University of California

Los Angeles

Scene Representations for Video Compression

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Georgios Georgiadis

2015

Abstract of the Dissertation

Scene Representations for Video Compression

by

Georgios Georgiadis

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2015

Professor Stefano Soatto, Chair

Video analysis has evolved into a fundamental research problem following an unprecedented growth of video content consumption in the last few years. Its applications span a wide range of tasks including video editing, indexing, classification, surveillance and autonomous driving. While for these tasks, representations that capture the main properties of the scene have been investigated, this has still not been done for the purpose of video compression. Current video compression systems largely ignore the underlying scene and instead only model its projection on the video frames. The goal of this thesis is to show that by modeling the scene further improvements can be achieved in compressing video data.

In the first part of the thesis, it is shown how a scene can be decomposed into two types of regions called "Visual Structures" and "Visual Textures". These regions are defined and algorithms that allow their inference are proposed. Based on such partition, a video compression system is designed that achieves a significant improvement over current state-of-the-art methods. In addition, the definition of textures is expanded to model transformations of the domain and range of the image and applications such as texture compression and segmentation are discussed.

In the second part, extensions of the video compression system are proposed that take advantage of further modeling of the scene, such homogeneity, occlusions and tight boundaries of regions. Finally, further applications of scene modeling are shown in various problems such as video hole-filling and independent motion detection.

The dissertation of Georgios Georgiadis is approved.

Alan Yuille

Ying Nian Wu

Stanley Osher

Stefano Soatto, Committee Chair

University of California, Los Angeles

2015

To Ewelina, my wife.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would first like to thank my advisor Prof. Stefano Soatto, for giving me the opportunity to work in the Vision Lab and for trusting me in making my own research decisions. His guidance has been invaluable and our discussions have been intellectually stimulating and will be forever memorable. I would also like to thank all members of my doctoral committee, Profs. Stanley Osher, Ying Nian Wu, Alan Yuille as well as Prof. Zhuowen Tu, a former member of the committee, for their constructive comments, suggestions and support. Special thanks also go to Profs. Ganesh Sundaramoorthi and Alessandro Chiuso for valuable discussions.

I am extremely grateful for having the opportunity to work alongside Alper Ayvaci, Avinash Ravichandran and Michalis Raptis, who have been guiding, supporting and helping me during my early years as a Ph.D. student. Special thanks go to Vasiliy Karasev. His help has been invaluable and his criticisms have unquestionably improved the quality of my work.

During my Ph.D. years at UCLA, I had the opportunity to meet a set of extraordinary people: Among others, former members Kamil Wnuk, Taehee Lee, Chaohui Wang, Jonathan Balzer and Joshua Hernandez, as well as current ones, Brian Taylor, Virginia Estellers, Konstantine Tsotsos, Nikos Karianakis and Jingming Dong. Working together will be an unforgettable experience.

I would also like to thank my parents, Dora and Nikos, and my sister Marilena who have supported me emotionally and enabled me to follow my dreams all these years. Finally, I would like to thank my wife, Ewelina who has endured me during my entire doctoral studies. Her patience and support were extraordinary. Without a doubt, my Ph.D. journey would have been extremely tedious and mundane without her by my side.

Vita

2008	M.Eng. in Electrical and Electronic Engineering, Imperial College, London
2009	Visiting Student Research Collaborator, Princeton University.
2010	M.S. in Electrical Engineering, Stanford University.
2012	Teaching Assistant, Computer Science Department, University of California, Los Angeles.
2013	M.S. in Computer Science, University of California, Los Angeles.
2013	Emerging Graphics Group Intern, Adobe Systems Inc.
2014	Technical Consultant, HBO Inc.
2014, 2015	Research Intern, Dolby Laboratories.

Publications

1. G. Georgiadis, A. Chiuso, S. Soatto, "Texture Representations for Image and Video Synthesis", In *IEEE Conference on Computer Vision and Pattern Recognition, 2015.*

2. G. Georgiadis, S. Soatto, "Exploiting Temporal Redundancy of Visual Structures for Video Compression", In *Data Compression Conference, 2015.*

3. G. Georgiadis, A. Chiuso, S. Soatto "Texture Compression", In *Data Compression Conference, 2013.*

4. G. Georgiadis, A. Ravichandran, S. Soatto and A. Chiuso "Encoding Scene Structures for Video Compression", In *Proc. SPIE Int. Soc. Opt. Eng. 8499, 2012.*

5. G. Georgiadis, A. Ayvaci and S. Soatto "Actionable Saliency Detection: Independent Motion Detection Without Independent Motion Estimation", In *IEEE Conference on Computer Vision and Pattern Recognition, 2012.*

6. G. Georgiadis and S. Soatto "Scene-Aware Video Modeling and Compression", In *Data Compression Conference, 2012.*

7. L. Lai, H. V. Poor, Y. Xin and G. Georgiadis "Quickest search over multiple sequences", In *IEEE Transactions on Information Theory, 2011.*

8. L. Lai, H. V. Poor, Y. Xin and G. Georgiadis "Quickest Sequential Opportunity Search in Multichannel Systems", In *International Workshop on Applied Probability, 2010.*

9. G. Leseur, N. Meunier, G. Georgiadis, L. Huang, J. DiCarlo, B. Wandell and P. B. Catrysse "High-speed document sensing and misprint detection in digital presses", In *Proc. SPIE Int. Soc. Opt. Eng. 7536, 2010.*

CHAPTER 1

Introduction

With video acquisition methods becoming more varied, complex and ubiquitous, the need to design algorithms for video analysis is leading the technology sector into new territories. Video content consumption is no longer limited through specific, traditional mediums such as the movie theater or the television set, but also include mobile phones, tablets and even virtual reality headsets. Both video content and viewing conditions are extremely varied and complicated to predict. This variety imposes an extremely challenging problem to video analysis algorithms as they are now faced with a lot of unknowns to deal with. However, such variety is in fact the reason why we need automation in the first place.

With a large amount of video content available on the Internet, users require better and faster retrieval mechanisms, better organization and more information about the content. To do that a series of problems needs to be solved, such as video retrieval, indexing, classification, recognition and localization. At the same time, similar problems need to be solved for video editing both in professional and amateur settings, autonomous driving and other applications. Such video analysis tools are no longer a desire, but instead a necessity. Recently, an increasing number of solutions have been appearing for most of these problems. However, an important problem remains unaddressed; Internet bandwidth is under unprecedented pressure from video transmission.

Cisco projects [CIS14] that by 2019, 105.2 Exabytes of video will be traveling the Internet every month. That is in fact equivalent to 5 million years of video content being played per month. In addition, Internet video traffic will comprise 77% of all Internet traffic. It is therefore undeniable that video content will dominate Internet traffic in the next few years. On top of that, higher video resolutions will place a huge pressure over the already over-burdened Internet bandwidth. The question that then arises is whether the current video compression systems can cope with such demand. To answer that, we would need to briefly overview the current video compression standards.

1.1 Overview of video compression systems

Since H.262/MPEG-2, ITU-T and ISO/IEC have been collaborating together to produce a universal video compression standard, that has helped the video industry standardize video playback mechanisms. Their most successful effort has lead to H.264/MPEG-4 AVC that was designed between 1999 and 2003. Recently, with the rise of higher video resolutions and more playback devices, the two groups have united to develop a new video compression standard called High Efficiency Video Coding (HEVC). From start, the new standard has been designed to cope with increased video resolutions and the use of parallel processing architectures [SOH12].

Since H.261, video compression standards have been adhering to a hybrid approach that performs inter/intra frame prediction and 2-D transform coding. Each frame is broken down into non-overlapping blocks. Each block could be coded using either inter-frame or intra-frame prediction, with the first type being the most common. The residual error is then transformed, scaled, quantized, entropy-coded and then transmitted to the receiver (decoder). In this architecture, the encoder contains the decoding processing loop in order to avoid any drift between the encoder and decoder [SOH12].

In these approaches, a large effort has been put into dividing the input to appropriate structures in order to maximize the compression gains. HEVC utilizes coding tree units (CTU), whereas previous standards depended on macroblocks. CTUs can be split into various sizes for better compression and can be bigger than traditional macroblocks. Motion compensation allows prediction of blocks from ones in other frames of the video, while intra-frame prediction supports 33 directional modes (instead of 8 in H.264) [SOH12].

Overall, traditional video compression systems operate directly on the pixel values, essentially ignoring the scene structure. Doing so, they optimize locally and ignore any semantic information that could be utilized for further improvements. This setting has been heavily explored in the last few decades and latest improvements return diminishing returns. Such "low-level" approaches could be argued that have reached a local optimal point in terms of performance. In order to break the so-called "compression-wall", video compression needs to move towards a different direction. For example, a different type of video modeling could be one where the encoder is not necessarily interested in reproducing exact intensity values of a frame, but instead is interested in reproducing what is perceivable by the human eye in that particular frame. In particular, we could consider a system that models the scene depicted in a video that is able to reconstruct each video frame up to a level that makes the reconstruction appear indistinguishable from the input frame. Such modeling could potentially provide not only further improvements in video compression, but it could also output useful information for other video analysis applications.

There have been a few approaches in the video compression literature that attempted to operate on the scene structure to achieve a better compression ratio. Even though the following chapters will provide a thorough discussion of related approaches, a brief introduction will be provided in this section to introduce the reader to the main topics in the literature. In particular, [NDK12] provides a good survey of such attempts. Perceptual-oriented video coding approaches rely on mechanisms that aim to optimize a perceptual metric that measures the similarity of input and reconstructed videos. Such methods typically follow the standard hybrid approach as far as the encoder is concerned, with the difference that certain regions of frames are not encoded. An image completion system is typically utilized at the decoding side to fill in the missing data [NBW09]. The image completion module is typically implemented using techniques from inpainting and/or texture synthesis. Parametric image completion approaches [PS00, HB95, DCW03] approximate the probability desnity function (pdf) of a texture using a compact parameter set, whereas non-parametric methods [KEB05, EL99] synthesize novel instances of the texture by sampling samples of the stochastic process directly from the input texture. The image completion module is responsible to respect boundary conditions and fill in the missing data in a way that is consistent with the already reconstructed video frames.

Since image completion methods do not reconstruct the original intensity values exactly, but only capture the region's statistical properties, it is important to utilize such systems only on regions that reconstructing their intensity values is perceptually irrelevant. These regions should be characterizable by their statistical properties, rather than by individual realizations. Therefore, perceptual-oriented video coding approaches need to be evaluated using a perceptual metric, since otherwise, they would perform poorly at regions where the image completion module was used. Approaches to approximate perceptual metrics such as the Structural Similarity index (SSIM) [WBS04] and Visual Information Fidelity (VIF) [SB06] have had some success, however none has replaced the peak signal-to-noise ratio (PSNR) in traditional comparisons. However, it is well known that PSNR is a poor predictor of perceptual similarity [WB09]. Methods that are interested in perceptual similarity typically employ human subjects to perform some type of subjective evaluation.

Other methods include works such as “Video Primal Sketch”, which is an extension of primal sketch to video [ZXZ11]. It partitions video frames to “structures” and “textures” and achieves improved performance over H.264. Structures are represented using a basis of functions, whereas textures are represented using a set of histograms. The parametric approach to both “structures” and “textures” limits the generalizability of the algorithm, since the parametric models chosen were generally hand-crafted for the specific dataset used. Other approaches that could be utilized in compression include texture modeling approaches such as [WHZ08] that are concerned with “inverting” the texture synthesis process.

1.2 Thesis outline

Video content consumption through the Internet (or through other bandwidth-sensitive mediums) became a reality due to the existence of lossy compression mechanisms. Lossy compression is feasible in video, due to image properties that allow relatively large reconstruction errors without significant penalty in perceptual similarity. This is unlike other data, that even small errors could render the compressed output corrupt. The current video compression standards rely therefore on lossy compression and modeling of pixel values. However, this second paradigm is not necessary for achieving a high perceptual similarity. Instead, it is possible to model higher level notions, such as the scene and still be able to achieve high quality reconstructed videos. This type of modeling could be able to offer higher compression ratios, thus allowing the infrastructure to cope with the ever-increasing demand.

The approach followed in this thesis is based on the proposal that video frames are a projection of the scene. As such, by modeling the scene, we should be able to reconstruct each video frame reliably and accurately. Certain characteristics of the scene are extremely important for video compression, such as photometry, dynamics, geometry, topology and statistical regularities. By modeling accurately such properties, it should be possible to achieve a higher compression ratio than by simply modeling the intensity values locally. This is based on the intuition that regions of the scene are repeatable in multiple frames or exhibit spatial regularity. By identifying such regions we could encode them once for the entire duration of the video and then simply use the model to reconstruct them at decoding time. In general, the projection of the scene onto video frames would be assumed to induce two types of regions: "structures" and "textures". In the rest of the thesis, it will be shown how each type of region can be modeled and how the two of them can be combined to provide a video compression encoder/decoder system.

In Chapter 2, a video coding system is presented that partitions the scene into "visual structures" and a residual "background" layer. The system exploits the temporal redundancy of visual structures to compress video sequences. A dictionary of track-templates is constructed, which corresponds to a representation of visual structures. A subset of the dictionary's elements is chosen to encode video frames using a Markov Random Field (MRF) formulation that places the track-templates in "depth" layers. The video coding system offers an improvement over H.265/H.264 and other methods in a rate-distortion comparison.

In Chapter 3, a general treatment on textures is proposed. Specifically, it is known that in texture synthesis and classification, algorithms require a small texture to be provided as an input, which is assumed to be representative of a larger region to be re-synthesized or categorized. Such textures are defined and their characterizations are automatically retrieved. Most works generate these small input textures manually by cropping, which does not ensure maximal compression, nor that the selection is the best representative of the original. A new representation is constructed that compactly summarizes a texture, while using less storage, that can be used for texture compression and synthesis. The representation is also integrated in a proposed video texture synthesis algorithm to generate novel instances of textures and video hole-filling. Finally, a novel criterion is proposed that measures structural and statistical dissimilarity between textures.

In Chapter 4, the definition of "visual textures" is expanded to handle general transformations of the domain and range of an image. Inference algorithms are proposed for estimating the "state" of such textures and their "variability". This mechanism represents the encoding stage. A non-parametric sampling scheme is proposed for decoding, by synthesizing textures from their encoding. While these are not faithful reproductions of the original textures (so they would fail a comparison test based on PSNR), they capture the statistical properties of the underlying process, as we demonstrate empirically. Finally, the tradeoff between fidelity (measured by a proxy of a perceptual score) and complexity is quantified.

In Chapter 5, an image segmentation algorithm is presented that generates multiple foreground/background partitions of the image by local region-based segmentations. The regions are evolved by comparing aggregated local statistics with their surrounding neighborhoods. But rather than aggregating statistics computed at each scale, a scale is selected based on the local image structure. The resulting multiple segmentations can be grouped into one partition using pairwise affinities. Results of the method are compared to human annotations and to alternate approaches using several common metrics.

In Chapter 6 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 is described. 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.

In Chapter 7 a video compression methodology is described 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 the scheme involves occlusion detection, optical flow computation, texture/structure partition, and a notion of proper sampling. Results are shown that exceed baseline compression performance, albeit at an increased computational cost.

In Chapter 8 a different application of video analysis is explored. A model and an algorithm to detect salient regions in video taken from a moving camera is presented. In particular, the focus of this study was on 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 existing schemes based on background subtraction (background motion too complex), multi-body motion estimation (insufficient parallax), and occlusion detection (uniformly textured background regions). A robust statistical inference approach is adopted to simultaneously estimate a maximally reduced regressor, and select regions that violate the null hypothesis (co-visibility under an epipolar domain deformation) as "salient". The 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. The algorithm is demonstrated on video footage from helicopters, airplanes, and ground vehicles.

In Chapter 9 a summary of the findings is provided. For video compression to move forward it is necessary to take into account the phenomenology of the data formation process. In particular, rather than compressing the images, one should compress the scene. Furthermore, scaling and occlusion phenomena play a critical role and they are thoroughly explored in this thesis.

CHAPTER 2

Visual Structures

With an ever-increasing video consumption rate on the Internet, we are faced with a continuously increasing pressure on available bandwidth [Ric10]. While the new H.265 [ITU] has improved performance over existing standards, the majority of video compression techniques have traditionally been confined in modeling and predicting pixel values of video frames. We consider an alternative to traditional coding schemes, where we assume that a video has been generated by an underlying scene. Our aim is to model and compress the source (scene) rather than the output (pixel values).

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

It has been previously argued that an image can be partitioned into structures and textures ([GZW03a, SCV02, BVS03]) based on statistics computed in that *one* image. We test whether image structures arise from properties of the scene, by leveraging on the notion of *proper sampling* [Soa10]. Proper sampling requires multiple images of the same scene to determine whether a structure is "real" in the sense of corresponding to something in the scene or an "alias", an artifact of nuisance factors in the image formation process. We model and compress those that satisfy this test and allow a standard video encoder to compress the rest. Finally, partitioning the scene into various types of regions for video coding has also been previously proposed [NBW09, GS12, WA93]. However these methods do not model "visual structures" to take advantage of their temporal redundancy.

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

2.1 Visual Structures

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

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

If we call $\nabla\psi \doteq \frac{\partial \psi}{\partial g}$ the gradient of the functional ψ with respect to (any) parametrization of the group, then under the "transversality" conditions $\det(\nabla\nabla\psi) \neq 0$, the equation $\nabla\psi = 0$ locally determines a unique function g (a *canonical representative*) of I, $g = \hat{g}(I)$, via the Implicit Function Theorem. If the canonical representative co-varies with the group, in the sense that $\hat{g}(I \circ g) = (\hat{g} \circ g)(I)$, then the functional ψ is called a *co-variant detector* (e.g. Laplacian-of-Gaussian (LoG) and the difference-of-Gaussians (DoG)). Varying σ produces a *scale-space*, whereby the locus of extrema of ψ describes a *graph* in $\mathbb{R}^3$, via $(p, \sigma) \mapsto \hat{p} = \hat{g}(I; \sigma)$. Starting from the smallest σ, one would have a large number of extrema; as σ increases, extrema will merge or disappear. Although in a two-dimensional scale space, extrema can also appear as well as split, they are increasingly rare as scale increases, so the locus of extrema as a function of scale is well approximated by a tree, called the *co-variant detection tree* [LS11]. A region $\omega \subset D$ is *canonizable* at scale σ if there exists a co-variant detector ψ that has one and only one isolated extremum in ω at that scale. We call this region a "visual structure". The region may be canonizable at multiple scales.

Canonization yields a number of regions each containing exactly one "structure". An image is properly sampled if any co-variant detector functional operating on the sampled image $\{I_{xy}\} \in \mathbb{R}^{X \times Y}$ yields the "same answer" (topology) that it would if ran on the "original" (continuous) image $I : D \to \mathbb{R}$. Assuming co-visibility, Lambertian reflection and constant illumination, topological equivalence of co-variant detector responses between the scene and the image can be replaced by that between *different images of the same scene.* Thus, two temporally adjacent images are *properly sampled* at scale σ_0 in a region ω if, for all scales $\sigma \geq \sigma_0$, there exists a one-to-one correspondence between covariant detection trees in ω [Soa10].

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

In practice, to determine the trackable regions, we use a feature point tracker such as [LS11]. Which regions are classified as trackable regions, depends on the detection threshold of each method. The effects of the threshold are visible in Fig. 2.1, where the number of tracks decreases by increasing the threshold. The ones that persist are usually the longest and most stable, a fact which we exploit for video compression.

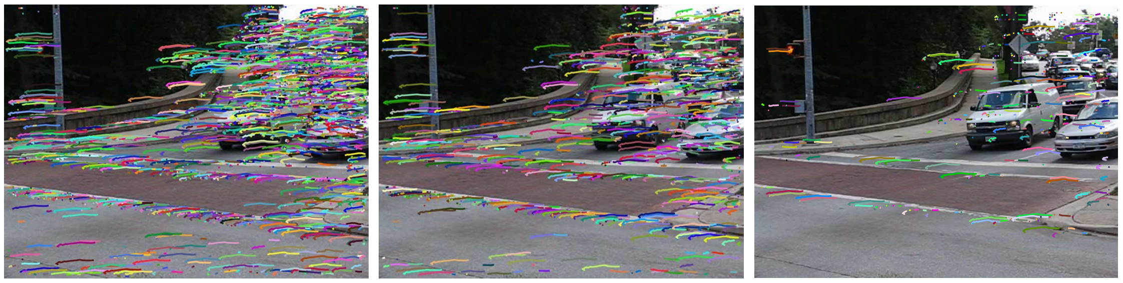

Figure 2.1: Varying the co-variant detection threshold produces different densities of trackable regions. There are typically three regions of interest, shown in the images. The tracks that persist through a wide a range of thresholds are typically the longest and most accurate.

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

2.1.1 Structure Representation

A trackable region, with index k, that appears in frames t_1 to t_2, can be represented losslessly by $\hat{F}_k = \{F_k(t_1), \ldots, F_k(t_2)\}$, where $F_k(t) \doteq \{I_{xy}(t), \forall(x, y) \in \omega_\sigma^k(t)\}$. $F_k(t)$ corresponds to the intensity values at pixel locations (x, y) in a neighborhood ω^k at scale (area) σ at time t. The feature point tracker [LS11] provides a set of regions $\mathcal{F} = \{\hat{F}_1, \ldots, \hat{F}_K\}$, where K is the number of trackable regions. We model the trackable regions (and structures) in a video, using a time-invariant dictionary element for each region that is of the same size as the region itself. We consider two alternative time-invariant representations, which we call the "track-template":

$$(a) \quad H_k^{(avg)}(\hat{F}_k) \doteq \frac{1}{\mathcal{T}} \sum_{t=t_1}^{t_2} F_k(t) \qquad\qquad (b) \quad H_k^{(fst)}(\hat{F}_k) \doteq F_k(1) \ , \tag{2.1}$$

where $\mathcal{T} = t_2 - t_1 + 1$. $H_k^{(fst)}$ is simply the intensity values of the track in the first frame it appears. In the mean-squared-error sense, the best one in minimizing the reconstruction error is $H_k^{(avg)}$. In Sec. 2.2.1, we show that by incorporating our method in H.265 we outperform H.265 in a rate-distortion comparison. For practical reasons (explained in Sec. 2.2.1), we are constrained to use $H_k^{(fst)}$.

One track-template is stored for each trackable region. The collection of all the track-templates, $\{H_1, \ldots, H_K\}$, from a video forms a dictionary, where the scale of each dictionary element is naturally selected to be the coarsest scale at which the track was detected. In Fig. 2.2, we show elements of the dictionary for a particular video. The track-templates introduce a compression gain at the expense of fidelity. For comparison, if we were to use $\hat{F}_k$ to represent the trackable regions, the distortion would have been 0, but the cost of encoding a track would have been $\beta \times (\sigma + 4) \times \mathcal{T}$, where β is a constant representing the cost of storing a double (i.e. $\beta = 8$ bytes), σ is the scale of the track in space and 4 is the number of parameters of the track (x, y, t, k). The track-template instead only requires $\beta \times (\sigma + 4\mathcal{T})$. Hence the compression ratio is $\xi = \frac{(\sigma+4)\mathcal{T}}{\sigma+4\mathcal{T}}$. Note that a compression gain is achieved (i.e. $\xi \geq 1$) for $\sigma \geq 0$ and for $\mathcal{T} \geq 1$. We measure the distortion introduced by computing the dissimilarity of the representation $H_k(\hat{F}_k)$ from each instance of the track $F_k(t)$:

$$q(H_k(\hat{F}_k), \hat{F}_k) = \frac{1}{\mathcal{T}} \frac{1}{\sigma} \sum_{t=t_1}^{t_2} \| H_k(\hat{F}_k) - F_k(t) \|^2 \ , \tag{2.2}$$

where $\|.\|$ denotes the Euclidean norm. This expression computes the average squared distance per pixel from the representation to the instances of the track. If

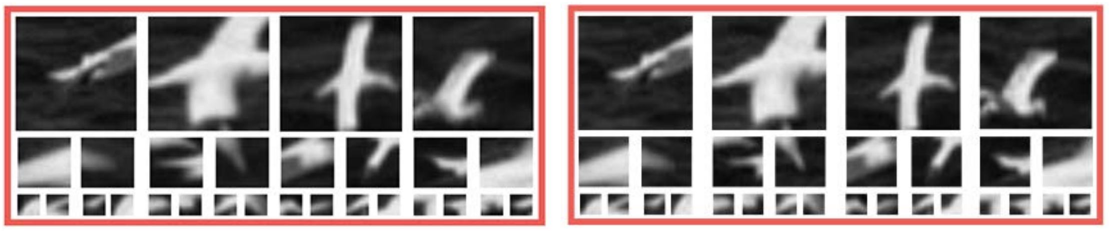

Figure 2.2: Examples of track-templates, $H_k^{(avg)}$ (left) and $H_k^{(fst)}$ (right). Each row shows track-templates at different scales (29×29, 15×15 and 7×7). $H_k^{(avg)}$ is smoother since the representation involves averaging, whereas $H_k^{(fst)}$ preserves image discontinuities better.

we are aiming for a specific fidelity, this function can be used to test whether the representation achieves it. In case it does not, we use a simple mechanism that would allow us to achieve that accuracy: We take each track and recursively break it in the middle, treating each half as an independent track. We stop the recursion when the desired fidelity is achieved for every track. The downside is that each split adds an additional element to the dictionary, which reduces the compression.

2.2 Encoding Track-Templates In a Video

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

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

Instead, we follow an alternative approach. We choose to assign an ordering of the track-templates by placing them on depth layers. A track-template placed on a layer with a smaller "depth" would be overlaid on top of another one placed on a layer with a larger "depth". Hence, by selecting an ordering of track-templates, we implicitly choose which of them to use to reconstruct the video frames. By following such a scheme, we would simply append a scalar to the track parameters, hence characterizing the track-templates with the parameter set (x, y, t, k, d), where d corresponds to the "depth" or ordering of that particular track-template (in each frame). This solution essentially performs a global optimization over all track-templates, which we propose to solve in one step. An example of the proposed solution is shown in Fig. 2.3.

To determine the ordering of the track-templates we base our solution on [WLP09], where the proposed model was used for segmentation, ordering and multi-object tracking. We use this model for video compression. Our solution modifies the original cost function, by dropping a number of terms that do not apply in our case (e.g. terms modeling spatial interactions for segmentation). Specifically, let $V_T = \{1, 2, \ldots, K\}$ be the index set of track-templates. Let $V_I = \{K+1, \ldots, K+N\}$ be the index set of intersections. Let H_k be the appearance of track-template $k \in V_T$ (i.e. either $H_k^{(fst)}$ or $H_k^{(avg)}$). Each intersection is defined to be a unique combination of track-templates overlapping. Let M_k be the index set of intersections which are occupied by track-template $k \in V_T$. Let d_k for $k \in V_T$

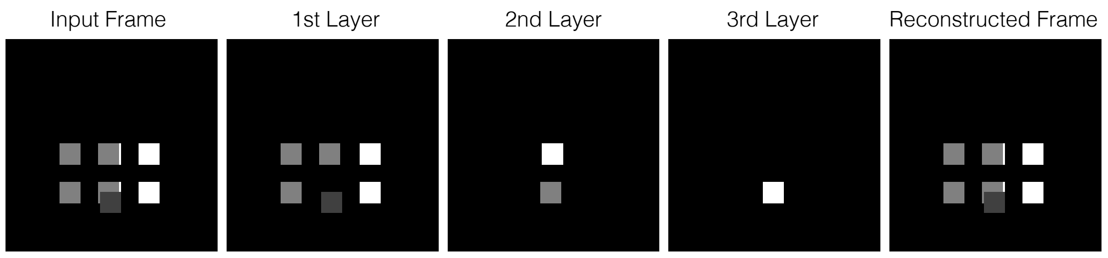

Figure 2.3: Encoding structures in a frame. Problem illustration. For this instance of the problem the dictionary is composed of 9 track-templates: 4 white, 4 light gray and 1 dark gray square. The original frame is decomposed into 3 layers. Occluded track-templates are pushed to the back layers. Our proposed solution retrieves the 3 middle frames, which along with the track-template parameters are used to reconstruct the input frame (right).

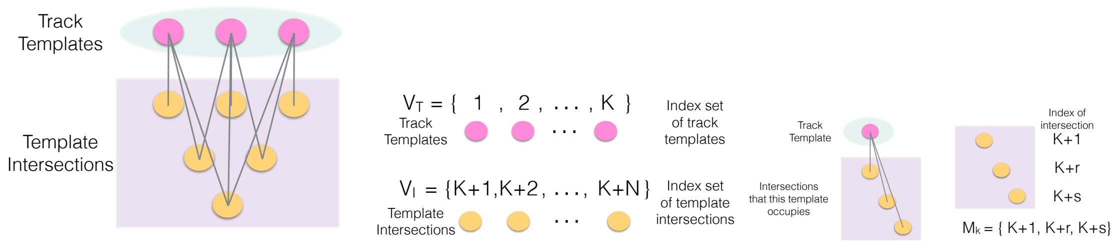
Track Templates
Template Intersections
V_T = { 1 , 2 , . . . , K }
Track Templates
Index set of track templates
V_I = {K+1,K+2, . . . , K+N}
Template Intersections
Index set of template intersections
Track Template
Intersections that this template occupies
Index of intersection
K+1
K+r
K+s
M_k = { K+1, K+r, K+s}

Figure 2.4: Left to right: (1) Model illustration. Top nodes represents 3 track-templates. Bottom nodes show the intersections of the 3 track-templates. Edges are drawn between every template and intersections occupied by them. (2) Index sets V_T and V_I. (3) M_k.

be the relative depth index (ordering) of the track-template. Assume that there are at most L layers, where $L \leq K$ and that $\mathcal{L} = \{0, 1, \ldots L-1\}$. Let $l_i \in V_T$ denote the index of the track-template assigned to intersection $i \in V_I$ (i.e. it is the "minimizer"). Fig. 2.4 illustrates these quantities. In addition, we introduce constraints A_1 and A_2. A_1 couples the track-template with the smallest depth with intersection i, by assigning its index to l_i. A_2 requires that the depth of one track-template is smaller than all others ("unique minimizer"):

$$A_1 : l_i = \underset{\{k|i \in M_k, k \in V_T\}}{\arg\min} d_k, \forall i \in V_I \ , \tag{2.3}$$

$$A_2 : \forall i \in V_I, \exists \tilde{k} \in \{k|i \in M_k, k \in V_T\} s.t. \forall k' \in \{k|i \in M_k, k \in V_T\} \backslash \{\tilde{k}\}, d_{\tilde{k}} < d_{k'} \tag{2.4}$$

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

$$A_3 : z_i = \min_{\{k|i \in M_k, k \in V_T\}} d_k, \forall i \in V_I \ . \tag{2.5}$$

A_3 requires that the depth of an intersection is the same as the depth of the "minimizer" of that intersection. It was shown by [WLP09] that the following equivalence holds: $\forall i \in V_I, A_1 \wedge A_2 \wedge A_3 \Leftrightarrow \wedge_{k \in V_T}(C_{1k} \wedge C_{2k} \wedge C_{3k})$ (for proof refer to [WLP09]), where:

$$C_{1k} : \neg((l_i = k) \wedge (z_i \neq d_k)) \ , \tag{2.6}$$

$$C_{2k} : \neg((l_i = k) \wedge (i \notin M_k)) \ , \tag{2.7}$$

$$C_{3k} : \neg((l_i \neq k) \wedge (z_i \geq d_k) \wedge (i \in M_k)) \ . \tag{2.8}$$

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

$$E(\gamma) = \sum_{i \in V} \phi_i(\gamma_i) + \sum_{(k,i) \in \mathcal{E}} \psi_{k,i}(\gamma_k, \gamma_i) \ . \tag{2.9}$$

The nodes have the following potentials:

$$\forall i \in V_I, \gamma_i = (l_i, z_i), \qquad \phi_i(\gamma_i) = ||I_i - H_{l_i}|| \ , \tag{2.10}$$

$$\forall i \in V_T, \gamma_i = d_i, \qquad \phi_i(\gamma_i) = \alpha |d_i| \ , \tag{2.11}$$

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

$$\psi_{k,i}(\gamma_k, \gamma_i) = \psi^1_{k,i}(\gamma_k, \gamma_i) + \psi^2_{k,i}(\gamma_k, \gamma_i) + \psi^3_{k,i}(\gamma_k, \gamma_i) \ , \tag{2.12}$$

$$\psi^1_{k,i}(\gamma_k, \gamma_i) = \lambda_1 \mathbb{1}((l_i = k) \wedge (z_i \neq d_k)) \ , \tag{2.13}$$

$$\psi^2_{k,i}(\gamma_k, \gamma_i) = \lambda_2 \mathbb{1}((l_i = k) \wedge (i \notin M_k)) \ , \tag{2.14}$$

$$\psi^3_{k,i}(\gamma_k, \gamma_i) = \lambda_3 \mathbb{1}((l_i \neq k) \wedge (z_i \geq d_k) \wedge (i \in M_k)) \ . \tag{2.15}$$

We solve $\gamma^{opt} = \arg\min_\gamma E(\gamma)$ using any standard inference method e.g. TRW-S [Kol06]. The optimization is performed for each frame independently. In the original work [WLP09], the set V_T corresponds to objects, rather than to track-templates and the set V_I corresponds to pixels rather than intersections. Intersections were introduced in our problem to improve compression. We fix $\alpha = 1$, $\lambda_1 = 50$, $\lambda_2 = 1$ and $\lambda_3 = 10$.

2.2.1 Integration With Standard Video Encoders

Once we retrieve the ordering of the track-templates using the previous step we can reconstruct the frames by transmitting the parameter set for each track-template: (x, y, t, k, d). The compression ratio of the representation to the uncompressed lossless one is $\xi = \frac{(\sigma+4)\mathcal{T}}{\sigma+5\mathcal{T}}$. For integer-valued σ and $\mathcal{T}$, $\xi \geq 1$ for $(\sigma > 1, \mathcal{T} > 1)$. We are therefore able to compress any track of length $\mathcal{T} \geq 2$ (i.e. the "trackable regions"). Using the depth d, we can reconstruct each frame by overlaying the structured regions with a smaller depth on top of others. In Fig. 2.5, we show how a frame from a video is decomposed into layers and then reconstructed to recover

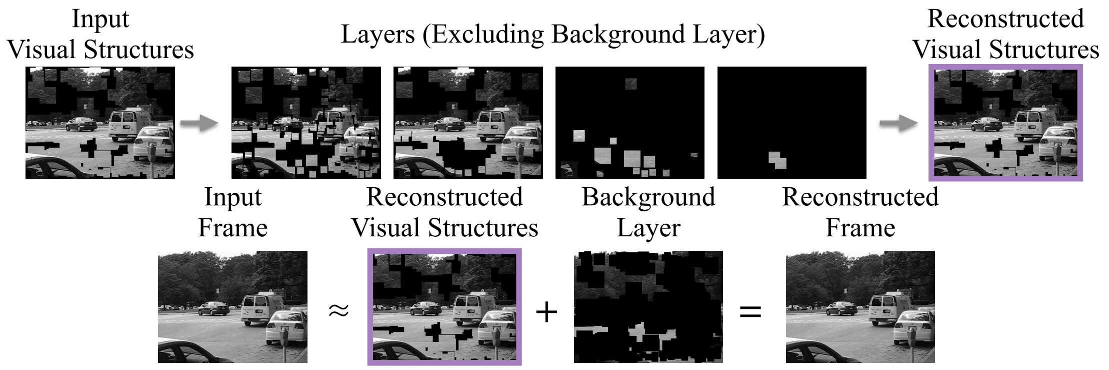

Figure 2.5: Reconstructing a frame. Visual structures are decomposed into depth layers and reconciled by overlaying them. The input frame is reconstructed by adding back to the visual structures the background layer.

all trackable regions.

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

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

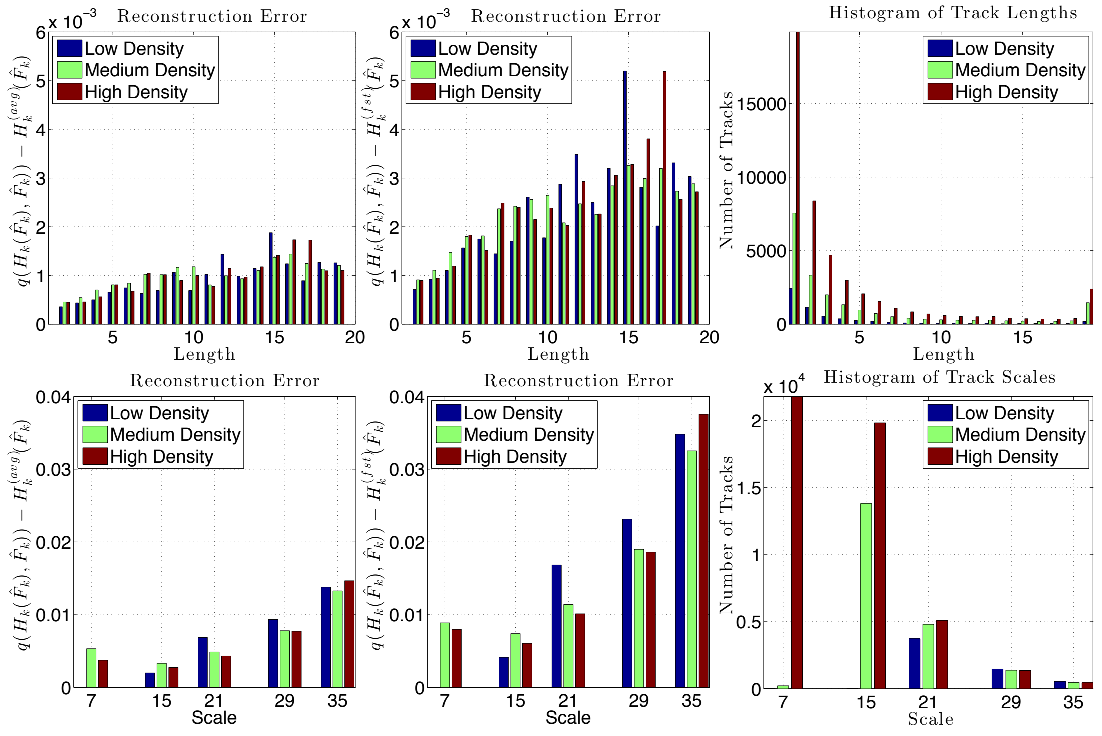

Figure 2.6: Results for tracks in MOSEG [BM10]. Top: $q(H_k^{(avg)}(\hat{F}_k), \hat{F}_k)$ and $q(H_k^{(fst)}(\hat{F}_k), \hat{F}_k)$ as a function of length and a histogram of track lengths. Bottom: $q(H_k^{(avg)}(\hat{F}_k), \hat{F}_k)$ and $q(H_k^{(fst)}(\hat{F}_k), \hat{F}_k)$ as a function of scale and the distribution of track scales.

(Fig. 2.5).

2.3 Experiments

We investigated how well the structure representation reconstructs the individual instances of the track, without applying the recursive, splitting method described in Sec. 2.1.1. Towards this end, we used the 10 car and 2 people sequences from the MOSEG dataset [BM10]. The sequences range from 19-60 frames, but for this experiment, we only used the first 19 frames for all videos to achieve uniformity in the results. We computed the structure representations and reconstructed the trackable regions of the videos using our proposed solution. For each track-template, we computed its average reconstruction error per pixel for each instance of the track according to Eq. 2.2 for both $H_k^{(avg)}$ and $H_k^{(fst)}$. We used 5 different scales for tracking with the smallest one being 7×7 and the largest being 35×35. In addition, we have varied the detection threshold of tracks and selected 3 representative levels. Typical distributions of tracks on the image domain, for the three thresholds, are shown in Fig. 2.1.

In Fig. 2.6, we show how $q(H_k(\hat{F}_k), \hat{F}_k)$ varies for both representations as a function of the length of the track and the scale of the track. We also show histograms of the distribution of tracks according to scale and length. For $H_k^{(avg)}$, the reconstruction error per pixel increases with increasing lengths and scales, but the increase is small and hence shows that the average can reliably represent tracks, even if they are long. For $H_k^{(fst)}$, the error increases slightly faster. The average reconstruction error over all tracks and thresholds is 9.2670×10^{-4} for $H_k^{(avg)}(\hat{F}_k)$ and 2.2×10^{-3} for $H_k^{(fst)}(\hat{F}_k)$.

We used our proposed system to encode the first 5 frames of the 12 video sequences from MOSEG. We used H.265/H.264 to encode the frames with the structure representations and background layers placed on them. We have also encoded the videos using H.265 (HM 16.2)[1], H.264 (JM 18.6 Reference Software [JM008]) and JPEG. Note that our method can be used along any other video encoding system, replacing H.265/H.264. In Fig. 2.7 we plot PSNR (dB) against bit rate (kbps) for our approach ("VS+h.265", "VS+h.264"), H.265, H.264 and JPEG. For better coverage of the image domain, we expanded the domain of each track-template by a factor of 3^2. To achieve varying fidelity for all methods, we varied the quantization levels.

We consistently outperform all other methods in all sequences. In these experiments, the representations achieve at least 25 dB in PSNR for each of the instances of the track they are representing (using our recursive, splitting algorithm),

[1] `https://hevc.hhi.fraunhofer.de/`

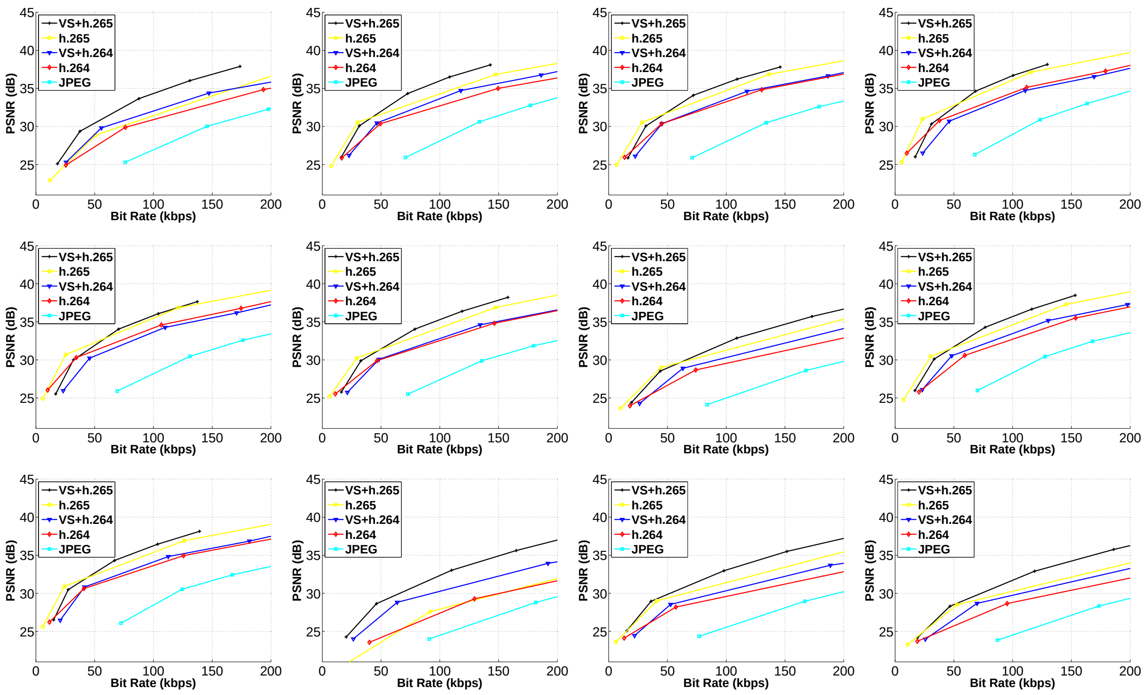

Figure 2.7: PSNR against bit rate. "VS+H.265"(black) and "VS+H.264"(blue) outperform respectively H.265(yellow) and H.264(red). Figures correspond to the sequences in Fig. 2.8.

before they are passed to H.265/H.264. At lower fidelity, the performance gain of our method diminishes due to the parameter overhead that needs to be transmitted. At higher fidelity our approach benefits from taking advantage of the temporal redundancy of the tracks and is much more efficient than competitive approaches.

Fig. 2.8 illustrates where the gain is achieved in our methods. For the last frame of the video sequences, we show which regions were predicted from previous frames (non-transparent regions) and which first appeared in this frame (semi-transparent). Generally, the larger the percentage of tracks that are temporally predicted, the larger the improvement is over other methods. While H.265/H.264 encodes the temporally predicted tracks, our encoder predicts them from previous frames. Our algorithm takes on average 96 seconds on the encoder side and 0.5 seconds on the decoder side per frame (excluding the computational time required

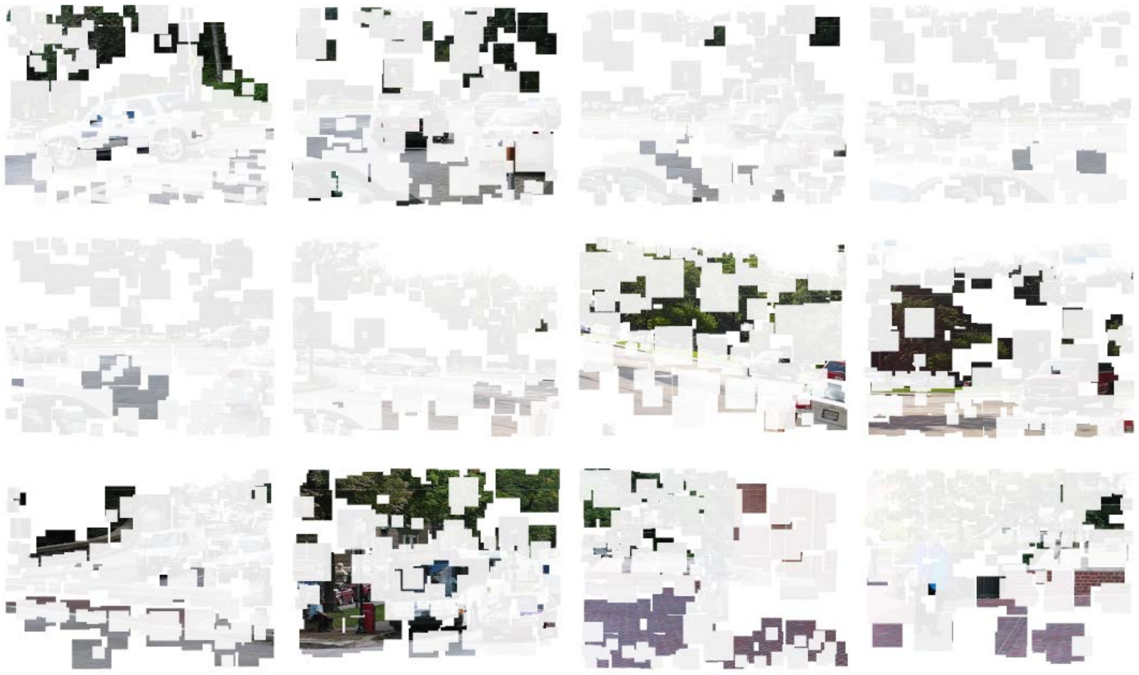

Figure 2.8: Propagated and newly-created tracks. Non-transparent tracks correspond to tracks that are motion-predicted from previous frames. Semi-transparent tracks are tracks that start in this frame. All results shown correspond to the fifth frame of each video.

by H.265/H.264), for a MATLAB/C++ implementation on an Intel 2.4 GHz dual core processor machine.

2.4 Discussion

We presented an alternative system to traditional video encoders, which was shown to exploit the temporal redundancy of visual structures. The frames were partitioned into structures and background layers. Structures are compressed using a time-invariant representation. They are then ordered in terms of reconstruction error and are used to reconstruct the video along with the background layers. Our method can be wrapped around standard encoders such as H.265 and H.264 and it outperforms both of them in a rate-distortion criterion.

CHAPTER 3

Visual Textures

"Visual textures" are regions of images that exhibit some form of spatial regularity. They include the so-called "regular" or "quasi-regular" textures (Fig. 3.2 left), "stochastic" textures (middle-left), possibly deformed either in the domain (middle-right) or range (right). Analysis of textures has been used for image and video representation [ZXZ11, GZW07], while synthesis has proven useful for image super-resolution [FL11], hole-filling [EL99] and compression [WWO08].

For such applications, large textures carry a high cost on storage and computation. State-of-the-art texture descriptors such as [CMK14, SM13] are computationally prohibitive to use on large textures. These issues are especially acute for video, where the amount of data is significantly larger.

Typically, these descriptors as well as texture synthesis algorithms assume that the size of the input texture is small, and yet large enough to compute statistics that are representative of the entire texture domain. Few works in the literature deal with how to infer automatically this smaller texture from an image and even fewer from a video. In most cases, it is assumed that the input texture is given, usually manually by cropping a larger one. Wei et. al. [WHZ08] propose an inverse texture synthesis algorithm, where given an input texture I, a compaction is synthesized that allows subsequent re-synthesis of a new instance $\hat{I}$. The method achieves good results, but it is semi-automatic, since it relies on external information such as a control map (e.g. an orientation field or other contextual information) to synthesize time-varying textures and on manual adjustment of the scale of neighborhoods for

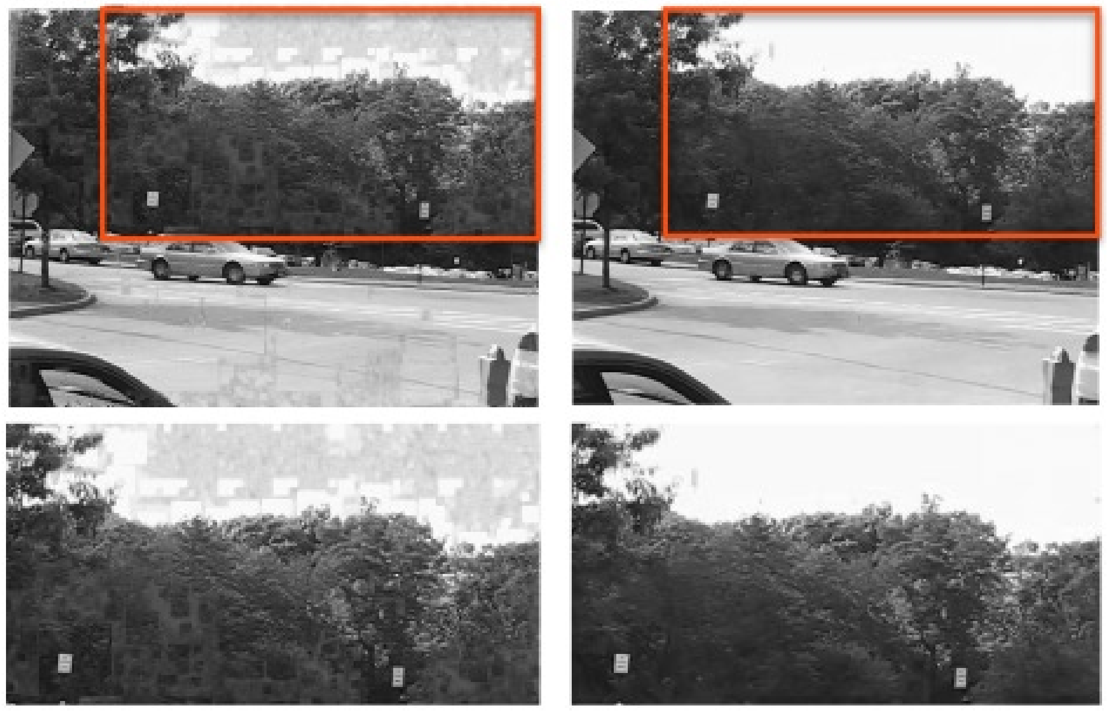

Figure 3.1: Reconstructed frame in a video at 40% compression. Left: Video Epitome [CFJ08], Right: Our approach. Below: Zoomed-in view of the red box. Our method improves reconstruction of both homogeneous and textured areas.

sampling from the compaction.

We propose an alternative scheme, which avoids using any external information by automatically inferring a compact time-varying texture representation. The algorithm also automatically determines the scale of local neighborhoods, which is necessary for texture synthesis [EL99]. Since our representation consists of samples from the input texture, for applications such as classification [CMK14, SM13], we are able to avoid synthesis biases that affect other methods [WHZ08].

Our contributions are to (i) summarize an image/video into a representation that requires significantly less storage than the input, (ii) use our representation for synthesis using the texture optimization technique [KEB05], (iii) extend this framework to video using a causal scheme, similar to [EL99] and show results for

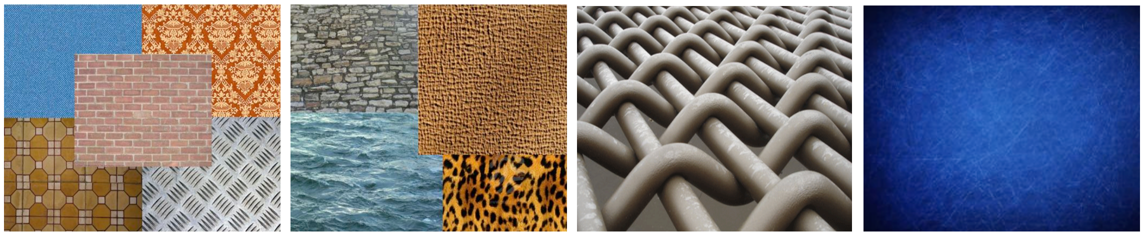

Figure 3.2: Regular, stochastic, domain- and range-deformed textures.

multiple time-varying textures, (iv) synthesize multiple textures simultaneously on video without explicitly computing a segmentation map, unlike [XBY09], which is useful for hole-filling and video compression, (see Fig. 3.1), and (v) propose a criterion ("Texture Qualitative Criterion" (TQC)) that measures structural and statistical dissimilarity between textures.

3.1 Related work

Our work relates to texture analysis ([MM02, LSP05, KEM09, LLH04] and references therein), perception ([MP90] and refs.), synthesis and mapping ([EL99] and refs.). We do not address phenomenology (*e.g.* "3-D textures" generated by the interplay of shape and illumination [SH98] vs. "decal textures" due to radiance) and the algorithms used to pool statistics (*e.g.* patch-based [VZ03], statistical [GG84], geometric [Zuc76] methods). We refer the reader to [Har05] for a more extensive survey.

Some work has been done in summarizing images (epitome [JFK03] and jigsaw [KWR06]) and video [CFJ08, WSI07]. These methods do not handle textures explicitly and as a result, their reconstructed textures suffer. [JFK03, KWR06, CFJ08] are also not able to extend textures to larger domains, since they rely on an explicit mapping between the input image/video and the summarization. Other schemes aim to compact the spectral energy into few coefficients [BAC96, Fen03, MP12]. [WWO08] compresses regular textures, but fails with stochastic ones. Our work models textures explicitly and hence achieves both high compression rates and high quality synthesis results.

In video texture synthesis, there are several works ranging from pixel-based ([BEL01], [WL00]), to patch-based [KSE03], to parametric [DCW03]. Our work falls in the patch-based category.

3.2 Textures

"Visual textures" are regions of images that exhibit spatial regularity. To characterize them, we use the notions of Markovianity, stationarity and ergodicity. We denote an image by $I : \Delta \to \mathbb{R}^+$, where $\Delta = (1,1) : (X,Y)$ is the pixel lattice. Statistics, or "features", map the image onto some vector space and *local* features operate on a subset of the image domain. Formally, a local statistic is defined on $\omega \subset \Delta$ as a function $\theta_\omega : \{I : \Delta \to \mathbb{R}^+\} \to \mathbb{R}^K;\ I \mapsto \theta_\omega(I)$ that only depends on the values of the image $I(x,y)$ for $(x,y) \in \omega$. A distribution on I, $dP(I)$ induces a distribution on θ_ω via $dP(\theta_\omega) \doteq dP(\theta_\omega(I))$.

In order to exploit spatial predictability, we leverage on the existence of a statistic that is *sufficient* to perform the prediction. This is captured by the notion of Markovianity. We say that a process $\{I\}$ is Markovian if every set $\Pi \subset \Delta$ admits a neighborhood $N(\Pi)$ and a statistic $\theta_{N(\Pi)}$ that makes $I(\Pi)$ independent of the "outside" $I(\Pi^c)$. Π^c is the complement of Π in a region $\Omega \subset \Delta$, where Ω corresponds to the texture region (see Fig. 3.3):

$$I(\Pi) \perp I(\Pi^c) \mid \theta_{N(\Pi)}. \tag{3.1}$$

This condition makes the process $\{I\}$ with measure $dP(I)$ a Markov Random Field (MRF), and establishes $\theta_{N(\Pi)}$ as a *sufficient statistic* for $I(\Pi)$. In general, $N(\Pi)$ could correspond to many regions, for instance $\omega = \Omega \backslash \Pi$. In describing a texture, we seek the *smallest* ω, in the sense of minimum area ("scale") $|\omega| = r$, so the corresponding θ_ω is a *minimal (Markov) sufficient statistic*. Of particular interest is the case when such a neighborhood is spatially homogeneous[1], as is the case in a *stationary* MRF.

A process $\{I\}$ is *stationary* if statistics are translation invariant. More formally, $\{I\}$ is stationary in θ_ω if,

$$\mathbb{E}(\theta_\omega) = \mathbb{E}(\theta_{\omega+T}), \quad T \in \mathbb{R}^2 \mid \omega + T \subset \Omega. \tag{3.2}$$

Such condition may be satisfied only after transformations of the image domain and range (as in *deformed textures*, Fig. 3.2)[2]. Note that unless the process is defined on the entire plane, one has to *restrict* the set of T to *allowable translations* to ensure Eq. (3.2) is computed where $\theta_{\omega+T}$ is defined.

To *test* whether a process is stationary from just one sample, we have to assume it is ergodic, meaning that the sample statistics converge to the ensemble ones:

$$\frac{1}{N}\sum_{i=1}^{N} \theta_{\omega+T_i} \xrightarrow{\text{a. } s.} \mathbb{E}(\theta_\omega), \tag{3.4}$$

for all admissible T_i. Given Ω, we can test whether the process $\{I\}$ is stationary in this region, by approximating statistics θ_ω using samples in a neighborhood $\bar{\omega} \subset \Omega$. The larger the size of $\bar{\omega}$, the better the approximation of the statistics, but the lower the compression achieved. Therefore, we seek the *smallest* possible $\bar{\omega}$ that allows inferring the statistics "sufficiently well". Assuming that such $\bar{\omega}$ is found, we can test for stationarity by *translating* it and testing whether the resulting statistics remain "sufficiently constant". This, however, can only be tested for *admissible* translations that keep $\bar{\omega}$ within Ω.

We define a texture as *a region Ω of an image I that can be rectified into a*

[1]Spatially homogeneous means that $\omega = \omega(\Pi)$ can be written in terms of neighborhoods of each pixel $\boldsymbol{x} = (x, y)$ within Π, and the smallest Markov neighborhood of a pixel, $\omega(\boldsymbol{x})$, is translation invariant, *i.e.,* its shape does not change as we translate $\boldsymbol{x}$: $\omega(\boldsymbol{x} + T) = \omega(\boldsymbol{x}) + T$.

[2]For a group G acting on the domain of I via $I(x, y) \mapsto I(g(x, y))$ and a group H acting on the range via $I(x, y) \mapsto h(I(x, y))$, we say that the process $\{I\}$ is $G - H$ stationary in $\theta_{g(\omega)}$ if,

$$\mathbb{E}(\theta_{g(\omega)}(h(I))) = \mathbb{E}(\theta_{g(\omega)+T}(h(I))), \quad T \in \mathbb{R}^2 \mid \omega + T \subset \Omega. \tag{3.3}$$

Such transformations can be inferred by a model selection criterion. Textures can be *rectified* by applying the inverse action g^{-1}, h^{-1} to the data.

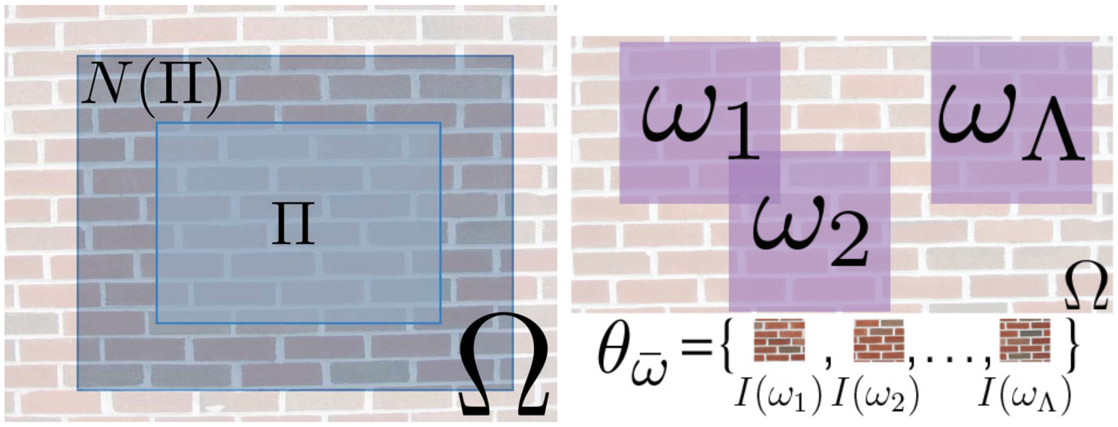

Figure 3.3: Left: A subset of the image domain, Π, with its local neighborhood. Ω denotes the domain of the texture. Right: Texture representation $\theta_{\bar{\omega}}$ and samples drawn from Ω.

sample of a stochastic process that is stationary, ergodic and Markovian. A texture is parametrized by the following quantities: (a) The Markov neighborhood ω and its Markov scale $r = |\omega|$, (b) the stationarity region $\bar{\omega}$ and its stationarity scale $\sigma = |\bar{\omega}|$, (c) the minimal sufficient statistic θ_ω defined on ω, and (d) Ω, the texture region. Note that $\omega \subset \bar{\omega} \subset \Omega$.

3.2.1 Texture Representation

We initially assume (and later relax) that $\Omega = \Delta$. The representation should have smaller complexity than the collection of pixel values in Ω and allow extrapolation beyond Ω, traits not shared by [CMK14, SM13], but the latter can be used to further reduce complexity in classification applications.

In a non-parametric setting, θ_ω is a collection of sample image values. $\bar{\omega} \doteq \bigcup_{\lambda=1,\dots,\Lambda} \omega_\lambda$ is the union of Λ sample regions ω_λ, each a Markov neighborhood of a pixel with coordinates (x_λ, y_λ). Collectively the neighborhoods capture the variability of the texture. Thus, a texture is represented by (a) ω_λ, chosen as a square for all λ with unknown area r, (b) $\bar{\omega}$, to be determined and (c)

$\theta_{\bar{\omega}} \doteq \{\theta_{\omega_\lambda}\}_{\lambda=1}^{\Lambda} \doteq \{I(\omega_\lambda)\}_{\lambda=1}^{\Lambda}$ (where $I(\omega_\lambda) \doteq \{I(x,y),\ \forall (x,y) \in \omega_\lambda\}$) that is uniquely specified by the image given r and ω_λ (Fig. 3.3).

Complexity controls the cardinality of $\bar{\omega}$. The best we can do is to select all patches ω from Ω and store them as $\bar{\omega}$. When complexity is fixed, however, for instance via a compression parameter ξ, we can only store $\xi \times (X \times Y)$ values, rather than $(X \times Y)$. This determines the cardinality of $\bar{\omega}$. The larger $r = |\omega_\lambda|$, the fewer the patches that can be stored in a given $\bar{\omega}$. There is a natural tradeoff where too small an r fails to capture the local neighborhood of the texture (Markov sufficient statistic) and too large an r fails to capture the statistical variability, as too few patches ω_λ can be contained in a given $\bar{\omega}$. Both have detrimental effects on extrapolating a texture beyond Ω, which we discuss next.

3.3 Inference

In this section we discuss how to infer a (minimal) representation $\{\omega, \bar{\omega}, \theta_{\bar{\omega}}\}$ from a given texture image $\{\Omega, I\}$, and how to synthesize a novel texture image $\hat{I}$ from it. We start from the latter since the algorithm that infers the representation utilizes the synthesis procedure.

3.3.1 Image Texture Synthesis

Given a representation $\{\omega, \bar{\omega}, \theta_{\bar{\omega}}\}$, we can synthesize novel instances of the texture by sampling from $dP(I(\omega))$ within $\bar{\omega}$. This is straightforward in a non-parametric setting, where the representation is itself a collection of samples. One can simply select neighborhoods ω_λ within $\bar{\omega}$, and populate a new lattice with patches $I(\omega_\lambda)$ ensuring compatibility along patch boundaries and intersections. Efros et. al. [EL99] proposed a *causal* sampling scheme that satisfies such compatibility conditions, but fails to respect the Markov structure of the underlying process (their $I(\omega_\lambda)$ are not a Markov sufficient statistic), which causes "blocky" artifacts

```
Algorithm 1: Texture Synthesis
1  Initialize $\hat{I}^{(0)}$ to a random texture;
2  Set $\nu_{\omega_s} = 1$ for $s = 1, \dots, S$ and $j_{max} = 20, b = 0.7$ ;
3  for $j = 1, \dots, j_{max}$ do
4      for $s = 1, \dots, S$ do
5          $\omega_s^{(j)} = \texttt{nrst\_nghbr}(\theta_{\bar{\omega}}, \bar{\omega}, \hat{I}^{(j-1)}(\hat{\omega}_s))$;
6      Let $\hat{I}^{(j)} = \arg\min_{\hat{I}} E(\hat{I}, \{\omega_s^{(j)}\}_{s=1}^S)$;
7      $\nu_{\hat{\omega}_s} = \|\hat{I}^{(j)}(\hat{\omega}_s) - I(\omega_s^{(j)})\|^{b-2}$ for $s = 1, \dots, S$ ;
8      if $(\forall \hat{\omega}_s \in \hat{\Omega}_S : \hat{I}^{(j)}(\hat{\omega}_s) = \hat{I}^{(j-1)}(\hat{\omega}_s))$ then
           break;

   Function nrst_nghbr($\theta_{\bar{\omega}}, \bar{\omega}, \hat{I}(\hat{\omega})$)
9      Let $s$ be the index of the the nearest neighbor of $\hat{I}(\hat{\omega})$ in $\theta_{\bar{\omega}}$ ;
10     Retrieve $\omega_s$ within $\bar{\omega}$ ;
11     return $\omega_s$ ;
```

and "drift." Instead, given $\{\omega, \bar{\omega}, \theta_{\bar{\omega}}\}$, we synthesize textures by choosing a subset of neighborhoods from $\bar{\omega}$ that satisfy the compatibility conditions and by construction also respect the Markov structure. We perform this selection and simultaneously also infer $\hat{I}$. We do so by first initializing $\hat{I}$ at random. We select neighborhoods $\hat{\omega}_s$ on a grid on the domain of the synthesized texture every $\frac{\sqrt{r}}{4}$. We let $\hat{\Omega}_S = \{\hat{\omega}_s\}_{s=1}^S$ denote the collection of the selected $\hat{\omega}_s$, $\Omega_S = \{\omega_s\}_{s=1}^S$ denote the chosen neighborhoods within $\bar{\omega}$ and $I(\omega_s) \in \theta_{\bar{\omega}}$ denote the nearest neighbor of $\hat{I}(\hat{\omega}_s)$. We minimize with respect to $\{\omega_s\}_{s=1}^S$ and $\hat{I}$ the function [KEB05]:

$$E(\hat{I}, \{\omega_s\}_{s=1}^S) = \sum_{\hat{\omega}_s \in \hat{\Omega}_S} \nu_{\hat{\omega}_s} \|\hat{I}(\hat{\omega}_s) - I(\omega_s)\|^2. \tag{3.5}$$

The procedure to minimize the above energy function is given in Alg. 1. An illustration of the quantities involved is shown in Fig. 3.4. $\nu_{\hat{\omega}_s}$, defined in Alg. 1, is used to reduce the effect of outliers, as done typically in iteratively re-weighted least

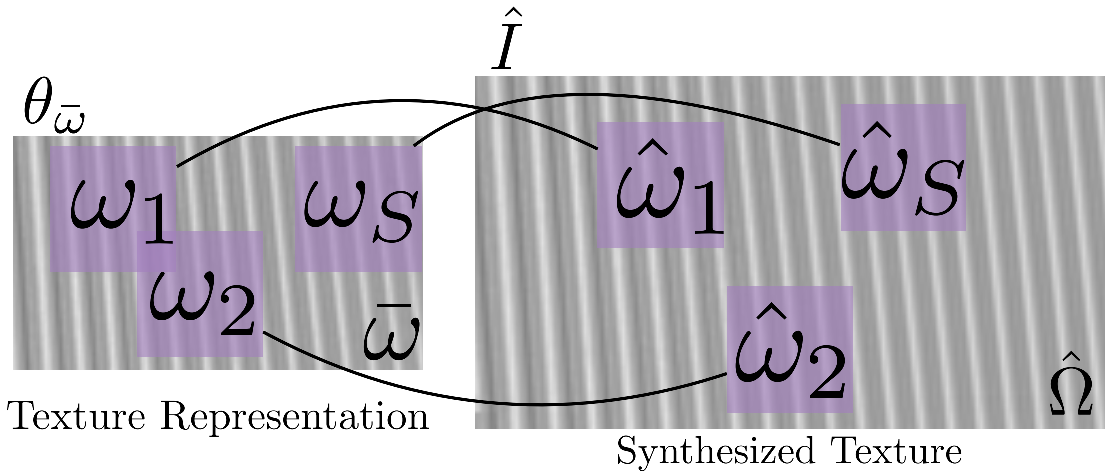

Figure 3.4: Image Texture Synthesis. For each neighborhood $\hat{\omega}_s$ in the synthesized texture, we find its nearest neighbor in $\bar{\omega}$.

squares [KEB05]. The process is performed in a multiscale fashion, by repeating the procedure over 3 neighborhood sizes: $|\hat{\omega}_s|, |\frac{\hat{\omega}_s}{2}|, |\frac{\hat{\omega}_s}{4}|$. By first synthesizing at scale $|\hat{\omega}_s| = r$, we capture the Markov structure of the texture. Subsequent repetitions refine the synthesized texture by adding finer details. We also repeat this process over a number of different output image sizes.

3.3.2 Video Texture Synthesis

The texture synthesis algorithm in [KEB05] was extended to temporal textures, which however relied on the availability of optical flow. Unfortunately, optical flow is expensive to store, as encoding it is more costly than encoding the original images. We propose a temporal texture synthesis algorithm that relies on neighborhoods ω_λ that extend in time.

We take the input video $\{I^t\}_{t=1}^T$, and compute a compact representation $\theta_{\bar{\omega}^t}$, from which we synthesize $\{\hat{I}^t\}_{t=1}^T$. In this section we assume we have $\theta_{\bar{\omega}^t}$ and in Sec. 3.3.5 we explain how to infer it. We re-define all quantities to have domains that extend in time. To reduce computational complexity we fix the temporal

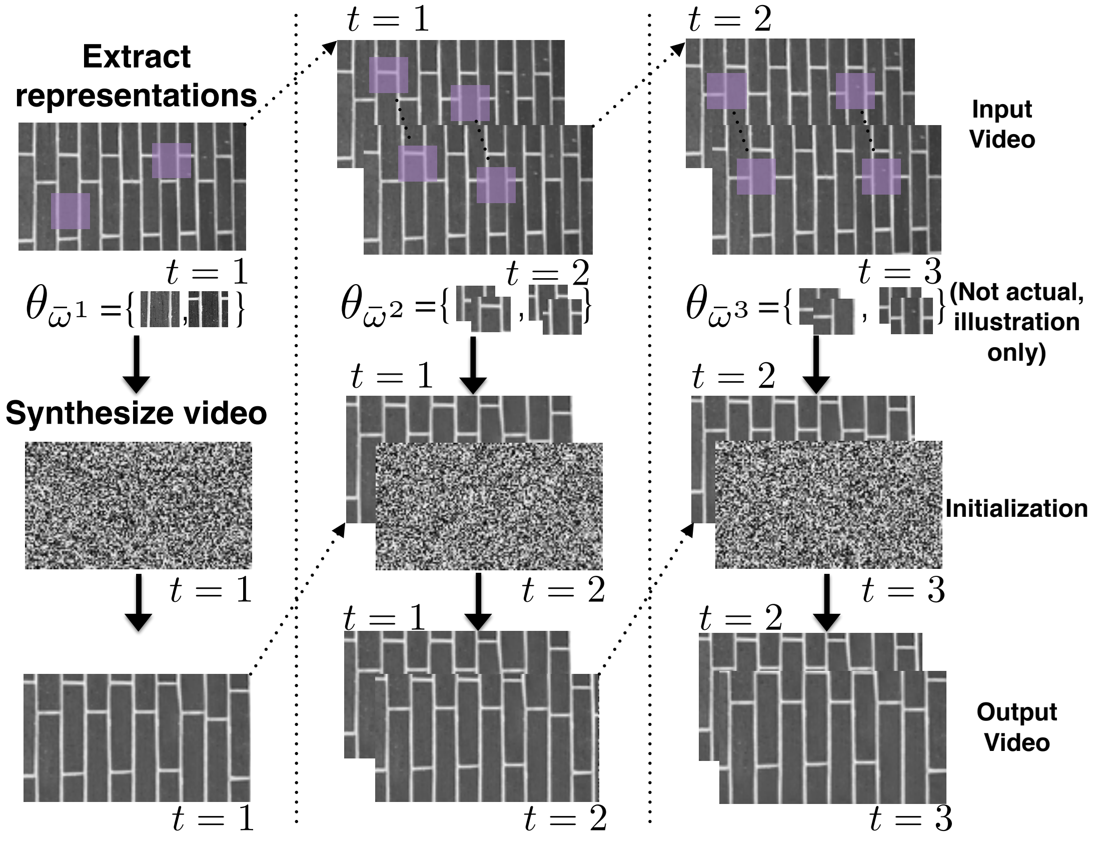

Figure 3.5: Temporal Texture Synthesis. We forward-synthesize the video from the texture representations of each frame using the previously synthesized frame as a boundary condition.

extension of the neighborhoods to 2 frames, although longer choices are possible. Hence for $t > 1$, $\omega_\lambda^t \subset (1, 1, t-1) : (X, Y, t)$, which makes it a 3-D neighborhood and $\bar{\omega}$ becomes $\bar{\omega}^t \doteq \bigcup_{\lambda=1,\ldots,\Lambda} \omega_\lambda^t$, a union of 3-D neighborhoods. $I^t(\omega_\lambda^t)$ is therefore defined on the 3-D lattice and $\theta_{\bar{\omega}^t} \doteq \{I^t(\omega_1^t), \ldots, I^t(\omega_\Lambda^t)\}$. For $t = 1$, $\omega_\lambda^{t=1}, \bar{\omega}^{t=1}$ and $\theta_{\bar{\omega}^{t=1}}$ remain 2-D.

We initialize the output video, $\hat{I}^t$, to a random texture and first synthesize frame $\hat{I}^{t=1}$ using Alg.1. We then let $t \leftarrow t + 1$ and synthesize frame t using a causal approach, similar to [EL99], by ensuring that the compatibility conditions with frame $t-1$ are satisfied: For each $\hat{\omega}_s^t \in \hat{\Omega}_S^t$ (where Ω_S^t corresponds to the set of selected $\hat{\omega}_s^t$ on the 3-D grid of the synthesized frames, extending temporally in

$[t-1, t]$), we mask (i.e., discount) the portion of the neighborhood that is in frame t (the unsynthesized part) and seek its nearest neighbor, ω_s^t, within $\theta_{\bar{\omega}^t}$ on *only* the unmasked region (i.e., on only the region that has already been synthesized). Once we get the nearest neighbors of all $\hat{\omega}_s^t$, we fix them and do not allow them to change. This is done in order to achieve compatibility with the already synthesized textures. We then unmask all neighborhoods and use them to minimize the energy function in Eq. (3.5) (see Fig. 3.5).

3.3.3 Synthesizing Multiple Textures Simultaneously

We demonstrate how multiple textures can be synthesized simultaneously for video and images without computing a segmentation map. This is useful for applications such as video compression (where $\{\omega, \bar{\omega}, \theta_{\bar{\omega}}\}$ can be used to synthesize the textures of the input video) or for image processing tasks such as hole-filling and frame interpolation.

To place the textures in their corresponding locations in a video (or image) we implicitly define their boundaries by partitioning each frame into two types of regions: Textures and their complementary region type, *structures.* Structures are regions of images that trigger isolated responses of a feature detector. These include blobs, corners, edges, junctions and other sparse features. We determine which regions are structures, by using a feature point tracker such as [LK81].

Partitioning images or video into two types of regions has been previously proposed by several works ([GZW03a, SCV02, BVS03]) using a single image. In our framework, if a region with a scale ϵ triggers an *isolated* response of a feature detector (i.e., it is a structure at scale ϵ), then the underlying process is, by definition, not stationary at the scale $|\bar{\omega}| = \epsilon$. Therefore, it is not a texture. It also implies that any region $\bar{\omega}$ of size $\epsilon = |\bar{\omega}|$ is not sufficient to predict the image outside that region. This of course does not prevent the region from being 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 $\bar{\omega}$. Of course, it could have multiple extrema, each isolated within a region of size $\epsilon << \sigma$. We conclude that, *for any given scale of observation σ, a region $\bar{\omega}$ with $|\bar{\omega}| = \sigma$ is either a structure or a texture.*

One must impose boundary conditions so that the texture regions fill around structure regions seamlessly. To perform texture extrapolation, we follow an approach similar to the one used for video texture synthesis. The video is initialized to a random texture. At locations where the structures were detected and tracked, we place the actual image (intensity) values. We select $\hat{\omega}_s^t \in \hat{\Omega}_S^t$ like before, on a 3-D grid of the synthesized frames, but with the added restriction that $\hat{\omega}_s^t$ needs to have at least one pixel in the texture domain (otherwise it is entirely determined i.e., it is a structure). The patches that are entirely lying in the texture domain need to be synthesized. The patches that straddle the texture/structure partition are used as boundary conditions and are synthesized causally.

We mask as before the portion of each neighborhood, $\hat{\omega}_s^t$, that is in frame t and is part of the texture domain (since this is the only unknown, unsynthesized part of the neighborhood). We then proceed as in the temporal texture synthesis algorithm: We find the nearest neighbor of the unmasked region within $\theta_{\bar{\omega}^t}$, fix it and do not allow it to change. Finally we unmask all the neighborhoods and use them to minimize the energy function in Eq. (3.5). Note that for $t = 1$, if $\hat{\omega}_s^t$ lies entirely in the texture domain, its nearest neighbor is permitted to change through the iterations, similar to Alg.1.

Finally, in addition to the structures, we could also use $\bar{\omega}^t$ to provide additional boundary conditions. Placing $I(\omega_\lambda^t)$ for $\omega_\lambda^t \subset \bar{\omega}^t$ on the image domain allows us to avoid explicitly synthesizing on these locations. These are already stored in $\theta_{\bar{\omega}^t}$, so we only need to additionally store the location of their central pixels. Examples of synthesizing multiple textures simultaneously are shown in Fig. 3.13.

3.3.4 Texture Qualitative Criterion

To evaluate the quality of the texture synthesis algorithm, we need a criterion that measures the similarity of the input, I, and synthesized, $\hat{I}$, textures. The peak signal-to-noise ratio ($PSNR$) is typically used as the criterion for evaluating the quality of a reconstruction. However, when the final user is the human visual system, $PSNR$ is known to be a poor criterion, especially for textures, as imperceptible differences can cause large $PSNR$ changes. Works such as [PH12, WBS04, TH94, SB06] operate on general images and do not exploit properties of textures. To address this issue, we introduce the Texture Qualitative Criterion (TQC), represented by E_{TQC}, which is composed of two terms. The first one, $E_1(\hat{I}, I)$, penalizes structural dissimilarity, whereas $E_2(\hat{I}, I)$ penalizes statistical dissimilarity. We let $\hat{\omega}_s/\omega_i$ be patches within $\hat{\Omega}/\Omega$, the domains of $\hat{I}/I$, and their nearest neighbors be $\omega_s/\hat{\omega}_i$, which are selected within the domains of $I/\hat{I}$. $I/\hat{I}$ can correspond to the input/synthesized textures, or simply two textures, which we wish to compare.

For $E_1(\hat{I}, I)$, we select N_S patches $\hat{\omega}_s \subset \hat{\Omega}$ and N_I patches $\omega_i \subset \Omega$ on a dense grid in the domain of the synthesized and input images respectively. We let $\hat{I}(\hat{\omega}_s)$ and $I(\omega_i)$ correspond to the intensity values in the synthesized and input neighborhoods respectively. We use the patches selected to compute the following cost function:

$$\begin{aligned} E_1(\hat{I}, I) = &\frac{1}{2N_I} \sum_{i=1}^{N_I} \frac{1}{|\omega_i|} \|\hat{I}(\hat{\omega}_i) - I(\omega_i)\|^2 + \\ &\frac{1}{2N_S} \sum_{s=1}^{N_S} \frac{1}{|\hat{\omega}_s|} \|\hat{I}(\hat{\omega}_s) - I(\omega_s)\|^2. \end{aligned} \tag{3.6}$$

Note that this expression resembles Eq. (3.5), with one change: There is an added summation in Eq. (3.6), which is over patches in the input image. The need of both

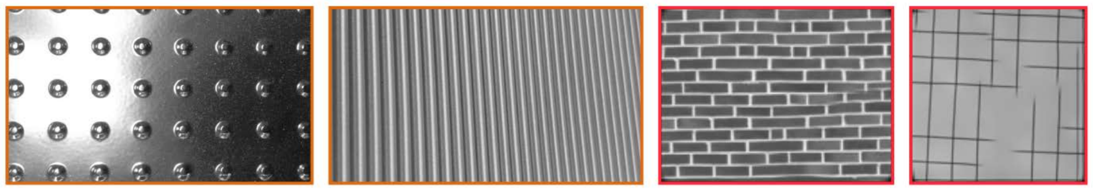

Figure 3.6: The first term in Eq. (3.6) identifies global range/domain transformations of the input texture (left images). The second term identifies erroneous texture synthesis results (right images).

of these terms has also been noted by others [WHZ08] and is illustrated in Fig. 3.6. The first term identifies domain/range deformations of the input texture, whereas the second term identifies artifacts in the synthesized texture. We compute this cost function over multiple scales (typically 3) and average over all scales. This makes the cost function more robust, as it is able to compute similarity of patches at multiple scales.

$E_2(\hat{I}, I)$ is based on a distance between histograms of filter responses, which allows us to capture the statistical differences between two textures:

$$E_2(\hat{I}, I) = \frac{1}{L} \sum_{l=1}^{L} \|\phi(g_l(I)) - \phi(g_l(\hat{I}))\|_{\chi^2}, \tag{3.7}$$

where $\|.\|_{\chi^2}$ is the χ^2 distance, $\phi(.)$ is a histogram of filter response values and $g_l(I), l = 1, \ldots, L$ are the responses of the L filters. We chose the filter bank of [LM01] [3]. Finally, TQC is given by:

$$E_{TQC}(\hat{I}, I) = E_1(\hat{I}, I) + E_2(\hat{I}, I). \tag{3.8}$$

3.3.5 Inference of Texture Representation

Given a complexity constraint ξ, we have a bound on the number, $\Lambda \doteq \Lambda(r)$, of samples, ω_λ, that can be stored in $\bar{\omega}$, which depends on r, the scale of ω_λ. To

[3]The filter bank consists of 48 filters: first and second derivates of Gaussians at 6 orientations and 3 scales, 8 Laplacian of Gaussian and 4 Gaussian filters. We used the default parameters of [LM01].

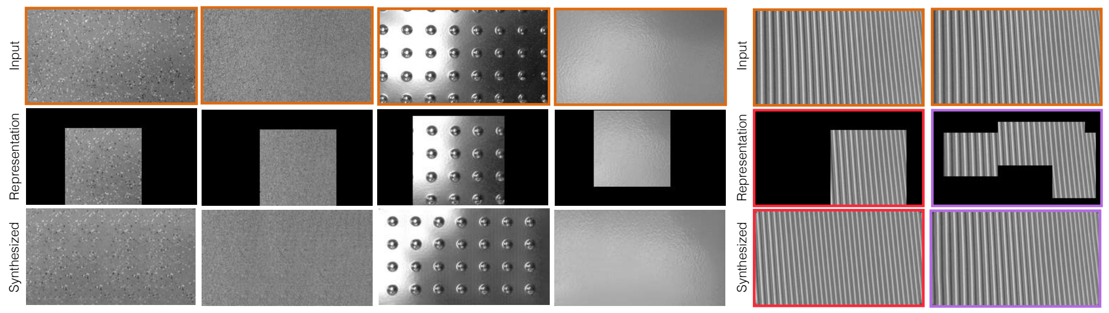

Figure 3.7: Texture representation, $\theta_{\bar{\omega}}$. Top: Input textures. Middle: Inferred representations. Bottom: Synthesized textures from the inferred representation. Right pair of images: Complexity ξ determines the representational power of $\theta_{\bar{\omega}}$. Increasing the number of stored samples, allows the representation to capture the domain transformation.

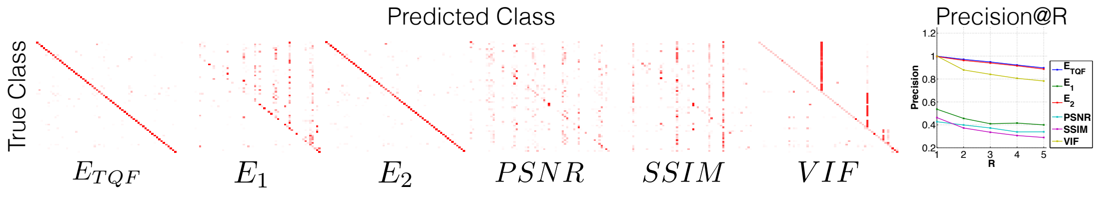

Figure 3.8: Left: Confusion tables for six competing methods. Right: Precision of methods for various values of retrieved nearest neighbors.

estimate r, the inference algorithm involves two levels of computation. The first level involves fixing r_{cand}, a candidate of r, and computing *which* samples $\omega_\lambda \subset \Omega$ should be stored in $\bar{\omega}$. This computation is repeated for a finite number of r_{cand}. To choose $\hat{r}$, an estimate of r, we use TQC to rank the representations and choose the best one according to this ranking. In this section, we describe this procedure in greater detail.

For each r_{cand}, we use Alg.1 (see Sec. 3.3.1) (or its variant if the input is a video) to synthesize a novel instance of the texture at just one scale, r_{cand}. Upon convergence, for each $\hat{\omega}_s$ there is an ω_s (its nearest neighbor), that is assigned to it. The set $\Omega_S = \{\omega_s\}_{s=1}^{S}$ denotes the collection of nearest neighbors within Ω and it is the entire data that the algorithm needs to synthesize the texture.

However, due to ξ, we need to choose which $\Lambda \leq S$ samples from Ω_S we store. To do so, we run the k-medians algorithm [BMS97], with an ℓ^2 distance and Ω_S as the input. The algorithm chooses the Λ cluster centers, $\omega_{\lambda_{cand}}$, from the set Ω_S that minimize the distance of all samples with their clusters. In this sense, they are the most "representative" samples of the underlying process. Since the cluster centers are samples from the distribution, if r_{cand} is the true Markov neighborhood scale, then the cluster centers approximate the variability of the texture. The cluster centers form $\bar{\omega}_{cand} \doteq \bigcup_{\lambda=1,\ldots,\Lambda} \omega_{\lambda_{cand}}$, which we can use to compute $\theta_{\bar{\omega}_{cand}}$. Using $\theta_{\bar{\omega}_{cand}}$, we re-synthesize the textures at multiple scales according to Sec. 3.3.1. We repeat this procedure for all r_{cand}. We choose $\hat{r}$ (an estimate of r) to be the scale that minimizes TQC. This ensures that the chosen $\hat{r}$ synthesizes the most similar texture among all r_{cand} and hence captures best the Markov neighborhood structure. We discretize the space of $\hat{r}$ to $[4^2, 8^2] \cup \{k \times 16^2\}_{k=1}^{K}$, where K is bounded by the scale of Ω. When Ω is unknown, we can bound the space of $\hat{r}$ to the image size, although empirically it can be set to a much lower value (e.g. 128×128 for 640×480 images). Once $\hat{r}$ is selected, we retrieve the corresponding estimates of $\bar{\omega}$ and $\theta_{\bar{\omega}}$ computed at scale $\hat{r}$. For video, we repeat this process on each frame independently.

3.3.6 Extending the Representation to Multiple Scales

To use the TQC, we need to assume that there is only one texture in the image domain. This poses issues when inferring the representation $\theta_{\bar{\omega}}$ and when evaluating the quality of reconstruction for images with more than one texture. To circumvent that, we partition the image domain into smaller regions and infer the texture representations on each one of these independently. We partitioned 640×480 frames into regions of 128×128, which in general gave us good results.

Since there is still no guarantee that there will not be more than one texture in each of these subsets, we allow the texture representation to be slightly more flexible: we allow ω_λ to take any of 3 different scales. Based on the restriction imposed by ξ, we distribute the available number of samples, Λ, to samples at 3 different scales i.e., $(\Lambda_r, \Lambda_{r/2}, \Lambda_{r/4})$, computed so that they satisfy the same complexity constraint. Since multiple choices would achieve this condition, we try all of them sequentially.

The above approach leads to forming a representation that is multi-scale: $\theta_{\bar{\omega}} = \{\theta_{\bar{\omega}_r}, \theta_{\bar{\omega}_{r/2}}, \theta_{\bar{\omega}_{r/4}}\}$, where the subscript on $\bar{\omega}$ denotes the scales of the samples ω_λ that belong to that particular $\bar{\omega}$. Like before, we repeat this process for various candidates of r and choose the best one according to TQC. Note that synthesis can still take place on the whole image domain and it is independent of the partitioning done for the inference of the texture representation.

3.4 Experiments

Texture Representation Analysis. To qualitatively evaluate the representational power of our scheme, we show in Fig. 3.7 a number of examples. With the exception of the example on the right, the algorithm picked in all others to represent the texture with just one sample, $\Lambda = 1$. $r = 96 \times 96$ for textures 1, 2, 4 and $r = 112 \times 112$ for texture 3. This is due to two reasons: (i) by storing ω_λ with large r, we can select neighborhoods at smaller scales within the stored ω_λ in texture synthesis, (ii) the textures shown with the exception of the last one, do not exhibit significant variation to require more than one sample to be stored. The first two exhibit little variation. In the next two, the representation captures the photometric variation by simply selecting (correctly) an ω_λ that exhibits this variation. The texture on the right exhibits a domain deformation. The algorithm is unable to identify this variability with $\Lambda = 1$ and $r = 96 \times 96$ (second to last

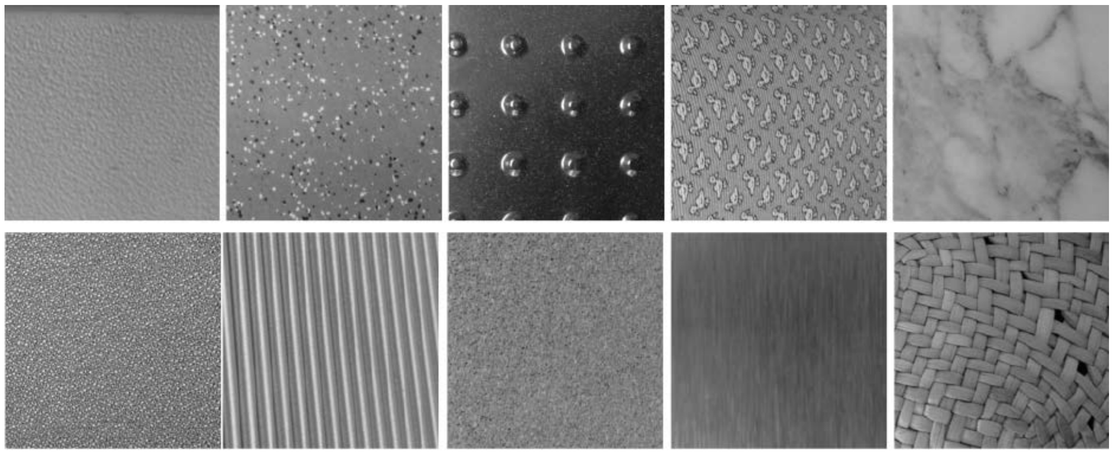

Figure 3.9: Texture dataset: 10 randomly selected examples.

column), but with more samples stored (rightmost column), it is able to do so. In that case, $\Lambda = 5$ and $r = 64 \times 64$. In Fig. 3.13, we show inferred representations for video. In such video sequences the structures provide boundary conditions between textures and the stored samples allow localization of the various transformations occurring on the textures.

Texture Qualitative Criterion. To evaluate *TQC*, we have constructed a dataset[4], made out of 61 classes of textures, with 10 samples in each class (10 randomly selected examples of classes are shown in Fig. 3.9). Each sample is compared against the other 609 texture images using six different quantities: E_{TQC}, E_1, E_2, $PSNR$, $SSIM$ [WBS04] and VIF [SB06]. For each image we retrieve R nearest neighbors using each of the six quantities and identify the class they belong to. We compute confusion matrices and show the result for $R = 5$ in Fig. 3.8, where the results are accumulated over all images. Furthermore, we plot the precision of each of the six methods also in Fig. 3.8, for $R = 1, \dots, 5$. E_{TQC} performs slightly better than E_2 and significantly better than the other methods. Note that E_1 is equivalent to [WHZ08], without using a control map.

To qualitatively evaluate *TQC*, we synthesized a number of textures using a

[4]`http://vision.ucla.edu/~giorgos/cvpr2015/`

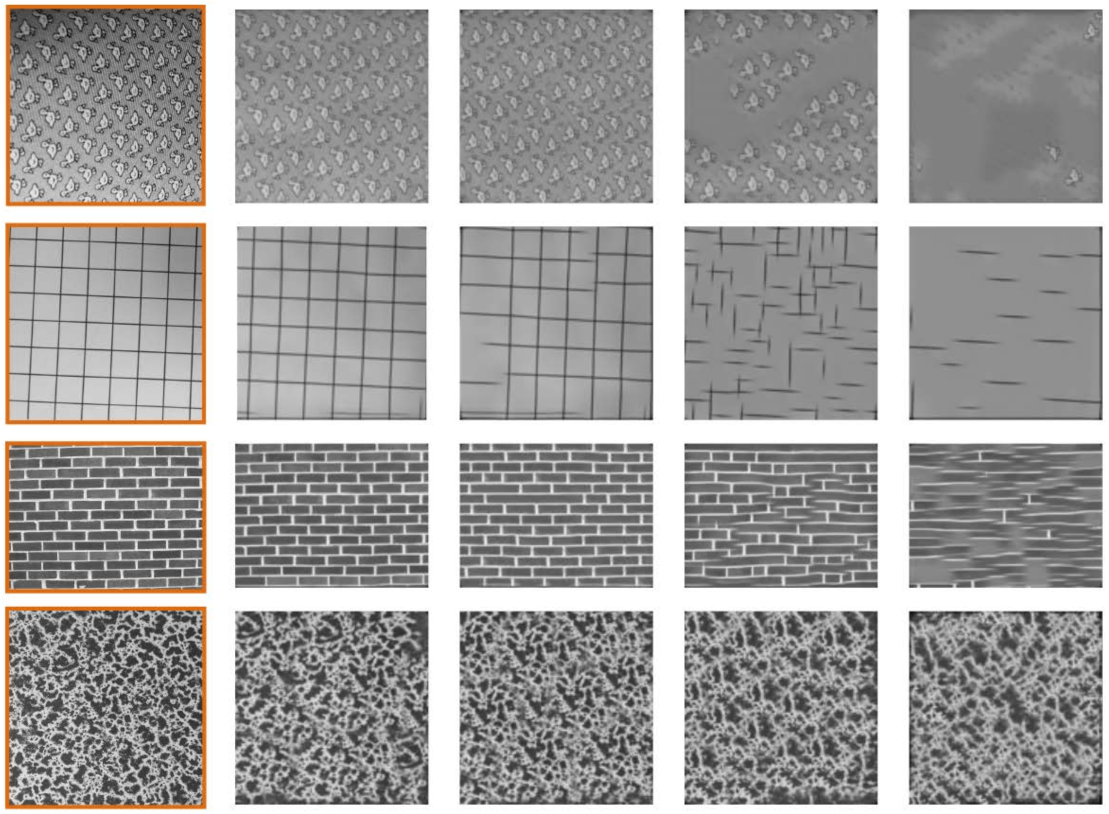

Figure 3.10: Texture Qualitative Function (TQC): Ordered synthesized textures using TQC. Left: Original textures. Right: Synthesized textures, left being the most similar to the input texture.

varying ξ and ω. We used TQC to order the synthesized textures and we show the ordered results in Fig. 3.10. The biggest benefit of E_{TQC} over E_2 is shown in Fig. 3.11. E_2 fails to detect artifacts in the synthesized textures, especially in regular ones, since the pooled statistics do not reveal any inconsistencies. On the other hand, E_{TQC} is able to identify these cases, since samples in the synthesized texture are artifacts of synthesis and hence are not present in the input texture.

Texture Synthesis. To evaluate the temporal extension of the synthesis algorithm, we decouple the representation from the synthesis procedure by letting $\xi = 1$. We initialize the output to a random video of the same length and size as the input. We use 20 frame long video input sequences to produce a novel instance

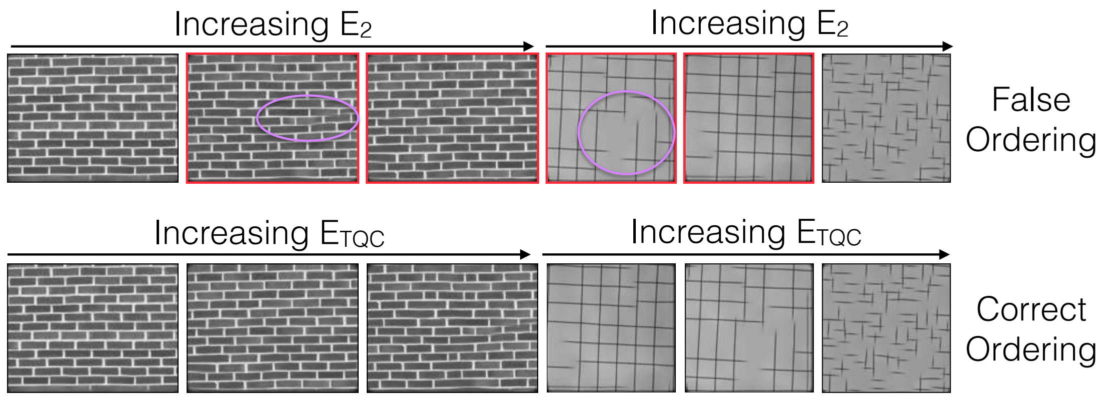

Figure 3.11: Two examples where E_2 fails and E_{TQC} succeeds in ordering the synthesized textures correctly with respect to the input texture. Images with a red outline have been incorrectly ordered. The issue arises mainly in regular textures. The regions within the purple ellipses are major sources of error.

of the texture (Fig. 3.12). Synthesized textures capture the statistics of the inputs and are temporally consistent.

Video Hole-Filling for Multiple Textures.

We synthesize multiple textures in the first 5 frames of 12 videos from the MOSEG dataset[BM10] at two different compression ratios using the inferred representations. We store the intensity values in 8-bit unsigned integers and the locations of the centers of ω in 16-bit unsigned integers. The space required to store the representation is $\Xi(\bar{\omega}) = \Lambda \times r \times 8 + \Lambda \times 2 \times 16$. The space required to store the texture regions without using the representation is $\Xi(\Omega) = 8 \times |\Omega|$. We let the compression ratio be $\xi = \frac{\Xi(\bar{\omega})}{\Xi(\Omega)}$. In Fig. 3.13, we show the last frame of the synthesized videos at $\xi = 40\%$. The results show that we can successfully compress textures in video using our representation; during decoding, the original video can be recovered by hole-filling. Note that we have also overlaid the detected structures on each frame. We control the number of feature points detected and hence the structure/texture partition, by empirically adjusting the detection threshold to retain in general long and stable tracks. In the same figure, we also show the results

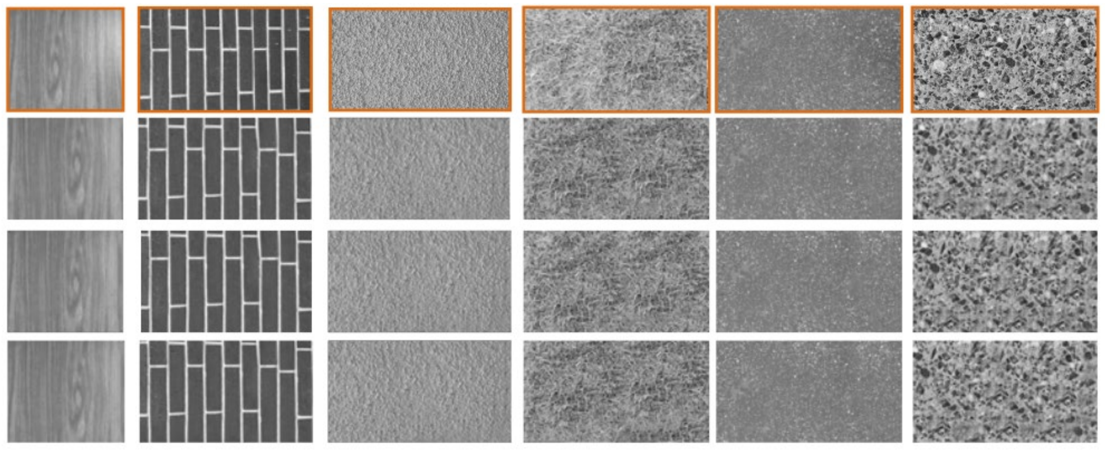

Figure 3.12: Temporal texture synthesis. Synthesizing a novel instance of a texture in a video sequence. The first frame of a 20-frame long input video sequence is shown in the top row. 1st, 10th and 20th synthesized frames are shown below each input image.

from [CFJ08], the most relevant work to ours, for the same target compression. We use their publicly available code and we have done our best to optimize the parameters to get the best results possible. Since their approach does not explicitly model textures, their method fails in synthesizing them well. For example, in the second sequence, the sky is only barely part of the "known" region and while our method is able to extrapolate and fill in the rest of it by synthesis, [CFJ08] is not able to do so from their summarization (epitome). In Table 3.1 we show the value of TQC for two target compression levels used in the experiments ($\xi = 40\%$ and $\xi = 60\%$) for each of the 12 video sequences. Examining the synthesized video sequences, it can be observed that the spatial artifacts for both methods are comparable (hence the values of TQC are approximately the same), but the temporal artifacts are significantly more for the higher compressed video, which TQC does not capture. An extension of this work would be to compute TQC on 3D samples, rather than 2D. For additional results, please refer to the project website[4].

Future Challenges. When the system is used at high compression ratios, syn-

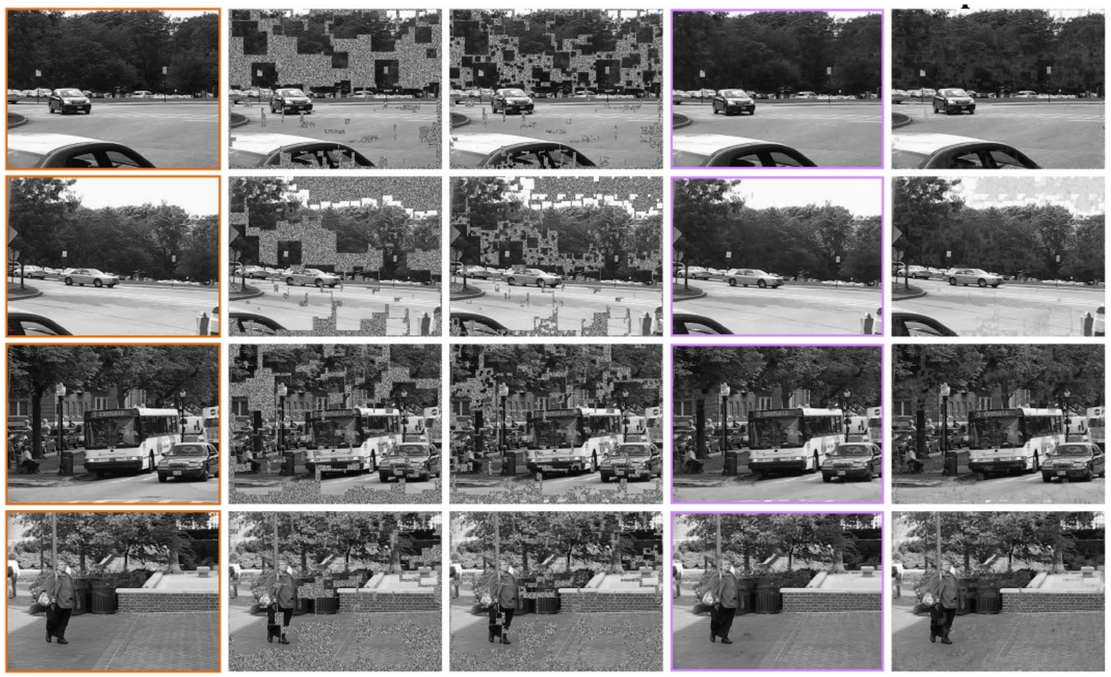

Figure 3.13: Video texture synthesis in natural images (Hole-filling). From left to right: (i) Last frame (5th) of input video, (ii) Structure regions, (iii) Structure / Texture regions, (iv) Synthesized frame (our result), (v) Video Epitome [CFJ08] (zoom in to view details).

thesizing large holes becomes challenging. Extending our coarse-to-fine approach to hole-filling could improve the results. In addition, to infer the representations in these videos, our algorithm requires an amount of time that is in the order of hours. Synthesis of frames typically takes 5-10 minutes. These times are comparable to similar methods [CFJ08, KSE03]. The reported time refers to a MATLAB implementation on an Intel 2.4 GHz dual core processor machine. In future work, we plan to find efficient approximations to speed up inference.

3.5 Discussion

A texture region exhibits a form of spatial regularity, hence it lends itself to spatial image compression techniques. We presented a new texture representation that

Seq-1	**Seq-2**	**Seq-3**	**Seq-4**	**Seq-5**	**Seq-6**
358(61%)	430(61%)	707(59%)	499(64%)	435(63%)	708(60%)
359(41%)	429(40%)	703(40%)	501(41%)	437(41%)	708(38%)
Seq-7	**Seq-8**	**Seq-9**	**Seq-10**	**Seq-11**	**Seq-12**
618(55%)	480(56%)	371(63%)	557(54%)	206(58%)	452(61%)
616(37%)	482(38%)	372(41%)	557(37%)	207(38%)	451(40%)

Table 3.1: Video texture synthesis evaluation. TQC for each of the 12 sequences on the texture regions (smaller is better). In brackets, we show the corresponding compression percentage achieved.

requires considerably less space to store than the input texture, while at the same time it retains many important characteristics: it can still be used as input to texture classification tasks and image and video texture synthesis. We presented a causal video compression scheme that utilizes the proposed representation and demonstrated how this can be used for hole-filling and texture compression in video. Finally, we proposed a new criterion to compare two textures that measures structural and statistical dissimilarity.

CHAPTER 4

Texture Compression

The notion of texture has a long history in visual perception, computer vision, computer graphics, computational geometry, content-based image retrieval, all with slightly different characterizations. In Chapter 3, we introduced and defined “Visual Textures”. In this chapter, we extend the definition so that domain and range transformations of the image are considered. Textures are defined in the context of *lossy compression.* To be more precise, we accept a loss, even a significant one, relative to the task of reproducing an exact replica of the original texture, as reflected for instance in the PSNR of their pixel-wise difference. However, ideally we would like our scheme to be *lossless* with respect to the task of *perception* by humans. Unfortunately, perceptual similarity is difficult to measure, and even more difficult to formalize analytically. Therefore, we postulate that images of textures are samples from some underlying stochastic process, and that perceptual similarity relates to the similarity between such processes. Similarity could be measured by some kind of distance between minimal sufficient statistics, if these could be computed. In particular, we assume that if two processes have the same sufficient statistics, the samples they generate are perceptually indistinguishable. This postulate is not unreasonable, provided we are willing to consider higher-order statistics, as the case of second-order [Jul62, Jul81] has long been refuted in psychophysical experiments [MP90]. So, if we could infer the minimal sufficient statistics of the underlying process from one sample of it (an image), we would encode and store them, and then at decoding just generate a random sample. This would be, by postulate, a (perceptually) lossless compression scheme.

Unfortunately, finite-dimensional minimal sufficient statistics exist only in special cases [JC73], and even then, they cannot be inferred from a finite sample. Therefore, our goal is to devise a *lossy* compression scheme by inferring *approximate sufficient statistics* of the underlying process, where fidelity is traded off with sample size. Since the "true" underlying process is not known, we cannot measure fidelity by comparing the estimated statistics with the true ones, but we can evaluate it by comparing samples generated from them. Therefore, ultimately the evaluation of a texture compression scheme has to be performed empirically, which we do in Sect. 4.4.

4.1 Prior related work and contributions

Texture compression relates to a vast literature on texture analysis ([MM02] and refs.), perception ([MP90] and refs.), synthesis and mapping ([EL99] and refs.), texture phenomenology ([SH98] and refs.), exemplar/patch based methods (structural vs. geometric vs. probabilistic [VZ03, Har05, ZWM98]) that we cannot realistically review in the limited scope of this paper. We refer the reader to [XMX08] for an overview. Our work is also naturally related to the Minimum Description Length (MDL) principle [Ris78], where the aim is to exploit regularities in the data to achieve compression. Whereas in traditional implementations of MDL and lossy MDL [MRY11, Bou11] the approach taken is to compress pixel values, in our work we aim to "compress" the source that generates them, i.e. the scene. Other traditional texture compression schemes typically 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 [Fen03, MP12].

Despite a wealth of work, however, there are relatively few attempts to clearly define and analytically characterize textures. In this work, we provide a definition based on standard concepts from stochastic processes such as stationarity, ergodicity, Markovianity (Sect. 4.2-4.3). The definition naturally lends itself to inference algorithms for encoding a texture, by inferring approximate sufficient statistics (Sect. 4.3.2), and decoding using non-parametric sampling, via a straightforward modification of [KEB05] that captures the correct Markov structure inferred from the data. We characterize the performance of our compression scheme empirically, and point to some challenges in the determination of the (multiple) intrinsic scales of textures. We assume that the domain where a texture region is defined is given to us, and focus on its compression (coding/decoding), as opposed to its segmentation from the non-texture region. Consequently, all examples we use in our experiments include images that contain exclusively texture regions.

4.2 Background

A (spatially) quantized image $\{I_{ij}\}_{(i,j)=1:(N,M)} \in \mathbb{R}^{M\times N}$ is obtained by averaging a function $I : D \subset \mathbb{R}^2 \to \mathbb{R};\ x \mapsto I(x)$ on a neighborhood of $x_{ij} \in D$ of size $\epsilon > 0$, $\mathcal{B}_\epsilon(x_{ij})$: $I_{ij} = \frac{1}{|\mathcal{B}_\epsilon|}\int_{\mathcal{B}_\epsilon(x_{ij})} I(x)dx$ where $|\mathcal{B}|$ is the area of $\mathcal{B}$. In general, $I_{ij} = I(x_{ij}) + n_{ij}$ where $n_{ij} = n_{ij}(I)$ is the quantization error.

In Chapter 3 we introduced and defined the notion of “Visual Textures”. In this chapter, we focus on how range and domain transformations of the image could affect the definitions introduced earlier. To do so, we re-introduce the main concepts (stationarity, ergodicity and Markovianity), but this time focus on the transformations of interest. As a result, certain concepts should appear familiar to the reader.

4.2.1 Stationarity

We interpret the quantized image as a sample (realization) from a process $\{I\}$ distributed according to a certain (unknown) distribution $I \sim dP(I)$. While the probabilistic description of this process can be technically problematic, sampling from it is straightforward, as it corresponds to measuring pixel values in certain subsets of the image domain. Consider a subset $\omega \subset \mathbb{Z}^2$, with cardinality $|\omega|$, and functions ϕ (statistics, or *"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: $dP(\phi_\omega) = dP(\phi(I(\omega))$. We also consider a group of planar transformations $g \in G$, with $g : \mathbb{R}^2 \to \mathbb{R}^2$. These represent "deformations" or "distortions" of the image due to, for instance, a change of vantage point, or a deformation of the scene [SPV09a]. 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 a $g \in G$ such that $\mathbb{E}(\phi_{g(\omega)})$ is translation-invariant, that is

$$\mathbb{E}(\phi_{g(\omega)}) = \mathbb{E}(\phi_{g(\omega)+T}), \quad T \in \mathbb{R}^2 \tag{4.1}$$

where $g(\omega) = \{g(x) \mid x \in \omega\} \cap \mathbb{Z}^2$ and $g(\omega) + T = \{g(x) + T \mid x \in \omega\} \cap \mathbb{Z}^2$. If (4.1) is satisfied only for T that belong to a discrete subgroup of planar translations (Frieze symmetries, see [LTL05]), then 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 $\bar{\omega} \supset \omega$, $dP(I)$ is *locally stationary* in $\bar{\omega}$ if (4.1) 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. 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*.

4.2.2 Ergodicity

A stationary process is ergodic if sample averages converge to ensemble averages (expectations):

$$\frac{1}{N}\sum_{i=1}^{N}\phi_{g(\omega)+T_i} \xrightarrow{\text{a. s.}} \mathbb{E}(\phi_{g(\omega)}) \tag{4.2}$$

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 the notion of Markovianity below.

4.2.3 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:

$$I(A) \perp I(A^c) \mid \phi_{\mathcal{N}(A)}. \tag{4.3}$$

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* (4.3) *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 (4.3) establishes $I(\mathcal{N}_\omega(A))$ 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.*

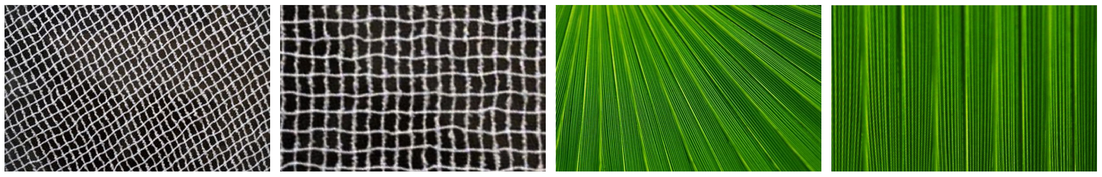

Figure 4.1: Affine and projective textures and their rectified versions. The transformation g can be determined in pre-processing via canonization [ZLG10, Soa10], or can be described to parametrize the statistic ϕ_ω and inferred as part of the compression process (i.e. in the search for ω).

4.3 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. 4.1), ergodic (Eq. 4.2), Markovian process (Eq. 4.3) 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* [Soa10]; for the specific case of the projective group, [ZLG10] provides a simple rank-minimization-based procedure (Fig. 4.1).

4.3.1 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, $\forall A \subset \Omega$,

$$I(A) \perp \underbrace{I(\Omega - A)}_{``outside''} \mid \underbrace{I(\mathcal{N}_\omega(A))}_{``inside''} \tag{4.4}$$

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

$$\mathbb{KL}\left(p(I(A)|I(\mathcal{N}_\omega(A))); p(I(A)|I(\Omega - A))\right) = 0 \tag{4.5}$$

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

$$\hat{\omega}(\beta) = \arg\min_{\omega} \left[sup_{A\in\Omega} H(I(A)|I(\mathcal{N}_\omega(A))) + \frac{1}{\beta}|\omega| \right]. \tag{4.6}$$

Note that this is a consequence of the Markovian assumption; it can be shown that the solution $\hat{\omega}(\beta)$ to (4.6), which can be seen as a version of the Information Bottleneck principle [TPB99], 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 (4.6) from sample data $H(I(A)|I(\mathcal{N}_\omega(A))) \simeq -\frac{1}{M}\sum_{i=1}^{M} \log p(I(A_i)|I(\mathcal{N}_\omega(A_i)))$, where $\mathcal{N}_\omega(A_i)$ 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 $\log p(I(A)|I(\mathcal{N}_\omega(A)))$, 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. [EL99] and section **??**) for fixed ω and $\bar{\omega}$. If we call $\hat{I}(A)$ the "synthesized" texture in A, this yields an estimator of the entropy $\hat{H}(I(A)|I(\mathcal{N}_\omega(A))) \doteq log\left(\frac{1}{M}\sum_{i=1}^{M} d(I(A_i),\hat{I}(A_i))\right)$. 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 (4.6) with a complexity cost on $\sigma = |\bar{\omega}|$:

$$\hat{r},\hat{\sigma} = \arg\min_{r=|\omega|,\sigma=|\bar{\omega}|} \hat{H}(I(A)|I(\mathcal{N}_\omega(A))) + \frac{1}{\beta}|\bar{\omega}|. \tag{4.7}$$

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 $p(I(A)|I(\mathcal{N}_\omega(A)))$.

4.3.2 Inference

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 [BNS10]. Because of the stationarity assumption, and given that we have chosen to parametrize ω with squared neighborhoods, we simply need to infer its size $|\omega|$. Consider therefore a region of growing size $|\omega_k| = 1, 2, \dots, n$ and any statistic ϕ computed in ω_k, $\phi(I(\omega_k))$. Its entropy as a function of k will follow a "staircase" behavior [BNS10], where the local minima correspond to $k_i = |\omega|$. This is consistent with the fact that textures exist at multiple scales (see Fig. 4.2). 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. 2.

Algorithm 2: Algorithm for inferring $\bar{\omega}$

```
Initialize a set $R = \emptyset$, and a threshold $\epsilon$
Sample $N$ patches $\{x_i : i = 1, \dots N\}$ of size $|\omega|$ from $\Omega$
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 squared sampled region from $\Omega$ of size $|R|$
```

We use Alg. 1 for texture synthesis.

4.4 Experiments

We show results of our texture compression algorithm in Fig.4.4. 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: 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

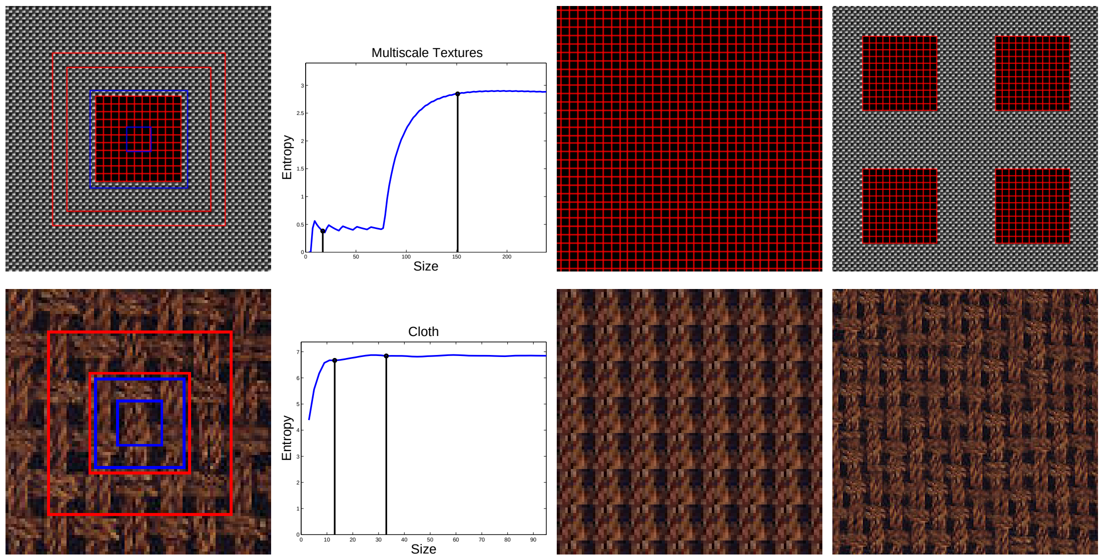

Figure 4.2: Multiscale analysis of textures. Top row, left to right: Texture "within" texture. Entropy plot. Synthesized texture at small scale, synthesized texture at a higher scale. Bottom row: Different textures appearing at different scales. The regions surrounded by the blue rectangles are the textures 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).

exceeds λ.

To determine $\bar{\omega}$, according to Sec 4.3.2, we need to accept "representatives" that are "close" to each patch sampled from Ω. In these experiments, we sample 3000 patches from Ω (which is related to the parameter $\theta = \theta(|\bar{\omega}|)$ mentioned in Sec. 4.2.2) and accept as a representative any patch that is less than 3×10^{-3} (per pixel distance) away from its nearest neighbor.

In Fig. 4.3 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 scale of ω. These sizes correspond to the small red boxes in Fig 4.4. Although the

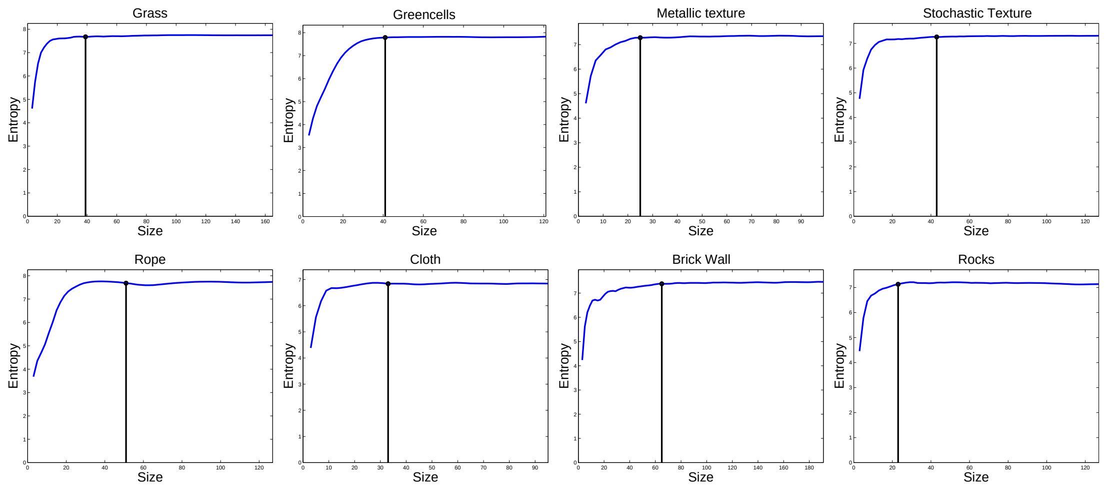

Figure 4.3: Entropy plots for the 8 textures shown in Fig 4.4. The black line indicates the scale (specifically the size of the side) of ω selected by our algorithm.

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. 4.2). 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. 4.2. 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

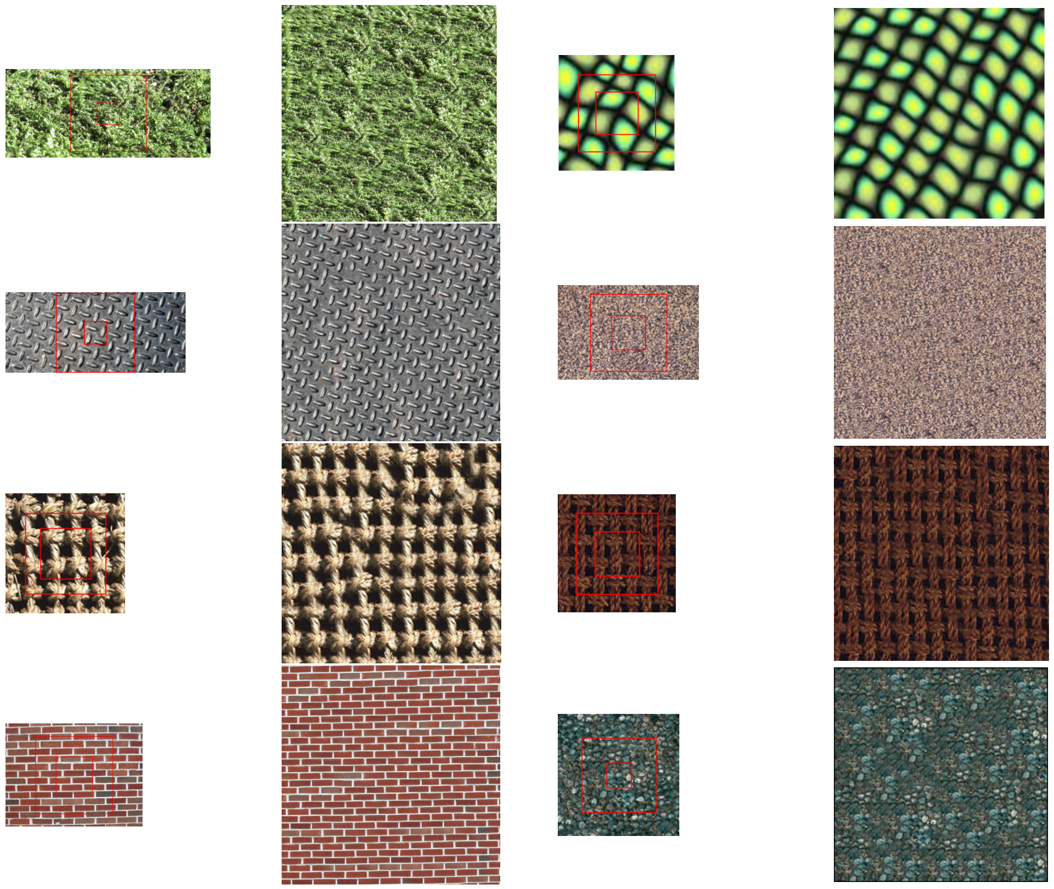

Figure 4.4: Odd Columns: Input texture. The large red box indicates the inferred scale of $\bar{\omega}$. The smaller red box indicates the inferred scale of ω. Even Columns: Synthesized textures from $\bar{\omega}$. The perceptual characteristics of the textures have been captured, indicating that $I(\bar{\omega})$ is indeed a Markov sufficient statistic, at least sufficient for the purpose of perceptual comparison.

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 limited use. Standard perceptual similarity scores, such as SSIM [WBS04], similarly fall short of capturing the perceived quality of the synthesized textures (see Fig. 4.5). We therefore filter the synthesized and original textures by a filter bank [LM01]; 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

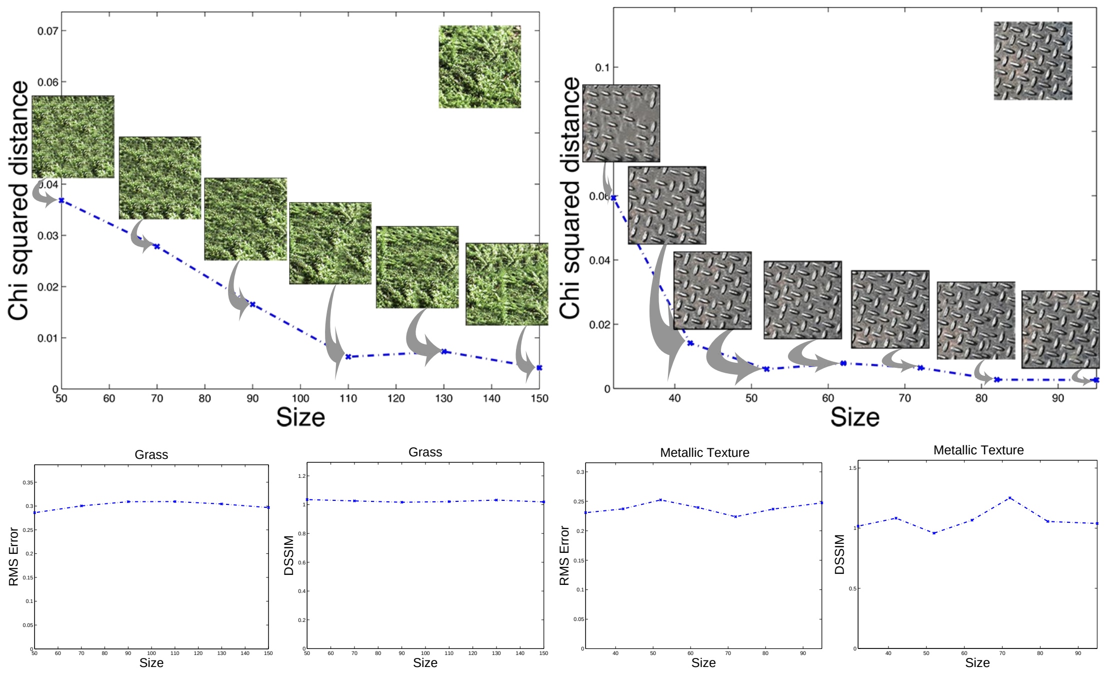

Figure 4.5: Rate-Distortion curves. Top: Mean distance of filter response histograms against $\bar{\omega}$ size. At each point, we show the synthesized texture given that scale of $\bar{\omega}$. At the top right we show the original texture. The qualitative behavior, as expected, indicates that larger size of $\bar{\omega}$ yields synthesized textures that are increasingly similar to the original sample. Bottom: Plots of RMS Error and DSSIM [WBS04] as 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.

textures. We take the texture (dis-)similarity to be the averaged result over all distances for each histogram pair (see Fig. 4.5). The distance decreases as the size of $\bar{\omega}$ increases indicating that the input and synthesized textures are becoming increasingly similar.

Fig. 4.6 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

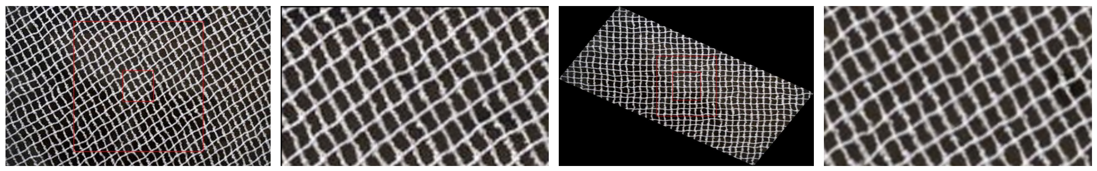

Figure 4.6: The texture in Fig. 4.1 is compressed and re-synthesized without prior rectification. (first and second figures). The texture is then rectified, compressed, re-synthesized and retransformed back with the inverse of the canonizing transformation (third and fourth figures). The two approaches achieve approximately the same perceptual quality but the rectified texture does so at a lower complexity cost ($|\bar{\omega}_{rectified}| \simeq |\frac{\bar{\omega}_{original}}{4}|$).

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

4.5 Discussion

We have presented a definition of textures in terms of standard concepts from stochastic processes such as stationarity, ergodicity, and Markovianity. We have then proposed algorithms to infer the constitutive elements of a texture, ω and $\bar{\omega}$, directly derived from the definitions. The inference yields a collection of different choices of Markov sufficient statistics, reflecting the multi-scale nature of textures.

Such statistics can then be used for compression purposes: the encoding is given by the statistics $I(\bar{\omega})$, and decoding is performed by texture synthesis via non-parametric sampling. Quantifying the performance of a texture compression scheme is non-trivial due to the absence of a universally accepted perceptual distortion score. We have used a score that quantitatively captures the perceived quality of the synthesized textures, and characterized the performance-complexity tradeoff empirically. In this work we have assumed that Ω (the stationarity domain) was given to us, or that equivalently the entire image or image patch is occupied by the texture. Next we will engage in the inference of Ω (texture segmentation).

CHAPTER 5

Texture Segmentation

Image segmentation concerns partitioning the domain of an image into coherent regions. We assume these are regions of space that project onto *simply connected* regions in the image, where *some* statistic is locally stationary and sufficiently different from its surroundings. Because of scaling phenomena in natural images, even the same statistic can exhibit rather different stationarity properties when aggregated in regions of different size. Thus, there is no single "correct" solution to segmentation, which remains an undecidable problem. Nevertheless, it is of great practical significance in a number of applications such as image editing, content-based retrieval and medical imaging.

In this paper, we exploit the robustness properties of region-based segmentation methods and mitigate their sensitivity to clutter by performing multiple bimodal/unilateral (foreground/background) partitions starting from multiple seeds. These can then be aggregated using a number of voting schemes or affinity clustering to produce a desired number of segments. Furthermore, we also put an emphasis on the issue of *scale.* Specifically, by adapting the scale parameter to the data we can employ a simpler local representation that does not require a multi-scale filter bank to capture the local statistics. An overview of our algorithm is shown in Fig. 5.1.

Because of the absence of ground truth, it has become customary to evaluate segmentation algorithms against human annotations. The most common evaluation schemes compare *a single* segmentation against multiple (often inconsistent) human

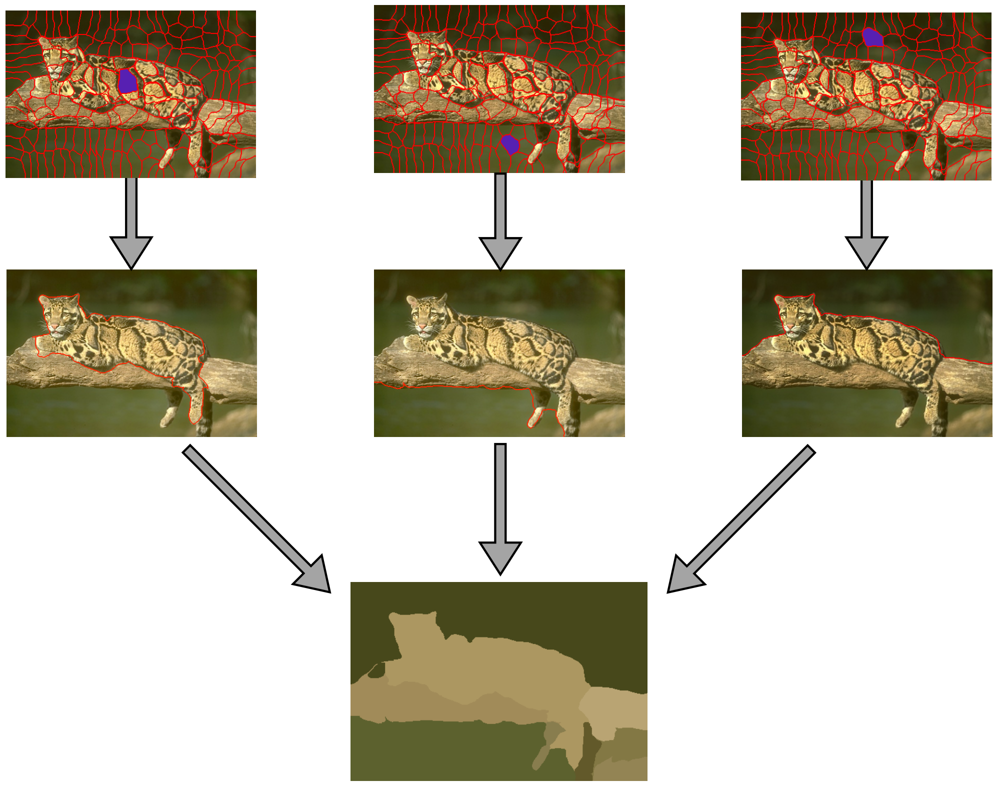

Figure 5.1: An overview of our method. Figure/ground partitions (middle) are computed for a number of seed regions (top, shown in purple). We then aggregate the boundary information deduced from each one of the partitions to produce the final segmentation (bottom). Boundaries of partitions are shown in red.

annotations. We compare our algorithm using this evaluation scheme with other approaches. We rank *first* in two common metrics, and *second* in the other one. Furthermore, we report sample results for a variety of natural images.

5.1 Related Work

Image segmentation in general has been approached in a variety of ways. Although the literature is too vast for us to review here, in broad strokes, these could be subdivided into contour-based or region-based methods. Arbelaez et al. [AMF11] falls in the first category. It first detects contours using brightness, color and texture cues and subsequently uses the Oriented Watershed Transform (OWT) to produce region boundaries. [RMY09] is a region based technique, in which the partition of the image domain is determined by greedily merging small coherent regions, *superpixels*, to minimize a cost function that depends on encoding regions and boundaries. Merging regions increases the coding cost of the combined region but removes boundaries and hence the optimal result is a trade-off of the two. Since regions are modeled by mixtures of Gaussians, the approach fails on non-GMM textures. Comaniciu et al. [CM02] partitions the image domain by finding modes of a probability density function and assigning each pixel to one of these modes. In [FH04], a graph is constructed, where nodes are pixels and edges measure dissimilarity between them. Regions are merged if an edge connecting the two regions is smaller than a certain threshold that depends on the two regions. Cour et al. [CBS05] is a generalization of [SM97], which is a multiscale spectral image segmentation algorithm. Affinity matrices at multiple scales are set up and the output image segmentation is a result of seeking a solution that is consistent across all scales, a requirement that is enforced by design constraints.

A number of works approach image segmentation through supervised learning. Carreira et al. [CS10] also compute figure/ground hypotheses. Once computed, figure/ground partitions are ranked using ground truth annotations. A model is trained that is used to produce one segmentation. A similar approach is also taken by [EH10]. [WT13] and [RS13] proposed methods that grow regions from an over-segmentation, but unlike us, both [WT13] and [RS13] require priors. Furthermore, [WT13] also requires class-specific shape priors aiming to produce a semantic image labeling, whereas we use a bottom-up approach, hence it is not directly comparable to our work.

Region-based binary (figure/ground) partition methods, for instance [CV01], seek to partition the image into two regions, by evolving a boundary model (represented, for instance, with a signed-distance function) in such a way that some statistics – computed in each of the two regions – are "maximally divergent" in some sense, for instance measured by the Kullback-Leibler divergence between color histograms. Thus, the two regions are "in competition" [ZLY95], and because of the bi-modal partition, the resulting optimization is convex [CE05]. However, while it is often the case that some object of interest or some part of it (which we may want to call "foreground") is well represented by the mode of a sample distribution of pixel values, this is often not the case for everything else (which we may want to collectively call "background" or "clutter"). Whatever (usually simple) statistics are computed in the background, once they are aggregated over the entire background, they lose discriminative power. This problem is well known and dates back to the Mumford-Shah functional and its application in medical imaging [TYW00]. This has been traditionally approached by performing multi-modal segmentation, which involved a far more complex logical combination of level set functions, which was no longer a convex problem. However, it is often the case that the statistics of the object of interest are sufficiently different from its immediate background [SSY10]. Thus we wish to leverage on the power and robustness of region-based methods to segment *multiple* possible foregrounds from their (local) background, and then perform a robust aggregation scheme to arrive at a global partition. Aggregation of multiple partitions has been found to improve localization compared to single segmentation approaches [ME07].

Donoser et al. [DUH09] follows a similar approach to us, by generating several figure/ground segmentations and then aggregating them to produce the final segmentation result. The algorithm first selects seed regions for the figure/ground segmentations by detecting salient regions in the image. To obtain the figure/ground segmentations a weighed total variation segmentation method is used, which minimizes a convex energy functional. Where the various segmentations are overlapping, they are combined using a maximum likelihood decision to determine the region membership of these pixels. The decision is made on estimated probability maps which are calculated during the figure/ground segmentations. This method suffers from a number of limitations: (i) Gaussian models are built for foreground/background regions that are used to determine the segmentations and the probability map, so it fails when non-Gaussian regions are encountered, (ii) the salient region detection is often not sufficient, since it is necessary as a post-processing step to initialize new seeds for unlabeled pixels, and (iii) all small unassigned regions are given a membership using an ad-hoc rule. Instead, we take a non-parametric approach, by representing regions with their histograms, which allows us to handle a larger set of regions that are not necessarily Gaussian. We avoid the salient region detection step, which is unreliable, and instead start from all possible (or unlabeled) seed regions, which in our formulation correspond to superpixels.

Once the figure/ground segmentations are computed, our algorithm aggregates them in one affinity matrix, which is then used in NCuts [SM97] to produce the final segmentation. In this sense, our work is also related to others that make use of the NCuts formulation. [MVM11] uses a weighted average of the Normalized Laplacian eigenvectors to correlate eigenvectors with a seed vector. In this way, it allows them to incorporate top-down priors or user input to produce a segmentation of the image. Our work differs from this, and also from [YS01], since it does not require any prior knowledge.

In our work we first formulate the problem of foreground/background partition and show how we can segment out the foreground region for individual seed regions. We then describe how these multiple foreground/background partitions can be combined to create an affinity matrix for pairwise region relationships, which is used to obtain the final segmentation.

5.2 Foreground / Background Partition

In order to benefit from the robustness and simplicity of region-based segmentation methods, but without making stringent assumptions on the clutter distribution, we employ a unilateral segmentation scheme starting from a "seed region", $s_i(0) \subset D$, where D is the domain of an image I, that is assigned to be foreground *by fiat*. On each iteration t, its boundary is evolved, driven by a distance between color distributions with its local neighborhood $n_i(t) \subset D$, until convergence. We call the final, *connected* region "foreground" and denote it by $s_i(\infty)$. We initialize the algorithm by partitioning D into superpixels by over-segmenting the image into a large number of small and coherent regions [RM03, MRE04] and use the superpixels as "seed regions". We use the publicly available code from [MRE04] with $N.sp = 200$. At the end of this stage, we are left with a collection of foreground/background partitions initialized from each superpixel, which are combined together using the technique explained in Sec. 5.4 to form one segmentation.

The notation $I(s_i(t)) \doteq \{I_{pq}, \forall x_{pq} \in s_i(t)\}$, where x_{pq} is a pixel at location $(p, q) \in D$, denotes the ensemble of intensity values in the subset $s_i(t)$. For color images, we indicate the channel index, k, using the notation $I^{(k)}$. We assume that the values $\{I_{pq}\}$ are in the range $[0, 1]$, possibly after contrast normalization. Furthermore, note that we define the local neighborhood $n_i(t)$ to consist of first and second-order superpixel neighbors of $s_i(t)$. First order neighbors of a region $s_i(t)$ are superpixels that are directly connected with $s_i(t)$ (i.e. share a boundary). Similarly, second order neighbors of $s_i(t)$ are superpixels that are directly connected to its first order neighbors. As $s_i(t)$ expands through the iterations, $n_i(t)$ adapts to the analogous local neighborhood of the current foreground region.

We choose as local discriminative statistics the normalized color histograms, $\phi^k_{s_i}(t) \doteq \phi(I^{(k)}(s_i(t))), \phi^k_{n_i}(t) \doteq \phi(I^{(k)}(n_i(t)))$ in $s_i(t)$ and $n_i(t)$ respectively for each of the $k = 1, \ldots, K$ channels (where $K = 1$ for grayscale and $K = 3$ for colored images), as a non-parametric estimate of the (marginal) distributions of intensity values in each color band. To determine the foreground region $s_i(\infty)$, we first define a function that captures the quality of a foreground/background segmentation:

$$Q(s_i(t)) = \sum_{k=1}^{K} \|\phi(I^{(k)}(s_i(t))) - \phi(I^{(k)}(n_i(t)))\|_{\chi^2} \tag{5.1}$$

As $s_i(t)$ evolves through the iterations, $Q(s_i(t))$ increases when superpixels added to the foreground increase the color histogram distance. These superpixels are those that have a color histogram similar to $s_i(t)$ and dissimilar to the rest of the background region $n_i(t)$. Hence the quality of the segmentation increases when the foreground is made dissimilar from the background. Note that driving the segmentation by making the foreground "maximally divergent" from the background is rooted in the literature of region-based segmentation [TYW00]. Therefore we seek to maximize $Q(s_i(t))$, while keeping $s_i(t)$ a *connected* set. We call the maximizer $s_i(\infty)$.

To determine $s_i(\infty)$, starting from $s_i(0)$ (a superpixel), we iteratively test the hypothesis, whether at iteration t a superpixel in the background region, $s_j \in n_i(t)$, in fact belongs to the foreground, $s_j \in s_i(t + 1)$, by testing if the quality of the segmentation *increases*:

$$\begin{aligned} Q(s_i(t) \cup s_j) \geq Q(s_i(t)) \Leftrightarrow \\ \sum_{k=1}^{K} \|\phi(I^{(k)}(s_i(t) \cup s_j)) - \phi(I^{(k)}(n_i(t) \setminus s_j))\|_{\chi^2} \geq \\ \sum_{k=1}^{K} \|\phi(I^{(k)}(s_i(t))) - \phi(I^{(k)}(n_i(t)))\|_{\chi^2} \end{aligned} \tag{5.2}$$

If this condition is satisfied, we include s_j in $s_i(t + 1)$ and remove it from

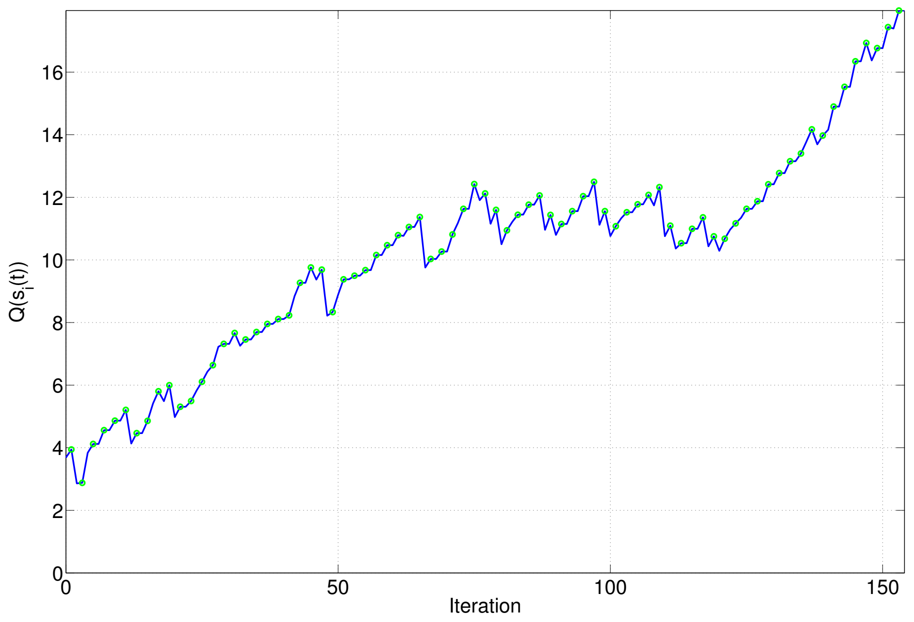

Figure 5.2: A typical run of a foreground expansion. Through the iterations $Q(s_i(t))$ increases when a superpixel is added (green points indicate inclusion of a superpixel). Whenever the local background is updated, $Q(s_i(t))$ may decrease.

$n_i(t+1)$. We also inflate $n_i(t)$ by letting $n_i(t+1)$ be the first and second order superpixel neighbors of $s_i(t+1)$. We then update the histograms of the two regions and repeat the test on other superpixels in n_i (using a lexicographic ordering for selection). The test for inclusion is done only if a superpixel is neighboring with $s_i(t)$ (*neighboring constraint*). An alternative to lexicographic ordering would be one that we test all superpixels and include the one that gives maximal increase. We found that such approach works as well as the one we follow, but it is much slower. The expansion is repeated until no other superpixel neighbors pass the test or until the image boundaries are reached; in which case we set the iteration index to $t = \infty$ and call the resulting region $s_i(\infty)$ a foreground hypothesis.

In Fig. 5.2 we show a run of the foreground/background partition for *one* superpixel. $Q(s_i(t))$ is computed at every iteration t. It is initialized by computing

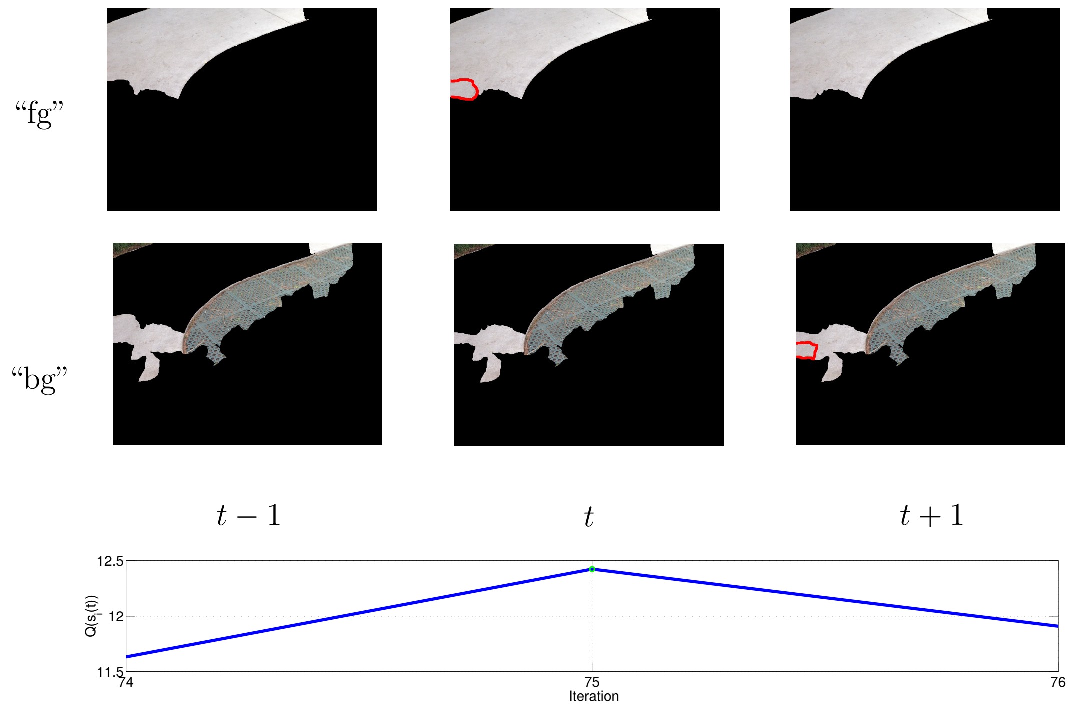

Figure 5.3: Top: The current foreground ("fg") and local background ("bg") shown for three consecutive iterations. At $(t-1)$ we show the current state of the two regions. At time t a superpixel is added (its boundary is shown in red in the top-middle figure), which makes the foreground to expand, whereas the background stays the same. This leads to an increase of $Q(s_i(t))$. At $(t+1)$, the local background is recomputed (the boundary of the superpixel added is shown in red in the bottom-right figure), and this (in this case) leads to a temporary decrease in the value of $Q(s_i(t+1))$. Bottom: $Q(s_i(t))$ for the three iterations of the foreground expansion shown in this figure.

$Q(s_i(0))$ for the seed region, i.e. for a superpixel. Every time a superpixel is added to the foreground $Q(s_i(t))$ *necessarily* increases. Once the foreground is updated

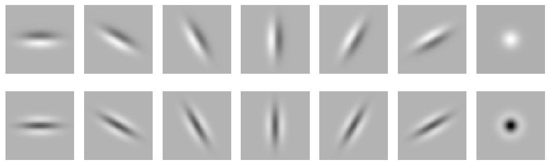

Figure 5.4: The filter bank used: edge, bar and blob filters at one scale.

with the addition of a superpixel, the local background is also updated to include the first and second order neighbors of the new foreground. Therefore this step changes the statistics of both the foreground and the background. This means that after the regions are updated, it could be possible that $Q(s_i(t))$ decreases. A decrease in $Q(s_i(t))$ indicates that the new local background now includes superpixels that have statistics that are similar to the foreground. In further iterations, these regions are tested for inclusion and when added they increase $Q(s_i(t))$, making the foreground dissimilar to the background.

5.3 Scale-adapted filter responses

To increase the discriminative power, we extend the hypothesis test to higher-order statistics than the histogram of color values, as customary in the analysis of texture [HB95, VZ03].

We choose a simple filter bank consisting of two directional filters at one scale and 6 different orientations, a Gaussian low-pass filter and a Laplacian of Gaussian filter, for a total of $L = 14$ filters. This is equivalent to a single-scale version of [LM01] (see Fig. 5.4). Thus, rather than applying a *multi-scale* ("texture") segmentation scheme, we perform content-specific *scale-selection* [Lin98], by adapting the scale of the filter bank to the size of *each* s_i we are expanding.

The selection of the scale of the seed region is not only useful to determine the size of the domain of the filter bank, but it is also critical to ensure sufficient sampling of the statistics used to compute the inclusion test. If the region is too small, the sample distribution will not be representative; if it is too large, with high probability it will straddle the boundary and produce a mixture distribution. Therefore, the adaptation of the scale is critical and should depend on the content of the image.

Scale selection has been investigated for natural texture, where the entropy of a statistic is computed on a region of growing area, and the radii corresponding to local extrema of entropy are taken as the "natural scale(s)" of the texture. In our case, we consider the area of the region $|s_i(0)| = \sigma$ as the scale for the current hypothesis. The scale of the domain of the filters is adapted to the size of $s_i(0)$ as $0.15 \times \sigma^{1/2}$. Test (5.2) then generalizes to:

$$\begin{aligned} Q(s_i(t) \cup s_j) \geq Q(s_i(t)) \Leftrightarrow \\ \sum_{k=1}^{K} \sum_{l=1}^{L+1} \|\phi\left(g_l\left(I^{(k)}\left(s_i(t) \cup s_j\right)\right)\right) - \\ \phi\left(g_l\left(I^{(k)}\left(n_i(t) \setminus s_j\right)\right)\right)\|_{\chi^2} \geq \\ \sum_{k=1}^{K} \sum_{l=1}^{L+1} \|\phi\left(g_l\left(I^{(k)}\left(s_i(t)\right)\right)\right) - \\ \phi\left(g_l\left(I^{(k)}\left(n_i(t)\right)\right)\right)\|_{\chi^2} \end{aligned} \quad (5.3)$$

where $\phi(.)$ generalizes to a histogram of intensity or filter response values, $g_1\left(I^{(k)}(\omega)\right) \doteq I^{(k)}(\omega)$, $g_l\left(I^{(k)}(\omega)\right), l = 2 \ldots L+1$ are the responses of the L filters. To partition the image into foreground/background for a particular seed region, we start from a superpixel s_i and run Alg. 3. Typical results of the foreground/background segmentation stage are shown in Fig. 5.5.

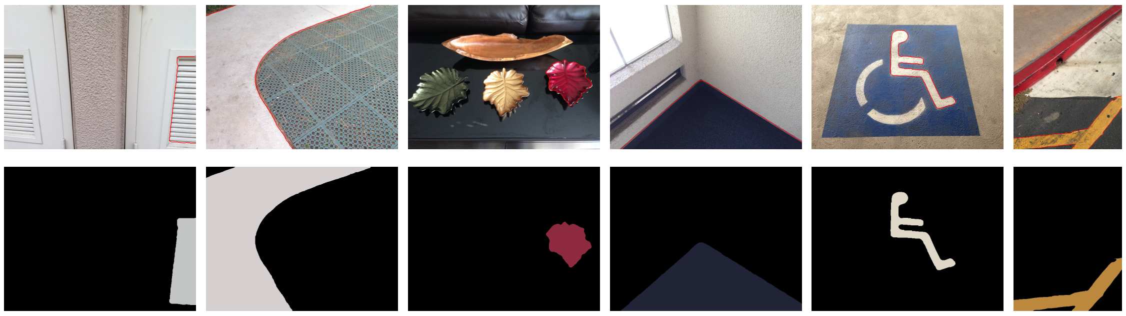
Figure 5.5: Foreground/Background partitions for selected seed locations. Given a seed region $s_i(0)$, the algorithm expands it to form $s_i(\infty)$. Top: Boundaries of segmented regions, $s_i(\infty)$, are shown in red. Bottom: $s_i(\infty)$ are represented by their mean intensity values.

Algorithm 3: $s_i(\infty) = \textit{expanded\_superpixel}(I, s_i)$

```
Input: Image $I$, and a seed $s_i$
Output: Segmented region $s_i(\infty) \subset \{(p,q)\}, (p,q) = 1:(N,M)$
  Let $t = 0$, initialize $s_i(0) = s_i$ and compute $n_i(0)$
  Let $\mathcal{N}(s_i(t)))$ be the first order superpixel neighbors of $s_i(t)$
  foreach $s_j \in \mathcal{N}(s_i(t))$ do
    Compute $d^- = Q(s_i(t))$
    Compute $d^+ = Q(s_i(t) \cup s_j)$
    if $d^+ \geq d^-$ then
      Update $s_i(t+1) := s_i(t) \cup s_j$
      Recompute $\mathcal{N}(s_i(t+1)))$ and $n_i(t+1)$
      Let $t := t+1$
Return $s_f(\infty) = s_i(t)$
```

5.4 Combining multiple partitions

Natural images do not contain “foregrounds” and “backgrounds”. Instead, there are multiple regions that, because of statistical homogeneity within, could be considered putative “foreground” regions. Therefore, it is neither realistic nor desirable to expect that the procedures above, initialized from multiple seeds, will converge to the same result. Therefore, there remains the question of how to combine the results of multiple seed evolutions to produce a partition of the image.

The resulting multiple segmentations are valid hypotheses of foreground regions. We utilize affinity to congeal the multiple hypotheses into one. For each $s_i(\infty)$, we build an affinity matrix in which pairwise relationships between $s_i(\infty)$ and its neighboring superpixels are encoded. We then aggregate these matrices into one and use that for the final segmentation.

Once all $s_i(\infty)$ are computed, we determine their neighboring superpixels, $\mathcal{N}(s_i(\infty))$. For each superpixel $s_m \in \mathcal{N}(s_i(\infty))$, we compute its distance from $s_i(\infty)$ using $d(s_m, s_i(\infty)) = \sum_{k=1}^{K} \sum_{l=1}^{L+1} \|\phi\left(g_l\left(I^{(k)}\left(s_m\right)\right)\right) - \phi\left(g_l\left(I^{(k)}\left(s_i(\infty)\right)\right)\right)\|_{\chi^2}$. For all pairs $s_m, s_n \in s_i(\infty)$, we set $W_i(s_m, s_n) = W_i(s_n, s_m) = 1$ (based on the evidence that all superpixels grouped in the foreground belong to the same region) and for all pairs $s_m \in \mathcal{N}(s_i(\infty))$ and $s_n \in s_i(\infty)$, we set $W_i(s_m, s_n) = W_i(s_n, s_m) = e^{-d(s_m, s_i(\infty))/\rho}$, where $\rho = 0.5$ in our experiments. We build affinity matrices using each $s_i(\infty)$ and as a last step, we let the aggregated affinity matrix $W = \overline{W_i}$ be the sample average. We estimate the number of regions from W, by calculating its eigenvalues, sorting them and automatically detecting the point at which the absolute value of the *first order difference* of the eigenvalues drops below a threshold ϵ. We choose as the number of regions the number of eigenvalues that have a first order difference larger than that threshold. Finally, we use NCuts [SM97] to calculate the final segmentation, by using the code available from [CYS04]. The overall algorithm is given in Alg. 4. To reduce the computational overhead we run Alg. 3 from a superpixel s_i if and only if it is not included in a region $s_j(\infty)$, i.e. if $s_i \notin s_j(\infty)$, $\forall j = 1, \ldots, i-1$. Note that typically there will be more than one region that a certain superpixel belongs to, hence multiple W_i will be contributing to its aggregated pairwise affinities.

Algorithm 4: Aggregating figure/ground partitions

```
Input: Image $I$
Output: $S$: Region labeling for image $I$
 Initialize $\rho = 0.5$
 Calculate superpixels of $I$ using [MRE04]
 Let $N_s$ be the total number of superpixels
 foreach $s_i \notin s_j(\infty), \forall j = 1, \ldots, i-1$ do
     Let $s_i(\infty) = expanded\_superpixel(I, s_i)$ (Alg. 3)
     Initialize matrix $W_i = 0^{N_s \times N_s}$
     $\forall s_m, s_n \in s_i(\infty)$, set $W_i(s_m, s_n) = W_i(s_n, s_m) = 1$
     $\forall s_m \in \mathcal{N}(s_i(\infty))$ and $s_n \in s_i(\infty)$,
         set $W_i(s_m, s_n) = W_i(s_n, s_m) = e^{-d(s_m, s_i(\infty))/\rho}$
 Let $W = \overline{W_i}$
 Estimate the number of regions $N$:
     Let $\lambda_i$ be the sorted eigenvalues of $W$ in decreasing magnitude
     Select as $N$, the number of eigenvalues that have a first order
         difference $\lambda_i - \lambda_{i+1} > \epsilon$
 Run NCuts [CYS04] with $W$ and $N$ for parameters to get the NCut
     eigenvectors
 Let $S(s_i), i = 1, \ldots, N_s$ be the membership of each superpixel
     given the discretized eigenvectors
Return $S$
```

5.5 Experiments

We discuss three applications of our algorithm: (1) Foreground/background partition, (2) Image segmentation and (3) Multi-scale segmentation.

Foreground/background partitions. In Fig. 5.5 we show a number of segments that have been computed using our foreground expansion method. By maximizing

BSDS300 [MFT01]	**Covering**	**PRI**	**VI**
Human	0.730	0.868	1.163
Our method	**0.597**	**0.813**	1.737
gPb-owt-ucm [AMF11]	0.590	0.810	**1.650**
Total Var. [DUH09]	0.570	0.780	1.810
TBES [RMY09]	0.540	0.780	1.860
Mean Shift [CM02]	0.540	0.780	1.830
Felz-Hutt [FH04]	0.510	0.770	2.150
NCuts [CBS05]	0.440	0.750	2.180

Table 5.1: Comparison of various segmentation methods in the BSDS300 testing set. The results of competing algorithms reported are taken from the study by [AMF11]. Best results (ignoring "Human") are shown in boldface. Our method outperforms all other methods in the PRI and Covering metrics and scores second in the VI metric.

the quality of the segmentation Q as defined, it allows us to expand the foreground region to include superpixels with similar histograms but exclude dissimilar ones. This allows the algorithm to handle some variation in the appearance of the foreground, since we do not penalize explicitly variability of the foreground (e.g. the foreground in the second image from left in Fig. 5.5). On the other hand, it always excludes dissimilar superpixels, ensuring that the superpixels included in the foreground region have similar visual properties. This observation lead us to the decision to define the affinities between superpixels in the foreground to be 1, as explained in Sec. 5.4.

Image segmentation. We test our algorithm on the Berkeley Segmentation Dataset (BSDS300), which consists of 200 training and 100 test images. Each image has multiple ground-truth segmentations. Our method is compared against 6 other methods. We use the following region-based metrics: (i) *Probabilistic*

Rand Index (PRI) [M 71], (ii) *Variation of Information* (VI) [Mei05] and (iii) *Covering* [AMF11]. *PRI* measures the agreement between two segmentations, $S = \{s_1, \ldots, s_r\}$ and $G = \{g_1, \ldots g_m\}$, of n pixels and it is simply the accuracy of the algorithm. It ranges between $[0, 1]$. *VI* is given by: $V(S, G) = H(S) + H(G) - 2 \times I(S, G)$, where $H(.)$ is the "entropy" of the segmentation and $I(.)$ is the mutual information of a pair of segmentations. *Covering* is given by $\mathcal{C}(S \to G) = \frac{1}{n} \sum_{g_i \in G} |g_i| \max_{s_i \in S} \mathcal{O}(g_i, s_i)$, where $\mathcal{O}(g_i, s_i)$ is the overlap between two regions g_i, s_i and is given by $\mathcal{O}(g_i, s_i) = |g_i \cap s_i| / |g_i \cup s_i|$. For multiple ground truth images we take the average over the computed values between our segmentation and each of the ground truth segmentations. For *PRI* and *Covering*, higher values indicate a higher agreement, whereas for *VI* lower scores are desired.

The results are reported in Table 5.1. Note that the result reported for "Humans" for each one of these metrics is calculated by computing each metric for each ground truth segmentation against the rest of them and then the result averaged over all images and humans. For qualitative evaluation of our method, we provide segmentation results in the training and testing set, shown in cvpr14/figs. 5.6, 5.8. Note that [DUH09] is the most similar method to us, which we show to outperform significantly in all three metrics. Overall, we rank first in two out of three metrics and second in the third one. By aggregating multiple foreground/background hypotheses for multiple seed regions allows us to build pairwise superpixel affinity relationships that are more robust than a single-pass image segmentation algorithm. Furthermore, the accuracy of the foreground segments plays a vital role in determining an accurate affinity matrix for image segmentation.

For the reported results, we found that for histograms of intensity values and filter responses, 25 bins gave us the best results. Using fewer bins yielded an increase in speed with only a slight performance degradation. The scale of the domain of the filters was chosen according to Sec. 5.3. For the rest of the filter parameter values, we used the largest scales of [VZ05]. We have chosen to compare

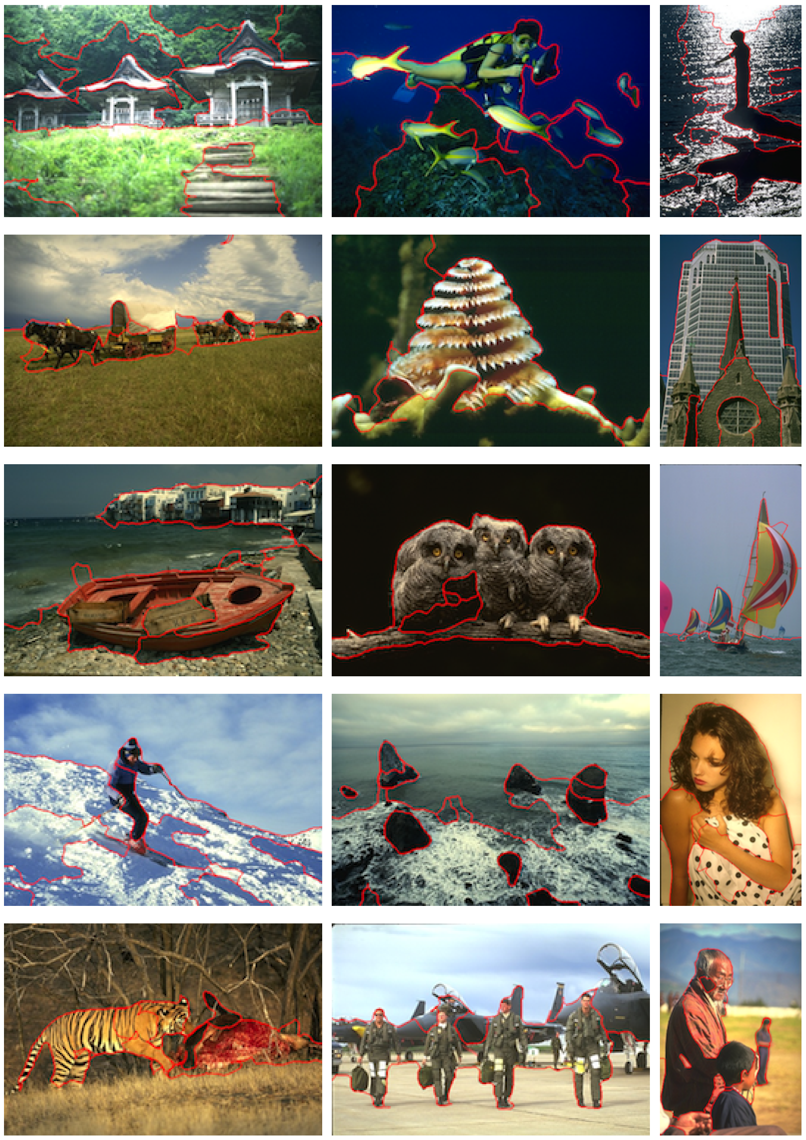

Figure 5.6: Segmentation results in the BSDS300 training set. Detected boundaries are shown in red.

histogram distances with χ^2 instead of l_1 for performance reasons. For a more extensive comparison between the two distances see [PW10]. As suggested by others [RMY09], all images were converted to the LAB colorspace [Wan95] giving us an extra performance boost. These parameters were selected by optimizing our results in the training set.

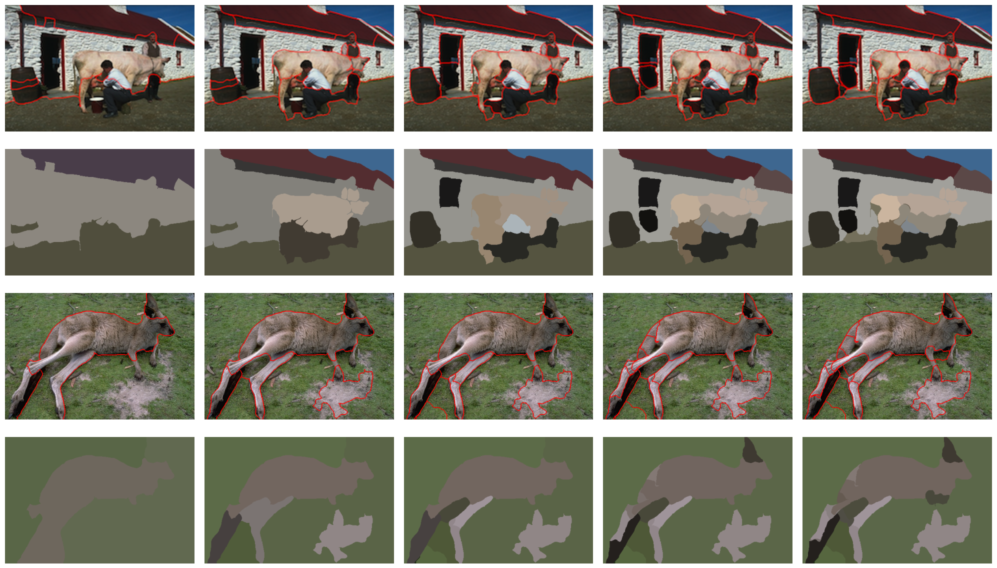
Figure 5.7: Multi-scale segmentation results. By varying the number of regions, we can produce segmentation results at different scales. From left to right: $3, 7, 11, 15, 19$ regions.

The algorithm takes on average 96 seconds to complete, excluding the computation of the superpixels, for a 481×321 sized image, with a MATLAB implementation on an Intel 2.4 GHz dual core processor machine. Note that this is significantly faster than the second best performing algorithm [AMF11]. The main computational burden comes from calculating the distance between the histograms, which is linear in the number of bins.

Multi-scale segmentation. Depending on the algorithm, multiple segmentations can be produced by varying one (model selection) parameter, for instance the expected number of regions. Instances of multiple segmentations generated by our algorithm are shown in Fig. 5.7. These are produced by allowing the number of regions N to vary in order to achieve a coarse-to-fine segmentation.

Failure cases. We show a representative sample of failure cases in the bottom row of Fig. 5.8. The main sources of error come from the following: (1) Estimating the number of regions that would yield the "best" segmentation is a problem that

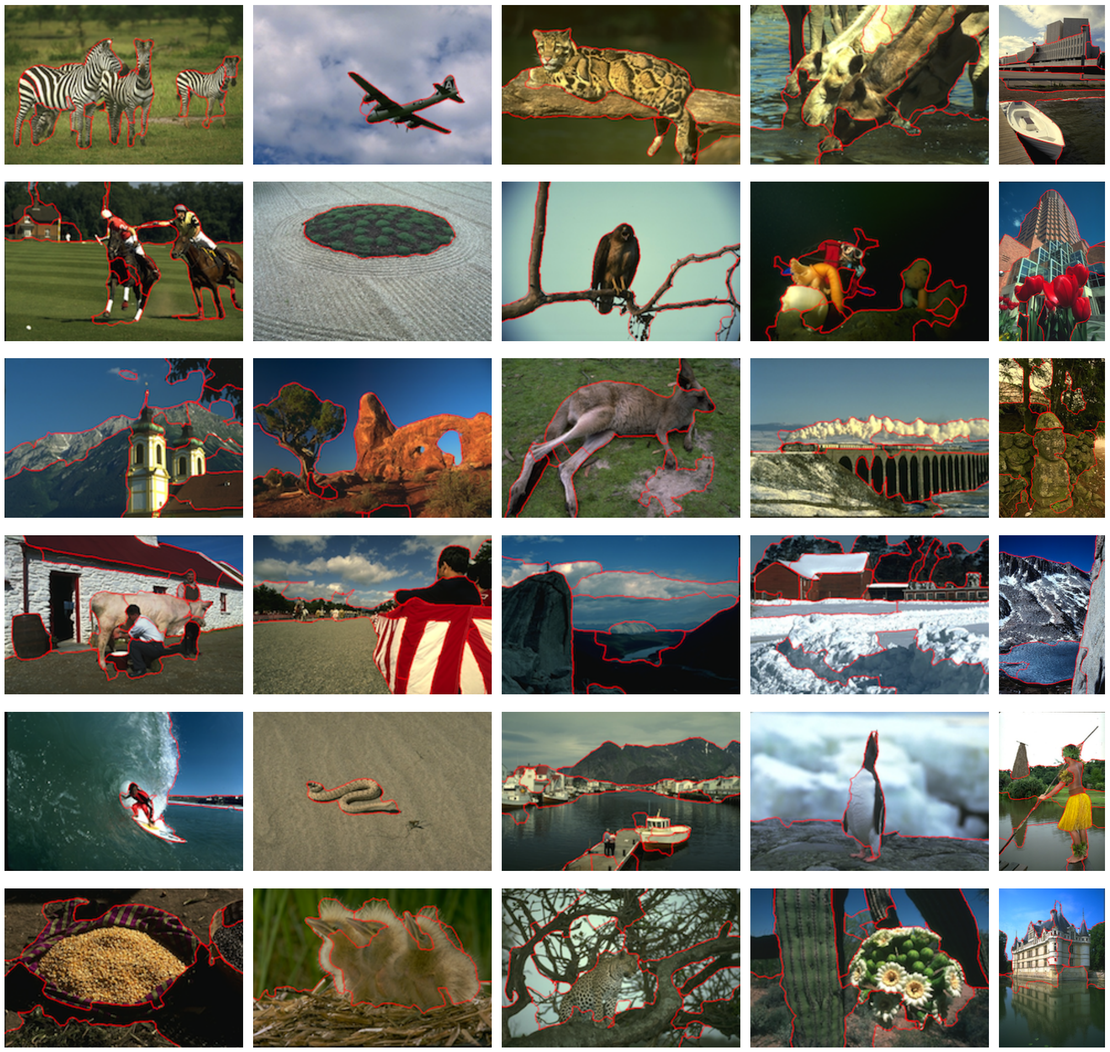
Figure 5.8: Segmentation results in the BSDS300 testing set. Bottom row: Failure cases. Detected boundaries are shown in red.

is not well defined. Therefore, a few of the final segmentations result with either too many or too few regions, (2) Some foregrounds have statistics that are "similar" to their backgrounds and since this is a low-level algorithm, if the visual properties of the two regions are similar, they cannot be discriminated. A possible way to improve the algorithm is to incorporate high-level semantics.

5.6 Discussion

Partitioning an image into multiple regions that exhibit local stationarity is a computationally complex problem. We have tackled this by breaking the problem down into a sequence of binary partitions of "foreground/local background" exploiting the robustness of region-based methods without being excessively penalized by their sensitivity to clutter. This is our first contribution. The second is in the handling of scale. Whereas most existing approaches start by feeding the image to a multi-scale filter bank, we let the local structure of the image determine what the scale of the local filters should be, that enables us to capture local statistics without excessive computational burden. Finally, we have shown how one can aggregate the multiple partitions into one segmentation, by computing pairwise affinities between superpixels in the foreground and background regions.

We have used standard benchmarks commonly found in the literature, where we have shown that, we outperform other schemes in two out of three metrics. We have provided numerous examples of success and failure.

CHAPTER 6

Encoding Scene Structures for Video Compression

The growth of video consumption over Internet and wireless [CIS11a] has recently rekindled interest in video compression, specifically towards breaking the compression ceiling of existing algorithms that encode video as a stream of blocks of pixels. 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, it has been noted that, under suitable assumptions, *"structures"* (regions of images that can be associated to a local reference frame) correspond to photometric or geometric characteristics of the *scene* if correspondence can be established across different images of the same scene [Soa10]. This has been exploited in previous chapters to show that the domain of each image can be partitioned into two disjoint (multiply-connected) sets: Structures and "textures," with the latter corresponding to spatially stationary regions. In this paper we propose a method to improve the spatial localization of such a partition.

In Chapter 3 we showed how Visual Structures and Visual Textures can be used to encode efficiently a video. The approach was based on partitioning the video frames into textures and structures and then letting the texture synthesis algorithm handle the boundary conditions. An alternative to that approach would be to explicitly specify the texture and structure regions using a segmentation map. The boundary conditions are handled explicitly not by the texture synthesis algorithm, but by matching boundary gradients to produce a gradient field that approximates properties of natural images [TJP10].

6.1 Related work

Our work is naturally related to video compression schemes as reflected in standards such as H.262, H.263, H.264 [ITU]. 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. 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 [ZXZ11] 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 chapter we propose a sequential approach whereby we first detect regions of the image that contain *structures* and are *properly sampled* (Section 2.1). We then exploit the complementarity of *structures* and *textures* Section 3.3.3 to encode the remainder of the image by performing texture segmentation (Section 6.3). We then infer a compressed representation of the textures (Chapter 4) that enables us to synthesize a perceptually similar realization at decoding. 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 6.4).

6.2 Texture / Structure partition

In this section we describe how to exploit temporal redundancy when possible (structures) and how to exploit spatial redundancy otherwise (textures). For completeness, we summarize previously described notions and provide a number of extensions.

6.2.1 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 [LS11], that requires a detection threshold. The effects of such a threshold are visible in Figure 2.1, 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 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. Note that this representation was previously introduced in Section 2.1.1.

6.2.2 Spatial redundancy

We exploit spatial redundancy by encoding stationary regions through their sample statistics. Following the texture representation introduced in Chapter 4, 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. In order to perform an empirical test, we need a larger region $\bar{\omega} \supset \omega$ where to aggregate the statistics. The region Ω is the *largest* admissible region where the stationarity assumption is satisfied. For a complete treatment of the properties and characterizations of textures, refer to Chapter 4.

6.2.3 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 (4.7). 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. 5).

Algorithm 5: Compression algorithm.

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

6.2.4 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 $\bar{\omega}$. We perform texture extrapolation using Alg. 1.

6.3 Segmentation

Given an image I we want to find Ω_i for the different textures in the image. The algorithm (Alg. 6) requires knowledge of the number of regions to be segmented (model selection) that can be determined using Agglomerative Information Bottleneck (AIB) [ST99]. 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 6.4.

Algorithm 6: Segmentation algorithm.

```
Initialize $N = 12, M = 30, K$
Sample (overlapping) square patches $\omega$ of sides $N$ on a dense grid from the
image
Let $\{\omega\}$ be the set of sampled $\omega$'s from previous step
foreach $\omega$ do
    Calculate a histogram of intensity values with $M$ bins for each patch
Calculate an empirical probability $p(M|\{\omega\})$
Use AIB [ST99] to sequentially merge $\omega$'s that that decrease $I(\{\omega\}, M)$ the
least
Cut the tree built by AIB in $K$ clusters to obtain K-clustering of the $\omega$'s
foreach pixel $x_i$ do
    Calculate in which $\omega$'s, $x_i$ falls in
    Find which cluster $K$, $x_i$ belongs to, using a max vote of the
    memberships of the $\omega$'s it falls in
Form regions $\Omega_i$ 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. 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. 2.1. 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.

6.4 Evaluation

Once we perform co-variant detection and proper sampling we encode the trackable regions by their corresponding dictionary element as discussed in Section 6.2.1. A typical result is shown in the first image in Figure 6.1. The rest of the domain of the frame is a candidate for texture, following the earlier discussion. We perform texture segmentation using Algorithm 6 and we obtain the second image of Figure 6.1. For AIB we use the implementation from [VF08]. 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

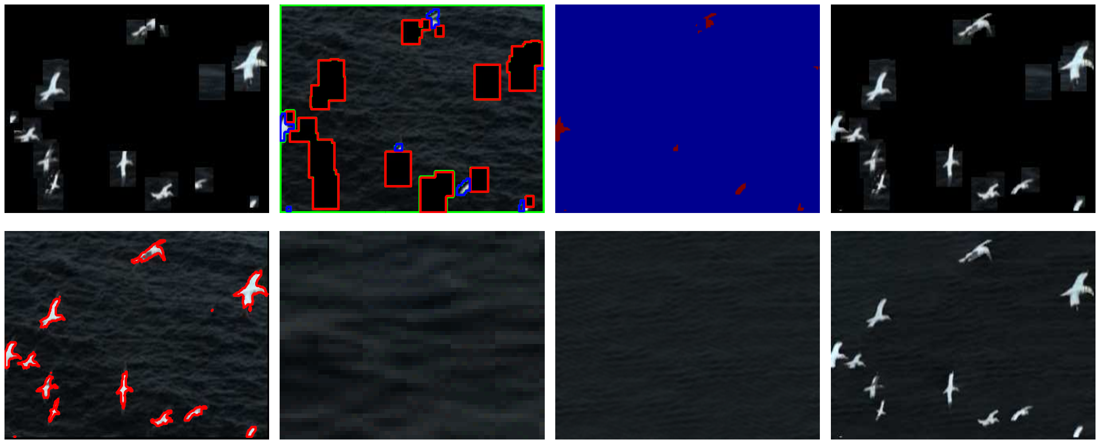

Figure 6.1: Pipeline. From left to right, top to bottom: (1) Tracked regions map. (2) Initial segmentation (red indicates boundaries of tracked regions, blue and green indicate boundaries of texture region candidates). (3) Regions of the image that did not satisfy the stationarity assumption underlying 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.

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 6.2.4 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 6.1 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.

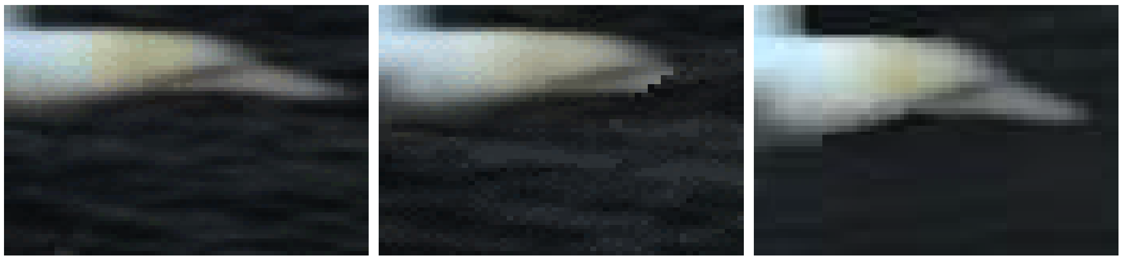

Figure 6.2: Reconstruction of structured regions. Left: Input frame, Center: Video Primal Sketch, Right: Our method. Our method successfully preserves salient regions.

The collection of trackable regions is then updated, and the final partition is shown in the fourth panel in Figure 6.1. We then reiterate the texture segmentation routine to obtain the different textured regions in the frame. We compress each texture region using Algorithm 5. A typical result is shown in the sixth panel of Figure 6.1. 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 6 (seventh image of Figure 6.1) and overlay the trackable regions using the representation stored in the dictionary. Representative results of this procedure are shown in the central column of Figure 6.3. 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 6.1. 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 [TJP10]. We use the algorithm proposed by [TJP10] and typical results are shown in the right column of Figure 6.3.

We also compare our method with [ZXZ11] in Figure 6.2. It can be seen that their method misses important structures that are detected by our approach.

6.5 Discussion

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 previous chapters by considering a tight partition respecting object boundaries and the statistics of natural images.

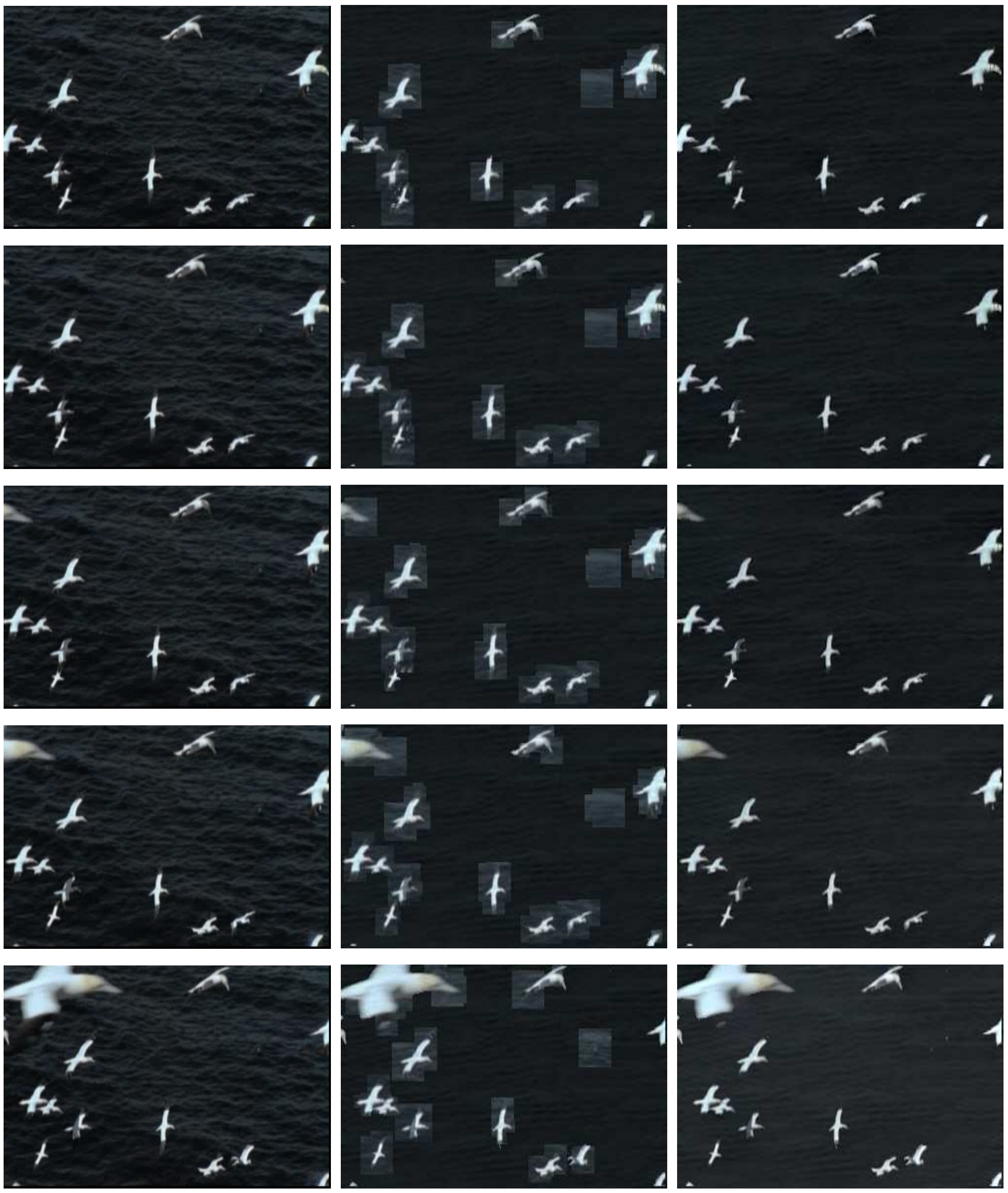

Figure 6.3: Samples from the "bird-sea" sequence. Left: Input sequence, Center: Overlaid tracked regions and synthesized textures. Right: Natural blending of synthesized textures and tracked regions.

CHAPTER 7

Scene-Aware Video Modeling and Compression

Video exhibits considerable redundancy in both space and time, and generic priors such as sparsity and regularity have been exploited with some success, but diminishing return. 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 [Soa10]. In this paper we focus on human fruition, like classical video compression, where performance is ideally computed with some perceptual score [TH94], but in practice with the (perceptually irrelevant) RMS or PSNR scores.

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.

[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 [CIS11b].

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. 7.3.1. 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. 7.3.2. This relates also to the notion of "texture" and addresses spatial redundancy.

In Sect. 7.4 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. 7.5 we point at limitations and challenges in our approach.

7.1 Related work

At the level of generality adopted thus far, this work relates to the entirety of computer vision. However, our goal is not to infer geometric, photometric, or dynamic properties of the scene from video. Instead, we aim to infer just enough of the phenomenology of the scene to be useful for compression, specifically occlusion and scaling. Even so, relevant prior work goes beyond what we can cover in conference submission. We refer the reader to representative work on *optical flow* [HS81, BBP04, LYT08], *occlusion detection* [ARS11, BA96], *tracking* e.g. KLT and particle filter [SBK10, TS94, ST06], *segmentation* e.g. gPb, mean shift [AMF11, CM02, FH04], *texture analysis and modeling* [EF01, LLX01, KEB05], including texture/structure partitioning [GZW03b, ZXZ11]. Finally, all video compression standards that offer end-to-end solutions are related to our work e.g. H.262, H.263, H.264 [ITU].

7.2 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 [MSK03], which can be shown to admit as closure the entire group of planar diffeomorphisms [SPV09b], 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:

$$I(x,t) = \kappa(t) \circ \rho \circ w(x,t) + n(x,t) \tag{7.1}$$

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 (7.1) 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.

7.3 Encoder

7.3.1 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 (7.1) 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. [ARS11] 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

$$\begin{aligned} \hat{w}(x,t), \hat{\Omega}(t) &= \arg\min_{w,\Omega} \|e_2\|_{\ell^2(D)} + \lambda \|e_1\|_{\ell^1(D\backslash\Omega)} \qquad (7.2) \\ \text{s. } t. \quad I(x,t) &= I(w(x,t), t-1) + e_1(x,t) + e_2(x,t) \end{aligned}$$

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 7.1, 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 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

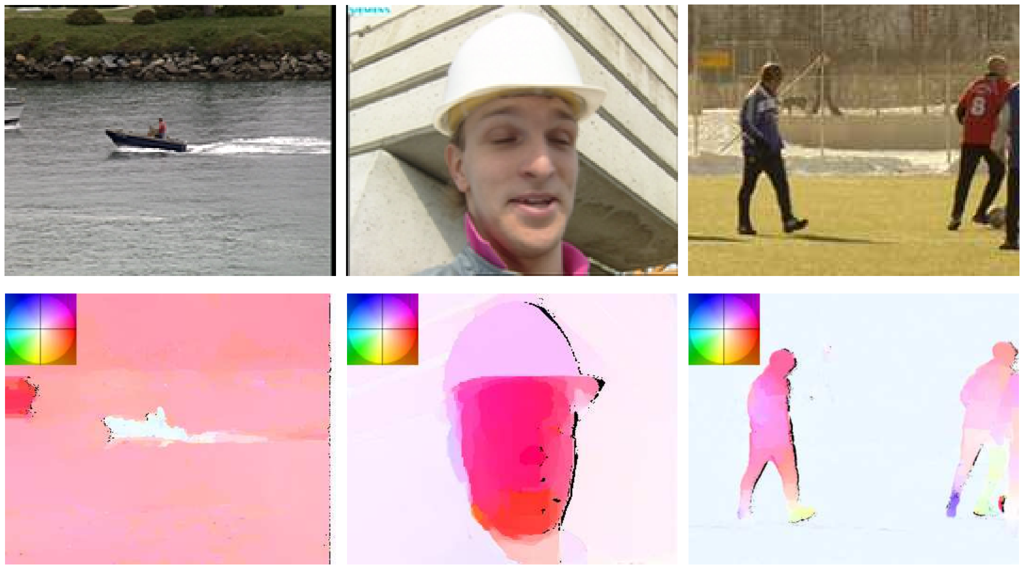

Figure 7.1: (COLOR) Occlusion detection and optical flow: Original images (top) and optical flow (bottom), visualized according to the color scheme shown in the top left corner of each image. Black regions are occluded, so no motion estimate is available since there is no region in image $I(x, t-1)$ that, transported with $w(x,t)$ yields $I(x,t)$.

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 [AMF11] for contour detection and segmentation. In Figure 7.2 we show representative samples of this process.

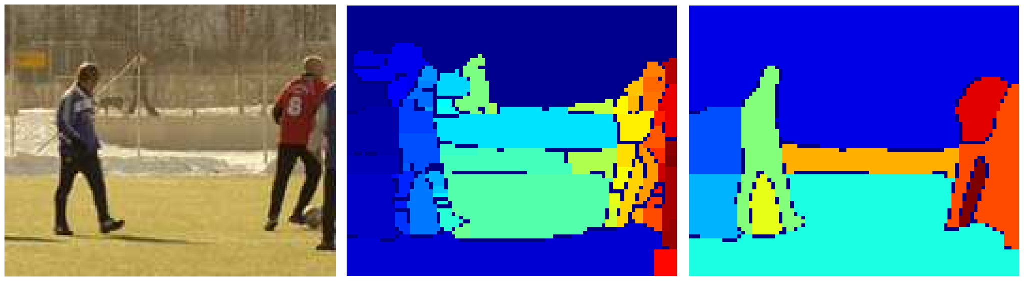
Figure 7.2: Segmentation [AMF11]. Varying a threshold yields structurally different segmentations. Original test frame (left). Segmentation using a low threshold (center). Segmentation with a higher threshold (right).

7.3.2 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* [Soa10]: Co-variant detector functionals [LS11] 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*,

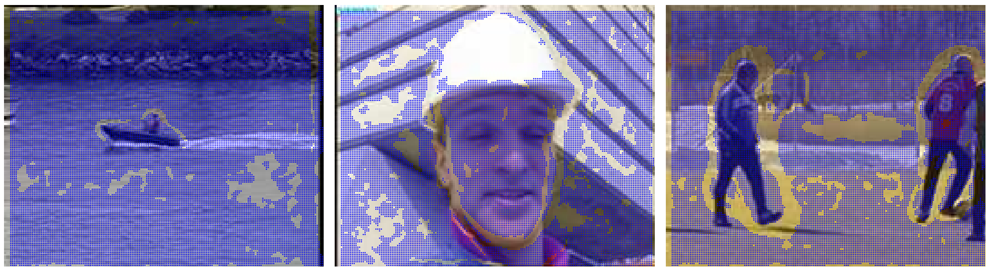

Figure 7.3: Covering of trackable regions [SBK10]. The rest is encoded by the texture module.

for instance $I(x,t), I(w(x,t), t-1)$. The portions of the regions that are properly sampled are then "trackable" [LS11], 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. 7.3.5. Additional processing can be performed to aggregate tracks over multiple views [SBK10]. In Figure 7.3 covering trajectories on sequences are shown.

7.3.3 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 regions and their description [LS11], 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 [WGZ08].

As discussed in Section 7.3.1, 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 7.4.

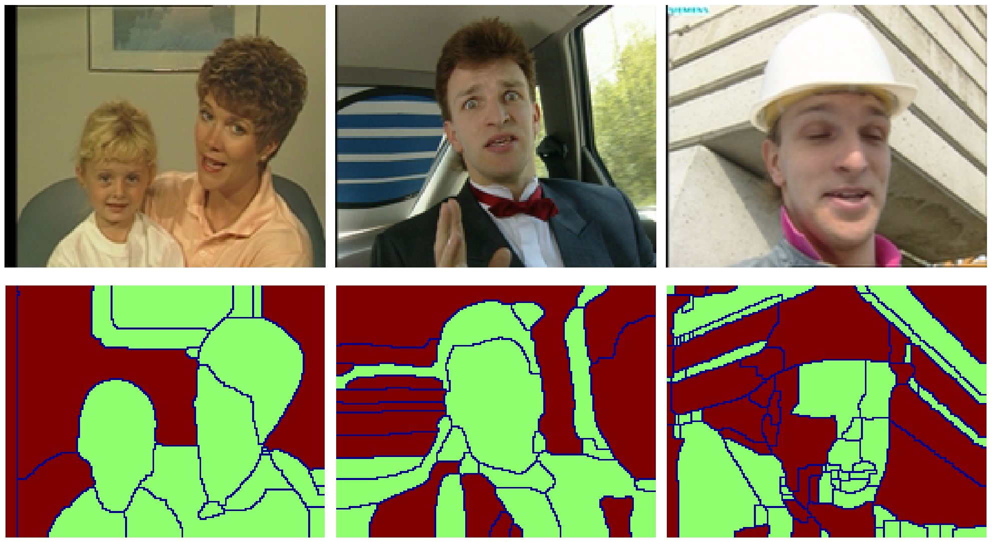

Figure 7.4: Original Test Sequences (top). Texture/Structure partition (bottom). Regions in green are labeled trackable regions and we can assign a motion vector to them. Textured regions are either homogenous regions or stochastic textures and it is preferable to spatially predict them since they cannot be tracked.

7.3.4 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.

7.3.5 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. 7.3.4. 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 [KEB05] 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. Note that the algorithm used is also a variant of Alg. 1 with an added term. This non-parametric sampling-based approach takes a subset of a texture region and expands it. This is accomplished by minimizing the following energy:

$$E_t(I_{|_B}; \{I_{|_{N(B)}}\}) = \sum_{p \in N(B)} \|I_p - \hat{I}_p\|^2 + \mu \sum_{p \in N(B)} \|DI_p - D\hat{I}_p\|^2 \tag{7.3}$$

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)). 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 7.5) 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

$$\hat{\alpha} = \arg\min_{\alpha_p} \|I_p - Y\alpha_p\|_2^2 + \lambda||\alpha_p||_1 \tag{7.4}$$

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 7.5. 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.

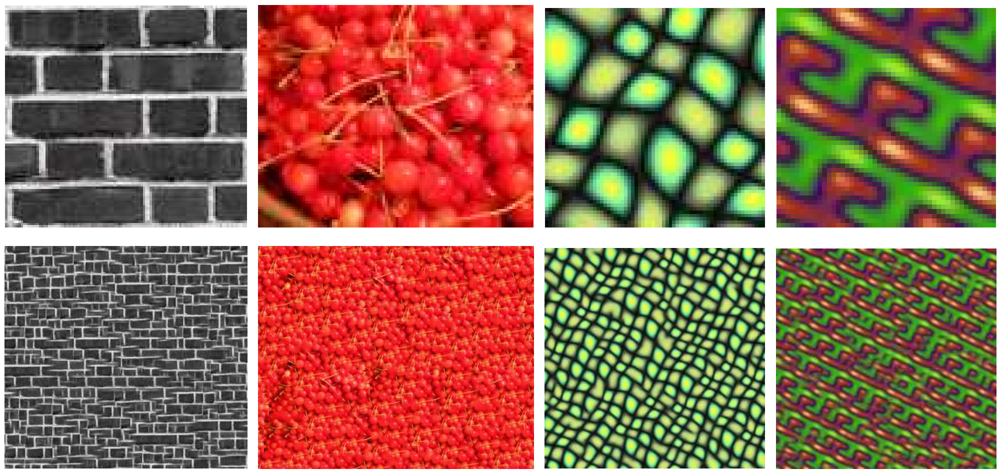
Figure 7.5: Low resolution image patches (top). Image Patches of higher resolution synthesized with our algorithm (bottom).

7.3.6 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 [ITU], 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.

7.4 Experiments

To evaluate our encoder we compared its performance with four different encoding methods. In the baseline method used (blue curve in Figure 7.7), 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 7.6 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 7.7 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.

[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/

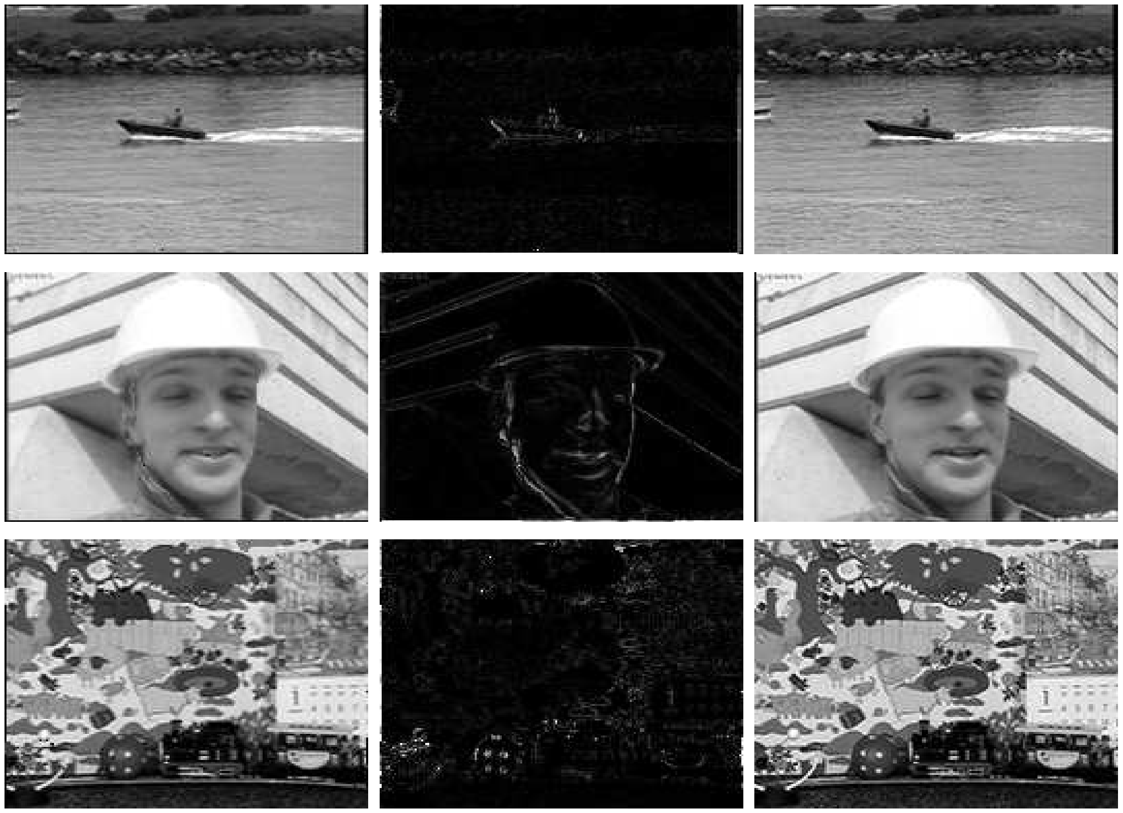

Figure 7.6: Qualitative results of our encoder. Predicted image using spatial, temporal and texture synthesis prediction modes (left). Residual error: black corresponds to zero error, white to large (center). Reconstructed video frames (right).

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.

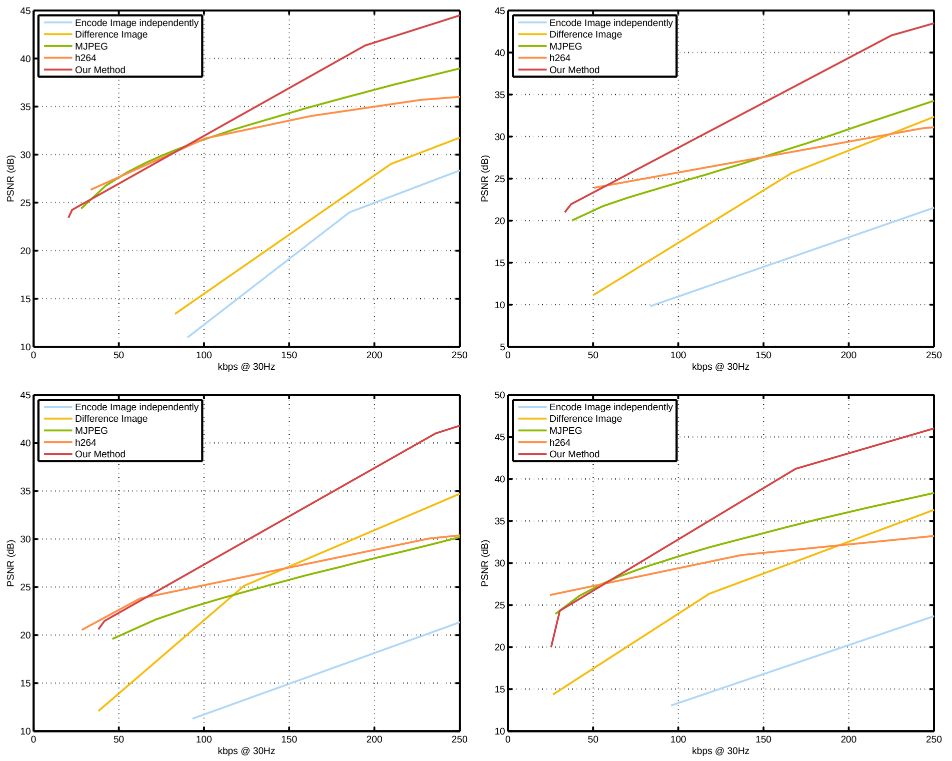

Figure 7.7: Quantitative comparison of our encoder with other methods. Our method significantly outperforms all other methods (including H.264) in a wide range of kbps in terms of PSNR. Results shown are for the sequences city (upper left), flower garden (upper right), mobile (bottom left) and coastguard (bottom right). Comparisons were ran for a duration of 50 frames.

7.5 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 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.

CHAPTER 8

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. 8.1), 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 [IB09] 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 [ISB99] 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" [SJK09] have improved efficiency compared to multi-rigid motion estimation, that was using algebraic geometric methods [Vid03] 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 [Soa09], that draws on ideas from Gibson's Ecological Approach to Visual Perception [Gib84], 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 *discovered* after a finite time interval represents

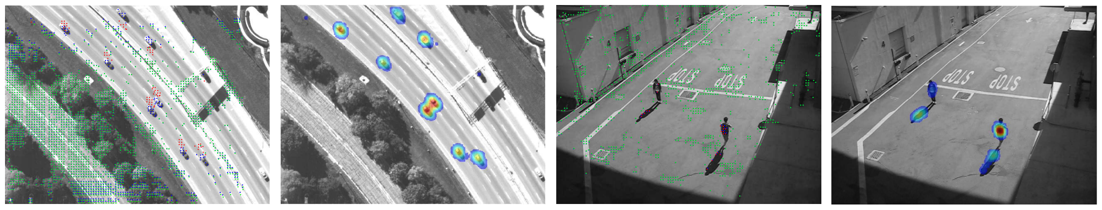

Figure 8.1: Detecting salient regions under camera motion: (1st, 3rd): Tracked feature points (blue) are classified as inliers (green) or outliers (red). (2nd, 4th): Estimated salient point density obtained by our algorithm.

the *"Actionable Information Increment"* provided by the "next image" [Soa09]. 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. 8.2 we propose an alternative scheme for detecting salient regions in videos.

8.1 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) [ARS12].

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:

$$H0 = \{\exists \text{ a } diffeo\ w : \Omega \to D \mid I(x, t + dt) - I(x + w(x), t) \overset{IID}{\sim} \mathcal{N}\} \tag{8.1}$$

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. 8.2,8.3).

8.2 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 (8.1) in the most efficient way, so that it would depend on the smallest possible number of free parameters. As we show in Sect. 8.3, 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:

$$\begin{aligned} H1 = &\{\exists\ V \in \mathbb{S}^2, \omega \in \mathbb{R}^3,\ Z : D \to \mathbb{R}^+ \mid \\ & w(x) = \pi(\widehat{\omega}\bar{x}Z(x) + V)\} \end{aligned} \tag{8.2}$$

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 [1]. 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

[1]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 $so(3) \doteq \{S | S^T = -S\}$.

up to a white residual. The two hypotheses can be tested simultaneously by substituting the expression of w in (8.2) into (8.1). 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 the literature on "dominant motion estimation" [IRP92] and robust motion estimation [SL03], and relates to robust statistics [Hub81] and outlier rejection in motion estimation [FB81].

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 [Vid03]. 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. 8.3.

8.3 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 (8.2). 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 [SFP94]:

$$y(x) = \mathcal{A}(x)\frac{V}{Z(x)} + \mathcal{B}(x)\omega \tag{8.3}$$

where:

$$\mathcal{A}(x) \doteq \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix} \tag{8.4}$$

$$\mathcal{B}(x) \doteq \begin{bmatrix} -x_1x_2 & 1+x_1^2 & -x_2 \\ -1-x_2^2 & x_1x_2 & x_1 \end{bmatrix} \tag{8.5}$$

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:

$$\hat{Z}, \hat{V}, \hat{\omega} = \arg\min \int_D \|y(x) - \mathcal{A}(x)\frac{V}{Z(x)} - \mathcal{B}(x)\omega\|_{\mathcal{H}} dx \tag{8.6}$$

where $\|\cdot\|_{\mathcal{H}}$ denotes a robust norm, for instance a Huber norm [Hub81]. After this is done, one would find the salient regions that violate this model, that is:

$$\Omega \doteq \{x \in D \mid \|y(x) - \mathcal{A}(x)\frac{\hat{V}}{\hat{Z}(x)} - \mathcal{B}(x)\hat{\omega}\| > \epsilon\} \tag{8.7}$$

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 (8.6) is solved in the continuum, regularization on Z has to be imposed (this is not necessary when (8.6) 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 [HJ92], 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 (8.6). This can be done easily since the model (8.3) 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 [CBS00] 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. 8.4) by rewriting the model (8.3) 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.

8.4 Computation of the optimal discriminant

Following the discussion in Sect. 8.3, salient points $x \in \Omega$ will be detected as a violation of the hypothesis provided by the model (8.3). 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, $x_i,\ i = 1, \ldots, N$ 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 (8.3). Writing (8.3) as a system of linear equations for all points and introducing $\mathcal{W}$ as a weight matrix we have:

$$\mathcal{W}Y(X) = \mathcal{W}C(V) \left[\begin{array}{cccc} p(x_1) & \ldots & p(x_N) & \omega \end{array} \right]^T \tag{8.8}$$

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

$$C(V) \doteq \begin{bmatrix} A(x_1)V & & & B(x_1) \\ & \ddots & & \vdots \\ & & A(x_N)V & B(x_N) \end{bmatrix} \tag{8.9}$$

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

$$\begin{aligned} [P, \omega]^T &= (\mathcal{W}C(V))^{\dagger}\mathcal{W}Y(X) \\ &\doteq (C(V)^T\mathcal{W}^T\mathcal{W}C(V))^{-1}C(V)^T\mathcal{W}^T\mathcal{W}Y(X) \end{aligned} \tag{8.10}$$

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:

$$\hat{C}(V)Y \doteq \left(I - \mathcal{W}C(\mathcal{W}C)^{\dagger}\right)\mathcal{W}Y = 0 \tag{8.11}$$

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:

$$\underset{V}{\text{minimize}} \quad \psi(V) = \frac{1}{2}||\hat{C}(V)Y||_2^2 \tag{8.12}$$

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.

where **1** is the indicator function. Note that this is a generalization of the case

Algorithm 7: 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
        $w_{2i-1}^{(k+1)} = w_{2i}^{(k+1)} = \frac{1}{||[e_{2i-1}^{(k)}, e_{2i}^{(k)}]||_2 + \varepsilon}$
    $\mathcal{W}^{(k+1)} = diag(w_1^{(k+1)}, \ldots, w_{2N}^{(k+1)})$
    $V^{(k+1)} := \hat{V}^{(k)} ash\|\hat{V}^{(k)}\|$
```

proposed by [HJ92]. The authors of [HJ92] minimized (8.12) 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 (8.12) are provided in Appendix 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 $i = 1, \ldots, N$. 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. 8.6.

8.5 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 (8.3) and rewrite it in homogeneous coordinates:

$$\begin{aligned}\begin{bmatrix} y \\ 0 \end{bmatrix} &= K \begin{bmatrix} \mathcal{A}V & \mathcal{B} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/Z \\ \omega \end{bmatrix} \\ &= \begin{bmatrix} fs_x & fs_\theta & O_x \\ 0 & fs_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{A}\frac{V}{Z} + \mathcal{B}\omega \\ 0 \end{bmatrix} \end{aligned} \tag{8.13}$$

From (8.13) it is obvious that $y(x)$ is independent of (O_x, O_y). On the other hand, also obvious from (8.13), 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 (8.13), similarly as in Sect. 8.4, we get:

$$\begin{aligned}\mathcal{W}Y(X) &= f\mathcal{W}\mathcal{K}C(V)[P,\omega]^T \\ &\doteq f\mathcal{D}[P,\omega]^T \end{aligned} \tag{8.14}$$

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:

$$\begin{aligned}[P,\omega]^T &= (f\mathcal{D})^\dagger \mathcal{W}Y(X) = (f^2\mathcal{D}^T\mathcal{D})^{-1} f\mathcal{D}^T\mathcal{W}Y(X) \\ \mathcal{W}Y(X) &= f\mathcal{D}(f^2\mathcal{D}^T\mathcal{D})^{-1} f\mathcal{D}^T\mathcal{W}Y(X) \\ &= \mathcal{D}(\mathcal{D}^T\mathcal{D})^{-1}\mathcal{D}^T\mathcal{W}Y(X) \end{aligned} \tag{8.15}$$

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.

8.6 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. 8.4 and Cars-2/06 are from the Hopkins 155 motion segmentation dataset [TV07] 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. [SJK09] 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. 8.2) 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. 8.3) were taken from an airliner approaching Boston Logan airport. People-3 (second row in Fig. 8.1) 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 [SBK10] that yields dense point trajectories. We used the Harris corner detector [HS88] 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

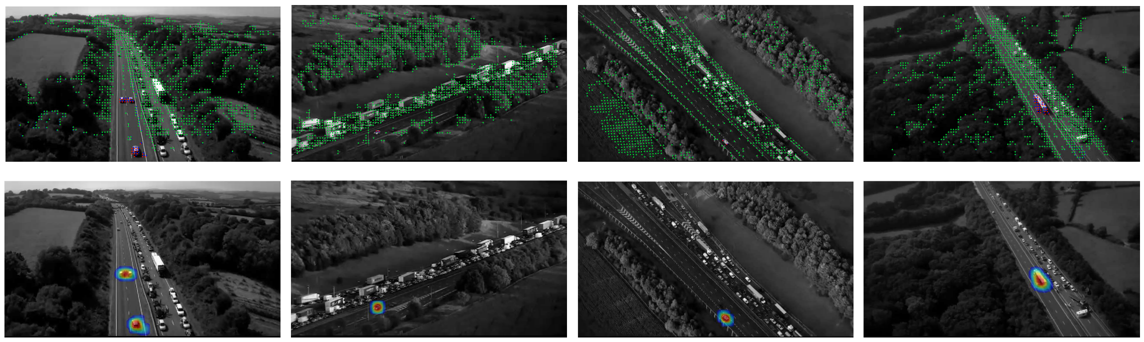
Figure 8.2: Four aerial views of a motorway. In all images cars in the right lane are stationary and cars in the left lane are moving. The true outliers in these cases are the moving cars in the left lane. The first row shows the dense tracked points in each image. The second row shows in color the detected outliers. Color convention is the same as in Fig. 8.4.

comparison with [SJK09]. 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 7 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. 8.1, 8.4, 8.2 and 8.3 we show the tracked regions and the salient regions as classified by our method. In Table 8.1, we compare the performance of our algorithm to three other methods using the F-measure: (i) RANSAC [FB81] 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 [HJ92] but minimized it with gradient descent rather than exhaustive

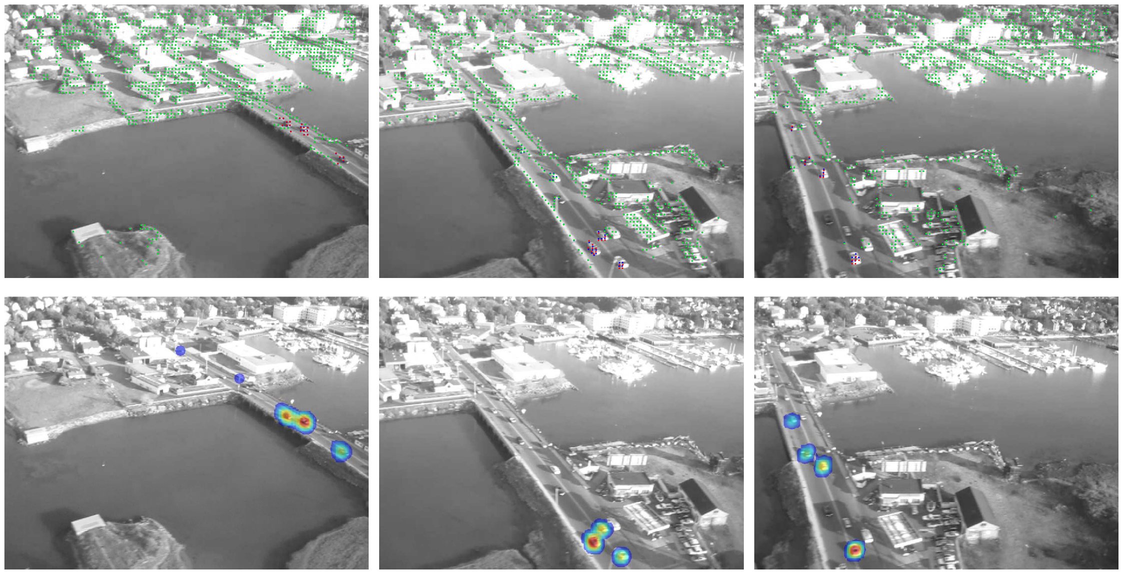
Figure 8.3: Three aerial views of a bridge taken from an airliner during a turn approaching Boston's airport. The first row shows the images with the tracked points and the second highlights the salient regions. The color codes are the same as in Fig. 8.4.

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. [SJK09] 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 8.1. Although [SJK09] 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 chooses the threshold value whereas for

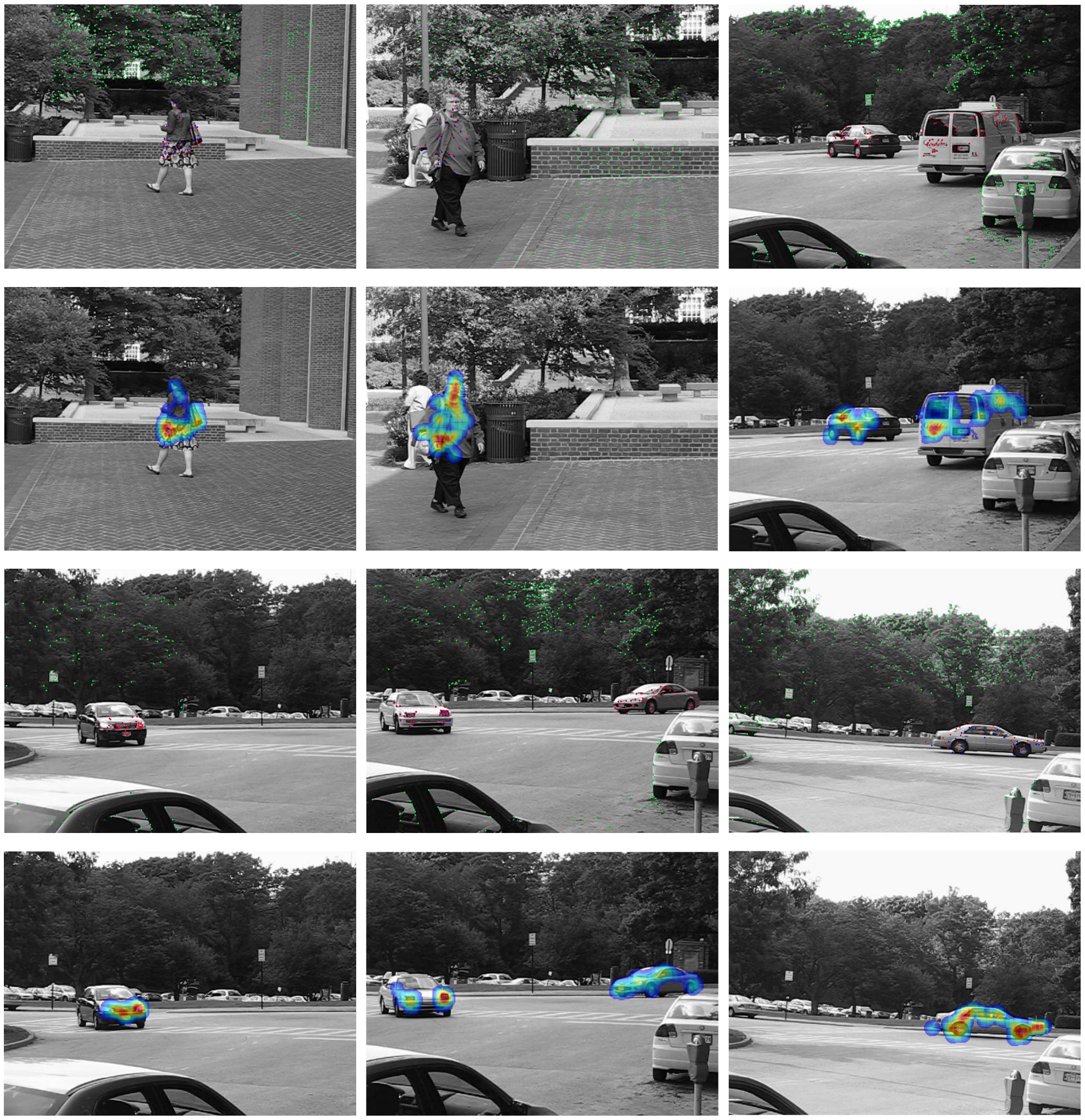

Figure 8.4: Sample results from the Hopkins 155 dataset: Odd rows: Images with tracked points. Red and green points show the locations of tracked points as predicted by the model. Points in green are the points that are classified as inliers and in red those that are classified as outliers. Blue dots (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.

RANSAC and [SJK09] 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 [SJK09] was 2.6 seconds. Experiments 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. 8.3. 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.

	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 [HJ92]	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. [SJK09]	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 [HJ92]	**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. [SJK09]	-	-	-	-	-	-	-

Table 8.1: Comparison on salient point detection performance of our algorithm against [HJ92], RANSAC under epipolar constraint and [SJK09] 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 [SBK10]. Our algorithm significantly outperforms all other 3 methods in almost all sequences.

8.7 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 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.

CHAPTER 9

Discussion

Video analysis has become a fundamental research problem following the rise of video consumption over multiple mediums. Users are more than ever interested in interacting with their medium and video consumption is becoming increasingly an active action. As a result, providing users with side information that stems out from video content is becoming a huge challenge for automatic mechanisms. Video analysis is required to perform scene/video understanding, infer semantics and encode such information in scene representations that could be utilized in a series of applications.

In this work, scene representations for video analysis were discussed. In particular, the goal was to design and infer representations that capture properties of the scene that could benefit video compression. Video frames were viewed as projections of an underlying scene. The projections were assumed to induce two types of regions, "structures" and "textures".

Representations for structures were designed and inferred when multiple structure regions are present in a frame. The first video coding system presented exploited only the temporal redundancy of visual structures; the remaining regions were left un-modeled and sent to a background layer. The approach was used as a wrapper around HEVC/H.264 and it was shown that it can outperform both in terms of a rate-distortion comparison.

Textures were introduced as the complementary region type to structures, which exhibit spatial regularity. A representation that exploits the spatial redundancy was designed that required considerably less space than the input texture. The representation is useful for texture synthesis in images and video and for applications such as video hole-filling. Hence, an image and video texture synthesis algorithm was proposed that is able to synthesize multiple textures simultaneously without the need of segmentation. A criterion that measures statistical and structural similarity was also proposed that can be used to measure similarity of two textures. Representations of structures and textures were then utilized simultaneously to form a complete video compression system. The texture representation was also extended to capture transformations in both the range and the domain of the texture. Doing so, can provide additional benefits in terms of compression. Further criterions that measure similarity of textures were also proposed.

In addition to the aforementioned applications, texture segmentation was also explored. By breaking the segmentation problem into a sequence of binary partitions and then uniting the region proposals using pairwise affinities, it was possible to segment textures in natural images in the presence of clutter and scale variability. Texture segmentation was utilized in a video encoding scheme to produce an extension to previous chapters, by considering a tight partition of the video frames respecting object boundaries and the statistics of natural images.

Furthermore, a video compression system that exploits additional properties of the scene was presented. In particular, occlusion and scale were discussed, structures/textures were modeled and an explicit prediction mode for homogeneous regions was introduced. Segmentation was utilized as part of the structure/texture partition inference and optical flow was introduced to aid the computation of motion vectors of the regions.

Finally, an additional application of video analysis was demonstrated, that of independent motion detection of objects moving over a homogeneous background. Under these conditions, the objects do not violate the co-visibility hypothesis, hence do no generate occlusions. Instead since these regions correspond to ones that violate a single rigid motion, they were detected by determining the outliers of the camera motion model. Regions that were determined as outliers were classified as “salient” regions, while the rest were classified as background.

While the focus of the thesis was on video compression, the representations presented have multiple utilities ranging from video/image hole-filling (video/image editing), texture synthesis (computer graphics), texture segmentation (computer vision, video compression) and independent motion detection (surveillance, monitoring, etc). While low-level approaches have driven research work in video compression, it is believed that further breakthroughs would come by instead modeling higher-level entities, such as the underlying scene. The latter approach can also provide information about the scene, useful for other applications that include video classification and retrieval.

APPENDIX A

Actionable Saliency Detection: Derivation of the Discrimant

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:

$$\frac{\partial\psi(V)}{\partial V} = \frac{\partial\psi(\hat{C})}{\partial \text{vec}(\hat{C})}\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)}\frac{\partial \text{vec}(\mathcal{W}C)}{\partial \text{vec}(C)}\frac{\partial \text{vec}(C(V))}{\partial V} \tag{A.1}$$

The four terms are given by the following equations:

$$\frac{\partial\psi(\hat{C})}{\partial \text{vec}(\hat{C})} = (\mathcal{W}x)^T \otimes (\mathcal{W}x)^T\hat{C} \tag{A.2}$$

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:

$$\begin{aligned}
\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)} &= \frac{\partial}{\partial \text{vec}(\mathcal{W}C)}\left(I - \mathcal{W}C(C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T\right) && \text{(A.3)}\\
&= -\frac{\partial}{\partial \text{vec}(\mathcal{W}C)}\mathcal{W}C(C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T && \text{(A.4)}\\
&= -\left(g_1(\mathcal{W}C)^T \otimes I_{2N}\right)f_1'(\mathcal{W}C) && \text{(A.5)}\\
&- \left(I_{2N} \otimes \mathcal{W}C\right)g_1'(\mathcal{W}C) && \text{(A.6)}
\end{aligned}$$

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 $f_2(\mathcal{W}C) =$

$\big((\mathcal{W}C)^T\mathcal{W}C\big)^{-1}$ 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:

$$g_1'(\mathcal{W}C) = (g_2(\mathcal{W}C)^T \otimes I_{N+3})f_2'(\mathcal{W}C) + (I_{2N} \otimes f_2(\mathcal{W}C))g_2'(\mathcal{W}C) \tag{A.7}$$

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

$$T_{N,M}\text{vec}(A) = \text{vec}(A^T) \tag{A.8}$$

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:

$$f_2'(\mathcal{W}C) = f_3'((\mathcal{W}C)^T\mathcal{W}C)g_3'(\mathcal{W}C) \tag{A.9}$$

From standard results:

$$\begin{aligned} f_3'((\mathcal{W}C)^T\mathcal{W}C) &= -(((\mathcal{W}C)^T\mathcal{W}C)^{-T} \otimes ((\mathcal{W}C)^T\mathcal{W}C)^{-1}) && \text{(A.10)} \\ g_3'(\mathcal{W}C) &= \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && \text{(A.11)} \end{aligned}$$

Therefore:

$$\begin{aligned}
f_2'(\mathcal{W}C) &= f_3'((\mathcal{W}C)^T\mathcal{W}C)g_3'(\mathcal{W}C) && \text{(A.12)}\\
&= -(((\mathcal{W}C)^T\mathcal{W}C)^{-T} \otimes ((\mathcal{W}C)^T\mathcal{W}C)^{-1}) && \text{(A.13)}\\
&\quad \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && \text{(A.14)}\\
g_1'(\mathcal{W}C) &= (g_2(\mathcal{W}C)^T \otimes I_{N+3})f_2'(\mathcal{W}C) && \text{(A.15)}\\
&\quad +(I_{2N} \otimes f_2(\mathcal{W}C))g_2'(\mathcal{W}C) && \text{(A.16)}\\
&= -\left(\mathcal{W}C \otimes I_{N+3}\right) && \text{(A.17)}\\
&\quad \times \left(\left((\mathcal{W}C)^T\mathcal{W}C\right)^{-T} \otimes \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}\right) && \text{(A.18)}\\
&\quad \times \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && \text{(A.19)}\\
&\quad + \left(I_{2N} \otimes \left((\mathcal{W}C)^T(\mathcal{W}C)\right)^{-1}\right) T_{2N,N+3} && \text{(A.20)}\\
\frac{\partial \text{vec}(\hat{C}(\mathcal{W}C))}{\partial \text{vec}(\mathcal{W}C)} &= -\left(g_1(\mathcal{W}C)^T \otimes I_{2N}\right) f_1'(\mathcal{W}C) && \text{(A.21)}\\
&\quad -\left(I_{2N} \otimes \mathcal{W}C\right) g_1'(\mathcal{W}C) && \text{(A.22)}\\
&= -\left[\left(((C^T\mathcal{W}^T\mathcal{W}C)^{-1}C^T\mathcal{W}^T)^T \otimes I_{2N}\right) I_{2N(N+3)}\right] && \text{(A.23)}\\
&\quad +\left(I_{2N} \otimes \mathcal{W}C\right) && \text{(A.24)}\\
&\quad \times \left(\mathcal{W}C \otimes I_{N+3}\right) && \text{(A.25)}\\
&\quad \times \left(\left((\mathcal{W}C)^T\mathcal{W}C\right)^{-T} \otimes \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}\right) && \text{(A.26)}\\
&\quad \times \left(I_{(N+3)^2} + T_{(N+3),(N+3)}\right)\left(I_{N+3} \otimes (\mathcal{W}C)^T\right) && \text{(A.27)}\\
&\quad + \left(I_{2N} \otimes \left((\mathcal{W}C)^T\mathcal{W}C\right)^{-1}\right) T_{2N,(N+3)} && \text{(A.28)}\\
& && \text{(A.29)}
\end{aligned}$$

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

$$\frac{\partial \text{vec}(\mathcal{W}C)}{\partial \text{vec}(C)} = I_{N+3} \otimes \mathcal{W} \quad \text{(A.30)}$$

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

$$\frac{\partial \text{vec}(C(V))}{\partial V} = \begin{bmatrix} A_1 \\ O_{2N,3} \\ A_2 \\ O_{2N,3} \\ \vdots \\ A_N \\ O_{6N,3} \end{bmatrix} \tag{A.31}$$

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.

References

[AMF11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. "Contour Detection and Hierarchical Image Segmentation." *TPAMI*, 2011.

[ARS11] A. Ayvaci, M. Raptis, and S. Soatto. "Sparse Occlusion Detection with Optical Flow." *IJCV*, 2011.

[ARS12] A. Ayvaci, M. Raptis, and S. Soatto. "Sparse Occlusion Detection with Optical Flow." *IJCV*, 2012.

[BA96] M. Black and P. Anandan. "The robust estimation of multiple motions: parametric and piecewise smooth flow fields." *Computer Vision and Image Understanding*, 1996.

[BAC96] Andrew C. Beers, Maneesh Agrawala, and Navin Chaddha. "Rendering from Compressed Textures." In *ACM SIGGRAPH*, 1996.

[BBP04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. "High accuracy optical flow estimation based on a theory for warping." *ECCV*, 2004.

[BEL01] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. "Texture mixing and texture movie synthesis using statistical learning." *TVCG*, 2001.

[BM10] T. Brox and J. Malik. "Object Segmentation by Long Term Analysis of Point Trajectories." *ECCV*, 2010.

[BMS97] P. S. Bradley, O. L. Mangasarian, and W. N. Street. "Clustering via Concave Minimization." *NIPS*, 1997.

[BNS10] S. Boltz, F. Nielsen, and S. Soatto. "Texture Regimes for Entropy-Based Multiscale Image Analysis." *ECCV*, 2010.

[Bou11] N. Bouguila. "Texture discrimination using local features and count data models." *CCCA*, 2011.

[BVS03] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. "Simultaneous structure and texture image inpainting." *TIP*, **12**(8):882–889, 2003.

[CBS00] A. Chiuso, R. Brockett, and S. Soatto. "Optimal structure from motion:local ambiguities and global estimates." *IJCV*, 2000.

[CBS05] T. Cour, F. Bénézit, and J. Shi. "Spectral Segmentation with Multiscale Graph Decomposition." *CVPR*, 2005.

[CE05] T.F. Chan and S. Esedoglu. “Aspects of Total Variation Regularized L1 Function Approximation.” *SIAP*, 2005.

[CFJ08] V. Cheung, B. J Frey, and N. Jojic. “Video epitomes.” *IJCV*, 2008.

[CIS11a] CISCO. “Entering the Zettabyte Era, Visual Networking Index.” CISCO VNI, 2011.

[CIS11b] CISCO. “Entering the Zettabyte Era, Visual Networking Index.” CISCO VNI, 2011.

[CIS14] CISCO. “Entering the Zettabyte Era, Visual Networking Index.” CISCO VNI, 2014.

[CM02] D. Comaniciu and P. Meer. “Mean Shift: A Robust Approach Toward Feature Space Analysis.” *TPAMI*, 2002.

[CMK14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. “Describing Textures in the Wild.” *CVPR*, 2014.

[CS10] J. Carreira and C. Sminchisescu. “Constrained Parametric Min-Cuts for Automatic Object Segmentation.” *CVPR*, 2010.

[CV01] T. F. Chan and L. A. Vese. “Active Contours without Edges.” *TIP*, 2001.

[CYS04] T. Cour, S. Yu, and J. Shi. “Normalized Cut Segmentation Code.” *University of Penn, Computer and Information Science Department*, 2004.

[DCW03] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. “Dynamic textures.” *IJCV*, 2003.

[DUH09] M. Donoser, M. Urschler, M. Hirzer, and H. Bischof. “Saliency Driven Total Variation Segmentation.” *ICCV*, 2009.

[EF01] Alexei A. Efros and William T. Freeman. “Image Quilting for Texture Synthesis and Transfer.” *Proc. of ACM SIGGRAPH*, 2001.

[EH10] Ian Endres and Derek Hoiem. “Category independent object proposals.” *ECCV*, 2010.

[EL99] A. Efros and T. Leung. “Texture Synthesis by Non-parametric Sampling.” *ICCV*, 1999.

[FB81] 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.

[Fen03] Simon Fenney. "Texture compression using low-frequency signal modulation." In *ACM SIGGRAPH*, 2003.

[FH04] Pedro Felzenszwalb and Daniel Huttenlocher. "Efficient Graph-Based Image Segmentation." *IJCV*, 2004.

[FL11] WT Freeman and C. Liu. "Markov random fields for super-resolution and texture synthesis." *Advances in Markov Random Fields for Vision and Image Processing*, 2011.

[GG84] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *TPAMI*, 1984.

[Gib84] J. J. Gibson. *The ecological approach to visual perception.* LEA, 1984.

[GS12] G. Georgiadis and S. Soatto. "Scene-Aware Video Modeling and Compression." In *Data Compression Conference.* April 2012.

[GZW03a] C. Guo, S. Zhu, and Y. N. Wu. "Toward a Mathematical Theory of Primal Sketch and Sketchability." *ICCV*, 2003.

[GZW03b] C. E. Guo, S. C. Zhu, and Y. N. Wu. "Towards a Mathematical Theory of Primal Sketch and Sketchability." *ICCV*, 2003.

[GZW07] C. Guo, S. C. Zhu, and Y.N. Wu. "Primal sketch: Integrating structure and texture." *CVIU*, 2007.

[Har05] R. M. Haralick. "Statistical and structural approaches to texture." *Proceedings of the IEEE*, 2005.

[HB95] D. J. Heeger and J. R. Bergen. "Pyramid-based texture analysis/synthesis." *ACM SIGGRAPH*, 1995.

[HJ92] D.J. Heeger and A.D. Jepson. "Subspace methods for recovering rigid motion I." *IJCV*, 1992.

[HS81] B.K.P. Horn and B.G. Schunck. "Determining optical flow." *Artificial Intelligence*, 1981.

[HS88] C. Harris and M. Stephens. "A combined corner and edge detector." *Alvey Vision*, 1988.

[Hub81] P.J. Huber. *Robust statistics.* Wiley, New York, 1981.

[IB09] L. Itti and P. Baldi. "Bayesian surprise attracts human attention." *Vision research*, 2009.

[IRP92] M. Irani, B. Rousso, and S. Peleg. "Detecting and tracking of multiple moving objects using temporal integration." *ECCV*, 1992.

[ISB99] Y. Ivanov, C. Stauffer, A. Bobick, and WEL Grimson. "Video surveillance of interactions." *CVPR*, 1999.

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

[JC73] W.S. Jewell and CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER. *Credible Means Are Exact Bayesian for Exponential Families.* Defense Technical Information Center, 1973.

[JFK03] N. Jojic, B. J. Frey, and A. Kannan. "Epitomic analysis of appearance and shape." *ICCV*, 2003.

[JM008] "H.264/AVC JM Reference Software.", August 2008.

[Jul62] B. Julesz. "Visual pattern discrimination." *IRE Trans info theory, IT-8*, 1962.

[Jul81] B. Julesz. "Textons, the elements of texture perception and their interactions." *Nature*, 1981.

[KEB05] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. "Texture optimization for example-based synthesis." *Proc. of ACM SIGGRAPH*, 2005.

[KEM09] I. Kokkinos, G. Evangelopoulos, and P. Maragos. "Texture analysis and segmentation using modulation features, generative models, and weighted curve evolution." *PAMI*, **31**(1):142–157, 2009.

[Kol06] V. Kolmogorov. "Convergent tree-reweighted message passing for energy minimization." *PAMI*, 2006.

[KSE03] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. "Graphcut Textures: Image and Video Synthesis Using Graph Cuts." *ACM Trans. Graph.*, 2003.

[KWR06] A. Kannan, J. Winn, and C. Rother. "Clustering Appearance and Shape by Learning Jigsaws." *NIPS*, 2006.

[Lin98] T. Lindeberg. *Edge Detection and Ridge Detection with Automatic Scale Selection.* Cambridge University Press, 1998.

[LK81] B. Lucas and T. Kanade. "An iterative image registration technique with an application to stereo vision." *IJCAI*, 1981.

[LLH04] Y. Liu, W-C Lin, and J. H. Hays. "Near-Regular Texture Analysis and Manipulation." *SIGGRAPH*, 2004.

[LLX01] Lin Liang, Ce Liu, Yingqing Xu, Baining Guo, and Heung yeung Shum. "Real-time texture synthesis by patch-based sampling." *Proc. of ACM Transactions on Graphics*, 2001.

[LM01] T. Leung and J. Malik. "Representing and recognizing the visual appearance of materials using three-dimensional textons." *IJCV*, 2001.

[LS11] T. Lee and S. Soatto. "Video-based descriptors for object recognition." *Image and Vision Computing*, 2011.

[LSP05] S. Lazebnik, C. Schmid, and J. Ponce. "A sparse texture representation using local affine regions." *PAMI*, 2005.

[LTL05] Y. Liu, Y. Tsin, and W. C. Lin. "The promise and perils of near-regular texture." *IJCV*, 2005.

[LYT08] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. "SIFT Flow: Dense Correspondence across Different Scenes." *ECCV*, 2008.

[M 71] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods." *JASA*, 1971.

[ME07] Tomasz Malisiewicz and Alexei A. Efros. "Improving Spatial Support for Objects via Multiple Segmentations." *BMVC*, 2007.

[Mei05] M. Meilă. "Comparing clusterings: an axiomatic view." *ICML*, 2005.

[MFT01] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics." *ICCV*, 2001.

[MM02] B. S. Manjunath and W. Y. Ma. "Texture features for browsing and retrieval of image data." *PAMI*, 2002.

[MP90] J. Malik and P. Perona. "Preattentive texture discrimination with early vision mechanisms." *JOSAA*, 1990.

[MP12] Pavlos Mavridis and Georgios Papaioannou. "Texture Compression using Wavelet Decomposition." *Proceedings of Pacific Graphics*, 2012.

[MRE04] G. Mori, X. Ren, A.A. Efros, and J. Malik. "Recovering Human Body Configurations: Combining Segmentation and Recognition." *CVPR*, 2004.

[MRY11] Hossein Mobahi, Shankar R. Rao, Allen Y. Yang, S. Shankar Sastry, and Yi Ma. "Segmentation of Natural Images by Texture and Boundary Compression." *IJCV*, 2011.

[MSK03] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An invitation to 3D vision, from images to models*. Springer Verlag, 2003.

[MVM11] S. Maji, N. K. Vishnoi, and J. Malik. "Biased normalized cuts." *CVPR*, 2011.

[NBW09] P. Ndjiki-Nya, D. Bull, and T. Wiegand. "Perception-oriented video coding based on texture analysis and synthesis." In *ICIP*, 2009.

[NDK12] Patrick Ndjiki-Nya, Dimitar Doshkov, Hagen Kaprykowsky, Fang Zhang, Dave Bull, and Thomas Wiegand. "Perception-oriented video coding based on image analysis and completion: A review." *Signal Processing: Image Communication*, **27**(6):579–594, 2012.

[PH12] M. Pedersen and J.Y. Hardeberg. "Full-reference image quality metrics: Classification and evaluation." *Found. Trends. Comp. Graphics and Vision.*, 2012.

[PS00] Javier Portilla and Eero P Simoncelli. "A parametric texture model based on joint statistics of complex wavelet coefficients." *International Journal of Computer Vision*, **40**(1):49–70, 2000.

[PW10] O. Pele and M. Werman. "The Quadratic-Chi Histogram Distance Family." *ECCV*, 2010.

[Ric10] E. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley, 2010.

[Ris78] J. Rissanen. "Modeling By Shortest Data Description." *Automatica*, 1978.

[RM03] X. Ren and J. Malik. "Learning a classification model for segmentation." *ICCV*, 2003.

[RMY09] S. R. Rao, H. Mobahi, A. Y. Yang, S. S. Sastry, and Yi Ma. "Natural Image Segmentation with Adaptive Texture and Boundary Encoding." *ACCV*, 2009.

[RS13] Zhile Ren and Gregory Shakhnarovich. "Image Segmentation by Cascaded Region Agglomeration." *CVPR*, 2013.

[SB06] H.R. Sheikh and A.C. Bovik. "Image information and visual quality." *IEEE TIP*, 2006.

[SBK10] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. "Dense point trajectories by GPU-accelerated large displacement optical flow." *ECCV*, 2010.

[SCV02] B. Sandberg, T.F. Chan, and L. Vese. "A level-set and Gabor-based active contour algorithm for segmenting textured images." *UCLA CAM report*, 2002.

[SFP94] S. Soatto, R. Frezza, and P. Perona. "Motion estimation via dynamic vision." *ECCV*, 1994.

[SH98] P. H. Suen and G. Healey. "Analyzing the bidirectional texture function." *CVPR*, 1998.

[SJK09] Yaser Sheikh, Omar Javed, and Takeo Kanade. "Background Subtraction for Freely Moving Cameras." *ICCV*, 2009.

[SL03] D. Skocaj and A. Leonardis. "Weighted and Robust Incremental Method for Subspace Learning." *ICCV*, 2003.

[SM97] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation." *TPAMI*, 1997.

[SM13] Laurent Sifre and Stéphane Mallat. "Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination." *CVPR*, 2013.

[Soa09] S. Soatto. "Actionable Information in Vision." *ICCV*, 2009.

[Soa10] S. Soatto. *Steps Toward a Theory of Visual Information.* ArXiv http://arxiv.org/abs/1110.2053, 2010.

[SOH12] G.J. Sullivan, J. Ohm, Woo-Jin Han, and T. Wiegand. "Overview of the High Efficiency Video Coding (HEVC) Standard." *Circuits and Systems for Video Technology, IEEE Transactions on*, **22**(12):1649–1668, Dec 2012.

[SPV09a] G. Sundaramoorthi, P. Petersen, V. S. Varadarajan, and S. Soatto. "On the set of images modulo viewpoint and contrast changes." *CVPR*, 2009.

[SPV09b] G. Sundaramoorthi, P. Petersen, V. S. Varadarajan, and S. Soatto. "On the set of images modulo viewpoint and contrast changes." *CVPR*, 2009.

[SSY10] G. Sundaramoorthi, S. Soatto, and A. Yezzi. "Curious snakes: A minimum latency solution to the cluttered background problem in active contours." *CVPR*, 2010.

[ST99] N. Slonim and N. Tishby. "Agglomerative information bottleneck." *NIPS*, 1999.

[ST06] P. Sand and S. Teller. "Particle Video: Long-Range Motion Estimation using Point Trajectories." *CVPR*, 2006.

[TH94] P.C. Teo and D.J. Heeger. “Perceptual image distortion.” *ICIP*, 1994.

[TJP10] Michael Tao, Micah Johnson, and Sylvain Paris. “Error-tolerant Image Compositing.” *ECCV*, 2010.

[TPB99] 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.

[TS94] C. Tomasi and J. Shi. “Good Features to Track.” *CVPR*, 1994.

[TV07] R. Tron and R. Vidal. “A benchmark for the comparison of 3-D motion segmentation algorithms.” *CVPR*, 2007.

[TYW00] A. Tsai, A. Yezzi, and A. Willsky. “A Curve Evolution Approach to Medical Image Magnification via the Mumford-Shah Functional.” *MICCAI*, 2000.

[VF08] A. Vedaldi and B. Fulkerson. “VLFeat: An Open and Portable Library of Computer Vision Algorithms.” 2008.

[Vid03] R. Vidal. “Generalized Principal Component Analysis (GPCA).” *CVPR*, 2003.

[VZ03] M. Varma and A. Zisserman. “Texture classification: Are filter banks necessary?” *CVPR*, 2003.

[VZ05] M. Varma and A. Zisserman. “A Statistical Approach to Texture Classification from Single Images.” *IJCV*, 2005.

[WA93] J. Wang and E. Adelson. “Layered representation for image sequence coding.” In *ICASSP*, 1993.

[Wan95] B. A. Wandell. *Foundations of Vision*. Sinauer Associates, Inc., 1995.

[WB09] Zhou Wang and A.C. Bovik. “Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures.” *Signal Processing Magazine, IEEE*, **26**(1):98–117, Jan 2009.

[WBS04] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. “Image quality assessment: From error visibility to structural similarity.” *IEEE TIP*, 2004.

[WGZ08] 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.

[WHZ08] L. Wei, J. Han, K. Zhou, H. Bao, B. Guo, and H. Shum. “Inverse Texture Synthesis.” *ACM Trans. Graph.*, 2008.

[WL00] L. Wei and M. Levoy. “Fast Texture Synthesis Using Tree-structured Vector Quantization.” *SIGGRAPH*, 2000.

[WLP09] C. Wang, M. de La Gorce, and N. Paragios. “Segmentation, ordering and multi-object tracking using graphical models.” In *ICCV*, 2009.

[WSI07] Y. Wexler, E. Shechtman, and M. Irani. “Space-time completion of video.” *PAMI*, 2007.

[WT13] David Weiss and Ben Taskar. “SCALPEL: Segmentation Cascades with Localized Priors and Efficient Learning.” *CVPR*, 2013.

[WWO08] Huamin Wang, Yonatan Wexler, Eyal Ofek, and Hugues Hoppe. “Factoring repeated content within and among images.” In *ACM TOG*, 2008.

[XBY09] S. Xiaowei, Y. Baocai, and S. Yunhui. “A Low Cost Video Coding Scheme Using Texture Synthesis.” In *ISP*, Oct 2009.

[XMX08] X. Xie, M. Mirmehdi, X. Xie, and J. Suri. “Handbook of Texture Analysis.” *ICP*, 2008.

[YS01] Stella X. Yu and Jianbo Shi. “Grouping with Bias.” *NIPS*, 2001.

[ZLG10] Z. Zhang, X. Liang, A. Ganesh, and Y. Ma. “TILT: Transform Invariant Low-Rank Textures.” In *ECCV*, 2010.

[ZLY95] S. Zhu, T. Lee, and A. Yuille. “Region competition: Unifying snakes, region growing, energy/Bayes/MDL for multi-band image segmentation.” *ICCV*, 1995.

[Zuc76] Steven W Zucker. “Toward a model of texture.” *CGIP*, 1976.

[ZWM98] S. C. Zhu, Y. N. Wu, and D. Mumford. “Filters, Random Fields and Maximum Entropy (FRAME): towards the unified theory for texture modeling.” *IJCV*, 1998.

[ZXZ11] H. Zhi, Z. Xu, and S.-C. Zhu. “Video Primal Sketch: A Generic Middle-Level Representation of Video.” *ICCV*, 2011.