)\frac{V}{Z(x)} - \mathcal{B}(x)\omega\|_{\mathcal{H}} dx \tag{27}$$

where $\|\cdot\|_{\mathcal{H}}$ denotes a robust norm, for instance a Huber norm [27]. After this is done, one would find the salient regions that violate this model, that is:

$$\Omega \doteq \{x \in D \mid \|y(x) - \mathcal{A}(x)\frac{\hat{V}}{\hat{Z}(x)} - \mathcal{B}(x)\hat{\omega}\| > \epsilon\} \tag{28}$$

where ϵ is related to the regularization parameter in the Huber norm. Note that the region Ω can, and in general will be *multiply-connected*, so even though this is a

binary classification problem, it enables detecting any number of independently moving objects, each projecting onto a different simply-connected subset of the image domain. Furthermore, when (27) is solved in the continuum, regularization on Z has to be imposed (this is not necessary when (27) is computed at a sparse set of locations). This is laborious, especially because the procedure of finding the motion $\hat{V}, \hat{\omega}$ and the range map $\hat{Z}$ has to be iterated once the outlier set Ω is removed, which in turn changes the motion and range estimates, resulting in a non-convex optimization problem.

Therefore, we resort to a trick introduced by Heeger and Jepson [25], whereby one solves the problem above for the case of the ℓ^2 norm, by exploiting the geometry of Hilbert spaces to "eliminate" the unknown depths $Z(x)$ and unknown rotational velocity ω from (27). This can be done easily since the model (24) is *linear* in $\frac{1}{Z}$ and ω, and therefore one can solve-and-substitute, thereby leaving a set of constraints on the unknown V alone. It has been shown [10] that this can be done without altering the topology of the solution space, in the sense that no spurious solutions are introduced by the algebraic manipulation.

Formally, this can be accomplished (Sect. 5.3.1) by rewriting the model (24) in terms of an operator $C(V)$ that multiplies all the unknown depths and rotational velocity, then multiplying by the orthogonal projector operator $\hat{C}(V)$ that eliminates the dependency on ω and Z, and leaves constraints on the unknown V only.

5.3.1 Computation of the optimal discriminant

Following the discussion in Sect. 5.3, salient points $x \in \Omega$ will be detected as a violation of the hypothesis provided by the model (24). Equivalently, one can seek to infer V in a robust fashion and detect Ω as the outlier set. We focus on a finite number of N sparse measurements, $x_i,\ i = 1, \dots, N$ and introduce a diagonal weight matrix $\mathcal{W} \in \mathbb{R}^{2N \times 2N}$. Ideally, $\mathcal{W}$ should be zero except for points that follow (24). Writing (24) as a system of linear equations for all points and introducing $\mathcal{W}$ as a weight matrix we have:

$$\mathcal{W}Y(X) = \mathcal{W}C(V) \begin{bmatrix} p(x_1) & \dots & p(x_N) & \omega \end{bmatrix}^T \tag{29}$$

where, $p(x_i) \doteq \frac{1}{Z(x_i)}$, $Y \doteq [y(x_1)^T, \dots, y(x_N)^T]^T$, $X \doteq [x_1^T, \dots, x_N^T]^T$, $\mathcal{W} \doteq diag(w_1, w_2, \dots, w_{2N-1}, w_{2N})$,

$$C(V) \doteq \begin{bmatrix} A(x_1)V & & & B(x_1) \\ & \ddots & & \vdots \\ & & A(x_N)V & B(x_N) \end{bmatrix} \tag{30}$$

For any (unknown) V, we can solve for $P \doteq [p(x_1) \dots p(x_N)]^T$ and ω using Least Squares:

$$\begin{aligned} [P, \omega]^T &= (\mathcal{W}C(V))^\dagger \mathcal{W}Y(X) \\ &\doteq (C(V)^T \mathcal{W}^T \mathcal{W} C(V))^{-1} C(V)^T \mathcal{W}^T \mathcal{W} Y(X) \end{aligned} \tag{31}$$

For readability purposes, we will henceforth drop the explicit dependence of C on V and of Y on X. We can then plug the solution of this equation back to the model to get

$\mathcal{W}Y = \mathcal{W}C(\mathcal{W}C)^{\dagger}\mathcal{W}Y$. Rearranging, we get:

$$\hat{C}(V)Y \doteq \left(I - \mathcal{W}C(\mathcal{W}C)^{\dagger}\right)\mathcal{W}Y = 0 \tag{32}$$

The above constraint is true even in the presence of outliers when the elements of $\mathcal{W}$ corresponding to those equations are 0. In practice, this cannot be achieved though, due to the presence of noise and unmodeled phenomena. Assuming that the error in motion estimation follows a Gaussian distribution the Least Squares estimation of P and ω is optimal. Hence it is important to calculate $\mathcal{W}$ properly so that inference of V is improved. To estimate V we solve the following minimization problem:

$$\underset{V}{\text{minimize}} \quad \psi(V) = \frac{1}{2}||\hat{C}(V)Y||_2^2 \tag{33}$$

where $V \in \mathbb{S}^2$. To calculate the weight matrix $\mathcal{W}$ we employ a more traditional M-estimator, as customary in robust statistics, that does not explicitly infer $\mathcal{W}$, but instead uses a composite norm residual where the weight of the outliers is reduced. This yields a minimal model, where the only unknowns are the directional coordinates of the translational velocity V, as discussed in the previous section.

Since we expect that most points in the scene will move rigidly, we anticipate that $\hat{C}(V)Y$ is sparse. We would hence want to choose the diagonal elements of $\mathcal{W}$ to enhance sparsity of the residual. In addition, every pair of elements of $\hat{C}(V)Y$, corresponds to the residual for a single point and hence this should also be taken into account when estimating $\mathcal{W}$. The outline of the algorithm is given below.

Algorithm 6: Iterative reweighted subspace minimization (IRWSM).

Initialize $\mathcal{W}^{(1)} = I, V^{(0)} = [1, 0, 0]^T$
foreach $k = 1, 2, 3, \ldots,$ *K* **do**
  Solve the following problem initializing with $V^{(k-1)}$:
    $\hat{V}^{(k)} = \arg\min_V \frac{1}{2}||\hat{C}(V, \mathcal{W}^{(k)})Y||_2^2$
  $e^{(k)} = \hat{C}(\hat{V}^{(k)}, I)Y$
  $\lambda = 1/\text{mean}(||e^{(k)}||)$
  foreach $i = 1, 2, 3, \ldots,$ *N* **do**
    $w_{2i-1}^{(k+1)} = w_{2i}^{(k+1)} = \frac{1}{|||[e_{2i-1}^{(k)}, e_{2i}^{(k)}]||_2 + \varepsilon}$
  $\mathcal{W}^{(k+1)} = diag(w_1^{(k+1)}, \ldots, w_{2N}^{(k+1)})$
  $V^{(k+1)} := \hat{V}^{(k)}/\|\hat{V}^{(k)}\|$

where $\mathbf{1}$ is the indicator function. Note that this is a generalization of the case proposed by [25]. The authors of [25] minimized (33) with $\mathcal{W} = I$ using exhaustive search. In that case the above problem is reduced to minimizing $C^{\perp}Y \doteq \left[I - C(C^TC)^{-1}C^T\right]Y$. By introducing $\mathcal{W}$, we solve this more general minimization problem to improve outlier rejection. Since the problem is non-convex, we use gradient descent with backtracking line search to estimate V. The details of the computation of the gradient of (33) are provided in A. We classify a point as an outlier as follows: define $\hat{e} = [\hat{e}_1 \ldots \hat{e}_{2N}]^T \doteq$

$\hat{C}(V^{(K)}, I)Y$ and $E_i \doteq [\hat{e}_{2i-1}, \hat{e}_{2i}]^T$ for $i = 1, \dots, N$. A point i is classified as an outlier when $||E_i||_2$ exceeds ϵ. The threshold ϵ can be determined using various techniques, one of which is explained in Sect. 5.4.

5.3.2 Effects of (mis)calibration

The model we have derived assumes that the image coordinates x_i and their corresponding velocities y_i are *calibrated*, that is they are available in metric units relative to the reference frame having origin at the principal point (intersection of the optical axis with the image plane), with the optical axis orthogonal to the image plane and aligned with the spatial Z axis. Most often, however, coordinates and velocities are given in *pixel* units, relative to, say, the top-left corner of the image. One cannot expect, in general, to just be able to plug the latter into the equation and get a sensible answer. Therefore, in this section we explore the effects of miscalibration on outlier detection.

We first show that knowledge of the *principal point* and the *focal length* does not affect the classification of outliers. We introduce the calibration matrix $K \in \mathbb{R}^{3\times 3}$ in (24) and rewrite it in homogeneous coordinates:

$$\begin{aligned} \begin{bmatrix} y \\ 0 \end{bmatrix} &= K \begin{bmatrix} \mathcal{A}V & \mathcal{B} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/Z \\ \omega \end{bmatrix} \\ &= \begin{bmatrix} fs_x & fs_\theta & O_x \\ 0 & fs_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{A}\frac{V}{Z} + \mathcal{B}\omega \\ 0 \end{bmatrix} \end{aligned} \tag{34}$$

From (34) it is obvious that $y(x)$ is independent of (O_x, O_y). On the other hand, also obvious from (34), the focal length and scale do indeed affect the estimation of the velocity. But the focal length does not affect the outlier distribution: writing the expressions of y, from (34), similarly as in Sect. 5.3.1, we get:

$$\begin{aligned} \mathcal{W}Y(X) &= f\mathcal{W}\mathcal{K}C(V)[P, \omega]^T \\ &\doteq f\mathcal{D}[P, \omega]^T \end{aligned} \tag{35}$$

where $\mathcal{K} \in \mathbb{R}^{2N\times 2N}$ is a block diagonal matrix with its block diagonal entries being $\hat{K} \doteq \begin{bmatrix} s_x & s_\theta \\ 0 & s_y \end{bmatrix} \in \mathbb{R}^{2\times 2}$. Solving for the same unknowns as before:

$$\begin{aligned} [P, \omega]^T &= (f\mathcal{D})^\dagger \mathcal{W}Y(X) = (f^2\mathcal{D}^T\mathcal{D})^{-1} f\mathcal{D}^T\mathcal{W}Y(X) \\ \mathcal{W}Y(X) &= f\mathcal{D}(f^2\mathcal{D}^T\mathcal{D})^{-1} f\mathcal{D}^T\mathcal{W}Y(X) \\ &= \mathcal{D}(\mathcal{D}^T\mathcal{D})^{-1}\mathcal{D}^T\mathcal{W}Y(X) \end{aligned} \tag{36}$$

Focal length is cancelled out in the expression and hence it is not necessary in order to employ our algorithm. In addition, since scale consists of two positive real numbers (or one number, if the pixels are square, or if the form factor of the pixel is known), one can simply augment the search from two parameters, corresponding to V, to four parameters, corresponding to s_x, s_y . In the following experiments we normalize the pixel coordinates to $[-1, 1]$. Regarding the *skew* of the pixel array, it can be assumed to be zero; that is, the pixels are rectangular, and not generic parallelograms.

5.4 Empirical evaluation

We tested our algorithm on 15 sequences. The sequences People-1, People-2, Cars-3, Cars-4, Cars-5, Cars-6 shown in this order in Fig. 22 and Cars-2/06 are from the Hopkins 155 motion segmentation dataset [67] and *ground truth* was provided. In addition, the trajectories of feature points are *provided* by the dataset and are available over the whole duration of the video sequence. This makes the dataset appropriate for comparison with Sheikh et al. [54] which requires the trajectories to be present in an extended period of time. These sequences contain objects that move slowly between consecutive frames, they are close to the camera and are moving independently from it.

The sequences Traffic-1,-2,-3,-4 (Fig. 23) were recorded from a helicopter monitoring a traffic jam. The motion of the camera covers a wide variety of translations and rotations. Bridge-1,-2,-3 (Fig. 24) were taken from an airliner approaching Boston Logan airport. People-3 (second row in Fig. 21) is an aerial view of closed distanced objects. These 8 sequences were *manually* annotated. In addition, to extract trajectories in *these* sequences we used the code provided by [61] that yields dense point trajectories. We used the Harris corner detector [23] to eliminate trajectories on textureless regions. Subsequently, an average of 1300 trajectories per frame are left. Since the extracted trajectories are not guaranteed to be present in all frames hence these sequences are not suitable for comparison with [54]. On the other hand, our algorithm is *not limited by the temporal support* of trajectories. Using the resulting trajectories for a pair of frames in a sequence (we use the middle pair), we calculate the optical flow i.e. $y(x_i)$ for $i = 1, \ldots, N$ which is then used as the input to Algorithm 6 to estimate V and determine the salient regions.

To distinguish between inliers and outliers, we calculate $||E_i||_2$ for each point x_i as its residual. We then construct the histogram of the residuals and find the local minimum nearest to the 0 residual bin. The residual value corresponding to this bin is selected as the threshold ϵ.

We successfully detect most of the salient regions in all sequences. In Fig. 21, 22, 23 and 24 we show the tracked regions and the salient regions as classified by our method. In Table 2, we compare the performance of our algorithm to three other methods using the F-measure: (i) RANSAC [18] with epipolar constraint. We fixed the number of iterations to 1000 and varied the threshold for each sequence to obtain the best results, (ii) we implemented the original method proposed by [25] but minimized it with gradient descent rather than exhaustive search i.e. we used $K = 1$ and $\mathcal{W} = I$ as parameters in our algorithm, and (iii) we implemented the outlier detection method proposed by Sheikh et al. [54] that enforces the rank constraint on trajectories using RANSAC. We also fixed the number of iterations to 1000 and varied the threshold to obtain the best results. This method requires trajectories available in an extended period of time which means it is only possible to compare with it on the Hopkins 155 dataset. For their method, in all experiments, we used either trajectories of length 30 or of the whole video, whichever was the smallest (27 frames on average).

Our algorithm significantly outperforms the other methods in 13 out of 15 sequences and achieves comparable results in the other two, Table 2. Although [54] uses a large number of frames to detect outliers, our algorithm still performs better even though we make use of just 2 frames to make a decision. In addition, our method automatically

Figure 22: *Sample results from the Hopkins 155 dataset: Odd rows: Images with tracked points. Red and green points show the locations of tracked points as predicted by the model. Points in green are the points that are classified as inliers and in red those that are classified as outliers. Blue dots (not visible for inlier points) are the true positions of tracked points. Even rows: Images showing in color the detected outliers. The color corresponds to a sum of Gaussians centered at each salient point.*

chooses the threshold value whereas for RANSAC and [54] we manually chose the best one. Even so, we still perform significantly better than both of these methods. For the Hopkins 155 dataset, it takes on average 32.6 seconds for a non-optimized MATLAB implementation of our algorithm to converge to a solution, whereas for the rest of the sequences, it takes 68 seconds with an average of 1300 tracked points. We terminate our minimization at each iteration when $||V^{(k)} - V^{(k-1)}||/||V^{(k-1)}|| < 10^{-3}$. The average runtime of RANSAC was 2.3 seconds and that of [54] was 2.6 seconds. Experiments

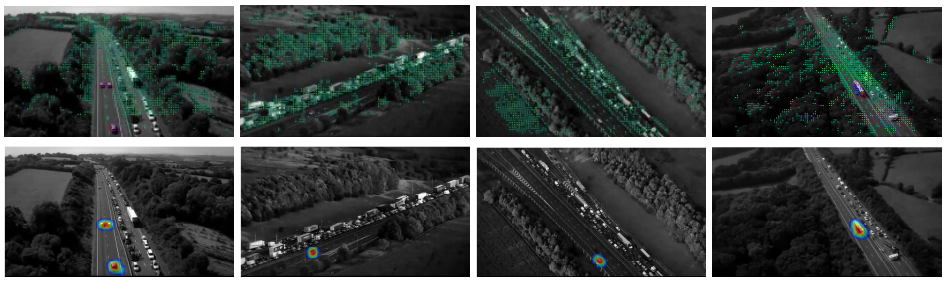

Figure 23: *Four aerial views of a motorway. In all images cars in the right lane are stationary and cars in the left lane are moving. The true outliers in these cases are the moving cars in the left lane. The first row shows the dense tracked points in each image. The second row shows in color the detected outliers. Color convention is the same as in Fig. 22.*

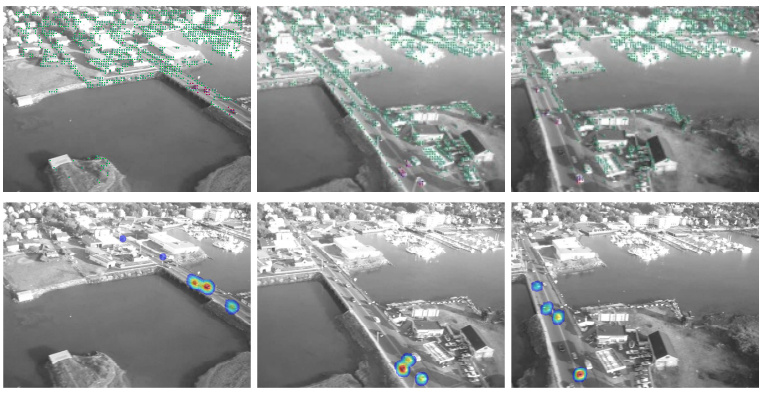

Figure 24: *Three aerial views of a bridge taken from an airliner during a turn approaching Boston's airport. The first row shows the images with the tracked points and the second highlights the salient regions. The color codes are the same as in Fig. 22.*

were ran on an Intel 2.4 GHz dual core processor machine.

Failure modes. The most significant failure case of our method is shown in Fig. 24. Moving cars at the far end of the bridge are not detected. This can be accounted to the fact that the outliers that are not detected are far from the camera and they appear stationary due to their relatively small motion.

5.5 Discussion

We have presented a model for detecting "salient" regions in an image that correspond to objects that are moving in a way that is incompatible with a single rigid motion. Note that, even if the motion is rigid, the deformation it induces on the domain of the image is, in general, as complex as a general diffeomorphism, depending on the shape

	People-1	People-2	Cars-2/06	Cars-3	Cars-4	Cars-5	Cars-6	People-3
Ours	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**0.91**
Heeger & Jepson [25]	0.35	0.06	0.43	0.24	0.22	0.69	0.11	0.88
RANSAC	0.64	0.77	0.54	0.69	0.21	0.65	0.56	0.69
Sheikh et al. [54]	0.91	0.68	0.95	0.90	0.94	0.93	0.80	-
	Traffic-1	Traffic-2	Traffic-3	Traffic-4	Bridge-1	Bridge-2	Bridge-3	
Ours	**0.78**	0.80	**1.00**	**0.93**	**0.55**	0.52	**0.63**	
Heeger & Jepson [25]	**0.78**	0.80	**1.00**	**0.93**	0.54	0.51	**0.63**	
RANSAC	0.11	**1.00**	0.11	0.35	0.49	**0.60**	0.50	
Sheikh et al. [54]	-	-	-	-	-	-	-	

Table 2: *Comparison on salient point detection performance of our algorithm against [25], RANSAC under epipolar constraint and [54] in terms of the F-measure. We compared the performance on* 15 *sequences. The ground truth and trajectories for the first* 7 *sequences were provided by the Hopkins 155 dataset. The last* 8 *were manually annotated by the authors and trajectories were extracted using [61]. Our algorithm significantly outperforms all other* 3 *methods in almost all sequences.*

of the scene, and even more complex if one considers occlusions. Therefore, simple "background subtraction" relative to a small-dimensional parametric motion model (such as an homography) does not work in general. Even occlusion detection, that in principle can be used for testing the co-visibility hypothesis, fails in the presence of objects moving on a homogeneous background.

Therefore, we have proposed a scheme to test for violations of co-visibility, relative to an epipolar domain deformation (as opposed to a general diffeomorphic domain deformation) using tools of robust statistics, and a simple expedient to eliminate motion and structure parameters that do not affect the outlier distribution.

We have also shown that accurate calibration of the camera is not necessary: while calibration error clearly affects the motion estimates, we have shown that some calibration parameters (principal point, focal length) do not affect the decision boundary between inlier and outlier, so they can be ignored for the purpose of saliency detection. Scale can either be coarsely calibrated, or estimated as a hidden variable in the regression/classification task.

Failure modes of our algorithm, illustrated in the experiments, include cases where the objects are too small or moving too slowly. As with any classification scheme, there is a dependency on a scalar parameter (detection threshold) that we have chosen using standard guidelines from robust statistics. Our algorithm is currently not operating in real time. However, the problem has significant structure that could be exploited to devise efficient implementations in hardware platforms in the near future.

6 Future Work and Timeline

- *Further investigation on representation of structures* (6 months):
  - Regions that are canonizable and properly sampled are currently represented by square patches. These patches often overlap on the image domain. A selection process is required to choose the best patches for each frame. In the existing scheme there is no selection process.
  - Currently only corners that appear in the image frames are tracked. In video compression and other applications though, there is benefit in tracking both corners and edges (optional direction).
- *Further investigation on textures* (9 months):
  - *Texture compression*: Revisit inference of $\bar{\omega}$. Revise and speed up algorithm. Possibly consider other possible representations for both $\bar{\omega}$ and ω.
  - *Texture synthesis*: Investigate possible directions that might enable speeding up texture synthesis.
  - *Texture segmentation*: Pixel-wise boundary segmentation of textures is required in order to take advantage of the other tools developed. Explore different directions such as expansion of regions from seeds or an MRF formulation.
  - *Dynamic Textures*: Textures synthesized for the purpose of video compression need to exhibit temporal continuity (i.e. sea needs to be moving from frame to frame rather than appear static).
  - *Texture recognition experiments* : Use the texture representation developed for texture recognition experiments.
- *Investigate texture / structure partition* (6 months): Once texture segmentation is completed, it is necessary to check whether structured regions exhibit spatial stationarity and hence might be part of a texture at a larger scale. Hence the texture / structure partition of an image frame needs to be further investigated and algorithms implementing it need to be developed.
- *Investigate perceptual metrics* (3 months): It is necessary to use a metric that will evaluate both the compressed representation of textures, the synthesized textures but also the reconstructed video based on our compression systems. Metrics proposed up to now in the literature are found to be inadequate for our needs. A need to develop a new metric arises and we will try to investigate whether a perceptual metric is possible to be designed that will be closer to the human visual system.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. *CVPR*, 2009. 21

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 2011. 5, 21, 36

[3] A. Ayvaci, M. Raptis, and S. Soatto. Sparse occlusion detection with optical flow. *IJCV*, 2012. 5, 34, 45

[4] A. C. Beers, M. Agrawala, and N. Chaddha. Rendering from compressed textures. In *ACM SIGGRAPH*, 1996. 5

[5] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *TIP*, 12(8):882–889, 2003. 5

[6] M. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise smooth flow fields. *Computer Vision and Image Understanding*, 1996. 5

[7] S. Boltz, F. Nielsen, and S. Soatto. Texture regimes for entropy-based multiscale image analysis. *ECCV*, 2010. 14

[8] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *ECCV*, 2004. 5

[9] T. Chan and L. Vese. Active contours without edges. *IEEE TIP*, 2001. 5

[10] A. Chiuso, R. Brockett, and S. Soatto. Optimal structure from motion:local ambiguities and global estimates. *IJCV*, 2000. 48

[11] CISCO. Entering the zettabyte era, visual networking index. CISCO VNI, 2011. 4

[12] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 2002. 5

[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991. 8

[14] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. *ICCV*, 1999. 5, 13

[15] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *Proc. of ACM SIGGRAPH*, 2001. 5, 6

[16] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 5

[17] S. Fenney. Texture compression using low-frequency signal modulation. In *ACM SIGGRAPH*, 2003. 5

[18] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 1981. 47, 51

[19] G. Georgiadis, A. Chiuso, and S. Soatto. Texture, structure and visual matching. *Submitted to NIPS*, 2012. 23, 26, 27, 28, 29, 32

[20] J. J. Gibson. *The ecological approach to visual perception*. LEA, 1984. 44

[21] C. E. Guo, S. C. Zhu, and Y. N. Wu. Towards a mathematical theory of primal sketch and sketchability. *ICCV*, 2003. 5, 8

[22] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 2005. 5

[23] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision*, 1988. 51

[24] E. Hayman and J. Eklundh. Statistical background subtraction for a mobile observer. *ICCV*, 2003. 6

[25] D. Heeger and A. Jepson. Subspace methods for recovering rigid motion i. *IJCV*, 1992. 48, 49, 51, 54

[26] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 1981. 5

[27] P. Huber. *Robust statistics*. Wiley, New York, 1981. 47

[28] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *TPAMI*, 1998. 6

[29] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking of multiple moving objects using temporal integration. *ECCV*, 1992. 47

[30] L. Itti and P. Baldi. Bayesian surprise attracts human attention. *Vision research*, 2009. 44

[31] ITUT-Recommendations. http://www.itu.int/itu-t/recommendations/. 1, 5, 23, 40

[32] Y. Ivanov, C. Stauffer, A. Bobick, and W. Grimson. Video surveillance of interactions. *CVPR*, 1999. 44

[33] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 1991. 5

[34] B. Julesz. *Foundations of Cyclopean Perception*. University of Chicago Press, 1971. 8

[35] A. Kumar, V. Kwatra, B. Singh, and S. Kapoor. Dynamic binary space partitioning for hidden surface removal. *Indian Conference on Computer Vision, Graphics and Image Processing*, 1998. 6

[36] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *Proc. of ACM SIGGRAPH*, 2005. 5, 6, 9, 15, 16, 28, 38

[37] T. Lee and S. Soatto. Video-based descriptors for object recognition. *Image and Vision Computing*, 2011. 10, 21, 24, 36, 37

[38] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 2001. 18

[39] W. Li-Yi, L. S. ans K. Vivek, and T. Greg. State of the art in example-based texture synthesis. *Eurographics*, 2009. 6

[40] L. Liang, C. Liu, Y. Xu, B. Guo, and H. yeung Shum. Real-time texture synthesis by patch-based sampling. *Proc. of ACM Transactions on Graphics*, 2001. 5, 6

[41] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. *ECCV*, 2008. 5

[42] Y. Liu, Y. Tsin, and W. C. Lin. The promise and perils of near-regular texture. *IJCV*, 2005. 11, 22

[43] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An invitation to 3D vision, from images to models*. Springer Verlag, 2003. 33

[44] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *JOSAA*, 1990. 6

[45] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *PAMI*, 2002. 5

[46] P. Mavridis and G. Papaioannou. Texture compression using wavelet decomposition. *Computer Graphics Forum (Proceedings of Pacific Graphics 2012)*, 2012. 5

[47] A. Moorthy and A. Bovik. Visual quality assessment algorithms: what does the future hold? *Multimedia Tools and Applications*, 2011. 6

[48] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, 2000. 6

[49] Y. Ren, C. Chua, and Y. Ho. Statistical background modeling for non-stationary camera. *PRL*, 2003. 6

[50] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *CVPR*, 2006. 5

[51] H. Sawhney, Y. Guo, and R. Kumar. Independent motion detection in 3d scenes. *TPAMI*, 2000. 6

[52] K. Seshadrinathan. *Video quality assessment based on motion models*. PhD thesis, The University of Texas at Austin, 2009. 6

[53] H. Sheikh and A. Bovik. Image information and visual quality. *Image Processing, IEEE Transactions on*, 2006. 6

[54] Y. Sheikh, O. Javed, and T. Kanade. Background subtraction for freely moving cameras. *ICCV*, 2009. 7, 44, 51, 52, 54

[55] D. Skocaj and A. Leonardis. Weighted and robust incremental method for subspace learning. *ICCV*, 2003. 47

[56] N. Slonim and N. Tishby. Agglomerative information bottleneck. *NIPS*, 1999. 16, 29

[57] S. Soatto. Actionable information in vision. *ICCV*, 2009. 44, 45

[58] S. Soatto. *Steps Toward a Theory of Visual Information*. ArXiv http://arxiv.org/abs/1110.2053, Technical Report UCLA-CSD100028, September 13, 2010. 4, 10, 12, 13, 23, 24, 36

[59] S. Soatto, R. Frezza, and P. Perona. Motion estimation via dynamic vision. *ECCV*, 1994. 47

[60] P. H. Suen and G. Healey. Analyzing the bidirectional texture function. *CVPR*, 1998. 5

[61] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. *ECCV*, 2010. 5, 36, 37, 51, 54

[62] G. Sundaramoorthi, P. Petersen, V. S. Varadarajan, and S. Soatto. On the set of images modulo viewpoint and contrast changes. *CVPR*, 2009. 33

[63] M. Tao, M. Johnson, and S. Paris. Error-tolerant image compositing. *ECCV*, 2010. 32

[64] P. Teo and D. Heeger. Perceptual image distortion. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*. IEEE, 1994. 6

[65] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 1999. 8, 13, 26

[66] C. Tomasi and J. Shi. Good features to track. *CVPR*, 1994. 5

[67] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. *CVPR*, 2007. 51

[68] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? *CVPR*, 2003. 5

[69] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. 2008. 30

[70] R. Vidal. Generalized principal component analysis (gpca). *CVPR*, 2003. 44, 47

[71] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Perceptual image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 2004. 6, 18, 20

[72] Y. N. Wu, C. Guo, and S. C. Zhu. From information scaling of natural images to regimes of statistical models. *Quarterly of Applied Mathematics*, 2008. 5, 8, 37

[73] T. Xin, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *ACM Press*, 2002. 6

[74] C. Yuan, G. Medioni, J. Kang, and I. Cohen. Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. *TPAMI*, 2007. 6

[75] Z. Zhang, X. Liang, A. Ganesh, and Y. Ma. Tilt: Transform invariant low-rank textures. In *ECCV*, 2010. 12, 13

[76] H. Zhi, Z. Xu, and S.-C. Zhu. Video primal sketch: A generic middle-level representation of video. *ICCV*, 2011. 5, 23, 32

[77] S. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (frame):towards the unified theory for texture modeling. *CVPR*, 1996. 17

A Appendix

To calculate $\nabla\psi(V) = \left[\frac{\partial\psi(V)}{\partial V}\right]^T$ we use the following conventions. The derivative of a function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is given by an $m \times n$ matrix of partial derivatives $[Df_{ij}] = \frac{\partial f_i(x)}{\partial x_j}$. For $A \in \mathbb{R}^{n\times m}$ and $f\colon \mathbb{R}^{n\times m} \to \mathbb{R}^{p\times q}$ the derivative is given by $\frac{\partial f(A)}{\partial A} = \frac{\partial \text{vec}(f(A))}{\partial \text{vec}(A)} \in \mathbb{R}^{pq\times mn}$ where the vec operator stacks the columns of a matrix on top of each other. Using these definitions and common rules for chain and product rules we can derive $\left[\frac{\partial\psi(V)}{\partial V}\right]^T$. We can decompose $\frac{\partial\psi(V)}{\partial V}$ as follows:

$$\frac{\partial\psi(V)}{\partial V} = \frac{\partial\psi(\hat{C})}{\partial \text{vec}(\hat{C})}\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)}\frac{\partial \text{vec}(\mathcal{W}C)}{\partial \text{vec}(C)}\frac{\partial \text{vec}(C(V))}{\partial V} \tag{37}$$

The four terms are given by the following equations:

$$\frac{\partial\psi(\hat{C})}{\partial \text{vec}(\hat{C})} = (\mathcal{W}x)^T \otimes (\mathcal{W}x)^T\hat{C} \tag{38}$$

Using the product rule we can get $\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)}$. Define $f_1(\mathcal{W}C) = \mathcal{W}C$ and $g_1(\mathcal{W}C) = (C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T$. We then have:

$$\begin{aligned}
\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)} &= \frac{\partial}{\partial \text{vec}(\mathcal{W}C)}\left(I - \mathcal{W}C(C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T\right) && (39)\\
&= -\frac{\partial}{\partial \text{vec}(\mathcal{W}C)}\mathcal{W}C(C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T && (40)\\
&= -\left[\left(g_1(\mathcal{W}C)^T \otimes I_{2N}\right) f_1'(\mathcal{W}C) + \left(I_{2N} \otimes \mathcal{W}C\right) g_1'(\mathcal{W}C)\right] && (41)
\end{aligned}$$

The derivative of $f_1(\mathcal{W}C)$ with respect to $\mathcal{W}C$ is simply $f_1'(\mathcal{W}C) = I_{2N(N+3)}$. For the derivative of $g_1(\mathcal{W}C)$ we need to use the product rule. Define $f_2(\mathcal{W}C) = \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}$ and $g_2(\mathcal{W}C) = (\mathcal{W}C)^T$. Then we can calculate the derivative of $g_1(\mathcal{W}C)$ using the following:

$$g_1'(\mathcal{W}C) = (g_2(\mathcal{W}C)^T \otimes I_{N+3})f_2'(\mathcal{W}C) + (I_{2N} \otimes f_2(\mathcal{W}C))g_2'(\mathcal{W}C) \tag{42}$$

The derivative of $g_2(\mathcal{W}C)$ is $g_2'(\mathcal{W}C) = T_{2N,N+3}$, where $T_{N,M} \in \mathbb{R}^{MN\times MN}$ is a permutation matrix such that

$$T_{N,M}\text{vec}(A) = \text{vec}(A^T) \tag{43}$$

To calculate the derivative of $f_2(\mathcal{W}C)$ we need to use the chain rule. Define $f_3(X) = X^{-1}$ and $g_3(\mathcal{W}C) = (\mathcal{W}C)^T\mathcal{W}C$. So, $f_3(g_3(\mathcal{W}C)) = f_2(\mathcal{W}C)$. The derivative of $f_2'(\mathcal{W}C)$ is:

$$f_2'(\mathcal{W}C) = f_3'((\mathcal{W}C)^T\mathcal{W}C)g_3'(\mathcal{W}C) \tag{44}$$

From standard results:

$$f_3'((\mathcal{W}C)^T\mathcal{W}C) = -(((\mathcal{W}C)^T\mathcal{W}C)^{-T} \otimes ((\mathcal{W}C)^T\mathcal{W}C)^{-1}) \quad (45)$$

$$g_3'(\mathcal{W}C) = \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) \quad (46)$$

Therefore:

$$\begin{aligned}
f_2'(\mathcal{W}C) &= f_3'((\mathcal{W}C)^T\mathcal{W}C)g_3'(\mathcal{W}C) && (47)\\
&= -(((\mathcal{W}C)^T\mathcal{W}C)^{-T} \otimes ((\mathcal{W}C)^T\mathcal{W}C)^{-1}) && (48)\\
&\quad \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && (49)\\
g_1'(\mathcal{W}C) &= (g_2(\mathcal{W}C)^T \otimes I_{N+3})f_2'(\mathcal{W}C) && (50)\\
&\quad +(I_{2N} \otimes f_2(\mathcal{W}C))g_2'(\mathcal{W}C) && (51)\\
&= -(\mathcal{W}C \otimes I_{N+3}) && (52)\\
&\quad \times \left(\left((\mathcal{W}C)^T\mathcal{W}C\right)^{-T} \otimes \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}\right) && (53)\\
&\quad \times \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && (54)\\
&\quad + \left(I_{2N} \otimes \left((\mathcal{W}C)^T(\mathcal{W}C)\right)^{-1}\right) T_{2N,N+3} && (55)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)} &= -\left(g_1(\mathcal{W}C)^T \otimes I_{2N}\right) f_1'(\mathcal{W}C) && (56)\\
&\quad -\left(I_{2N} \otimes \mathcal{W}C\right) g_1'(\mathcal{W}C) && (57)\\
&= -\left[\left((C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T\right)^T \otimes I_{2N}\right) I_{2N(N+3)}] && (58)\\
&\quad +(I_{2N} \otimes \mathcal{W}C) && (59)\\
&\quad \times (\mathcal{W}C \otimes I_{N+3}) && (60)\\
&\quad \times \left(\left((\mathcal{W}C)^T\mathcal{W}C\right)^{-T} \otimes \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}\right) && (61)\\
&\quad \times \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && (62)\\
&\quad + \left(I_{2N} \otimes \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}\right) T_{2N,(N+3)} && (63)\\
& && (64)
\end{aligned}$$

The derivative $\frac{\partial \text{vec}(\mathcal{W}C)}{\partial \text{vec}(C)}$ is given by:

$$\frac{\partial \text{vec}(\mathcal{W}C)}{\partial \text{vec}(C)} = I_{N+3} \otimes \mathcal{W} \quad (65)$$

In addition $\frac{\partial \text{vec}(C(V))}{\partial V}$ is given by

$$\frac{\partial \text{vec}(C(V))}{\partial V} = \begin{bmatrix} A_1 \\ O_{2N,3} \\ A_2 \\ O_{2N,3} \\ \vdots \\ A_N \\ O_{6N,3} \end{bmatrix} \quad (66)$$

which is the final term of the derivative of $\frac{\partial \psi(V)}{\partial V}$. $O_{M,N}$ is an $M \times N$ matrix with all elements equal to 0.