Two-Stream Convolutional Networks for Dynamic Texture Synthesis

MATTHEW TESFALDET

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

August, 2018

Abstract

This thesis introduces a two-stream model for dynamic texture synthesis. The model is based on pre-trained convolutional networks (ConvNets) that target two independent tasks: (i) object recognition, and (ii) optical flow regression. Given an input dynamic texture, statistics of filter responses from the object recognition and optical flow ConvNets encapsulate the per-frame appearance and dynamics of the input texture, respectively. To synthesize a dynamic texture, a randomly initialized input sequence is optimized to match the feature statistics from each stream of an example texture. In addition, the synthesis approach is applied to combine the texture appearance from one texture with the dynamics of another to generate entirely novel dynamic textures. Overall, the proposed approach generates high quality samples that match both the framewise appearance and temporal evolution of input texture. Finally, a quantitative evaluation of the proposed dynamic texture synthesis approach is performed via a large-scale user study.

Acknowledgements

First and foremost, I am grateful to my parents, Bereket and Simret Tesfaldet, for their unending commitment to my well-being and utmost support during trying moments of my research.

I am thankful to my mentors and supervisors, Marcus Brubaker and Kosta Derpanis, for their support and patience in guiding me these past two years. I am also thankful for the many opportunities they have given me for engaging with the vibrant community of computer vision and deep learning researchers and practitioners. I look forward to continuing my research with them.

I am grateful to Rick Wildes and Michael Brown for the many fruitful and enjoyable discussions we have had about research during my time at York University. I thank Rick for his comments and suggestions regarding the thesis manuscript.

I acknowledge the financial support of the Canadian Graduate Scholarship (CGS) awarded by the Natural Sciences and Engineering Research Council of Canada (NSERC). This research was undertaken as part of the Vision: Science to Applications program, thanks in part to funding from the Canada First Research Excellence Fund.

I reserve an additional, special thanks to Kosta Derpanis for his personal and academic support over the past five years, and for introducing me to the wonderful world of computer vision and deep learning. Your inspiring commitment to the success of your students, both in the classroom and in the lab, has helped this "young padawan" take a large step towards realizing his goals.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The natural world is rich in visual texture. While a precise definition of texture remains to be found, most research consider it as visual patterns that exhibit local variations while maintaining global homogeneity, as shown in Fig. 1.1.

Textures can be static or dynamic: static textures exist in two-dimensional (2D) image space (*e.g.*, grass and water) while dynamic textures extend the notion across time (*e.g.*, fluttering grass and wavy water). As a result, local spatial variations and global homogeneity extend across space *and* time. These temporal patterns have previously been studied under a variety of names, including turbulent flow [32] (for extracting optical flow from fluids undergoing irregular fluctuations), temporal textures [47] (for recognition of moving patterns such as windblown trees or rippling water), time-varying textures [5] (for synthesizing stochastically-moving patterns), dynamic textures [13] (for modelling and synthesizing stochastically-moving patterns), textured motion [64] (for modelling and

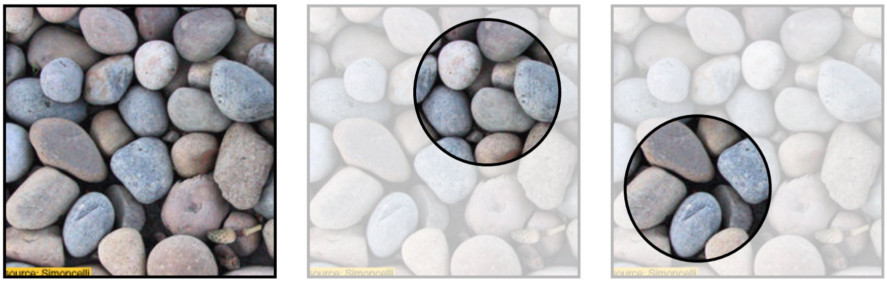


Figure 1.1: Definition of a static texture. A static texture is a visual pattern that exhibits local spatial variations while maintaining global homogeneity. Observing small apertures across the texture should reveal visual content that appears roughly the same, captured as feature statistics.

synthesizing patterns undergoing stochastic or consistent motion), and spacetime textures [12] (for classifying dynamic patterns). In this thesis, the term “dynamic texture” is adopted.

Both static and dynamic texture cues play important roles in our perception of surfaces. Understanding and characterizing these patterns has long been a problem of interest in human perception, computer vision, and computer graphics. In computer vision, studying the underlying statistics of textures allows us to gain insight as to how these complex structures can be interpreted and how we may be able to leverage this knowledge to inform certain vision-related tasks. Examples of such tasks include shape-from-texture [25], material recognition [10, 2], texture synthesis [31], and more recently, image style transfer [22]. In terms of specific applications, there are many in the creative-industry including, but not limited to, computer-generated imagery, digital painting, and image editing.

Shape-from-texture involves recovering the three-dimensional (3D) shape of an object from a 2D image by using texture as a cue. Gibson [25] proposed the *texture gradient* as the primary basis of surface perception by humans. He conjectured that neighbouring areas on a textured surface are perceived differently only due

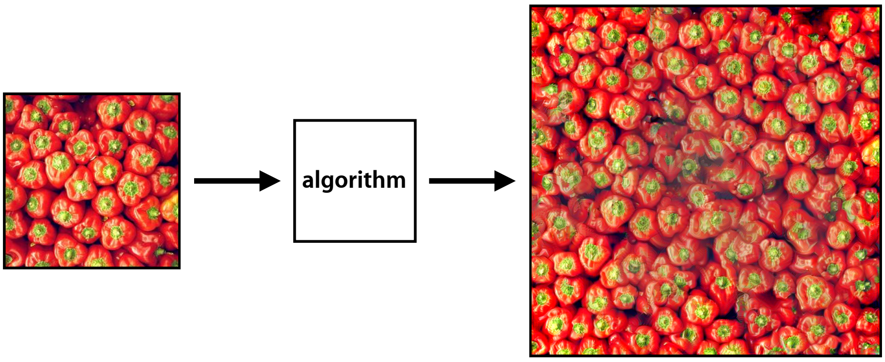


Figure 1.2: Texture synthesis is the process of algorithmically constructing a texture (right) that matches or extends a given source texture (left) by taking advantage of its structural content.

to differences in surface orientation and distance from the observer.

Material recognition in computer vision involves recognizing material categories (*e.g.*, fabric, water, and wood) from an image based on the visual appearance of surfaces. The visual appearance of a surface depends on several factors [10, 2], such as illumination, geometric structure at various scales, viewing direction, and surface reflectance properties, *e.g.*, the Bidirectional Reflectance Distribution Function (BRDF) [46]. Notably, texture can be useful for distinguishing materials. For example, wood and water each have unique texture that easily distinguishes the two. Dana *et al.* [10] introduced the Bidirectional Texture Function (BTF) and demonstrated that the visual appearance of materials can be characterized by measuring texture.

Texture synthesis (Fig. 1.2) is the process of algorithmically constructing a texture that matches or extends a given source texture by taking advantage of its structural content. Heeger and Bergen [31] took advantage of the fact that two textures are often difficult to discriminate when they produce a similar distribution of responses from a bank of linear filters. They used a combination of Laplacian and

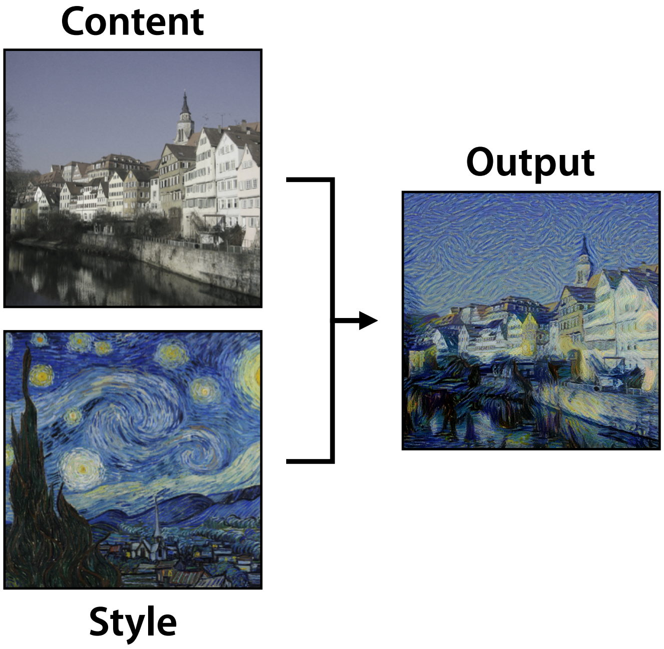


Figure 1.3: Image style transfer. The goal is to synthesize a texture (right) from an input "style" image (bottom-left) while constraining the process in order to preserve the semantic content of an input "content" image (top-left).

steerable pyramids to deconstruct a given texture and synthesized a new texture by matching the distributions of responses from each pyramid level. Portilla and Simoncelli [50] extended this approach by including complex "analytical" filters that allowed them to utilize measures of local phase and energy in their texture descriptors. More recently, Gatys *et al.* [21] demonstrated impressive results for texture synthesis by using a convolutional network (ConvNet) instead of a linear bank of filters to model the non-linear spatial statistics of a given texture.

Image style transfer (Fig. 1.3) is a recent technique where the goal is to recompose an image in the "style" (*e.g.*, texture) of another image. This can be considered as a texture transfer problem, as previously demonstrated by Efros *et al.* [15], where they transferred a given texture to another image by stitching together small patches of the given texture while conforming to the luminance of the other image. Although the simplicity of their approach was attractive, it failed for highly structured textures due to patch boundary inconsistencies, limiting the selection of acceptable textures. Gatys *et al.* [21] modified their previous ConvNet for texture synthesis to support texture transfer by including an additional objective that enforced the synthesized texture to match the semantic content of a given image, resulting in an image style transfer [22]. Unlike the patch-based method of Efros *et al.* [15], Gatys *et al.*'s approach of using a ConvNet was more robust to textures with long-range consistencies.

Motivated by the ConvNet model of Gatys *et al.* [21] for texture synthesis, the focus of this thesis is on the synthesis of dynamic texture samples, as captured in video, based on a single exemplar through the use of ConvNets. Inspired by Gatys *et al.*'s [22] method of image style transfer, a novel form of style transfer for dynamic textures is presented as well.

1.2 Summary of thesis

Many common dynamic textures are naturally described by the ensemble of appearance and dynamics (*i.e.*, spatial and temporal pattern variation) of their constituent elements. In this thesis, a factored analysis of dynamic textures in terms of their appearance and dynamics is proposed. This factored analysis is then used to enable dynamic texture synthesis based on an example dynamic texture as input. It also enables a novel form of style transfer where the target appearance and dynamics can be taken from different sources—termed *dynamics style transfer*. An overview of dynamic texture synthesis and dynamics style transfer is shown in Fig. 1.4.

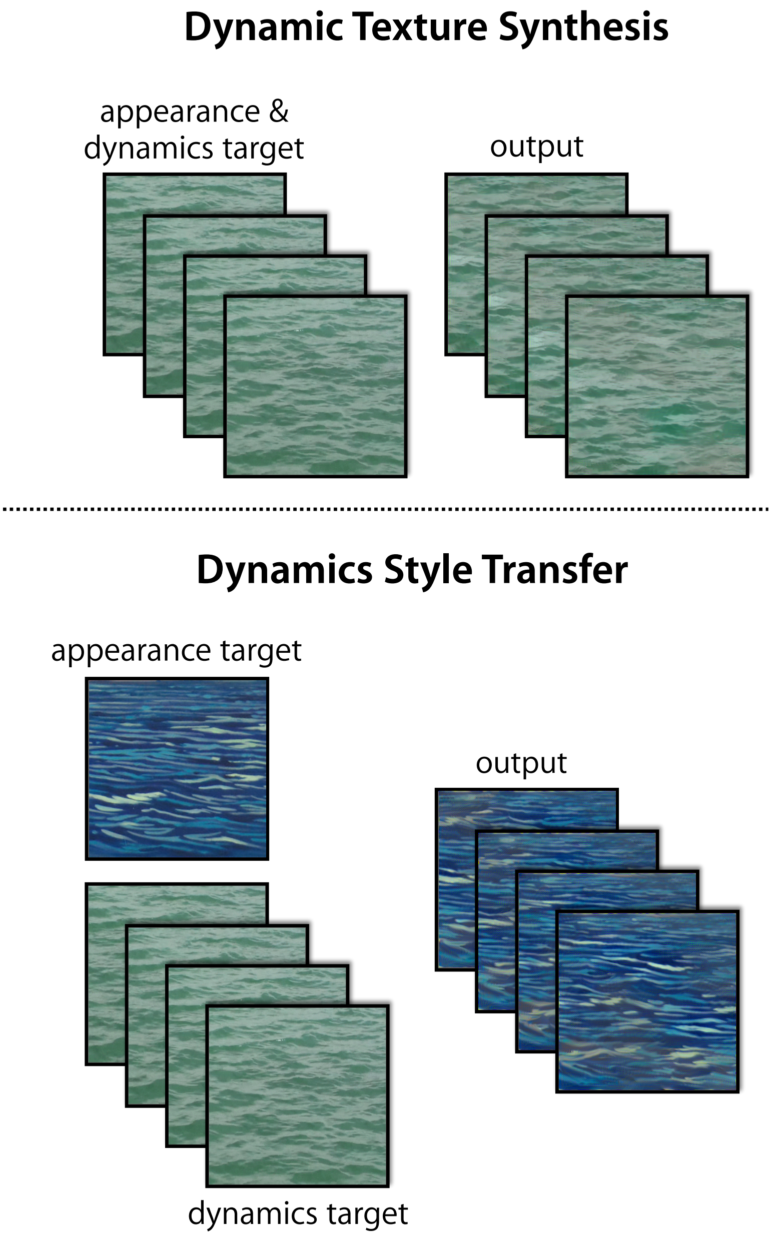


Figure 1.4: Dynamic texture synthesis and dynamics style transfer. (top) Given an input dynamic texture as the target, the two-stream model synthesizes a novel dynamic texture that preserves the target's appearance and dynamics characteristics. (bottom) The two-stream approach enables synthesis that combines the texture appearance from one target with the dynamics from another, resulting in a composition of the two.

The proposed model is constructed from two ConvNets: an appearance stream and a dynamics stream, which have been pre-trained for object recognition and optical flow regression, respectively. Similar to previous work on spatial textures [21, 31, 50], an input dynamic texture is summarized in terms of a set of spatiotemporal statistics of filter outputs from each stream. The appearance stream models the per-frame appearance of the input texture, while the dynamics stream models its temporal dynamics. The synthesis process consists of optimizing a randomly initialized noise pattern such that its spatiotemporal statistics from each stream match those of the input texture. The architecture is inspired by insights from human perception and neuroscience. In particular, psychophysical studies [9] show that humans are able to perceive the structure of a dynamic texture even in the absence of appearance cues, suggesting that the two streams are effectively independent. Similarly, the two-stream hypothesis [26] models the human visual cortex in terms of two pathways, the ventral stream (involved with object recognition) and the dorsal stream (involved with motion processing). Two-stream networks have also been used for video understanding tasks in computer vision, with particular attention to action recognition [57, 18].

In this thesis, the two-stream analysis of dynamic textures is applied to texture synthesis. A range of dynamic textures are considered and it is demonstrated that the proposed approach generates novel, high quality samples that match both the frame-wise appearance and temporal evolution of an input example. Further, as stated previously, the factorization of appearance and dynamics enables a novel form of style transfer, where the dynamics of one texture are combined with the appearance of a different one, *cf*. [22]. This can even be done using a single image as an appearance target, which allows static images to be animated. Finally, the perceived realism of the generated textures is validated through an extensive user study.

1.3 Contributions

The contributions of this work span both theory and application. Specifically, this thesis presents the key components for creating the proposed dynamic texture synthesis model, resulting in four primary contributions to the dynamic texture literature.

1. **Factored representation of appearance and dynamics**. First, theoretical insight into the characterization of dynamic textures is provided by building a novel factored representation of both appearance and dynamics. Qualitatively, the two-stream representation is effective in generating visually compelling, novel instances of a wide range of dynamic textures.

2. **Motion energy representation of dynamics via a ConvNet**. Second, for the representation of dynamics, a novel ConvNet based on a "marginalized" motion energy model [11, 12] is constructed and trained on the proxy task of optical flow regression. This representation of dynamics provides a substantial improvement to the quality of synthesized textures when compared to using optical flow directly.

3. **Dynamics style transfer**. Third, a novel form of style transfer is demonstrated, where the dynamics of a dynamic texture can be mixed with the spatial appearance of a different (static or dynamic) texture. This is enabled by the proposed factored representation.

4. **Quantitative evaluation via user study**. Finally, a quantitative evaluation on the limitations of the method is performed through the inclusion of a broad range of textures and an extensive user study. This analysis provides insight for the types of characteristics of temporal imagery that may cause a breakdown of the proposed model. These insights may point to future work to address limitations of the proposed model.

Chapter 2

Background

This chapter aims to summarize the relevant theory and mathematics of convolutional networks, static and dynamic texture synthesis, image style transfer, and representations of dynamics so as to provide sufficient background information for the following chapters. Research related to this thesis is covered simultaneously.

2.1 Convolutional networks

This section provides a brief summary of convolutional networks; a more thorough overview can be found in texts (*e.g.*, [27]) and review articles (*e.g.*, [29]).

A convolutional network (ConvNet) is a feed-forward computational graph of processing nodes for approximating non-linear functions. It is commonly used in analyzing visual imagery and is a class of artificial neural networks (ANNs), which are computational systems inspired by biological neural networks. ConvNets perform tasks (*e.g.*, classifying objects or regressing the angle of rotated handwritten digits) by processing an input, $\mathbf{X} \in \mathbb{R}^{H_x \times W_x \times C_x}$, in a feed-forward manner via a series of linear and non-linear transformations and producing an output, $\mathbf{Y} \in \mathbb{R}^{H_y \times W_y \times C_y}$, relevant to the task (such as the class of an object). Here, $H_* \times W_*$ represents the spatial dimensions and C_* represents the number of channels (*i.e.*, the "depth" of the input). For example, a 3-channel colour image input with 256×256 spatial locations is represented as $\mathbf{X} \in \mathbb{R}^{256 \times 256 \times 3}$, where each spatial location contains three pixel intensities, corresponding to the amount of intensity of the colours Red, Green, and Blue (RGB), respectively. Each non-linear transformation acts as a point of demarcation in the network known as a *layer*. ConvNets typically consist of multiple layers, each containing a collection of nodes sometimes called *neurons*. At each spatial-channel location of the input to a layer, a neuron computes local non-linear transformations, *e.g.*, $\bar{\phi}(\mathbf{x}) = \max(0, \phi(\mathbf{x}))$ (known as the *rectified linear unit* or ReLU [45]). At a single location of the input, $\mathbf{x} \equiv (x, y, z)$, the non-linear transformation performed by this neuron produces an output called a *feature activation*, $\bar{\phi}(\mathbf{x}) \in \mathbb{R}$. The set of activations produced by a neuron at every location of the input is known as an *activation* or *feature map*, $\bar{\phi} \in \mathbb{R}^{H_{\bar{\phi}} \times W_{\bar{\phi}} \times C_{\bar{\phi}}}$. At the l-th layer of the network and for each location $\mathbf{x}$ of the input map, the input to each neuron is the weighted linear combination of activations from local, neighbouring neurons at the previous layer, $l-1$:

$$\begin{aligned} \phi^l(\mathbf{x}) &= \left(\mathbf{w}^l * \bar{\phi}^{l-1}(\mathbf{x})\right) + b^l \\ &= \left(\sum_{(i,j,k) \in \Omega} \mathbf{w}^l(i,j,k) \bar{\phi}^{l-1}(x-i, y-j, z-k) \right) + b^l \ , \end{aligned} \tag{2.1}$$

where $\mathbf{w}^l$ are the *weights* (or *filter*) applied to the input $\bar{\phi}^{l-1}(\mathbf{x})$, Ω is a spatial-channel neighbourhood centered about $\mathbf{x}$, b^l is an offset term known as the *bias*, and $*$ represents the *convolution* (or *filtering*) operator. Colloquially, the term "convolution" is often used to describe the combined process of convolving over an input and subsequently computing its activation. Convolutions are performed at each location of the input, operating on a localized area of influence called its *receptive field*. ConvNets are distinguished from most other ANNs in that the weights associated with a convolution are shared across the entire input. Specifically, when convolving over an input, instead of using a unique filter for each location, the same filter is reused—this is called *weight sharing*. The idea is that if a filter is important for capturing a feature (*e.g.*, edge, corner, face, etc.) at a particular location, then it is assumed that the filter is important at other locations as well since the same feature may appear elsewhere. At the base of the ConvNet, inputs are typically images while inputs at intermediate layers are activation maps ($e.g.$, $\bar{\phi} \in \mathbb{R}^{H_{\bar{\phi}} \times W_{\bar{\phi}} \times C_{\bar{\phi}}}$).

Pooling

ConvNets commonly include *pooling* layers that combine regions of activations into a single activation in the next layer. There are two commonly used types of pooling: *average pooling*, which uses the average value from each of region of activations at the previous layer, and *max pooling*, which uses the max value instead. Pooling provides a degree of translation-invariance to ConvNets by making them less affected by small changes in the positions of input activations. It is typically followed by a downsample operation to reduce spatial resolution.

Normalization

Widely used in ConvNets are *normalization* layers that serve to inhibit or bind local activations to a certain range. Normalization is typically done across channels, such as with the commonly used *divisive normalization*:

$$\hat{\phi}_i^l(\mathbf{x}) = \frac{\bar{\phi}_i^l(\mathbf{x})}{\sum_{j=1}^{C} \bar{\phi}_j^l(\mathbf{x}) + \epsilon} \ , \tag{2.2}$$

where $\bar{\phi}_i^l(\mathbf{x})$ is an activation from the i-th channel at spatial position $\mathbf{x}$, C is the total number of channels, and ϵ is a small value to prevent division by zero. An issue with using unbounded activation functions (*e.g.*, ReLU) with convolutions in a ConvNet is that their outputs keep increasing with increasing contrast of the input. This dependency on contrast magnitude makes it difficult to determine whether a high response is indicative of a salient feature or high input contrast. Thus, divisive normalization binds unbounded activations across channels such that information about the magnitude of contrast is discarded in favour of a representation based only on relative variations in contrast across input activations, *i.e.*, the underlying feature patterns of the input.

Training a convolutional network

ConvNets "learn" to perform tasks through an iterative process called *training*, which involves optimizing their weights based on an objective over training data (*e.g.*, input images with corresponding expected outputs). At each iteration, an input is fed through the network to produce an output that is subsequently evaluated against the expected, *i.e.*, groundtruth, output for the given input. This evaluation is known as the *loss function* and represents the network's performance on the task. Implicitly, it also represents the objective the network must achieve, *e.g.*, minimizing classification or regression error. Starting from the loss, the network adjusts its weights and biases at each layer via the gradient of the loss with respect to the weights and biases at that layer. The adjustment of weights and biases via the gradient of the loss function is called *backpropagation* [27]. After a suitable amount of training iterations, this *gradient descent* process is terminated.

2.1.1 Two-stream convolutional networks

In the context of video analysis, two-stream ConvNets [57, 18] are a class of ConvNets that separate processing of input temporal imagery into two recognition streams (spatial and temporal), each with its own task. They are inspired by the two-stream hypothesis [26] that models the human visual cortex in terms of two pathways, the ventral stream (involved with object recognition) and the dorsal stream (involved with motion processing).

Both streams are implemented as ConvNets with the overall intent to decouple processing of spatial and temporal information. This decoupling allows each network to be trained on separate tasks, *e.g.*, the spatial ConvNet can be trained on object recognition and the temporal ConvNet can be trained on motion recognition. Furthermore, the decoupled spatial ConvNet can take advantage of the availability of large amounts of annotated image data for training, *e.g.*, the ImageNet dataset [54]. Two-stream ConvNets have been successfully used for video understanding tasks in computer vision, with particular attention to action recognition [57, 18].

2.2 Parametric texture synthesis

Texture synthesis is the process of algorithmically constructing a texture that matches or extends a given source texture by taking advantage of its structural content. There are two general approaches that have dominated the texture synthesis literature: non-parametric sampling approaches that synthesize a texture by sampling pixels of a given source texture [16, 39, 55, 66], and statistical parametric models that aim to synthesize a texture by sampling from a parameterized model of the source texture. As the proposed approach is an instance of a parametric model, this section will focus on these parametric approaches.

The statistical characterization of visual textures was introduced in the seminal work of Julesz [36]. He conjectured that particular statistics of pixel intensities were sufficient to partition textures into metameric (*i.e.*, perceptually indistinguishable) classes. Later work leveraged this notion for static texture synthesis [31, 50]. In particular, inspired by models of the early stages of visual processing, statistics of (handcrafted) multi-scale oriented filter responses were used to optimize an initial noise pattern to match the filter response statistics of an input texture.

More recently, Gatys *et al.* [21] demonstrated impressive results by replacing the handcrafted linear filter bank with the learned filters from the VGG-19 [58] ConvNet pre-trained on the ImageNet [54] dataset for the task of object recognition. Textures were modelled in terms of the normalized correlations between activation maps within several layers of the network.

2.2.1 Texture synthesis using a convolutional network

Since the two-stream approach to dynamic texture synthesis proposed in this thesis is an extension of the Gatys *et al.* [21] texture synthesis model, it is useful to describe their approach here. Given a target texture as input, let $\mathbf{A}^l \in \mathbb{R}^{N_l \times M_l}$ be its row-vectorized activation maps at the l-th layer of a ConvNet, where N_l and M_l denote the number of activation maps and the number of spatial locations, respectively (in the case of Gatys *et al.* [21], they used the VGG-19 ConvNet, and they normalized the network by scaling its weights such that the mean activation of each convolutional filter over images and positions is equal to one). The normalized correlations between activation maps within a layer are encapsulated by a Gram matrix, $\mathbf{G}^l \in \mathbb{R}^{N_l \times N_l}$, whose entries are given by:

$$G^l_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} A^l_{ik} A^l_{jk} \ , \tag{2.3}$$

where A^l_{ik} denotes the activation of feature i at location k in layer l on the target texture. Given a synthesized texture as input, similarly, let its row-vectorized activation maps be $\hat{\mathbf{A}}^l \in \mathbb{R}^{N_l \times M_l}$ and its normalized activation map correlations be the Gram matrix, $\hat{\mathbf{G}}^l \in \mathbb{R}^{N_l \times N_l}$, whose entries are given by:

$$\hat{G}^l_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} \hat{A}^l_{ik} \hat{A}^l_{jk} \ . \tag{2.4}$$

The final objective is defined as the average of the mean squared error between the Gram matrices of the target texture and that of the synthesized texture:

$$\mathcal{L} = \frac{1}{L} \sum_{l} \|\mathbf{G}^l - \hat{\mathbf{G}}^l\|^2_F \ , \tag{2.5}$$

where L is the number of ConvNet layers used when computing Gram matrices and $\|\cdot\|_F$ is the Frobenius norm. In the case of Gatys *et al.* [21], Gram matrices were computed on layers *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4*. Through systematic evaluation, they reported these layers to be qualitatively superior to other layer subsets for texture synthesis. To note, the Gram matrix is positive semidefinite and thus exists in a non-Euclidean space. Non-Euclidean metrics (*e.g.*, the Log-Euclidean [4]) may be more suitable than the Frobenius norm, however, following Gatys *et al.*'s [21] implementation, the Frobenius norm is used here instead. Gatys *et al.* did not investigate using other metrics, so this may be worth exploring for future work.

Before synthesizing a texture, an initial forward pass through the ConvNet is performed with the target texture as input. The target texture's Gram matrices across various layers in the network are computed and stored to be used in the final objective for the synthesis process, Eq. 2.5. Then the synthesized texture is initialized with Independent and Identically Distributed (IID) Gaussian noise. The final objective, Eq. 2.5, is minimized with respect to the synthesized texture. With each iteration of the optimization process, the pixel values of the synthesized texture are updated to appear increasingly perceptually similar to the target texture. An overview of this process is presented in Fig. 2.1.

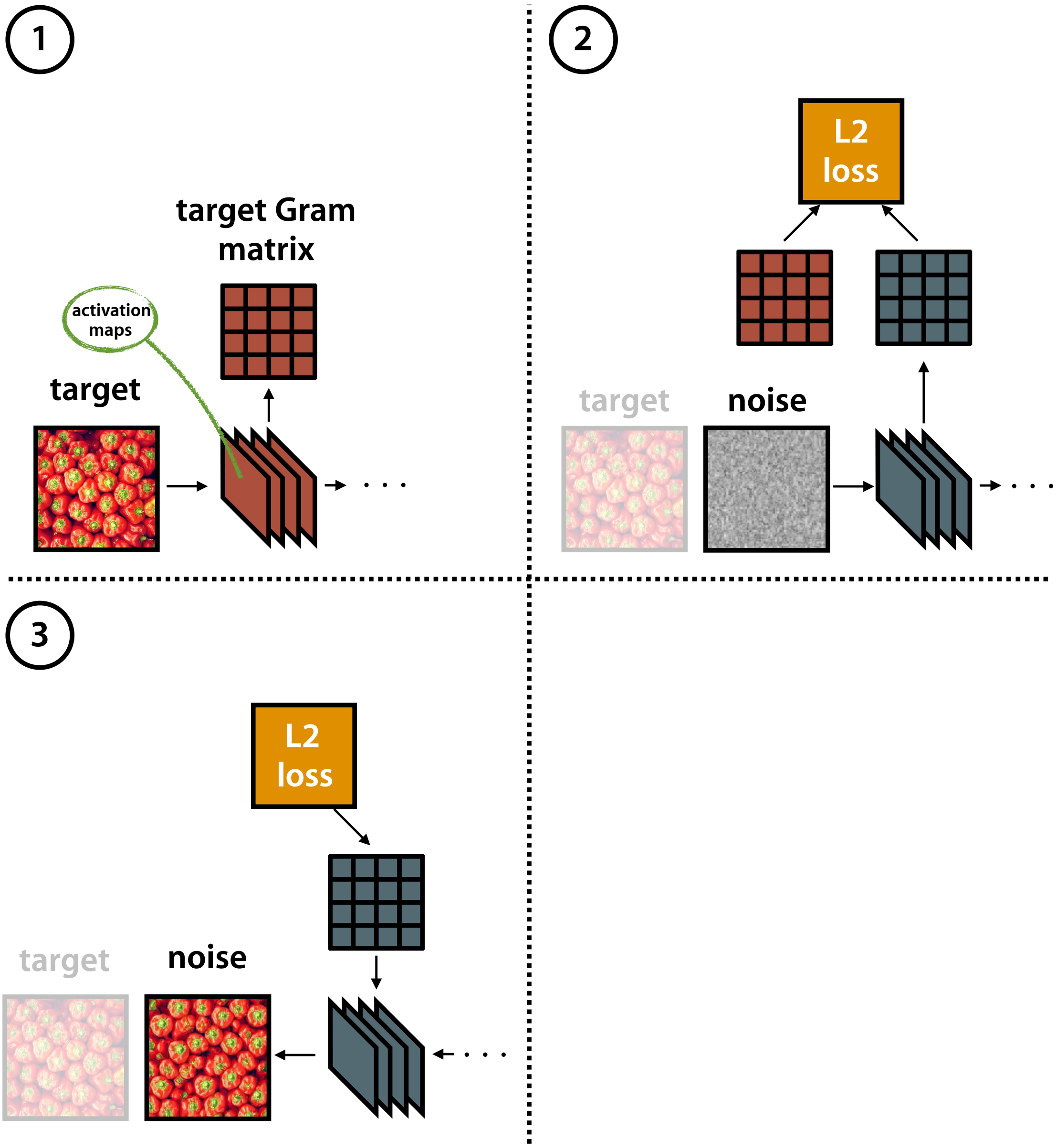


Figure 2.1: Gatys *et al.*'s [21] approach to static texture synthesis with the VGG-19 [58] convolutional network. Only the first layer of VGG-19 is shown. (1) An initial forward pass is performed with the target texture. Its Gram matrices across various layers are computed and stored. (2) The total L2 loss between the Gram matrices of the synthesized texture and the target is computed. (3) The loss is optimized with respect to the synthesized texture (with the weights of VGG-19 fixed), updating it to appear as perceptually similar to the target as possible.

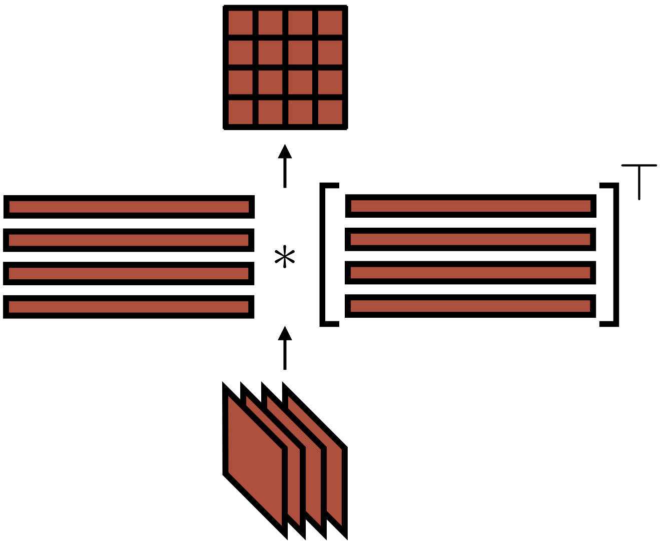


Figure 2.2: Computing the Gram matrix from the activation maps of a layer. Activation maps are first reshaped to row vectors, then the Gram matrix is realized by computing a matrix multiplication between the row-vectorized activation maps and their transpose.

2.2.2 The Gram matrix as a texture metric

Before explaining the Gram matrix as a suitable texture metric, it is necessary to first understand the mathematics behind it. The Gram matrix, $\mathbf{G} \in \mathbb{R}^{n \times n}$, of a set of m-dimensional vectors, $v_1, \ldots, v_n \in \mathbb{R}^m$, is the symmetric matrix of inner products, whose entries are given by $G_{ij} = \langle v_i, v_j \rangle$. Essentially, the Gram matrix is a covariance matrix describing which of its input vectors are correlated with each other. In the case of Gatys *et al.*'s [21] texture synthesis with a ConvNet, the set of vectors used to compute the Gram matrix are row-vectorized activation maps (Fig. 2.2).

The hierarchical feature representations learned by ConvNets have been shown to be powerful for difficult visual perceptive tasks such as object recognition [38, 58], significantly outperforming previous models that have relied on handcrafted approaches. Significantly, Movshon and Simoncelli [44] have suggested that given the texture-like features that can be captured in the intermediate stages of these ConvNets, images synthesized from these representations may prove useful as stimuli in perceptual or physiological investigations. This suggests that ConvNets may be a suitable basis for modelling textures, and consequently, texture synthesis.

In texture synthesis, the Gram matrix computed on activations measures the amount that co-located features tend to activate together. It transforms the spatially-varying feature space into a stationary, spatially-invariant one. Since textures exhibit stationary statistics (*i.e.*, the visual content is spatially homogeneous), by computing Gram matrices across several layers, a stationary, multi-scale representation of the input image in terms of its texture information is achieved.

2.2.3 Image style transfer

In subsequent work, Gatys *et al.*'s [22] texture model was used in image style transfer, where the style of one image was combined with the image content of another to produce a new image. This was achieved by appending an additional term to Eq. 2.5 that enforced the synthesized texture to match the semantic content of the given content image. Specifically,

$$\mathcal{L} = \frac{1}{L_{style}} \sum_{l} \|\mathbf{G}^l - \hat{\mathbf{G}}^l\|_F^2 \; + \frac{1}{L_{content}} \sum_{l} \|\mathbf{A}^l - \hat{\mathbf{A}}^l\|_F^2 \;, \tag{2.6}$$

where L_{style} and $L_{content}$ are the number of VGG-19 layers used when computing Gram matrices and activation maps, respectively. Gram matrices are computed on the same layers as before, and activation maps are computed on layer *conv4_2*.

To briefly review, the Gram matrix of activation maps conveys a notion of texture, or “style”, describing which features tend to activate together. Although the style of the style image is preserved, the global arrangement of its features are not. By including the objective of matching the features of the content image, however, the global arrangement of semantic image content from the content image is preserved. This results in a synthesized image that contains the content of the content image and the style of the style image. The idea of transferring style from one image to another serves as a loose inspiration for the novel dynamics style transfer enabled by the proposed two-stream model. Although with dynamics style transfer, content is not preserved, as it is only a transfer of texture dynamics encompassed by a texture synthesis process.

2.3 Dynamic texture synthesis

Dynamic textures extend from static textures with an additional temporal dimension. The stationarity of spatial statistics of static textures also applies to the temporal domain of dynamic textures.

Unlike static texture synthesis, dynamic texture synthesis has not been as deeply explored. Loosely related—although tangential to dynamic texture synthesis and the proposed dynamics style transfer—Ruder *et al.* [53] extended the image style transfer model of Gatys *et al.* [22] to video by using optical flow to enforce temporal consistency of the resulting imagery. Although their model produced a video output, their core approach focused on an analysis of static style on a per-frame basis. This is tangential to the proposed approach since dynamic textures require an analysis across space *and time*.

Variants of linear autoregressive models have been studied [60, 13, 64, 19] that jointly model the appearance and dynamics of spatiotemporal patterns. More recent work has considered ConvNets as a basis for modelling dynamic textures. Xie *et al.* [67] proposed a spatiotemporal generative model where each dynamic texture is modelled as a random field defined by multiscale, spatiotemporal ConvNet filter responses and dynamic textures are realized by sampling the model. Unlike the proposed approach, which assumes pre-trained fixed networks, Xie *et al.*'s [67] approach requires their ConvNet weights to be trained using the input texture prior to synthesis. The manner in which they model dynamic textures appears to limit synthesis to a reconstruction, not an extrapolation, of the original sequence, limiting the generalizability of their approach, *e.g.*, synthesizing textures beyond the spatiotemporal extent of the input. A recent unpublished work by Funke *et al.* [20] described preliminary results extending the framework of Gatys *et al.* [21] to model and synthesize dynamic textures by computing a Gram matrix of filter activations over a small spatiotemporal window. In contrast, the proposed two-stream filtering architecture is more expressive as the dynamics stream is specifically tuned to spatiotemporal dynamics. Moreover, the factorization in terms of appearance and dynamics enables a novel form of style transfer, where the dynamics of one pattern are transferred to the appearance of another to generate an entirely new dynamic texture. This work is the first to demonstrate this form of style transfer.

2.4 Representations of dynamics

Numerous representations of dynamics in temporal imagery have been explored, each with their own limitations and level of abstraction. Figure 2.3 illustrates an

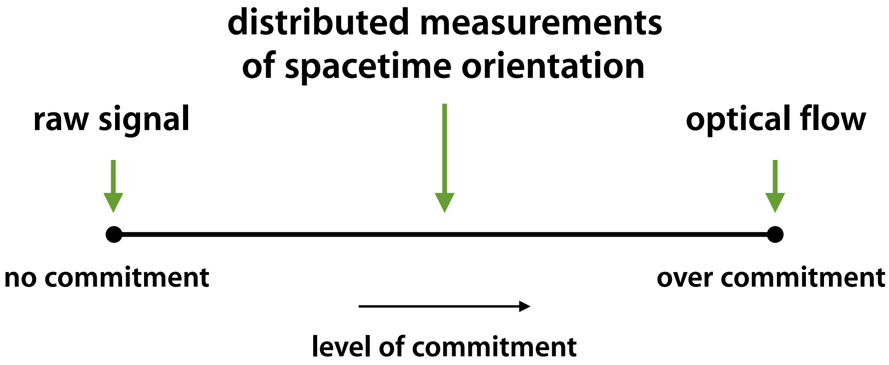


Figure 2.3: Dynamics representation spectrum (adapted from Derpanis [11]). Common abstractions of dynamics of temporal imagery and their respective level of commitment to an underlying model.

organization of several extant representations of temporal imagery dynamics. At one extreme, no commitment to an abstraction is made, the raw pixelwise intensity is used directly. This representation fails to leverage the rich underlying structure in the data. The remaining representations are discussed below.

2.4.1 Optical flow

At the other extreme of Fig. 2.3, a two-dimensional (2D) vector field is used to represent the dynamics of the input temporal imagery. This vector field is known as *optical flow*. It is used to represent the apparent motion of image pixels between two consecutive frames that is caused by the movement of objects or the camera. Each vector in the 2D vector field represents a displacement consisting of a horizontal and vertical component, describing the movement of pixels from one frame to the next. Figure 2.4 provides a visualization of optical flow. The recovery of optical flow from temporal imagery has long been studied in computer vision. Traditionally, it has been addressed by handcrafted approaches *e.g.*, [33, 43, 52]. Recently, ConvNet approaches have been demonstrated as viable

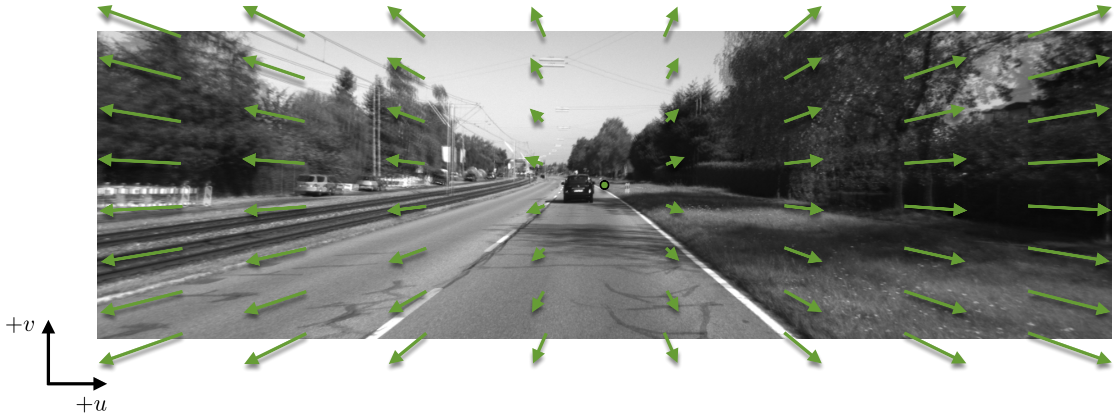


Figure 2.4: Optical flow visualization. Optical flow is a 2D displacement vector field used to represent the apparent motion of image pixels between two consecutive frames, caused by the movement of objects or the camera. Pictured are two, super-imposed, consecutive frames taken from the KITTI dataset [24]. The corresponding optical flow is visualized as an array of green arrows. Note that this is just a sample of the motions that optical flow can characterize.

alternatives [14, 34, 51, 68].

A limitation of optical flow is its reliance on a single coherent movement for each pixel and its underlying assumption on brightness constancy, which is difficult to justify for the spectrum of dynamics one encounters in the real world. Examples of dynamics that optical flow fails to capture include flickering, semi-transparent motion, and stochastic dynamics. These are some of the dynamics typically exhibited by dynamic textures. Therefore, optical flow is not a suitable substrate for representing the spectrum of dynamics in dynamic textures.

2.4.2 Marginalized spacetime oriented energies

Between the two extremes lies the representation of dynamics that aims to capture a distribution of measurements of spacetime orientations in the input temporal imagery. Unlike flow-based analyses that focus on the apparent motion (*i.e.*, translation) present in the data, measurements of spacetime orientations take a geometric and generalized approach in capturing *spacetime structures*: oriented structures in the spatiotemporal domain that manifest themselves as motion or non-motion, *e.g.*, flickering, stochastic dynamics, etc.

Previous work [3, 17, 30, 56, 65, 48, 12] has shown that the velocity of image content (*i.e.*, motion) can be interpreted as a 3D oriented structure in the x-y-t spatiotemporal domain. Furthermore, in the frequency domain, the signal energy of these oriented structures lie on a plane through the origin where the slant of the plane is defined by the velocity of the image content. For example, in the case where a spacetime structure is defined by the image velocity $(u, v)^\top$, the unit normal of the plane is given by $\hat{\mathbf{n}} = (u, v, 1)^\top / ||(u, v, 1)^\top||$. Hence, energy models of visual motion, like those presented in these works, are described as "oriented energy" or "motion energy" models, and they attempt to identify this orientation-plane (and hence the pattern's velocity) via a set of image filtering operations. Specifically, given an input image sequence, these models consist of an alternating sequence of linear and non-linear operations that yield a distributed representation (*i.e.*, implicitly coded) of pixelwise optical flow. These models have been motivated and studied in a variety of contexts, including computer vision, visual neuroscience, and visual psychology

Whereas motion indicates a single, dominant orientation in the spatiotemporal and frequency domains, non-motion can indicate an unconstrained, underconstrained, multi-dominant, heterogeneous, or isotropic orientation. For example, an unconstrained orientation corresponds to structure-less imagery (*e.g.*, image of a clear sky) in the spatiotemporal domain and point energy response in the origin in the frequency domain; and a multi-dominant orientation corresponds to multiple, super-imposed spacetime structures in the spatiotemporal domain (*e.g.*, waterfall over a stationary background) and multiple, super-imposed oriented planes in the frequency domain. These, along with the other orientations, are visualized in Fig. 2.5.

This thesis adopts the Marginalized Spacetime Oriented Energy (MSOE) approach of Derpanis and Wildes [12] in representing the observed distribution of dynamics (*i.e.* motion and non-motion) of an input dynamic texture. They conjectured that the constituent spacetime orientations for a spectrum of common visual patterns (*e.g.*, dynamic textures) can serve as a basis for describing the temporal variation of an image sequence. They successfully applied their model for the task of dynamic texture recognition; here it is used for the task of dynamic texture synthesis. Significantly, a completely analytically-defined oriented energy ConvNet model provides the current state-of-the-art for the related task of dynamic texture recognition [28]. The proposed two-stream architecture adopts the MSOE model by encoding it as a ConvNet that serves as the representation of observed dynamics of input dynamic textures—the dynamics stream. The same computational steps are used, however, the handcrafted filters of the MSOE model are not used and are learned instead so that they are better tuned to deal with the noise distributions encountered in natural imagery. The construction of the ConvNet is discussed in the next chapter and the MSOE model is reviewed here.

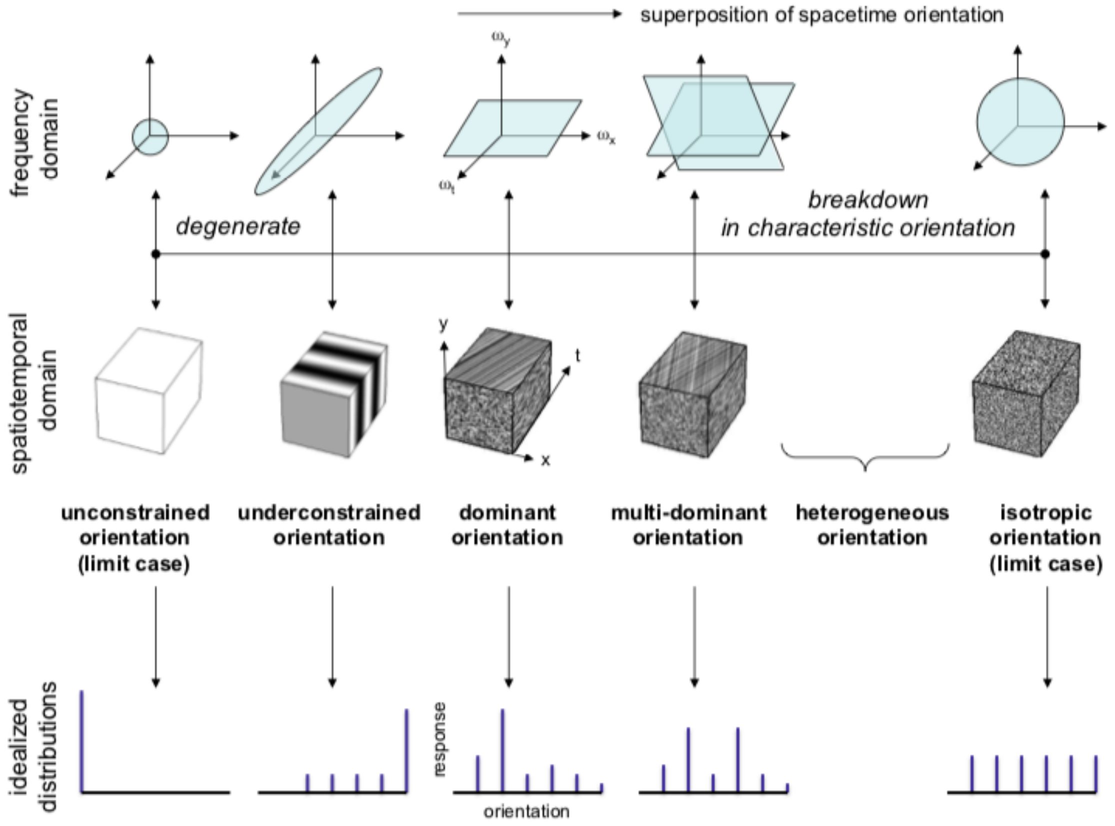

superposition of spacetime orientation
frequency domain
ω_y
ω_x
ω_t
degenerate
breakdown in characteristic orientation
spatiotemporal domain
y
t
x
unconstrained orientation (limit case)
underconstrained orientation
dominant orientation
multi-dominant orientation
heterogeneous orientation
isotropic orientation (limit case)
idealized distributions
response
orientation

Figure 2.5: Dynamic texture spectrum (figure from Derpanis and Wildes [12], Copyright © 2012, IEEE). The top and middle rows depict prototypical dynamic textures in the frequency and spatiotemporal domains, respectively. From left-to-right, an increasing amount of spacetime structures are superimposed in a texture. The bottom row depicts a seven bin histogram of the relative spacetime-oriented structure (or lack thereof) present in each dynamic texture. The first histogram bin captures lack of structure. The remaining histogram bins from left-to-right correspond to spacetime orientations selective for static, rightward motion, upward motion, leftward motion, downward motion and flicker structure.

Given input temporal imagery, $\mathbf{I} \in \mathbb{R}^{T \times H \times W}$ (time × height × width), a bank of oriented 3D filters, *e.g.*, Gaussian third derivative filters $G_3 \in \mathbb{R}^{T \times H \times W}$, which are sensitive to a range of spatiotemporal orientations, are each applied:

$$E_{\hat{\theta}} = G_{3_{\hat{\theta}}} * \mathbf{I} \ , \tag{2.7}$$

where $*$ denotes convolution, and $G_{3_{\hat{\theta}}}$ is a Gaussian third derivative filter oriented in the direction of the 3D unit vector $\hat{\theta}$ which lies along the filter's symmetry axis. Each of these filtering operations results in a spacetime volume of filter responses, $E_{\hat{\theta}}$. These filter responses are then rectified (squared) and pooled over local spacetime regions to make the responses robust to the phase of the input signal, *i.e.*, robust to the alignment of the filter with the underlying image structure:

$$\bar{E}_{\hat{\theta}} = \sum_{(x,y,t) \in \Omega} E_{\hat{\theta}}(x,y,t)^2 \ . \tag{2.8}$$

At this point, each oriented energy measurement includes measurement of spatial orientation. This means that spatial image structures will affect the responses of the bank of oriented filters, making them dependent on spatial appearance. This dependence is unwanted as it can occur at an otherwise coherent dynamic region, *e.g.*, a surface with varying spatial appearance exhibiting a single, dominant motion. Thus, a description consisting purely of pattern dynamics is sought. To remove this difficulty, the spatial orientation component of each filter is discounted via "marginalization". Specifically, filter responses consistent with the same temporal orientation (not necessarily the same spatial orientation), $\hat{\theta}_i$, are summed:

$$E_{\hat{\mathbf{n}}} = \sum_{i=1}^{N} E_{\hat{\theta}_i} \ , \tag{2.9}$$

where $\hat{\mathbf{n}}$ denotes the unit normal of the plane in frequency-space that the spacetime structures captured by these filters lie upon (implicitly describing a single temporal orientation), and N denotes the number of these filters. These responses provide a pixelwise distributed measure of which spacetime structures (discounting spatial information) are present in the input. However, these responses are confounded by local image contrast that makes it difficult to determine whether a high response is indicative of the presence of a spacetime structure or simply due to high image contrast. To address this ambiguity, an L_1 normalization is applied across oriented filter responses which results in a representation that is robust to local appearance variations but highly selective to spacetime orientation:

$$\hat{E}_{\hat{\mathbf{n}}_i} = \frac{E_{\hat{\mathbf{n}}_i}}{\sum_{j=1}^{M} E_{\hat{\mathbf{n}}_j} + \epsilon} \ , \tag{2.10}$$

where $\hat{E}_{\hat{\mathbf{n}}_i}$ denotes an oriented filter response from Eq. 2.9 corresponding to a plane in frequency-space with unit normal $\hat{\mathbf{n}}_i$, and ϵ is a small value (*e.g.*, 1e$-$12) to avoid division by zero.

Chapter 3

Technical approach

The proposed two-stream approach consists of an appearance stream, representing the static (texture) appearance of each frame, and a dynamics stream, representing temporal variations between frames. Each stream consists of a ConvNet whose activation statistics are used to characterize the dynamic texture. Synthesizing a dynamic texture is formulated as an optimization problem with the objective of matching activation statistics between the target and synthesized textures. The dynamic texture synthesis approach is summarized in Fig. 3.1 and the individual pieces are described in turn in the following sections.

3.1 Texture model: Appearance stream

The appearance stream follows the static texture model introduced by Gatys *et al.* [21] which was summarized in the previous chapter (Sec. 2.2.1). To briefly review, the key idea is that correlations between activation maps (*i.e.*, normalized Gram matrices) in a ConvNet trained for object recognition (*e.g.*, VGG-19 [58]) capture

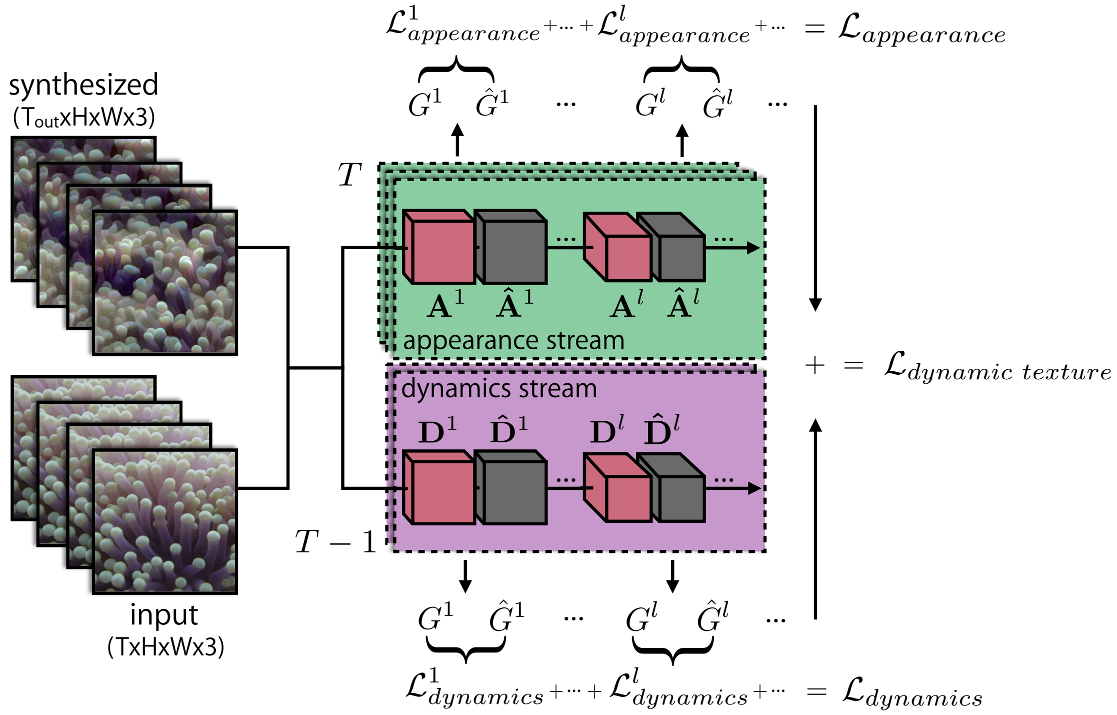

$\mathcal{L}^1_{appearance} + \cdots + \mathcal{L}^l_{appearance} + \cdots = \mathcal{L}_{appearance}$
G^1 $\hat{G}^1$ ... G^l $\hat{G}^l$ ...
synthesized
(ToutxHxWx3)
T
$\mathbf{A}^1$ $\hat{\mathbf{A}}^1$ $\mathbf{A}^l$ $\hat{\mathbf{A}}^l$
appearance stream
$+ = \mathcal{L}_{dynamic\ texture}$
dynamics stream
$\mathbf{D}^1$ $\hat{\mathbf{D}}^1$ $\mathbf{D}^l$ $\hat{\mathbf{D}}^l$
$T-1$
input
(TxHxWx3)
G^1 $\hat{G}^1$ ... G^l $\hat{G}^l$ ...
$\mathcal{L}^1_{dynamics} + \cdots + \mathcal{L}^l_{dynamics} + \cdots = \mathcal{L}_{dynamics}$

Figure 3.1: Two-stream dynamic texture generation. Two sets of Gram matrices represent a dynamic texture's appearance and dynamics. Matching these statistics allows for the generation of novel textures as well as style transfer between textures. Here, G^l and $\hat{G}^l$ are the Gram matrices of activations A^l and $\hat{A}^l$ (or D^l and $\hat{D}^l$) corresponding to the target and synthesized sequence, respectively, computed at layer l of the appearance stream (or dynamics stream) and averaged over time T (or $T-1$). $\mathcal{L}^l_{\text{appearance}}$ is the appearance loss at layer l, computed as the squared Frobenius norm between G^l and $\hat{G}^l$ from the appearance stream. Similarly, $\mathcal{L}^l_{\text{dynamics}}$ is the dynamics loss at layer l for the dynamics stream. By summing each loss computed at various layers, we arrive at $\mathcal{L}_{\text{appearance}}$ and $\mathcal{L}_{\text{dynamics}}$, which, when summed, form the combined dynamic texture loss, $\mathcal{L}_{\text{dynamic texture}}$, that is to be minimized.

texture appearance. The same publicly available normalized VGG-19 ConvNet [58] used by Gatys *et al.* [21] is used here. The proposed appearance stream utilizes this model by simply applying it indepedently to each frame of the synthesized and target texture.

3.1.1 Target texture appearance

To capture the appearance of an input dynamic texture, an initial forward pass through VGG-19 is performed with each frame of the image sequence to compute the feature activations (filter responses), $\mathbf{A}^{lt} \in \mathbb{R}^{N_l \times M_l}$, for various levels in the network, where N_l and M_l denote the number of feature activations and the number of spatial locations of layer l at time t, respectively. The auto-correlations of the filter responses in a particular layer are averaged over the frames and encapsulated by a Gram matrix, $\mathbf{G}^l \in \mathbb{R}^{N_l \times N_l}$, whose entries are given by:

$$G^l_{ij} = \frac{1}{TN_lM_l} \sum_{t=1}^{T} \sum_{k=1}^{M_l} A^{lt}_{ik} A^{lt}_{jk} \ , \tag{3.1}$$

where T denotes the number of input frames and A^{lt}_{ik} denotes the activation of feature i at location k in layer l on the target frame t.

3.1.2 Synthesized texture appearance

The synthesized texture appearance is similarly represented by a Gram matrix, $\hat{\mathbf{G}}^{lt} \in \mathbb{R}^{N_l \times N_l}$, whose activations are given by:

$$\hat{G}^{lt}_{ij} = \frac{1}{N_lM_l} \sum_{k=1}^{M_l} \hat{A}^{lt}_{ik} \hat{A}^{lt}_{jk} \ , \tag{3.2}$$

where $\hat{A}^{lt}_{ik}$ denotes the activation of feature i at location k in layer l on the synthesized frame t.

3.1.3 Appearance loss

The appearance loss, $\mathcal{L}_{\text{appearance}}$, is defined as the temporal average of the mean squared error between the Gram matrix of the input texture and that of the synthesized texture computed at each frame:

$$\mathcal{L}_{\text{appearance}} = \frac{1}{L_{\text{app}} T_{\text{out}}} \sum_{t=1}^{T_{\text{out}}} \sum_{l} \|\mathbf{G}^l - \hat{\mathbf{G}}^{lt}\|_F^2 \ , \tag{3.3}$$

where L_{app} is the number of VGG-19 layers used to compute Gram matrices, T_{out} is the number of frames being generated in the output, and $\|\cdot\|_F$ is the Frobenius norm. Consistent with previous work [21], Gram matrices are computed on the following layers: *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4*. Through systematic evaluation, Gatys *et al.* [21] reported these layers were qualitatively superior to other layer subsets for static texture synthesis. Furthermore, they reported that only a subset of layers are required to synthesize textures that are almost indistinguishable from the input texture, so long as the chosen layers span a wide spectrum of receptive fields (*i.e.*, using early, mid, and later layers of the network).

3.2 Texture model: Dynamics stream

Parallel to the appearance stream ConvNet is a ConvNet designed for capturing texture dynamics. There are three primary goals in designing this dynamics stream.

1. The activations of the ConvNet should represent the temporal variation of the input pattern.

2. The activations should be largely invariant to the appearance (*i.e.*, spatial content) of the images (which should be characterized by the appearance stream described above).

3. The dynamics representation should be differentiable to enable synthesis via a ConvNet.

By analogy to the appearance stream, an obvious choice is a ConvNet architecture suited for computing optical flow (*e.g.*, [14, 34]) which is naturally differentiable. However, with most such models it is unclear how invariant their layers are to appearance. Instead, a novel network architecture is proposed which is motivated by the spacetime-oriented energy model [12, 56].

3.2.1 Review: Marginalized spacetime oriented energies

This section is only intended to briefly review the aspects of the Marginalized Spacetime Oriented Energies (MSOE) model [12] that are most relevant to the following section; a more thorough overview can be found in the previous chapter (Sec. 2.4.2).

A significant limitation of optical flow is its reliance on a single coherent movement for each pixel and its underlying assumption on brightness constancy, which only partially describes the dynamics one may encounter in the real world, and thus in dynamic textures. In response, Derpanis and Wildes [12] showed that the constituent spacetime orientations for a spectrum of common visual patterns can serve as a basis for describing the temporal variation of an image sequence. Their observation motivated a motion oriented-energy approach to representing dynamics, rather than a flow-based approach. By constructing a set of 3D oriented filters designed to capture spacetime structures beyond just translational motion, they demonstrated a successful application of the approach for dynamic texture recognition [12].

Motion energy models may form an ideal basis for the dynamics stream of the proposed dynamic texture synthesis ConvNet. As such, the MSOE model proposed by Derpanis and Wildes [12] is used to motivate the network architecture.

3.2.2 ConvNet architecture

Using this model as the basis, the following convolutional network is proposed. The ConvNet input is a pair of temporally consecutive greyscale images, $\mathbf{I} \in \mathbb{R}^{T\times H\times W\times C}$ (time × height × width × channels), where $C = 1$ and $T = 2$. From here forth, the channel dimension (C) will be omitted for simplicity. Each input pair is first normalized to have zero-mean and unit variance (*i.e.*, contrast normalization or "instance normalization" [63]), as follows:

$$\mathbf{I}_N = \frac{\mathbf{I} - \mu}{\sigma + \eta} \ , \tag{3.4}$$

where μ is the average pixel value of the input pair, σ is the standard deviation of the input pair, and η is a small value (1e−12) to prevent dividing by zero. This step provides a level of invariance to overall brightness and contrast (*i.e.*, global additive and multiplicative signal variations) as well as eases the training process of the ConvNet [40]. The first layer consists of a 3D convolution over the normalized input pair with a bank of 32 3D filters of size $2 \times 11 \times 11$ (time × height × width), resulting in an output of spacetime oriented energy measurements:

$$E_F(\mathbf{x}) = F * \mathbf{I}_N(\mathbf{x}) \ , \tag{3.5}$$

where E_F denotes the response of filter F (of size $2 \times 11 \times 11$) after a convolution, $*$, centered about $\mathbf{x} \equiv (t, x, y)$. In handcrafted approaches (*e.g.*, [12]), a bank of oriented 3D Gaussian third derivative filters is often used, which require only 10 orientations as a spanning basis. Moreover, these filters typically exceed a temporal extent of $T = 2$ to capture a wider range of temporal frequencies. Here, however, an overcomplete bank of 32 *learned* 3D filters with a temporal extent of $T = 2$ is used. This approach is taken for two reasons. First, 3D Gaussian third derivative filters are one of the many types of filters one can use for measuring oriented energy (*e.g.*, one can use 3D Gaussian fourth derivative, Gabor, lognormal, or causal-time filters [11]), so it is not required to restrict the network to a smaller filter bank. Specifically, since these filters are being learned, it is within the capacity of the network to learn a wide variety of oriented filters beyond 3D Gaussian third derivative filters. Second, due to GPU memory limitations, the temporal extent of filters are limited to the temporal extent common for optical flow groundtruth imagery, $T = 2$. Although this restriction limits the range of temporal frequencies that can be captured in dynamic textures, it is still effective in enabling dynamic texture synthesis with dynamic textures spanning a wide range of dynamics, as shown in the next chapter.

After computing spacetime oriented energy measurements, a squaring activation function and 5×5 spatial max-pooling (with a stride of one) is applied to make the responses robust to local signal phase:

$$\bar{E}_F(\mathbf{x}) = \max_{i \in \Omega}\{E_F(i)^2\} \ , \tag{3.6}$$

where Ω is a 5×5 spatial neighbourhood centered about $\mathbf{x}$. A 2D convolution follows with 64 filters of size 1×1 that combines energy measurements that are consistent with the same frequency domain plane:

$$E_G(\mathbf{x}) = G * \bar{E}_F(\mathbf{x}) \ , \tag{3.7}$$

where E_G denotes the response of filter G (of size 1×1) after a convolution. Finally, to remove local contrast dependence, an L_1 divisive normalization is applied to each spatial location:

$$\bar{E}_G(\mathbf{x}) = \frac{E_G(\mathbf{x})}{\|E_G(\mathbf{x})\|_1 + \epsilon} \ , \tag{3.8}$$

where $\| \cdot \|_1$ is the L_1 norm computed over the filter responses of all filters and ϵ is a small value (1e$-$12) to prevent dividing by zero.

To capture spacetime orientations beyond those capable with the limited receptive fields used in the initial layer, a five-level spatial Gaussian pyramid is computed. Each pyramid level is processed independently with the same spacetime-oriented energy model and then bilinearly upsampled to the original resolution and concatenated:

$$E(\mathbf{x}) = \left(\bar{E}_G(\mathbf{x}), \bar{E}_G(\mathbf{x}_{\downarrow\times 2})_{\uparrow\times 2}, \ldots, \bar{E}_G(\mathbf{x}_{\downarrow\times 2^{k-1}})_{\uparrow\times 2^{k-1}}, \bar{E}_G(\mathbf{x}_{\downarrow\times 2^{k}})_{\uparrow\times 2^{k}}\right) \ , \tag{3.9}$$

where $(\cdot)_{\downarrow\times n}$ and $(\cdot)_{\uparrow\times n}$ denote an n-times downsample and upsample, respectively,

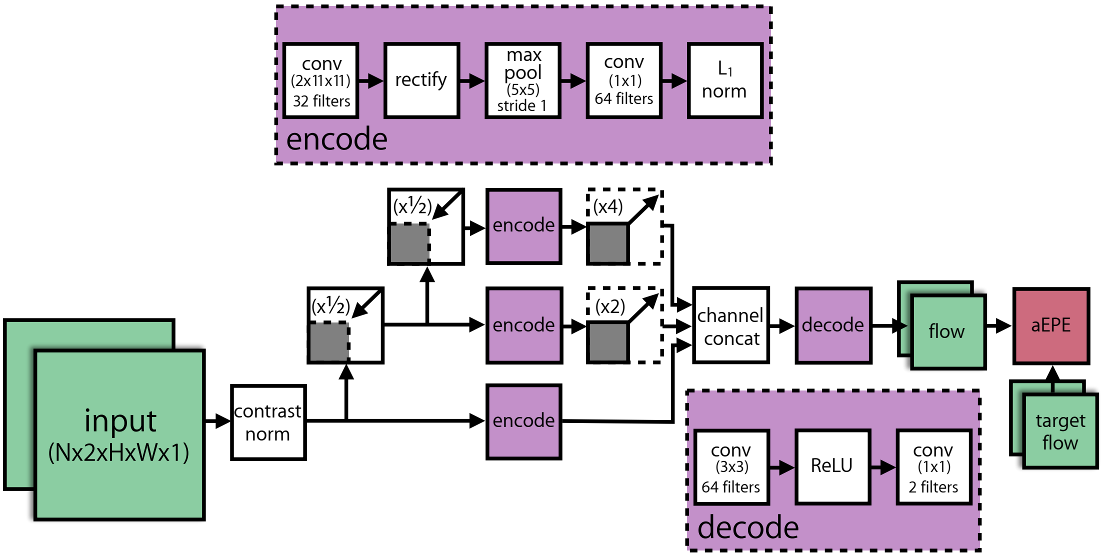


Figure 3.2: Dynamics stream ConvNet. The ConvNet is based on a spacetime-oriented energy model [12, 56] and is trained for optical flow regression. Three scales are shown for illustration; in practice five scales are used.

and $k+1$ denotes the number of pyramid levels. This final output of the dynamics encoding stage is named the "concatenation layer".

Training

Prior energy model instantiations (*e.g.*, [3, 12, 56]) used handcrafted filter weights. While a similar approach could be followed here, instead the weights are learned so that they better deal with the noise distributions encountered in natural imagery. To train the network weights, additional decoding layers are added that take the concatenated distributed representation from the concatenation layer and apply a 3×3 convolution (with 64 filters), ReLU activation, and a 1×1 convolution (with 2 filters) that yields a two channel output encoding the optical flow directly. The proposed architecture is illustrated in Fig. 3.2.

For training, the standard average endpoint error (aEPE) flow metric (*i.e.*,

L_2 norm) is used between the predicted flow and the ground truth flow as the loss. Since no large-scale optical flow dataset exists that captures natural imagery with groundtruth flow, the assembly of a new dataset was necessary. Videos from an unlabeled video dataset are fed through an existing flow estimator to produce optical flow groundtruth for training, *cf*. [61]. For the unlabeled video dataset, the UCF101 dataset for action recognition [59] is used as it contains a wide variety of complex movements of natural imagery. The synthetic Flying Chairs dataset [14] was also considered as it contained ground truth optical flow; however, training the dynamics stream on this dataset reduced the overall quality of synthesized dynamic textures. This can be explained by the limited motions and appearances exhibited by the rigid objects in Flying Chairs, which is undesirable for estimating motion of dynamic textures. For producing the optical flow groundtruth, the EpicFlow [52] model is used for its state-of-the-art performance (at the time of experimentation) on optical flow regression.

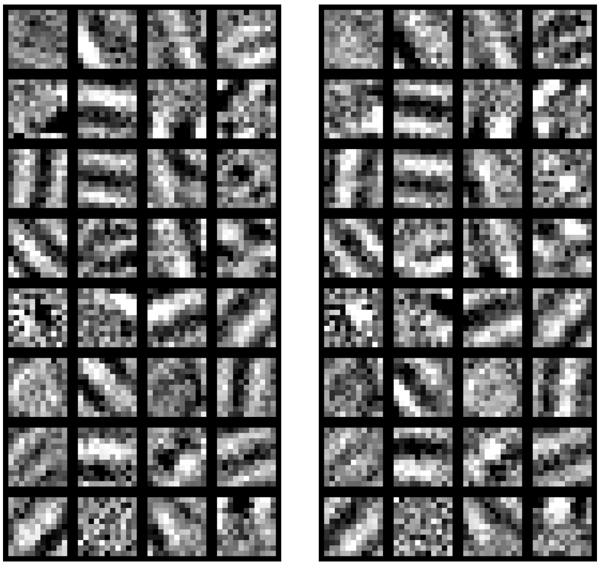
The distribution of movement directions in UCF101 is biased to left-to-right and right-to-left motions, which is undesirable as dynamic textures are not necessarily restricted to certain directions of motion. To combat this dataset bias, geometric data augmentations similar to those used by FlowNet [14] are used to equalize the distribution of movement directions in the generated dataset. Additionally, photometric data augmentations similar to those used by FlowNet [14] are used here as well. These augmentations include an image rotation with a rotation amount uniformly sampled from the range $[-180°, 180°]$; left-right and up-down flipping with a 50% chance; additive gaussian noise with a sigma uniformly sampled from the range $[0, 0.04 * 255]$; gamma correction with a gamma value uniformly sampled from the range $[0.7, 1.5]$; additive brightness change with the additive value normally sampled from the distribution $\mathcal{N}(\mu = 0, \sigma = 0.2 * 255)$; and a multiplicative brightness change with the multiplicative value uniformly sampled from the range $[0.2, 1.4]$. Each of these augmentations are done in random order. The aEPE loss is optimized using Adam [37].

Optical flow is chosen as the proxy task (as opposed to dynamic texture recognition [12]) for learning the multiscale distributed representation of dynamics because of the ease of obtaining large amounts of optical flow groundtruth for training. Although optical flow is not a suitable representation of the dynamics in dynamic textures, evidence suggests that it is sufficient enough to induce the encoding stage to learn the MSOE model within its representational capacity. For example, inspection of the learned filters in the initial layer of the encoding stage showed evidence of spacetime-oriented filters, consistent with the handcrafted filters used in previous work [12]. This point is illustrated in Fig. 3.3. Furthermore, and shown in the next chapter (Fig. 4.4), there is evidence that the learned representation of dynamics is largely invariant to appearance, another indication that an MSOE model has been learned.

3.2.3 Target texture dynamics

Similar to the appearance stream, filter response correlations in a particular layer of the dynamics stream are averaged over the number of image frame pairs and encapsulated by a Gram matrix, $\mathbf{G}^l \in \mathbb{R}^{N_l \times N_l}$, whose entries are given by:

$$G^l_{ij} = \frac{1}{(T-1)N_l M_l} \sum_{t=1}^{T-1} \sum_{k=1}^{M_l} D^{l(t,\ t+1)}_{ik} D^{l(t,\ t+1)}_{jk} \ , \tag{3.10}$$



(a) Frame 1 (b) Frame 2

Figure 3.3: Learned spatiotemporal filters in the first layer of the dynamics stream. (a) and (b) each depict a temporal slice of the learned filters ($2 \times 11 \times 11$), operating on the first and second frame of an input pair, respectively. Inspection of the learned filters reveals structures consistent with the handcrafted temporal derivative filters used in previous work [12] (*e.g.*, row 3, col 1 captures rightward movement and row 8, col 1 captures down-right movement).

where $D^{l(t,\ t+1)}_{ik}$ denotes the activation of feature i at location k in layer l on the target frames t and $t+1$.

3.2.4 Synthesized texture dynamics

The dynamics of the synthesized texture is represented by a Gram matrix of filter response correlations computed separately for each pair of frames, $\hat{\mathbf{G}}^{l(t,\ t+1)} \in \mathbb{R}^{N_l \times N_l}$, with entries:

$$\hat{G}^{l(t,\ t+1)}_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} \hat{D}^{l(t,\ t+1)}_{ik} \hat{D}^{l(t,\ t+1)}_{jk} \ , \tag{3.11}$$

where $\hat{D}^{l(t,\ t+1)}_{ik}$ denotes the activation of feature i at location k in layer l on the synthesized frames t and $t+1$.

3.2.5 Dynamics loss

The dynamics loss, $\mathcal{L}_{\text{dynamics}}$, is defined as the average of the mean squared error between the Gram matrices of the input texture and those of the generated texture:

$$\mathcal{L}_{\text{dynamics}} = \frac{1}{L_{\text{dyn}}(T_{\text{out}} - 1)} \sum_{t=1}^{T_{\text{out}}-1} \sum_{l} \|\mathbf{G}^l - \hat{\mathbf{G}}^{l(t,\ t+1)}\|_F^2 \ , \tag{3.12}$$

where L_{dyn} is the number of ConvNet layers being used in the dynamics stream to compute Gram matrices.

The Gram matrix is computed on the output of the concatenation layer, where the multiscale distributed representation of orientations is stored. While it is tempting to use the predicted optical flow output from the network's decoder stage, this generally yields poor results as shown in the evaluation. Due to the complex, temporal variation present in dynamic textures, they contain a variety of local spacetime orientations rather than a single dominant orientation. As a result, the flow estimates will tend to be an average of the underlying orientation measurements and consequently not descriptive. A comparison between the texture synthesis results using the concatenation layer and the predicted flow output is provided in Chapter 4.

3.3 Dynamic texture synthesis

The overall dynamic texture loss consists of the combination of the appearance loss, Eq. (3.3), and the dynamics loss, Eq. (3.12):

$$\mathcal{L}_{\text{dynamic texture}} = \alpha \mathcal{L}_{\text{appearance}} + \beta \mathcal{L}_{\text{dynamics}} \ , \tag{3.13}$$

where $\alpha = 1\text{e}9$ and $\beta = 1\text{e}15$ are the weighting factors for the appearance and dynamics content, respectively. Dynamic textures are implicitly defined as the (local) minima of this loss. Textures are generated by optimizing Eq. (3.13) with respect to the synthesized spacetime volume, *i.e.*, the pixels of the video. Variations in the resulting texture are found by initializing the optimization process using IID Gaussian noise. Consistent with previous work [21], L-BFGS [42] is used for optimization. Dynamic texture synthesis results are provided in Chapter 4.

3.3.1 Incremental texture synthesis

Naive application of the outlined approach will consume increasing amounts of memory (in this case, GPU memory) used by the ConvNet as the temporal extent (*i.e.*, number of frames) of the dynamic texture grows; this fact makes it impractical to process and generate longer sequences. Instead, long sequences can be incrementally generated by separating the sequence into subsequences and optimizing them sequentially. Fig. 3.4 shows a visualization of the incremental texture synthesis process.

This process is realized by initializing the first frame of a subsequence as the last frame from the previous subsequence and keeping it fixed throughout the optimization. The remaining frames of the subsequence are initialized randomly and optimized as above. This approach ensures temporal consistency across synthesized subsequences and can be viewed as a form of coordinate descent for the full sequence objective. Specifically, each subsequence can be viewed as a coordinate/direction of the full sequence objective that is to be minimized over, while keeping the other coordinates (*i.e.*, other subsequences) fixed. This approach can also be viewed as a form of non-linear autoregression where the output variable (in this case, the current subsequence) non-linearly depends on its previous values (previously synthesized subsequence) and a stochastic term (randomly initialized frames of the current subsequence). The flexibility of this framework allows other texture generation problems to be handled simply by altering the initialization of frames and controlling which frames or frame regions are updated.

3.3.2 Temporally-endless texture synthesis

An interesting extension that was explored were dynamic textures where there is no discernible temporal seam between the last and first frames. Played as a loop, these textures appear to be temporally endless. This is trivially achieved by

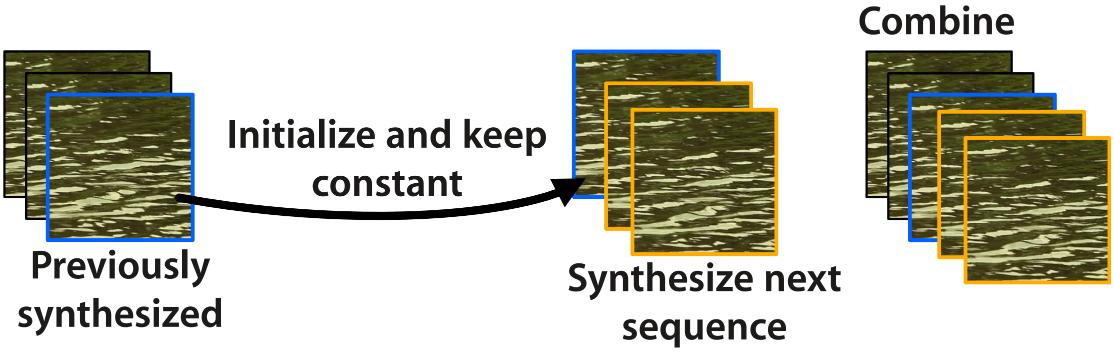


Figure 3.4: Incremental texture synthesis. Long sequences can be incrementally generated by separating the sequence into subsequences and optimizing them sequentially. The last frame (blue border) of a previously synthesized subsequence (left) is used to initialize the first frame of the next subsequence (middle) and is kept fixed throughout optimization. The remaining frames of the subsequence (yellow border) are initialized randomly and optimized as per usual. Finally, the two subsequences are combined to produce a longer sequence (right).

adding an additional term to the dynamics loss (Eq. 3.12) that ties the last frame to the first:

$$\mathcal{L}_{\text{dynamics}} = \frac{1}{L_{\text{dyn}} T_{\text{out}}} \left(\sum_{t=1}^{T_{\text{out}}-1} \sum_{l} \|\mathbf{G}^{l} - \hat{\mathbf{G}}^{l(t,\ t+1)}\|_F^2 + \sum_{l} \|\mathbf{G}^{l} - \hat{\mathbf{G}}^{l(T_{\text{out}},\ 1)}\|_F^2 \right). \tag{3.14}$$

3.3.3 Dynamics style transfer

The underlying assumption of the proposed model is that the appearance and dynamics of dynamic textures can be factorized. As such, it should allow for the transfer of the dynamics of one texture onto the appearance of another. This operation can be likened to “style transfer”, where style, in this case, is characterized by dynamics. Style transfer has previously been explored for static imagery, called image style transfer [22, 7, 23]. Dynamics style transfer is accomplished with the proposed model by performing the same optimization as described in Sec. 3.3, but with the target Gram matrices for appearance and dynamics computed from two different textures, respectively.

3.4 Summary

This chapter presented the technical approach of the proposed ConvNet for dynamic texture synthesis, including its various extensions. The ConvNet implemented a factored approach in separately modelling the appearance (*i.e.*, spatial content) and dynamics information (*i.e.*, temporal variation) of an input dynamic texture. It consisted of two main components: an appearance stream and a dynamics stream. Each stream consisted of a ConvNet whose activation correlations (*i.e.*, Gram matrices) were used to characterize the dynamic texture. The appearance stream was an adaptation of the static texture model introduced by Gatys *et al.* [21]. Here it was utilized by applying it independently to each frame of the dynamic texture, arriving at a purely appearance-wise description of it.

Parallel to the appearance stream is the dynamics stream, which was designed for purely capturing texture dynamics while forgoing appearance. Dynamic textures exhibit complex temporal variations that can not be adequately modelled by optical flow alone. Thus, a novel ConvNet was proposed, taking inspiration from spacetime-oriented energy models [12, 56] that can characterize temporal imagery beyond the capabilities of optical flow. Namely, the Marginalized Spatiotemporal Oriented Energies (MSOE) model introduced by Derpanis and Wildes [12] was adapted to a ConvNet, which was trained through the proxy task of predicting optical flow.

Dynamic textures were synthesized by optimizing each stream's objective with respect to the synthesized texture. Both streams followed the objective of minimizing the distances between activation correlations of the synthesized dynamic texture and target dynamic texture.

Finally, this chapter presented three extensions to the proposed model: incremental texture synthesis, temporally-endless texture synthesis, and dynamics style transfer. Incremental texture synthesis provided a solution to memory-constraints brought upon by the synthesis of long dynamic textures by introducing an incremental synthesis process over subsequences of the entire dynamic texture. Temporally-endless texture synthesis included an additional constraint on the dynamics objective that tied the last and first frame of the synthesized dynamic texture, producing dynamic textures that appeared to be endlessly looping. Dynamics style transfer took advantage of the two-stream factorization of appearance and dynamics to synthesize dynamic textures that combine the texture appearance from one target with the dynamics from another.

Chapter 4

Evaluation

The goal of (dynamic) texture synthesis is to generate samples that are indistinguishable from the real input target texture by a human observer. In this chapter, a variety of synthesis results are presented, including qualitative comparisons with extant methods and a user study to quantitatively evaluate the realism of the synthesized results. Given their temporal nature, the results are best viewed as videos, which are available on the project page: `ryersonvisionlab.github.io/two-stream-projpage`. The two-stream architecture was implemented using TensorFlow [1], an open source machine learning framework. Results were synthesized using an NVIDIA Titan X (Pascal) GPU and synthesis times ranged between one to three hours (corresponding to $6,000$ optimization iterations) to generate 12 frames with an image resolution of 256×256. The parameters of the dynamics stream were set as follows: $\alpha = 1\text{e}9$, $\beta = 1\text{e}15$, $\epsilon = 1\text{e}{-12}$, and $\eta = 1\text{e}{-12}$. For the full synthesis results and source code, please refer to the supplemental material available on the project page.

4.1 Qualitative results

In this section, a qualitative analysis is performed on the tasks of dynamic texture synthesis, incremental texture synthesis, temporally-endless texture synthesis, and dynamics style transfer.

4.1.1 Dynamic texture synthesis

The dynamic texture synthesis process was applied to a wide range of textures selected from the DynTex [49] database and others that were collected in-the-wild. The collected textures include ones that adhere to the texture assumptions of this thesis (*i.e.*, spatiotemporal homogeneity) and ones that do not. Included in the supplemental material are synthesized results of nearly 60 different textures that encapsulate a range of phenomena, such as flowing water, waves, clouds, fire, rippling flags, waving plants, and schools of fish. Some sample frames are shown in Fig. 4.1 and Fig. 4.2 but readers are encouraged to view the videos to fully appreciate the results.

Figure 4.1 and Fig. 4.2 show some example success cases with the two-stream method where appearance and dynamics characteristics from the target are reliably preserved in the synthesized result. For example, the upward fiery dynamics and flickering appearance of `fireplace_1`, the outward dynamics of the explosive splash of `lava`, the wispy fluid dynamics of `smoke_1`, the flowing vegetation in `underwater_vegetation_1`, and the downward rippling flow of `water_3`, are all captured in the synthesized results.

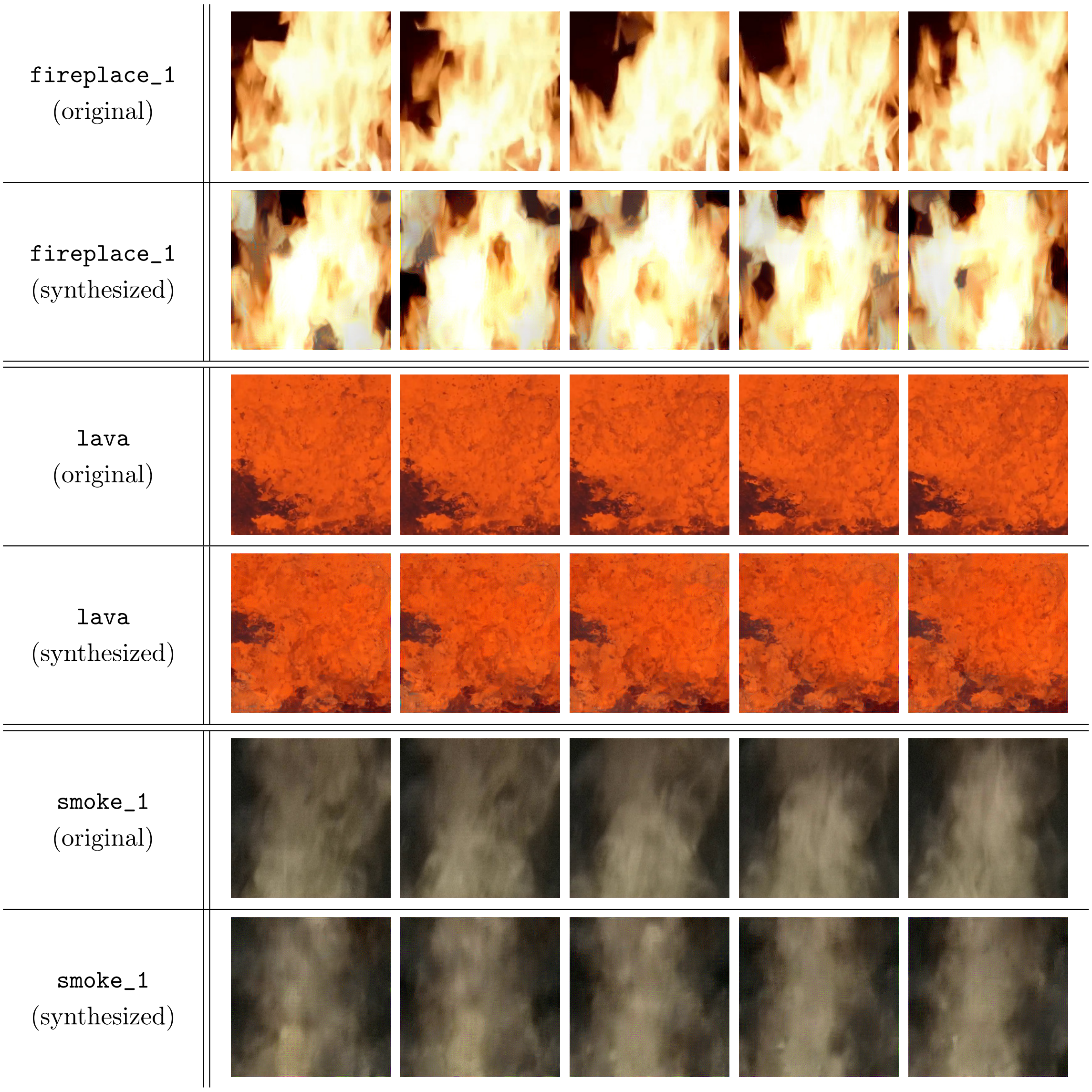


Figure 4.1: Dynamic texture synthesis success examples. Names correspond to files in the supplemental material.

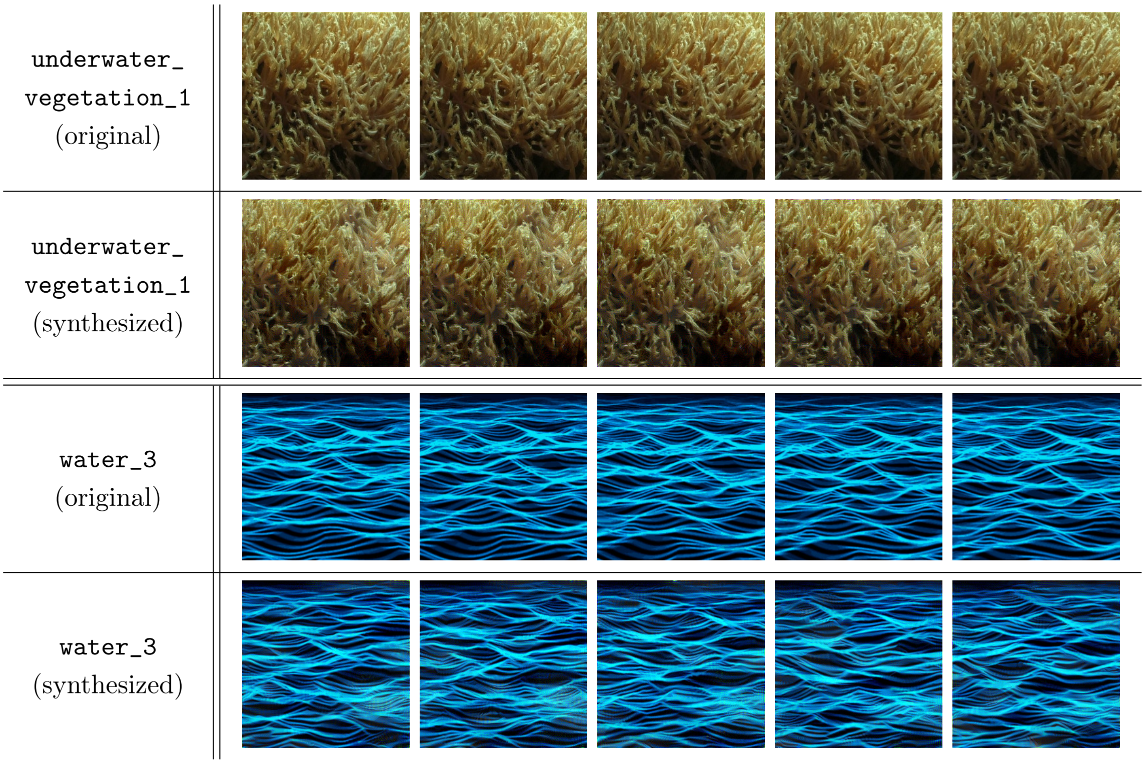

underwater_ vegetation_1 (original)
underwater_ vegetation_1 (synthesized)
water_3 (original)
water_3 (synthesized)

Figure 4.2: Dynamic texture synthesis success examples. Names correspond to files in the supplemental material.

Failure modes

Example failure modes of the two-stream method are presented in Fig. 4.3. In general, most failures result from inputs that violate the underlying assumption of a dynamic texture, *i.e.*, the appearance and/or dynamics are not spatiotemporally homogeneous. In the case of the `escalator` example, the long edge structures in the appearance are not spatially homogeneous, and the dynamics vary due to perspective effects that change the motion from purely downward to downward and outward. The resulting synthesized texture captures an overall downward motion but lacks the perspective effects and is unable to consistently reproduce the long edge structures. This is consistent with previous observations on static texture synthesis [21] and suggests it is a limitation of the Gram matrix representation used in the appearance stream.

Another example is the `flag` sequence where the rippling dynamics are relatively homogeneous across the pattern but the appearance varies spatially. As expected, the synthesized texture does not faithfully reproduce the appearance; however, it does exhibit plausible rippling dynamics.

Also shown in Fig. 4.3 is the `cranberries` sequence, which consists of a combination of swirling and wave dynamics. The model faithfully reproduces the appearance but is unable to capture the spatially varying dynamics. Interestingly, it still produces a result which is statistically indistinguishable from real in the user study discussed in Sec. 4.2.

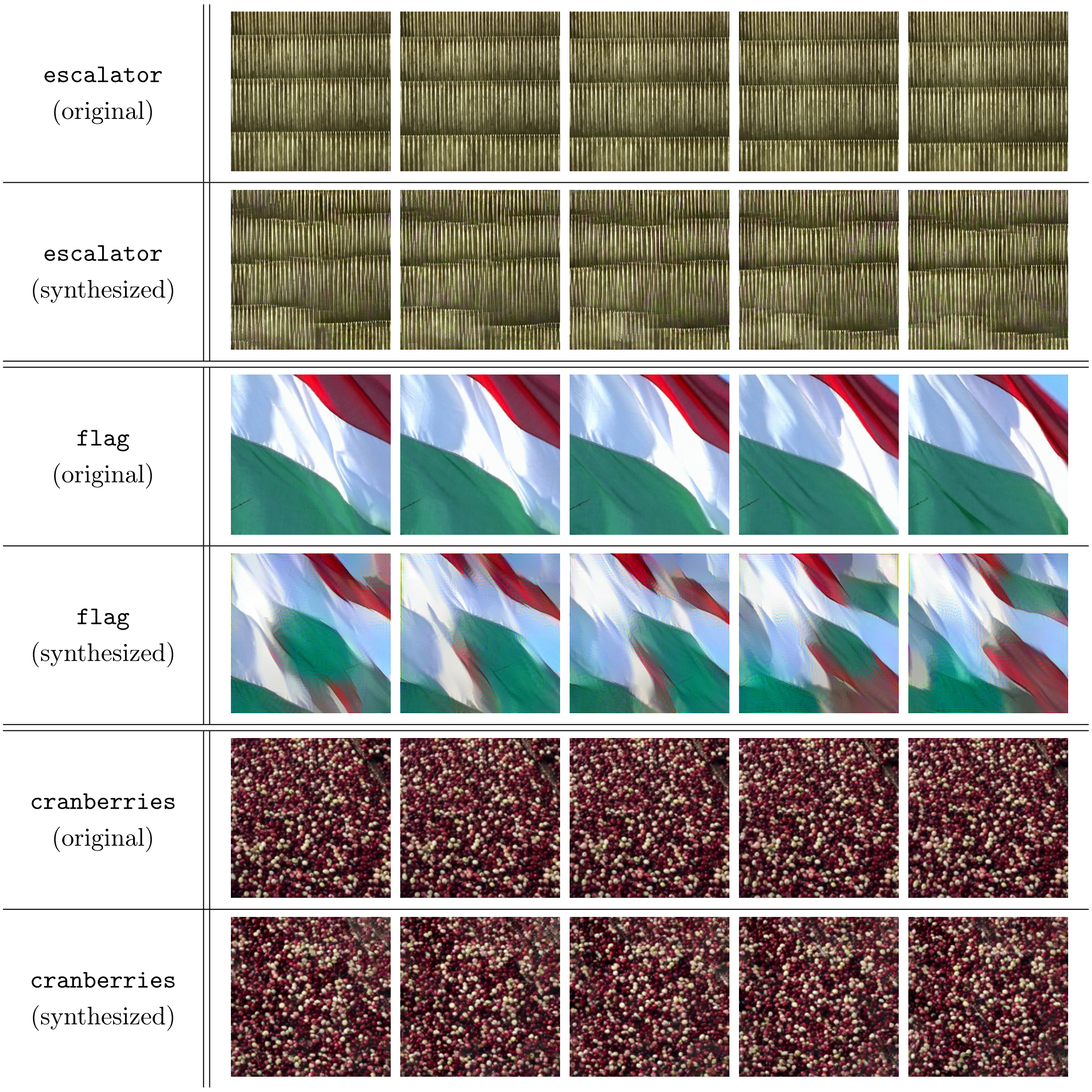


Figure 4.3: Dynamic texture synthesis failure examples. In these cases, the failures are attributed to either the appearance or the dynamics not being homogeneous.

Appearance vs. dynamics streams

Two experiments were conducted to verify that the appearance and dynamics streams were capturing complementary information. To validate that the texture generation of multiple frames would not induce dynamics consistent with the input, frames were synthesized starting from randomly generated noise but only using the appearance statistics and corresponding loss, *i.e.*, Eq. 3.3. As expected, this produced frames that were valid textures but with no coherent dynamics present. To examine the dynamics, see `fish` in the supplemental material.

Similarly, to validate that the dynamics stream did not inadvertently include appearance information, dynamic textures were synthesized using the dynamics loss only, *i.e.*, Eq. 3.12. The resulting frames had no visible appearance and had an extremely low dynamic range, *i.e.*, the standard deviation of pixel intensities was 10 for values in $[0, 255]$. This indicates a general invariance to appearance and suggests that the two-stream dynamic texture representation has factored appearance and dynamics, as desired. Results for a sequence containing a school of fish are shown in Fig. 4.4 with enhanced contrast.

4.1.2 Incremental texture synthesis

Dynamic textures synthesized incrementally, as described in Sec. 3.3.1, are included in the supplemental. Sequences as long as 122 frames (using 11 frame subsequences) were synthesized with no observed divergence or degradation. The resulting textures were perceptually indistinguishable from those synthesized with the typical batch process.

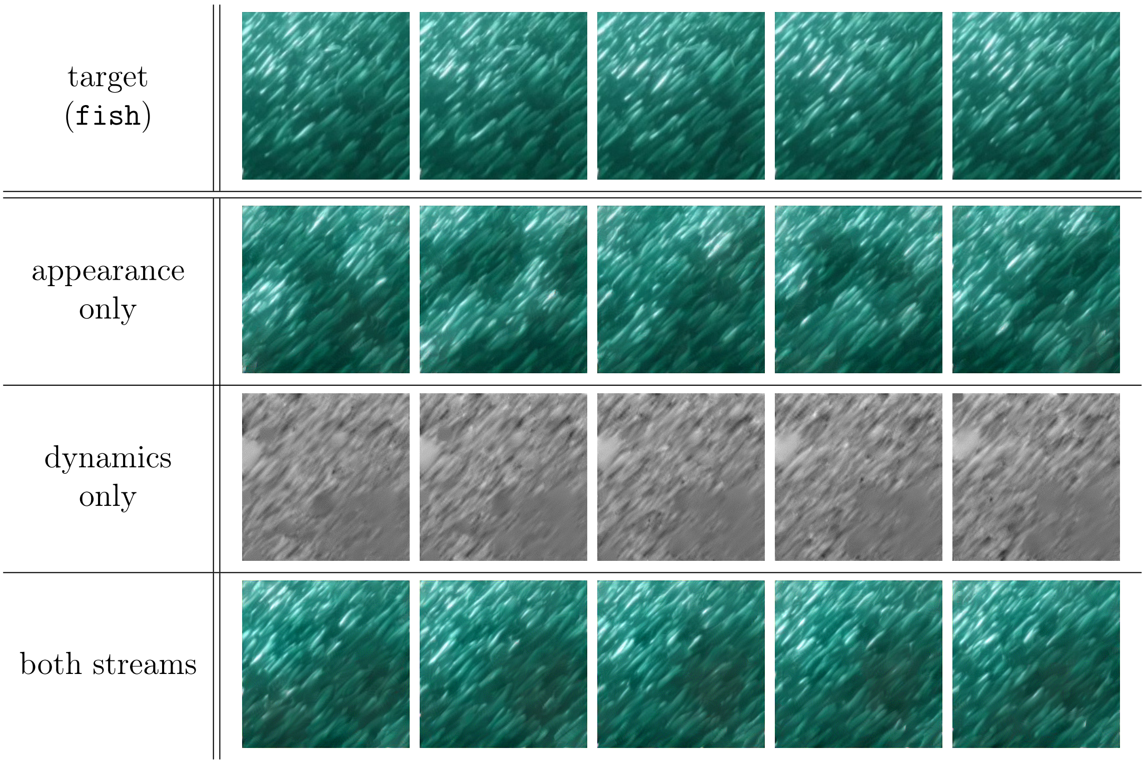


Figure 4.4: Two-stream dynamic texture synthesis versus dynamics-only and appearance-only texture synthesis. (top row) Target dynamic texture. (second row) Texture synthesis without dynamics constraints shows consistent per-frame appearance but no temporal coherence. (third row) Texture synthesis without appearance constraints shows limited per-frame appearance with pixel intensities having a standard deviation of 10. (bottom row) Including both streams induces consistent appearance and dynamics.

4.1.3 Temporally-endless texture synthesis

An example of a synthesized temporally-endless dynamic texture is shown in Fig. 4.5. As described in Sec. 3.3.2, the dynamic texture appears temporally endless, *i.e.*, there is no apparent temporal discontinuity between the last and first frames.

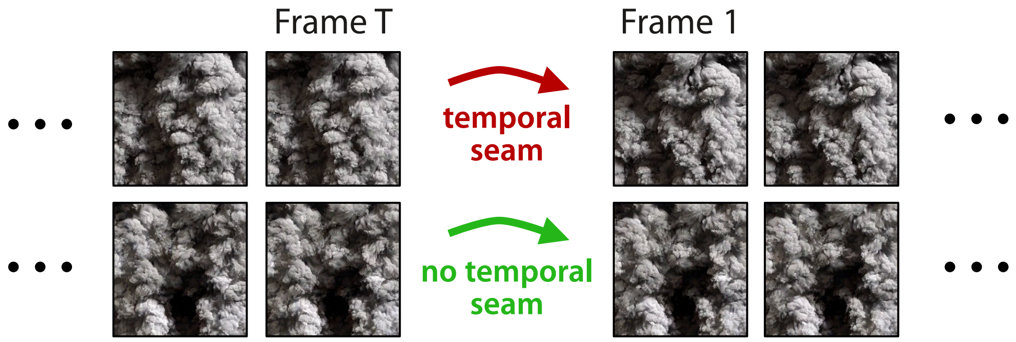


Figure 4.5: Temporally-endless texture synthesis. (top row) Target texture. (bottom row) Synthesized texture. By adding an additional loss to the dynamics stream that ties the last frame to the first, the synthesized dynamic texture appears to be temporally endless. Note the lack of an abrupt appearance change (*i.e.*, temporal seam) between the last frame and the first frame of the synthesized dynamic texture.

4.1.4 Dynamics style transfer

A dynamics style transfer results is shown in Fig. 4.7 (top row), using two real videos as the appearance and dynamics target, respectively. Additional examples are available in the supplemental material. When performing dynamics style transfer it is important that the appearance structure of both targets to be similar in scale and semantics, otherwise, the synthesized dynamic textures will look unnatural. For instance, transferring the dynamics of a flame onto a water scene will generally produce implausible results (Fig. 4.6).

The dynamics of a texture can also be applied to a static input image, as the target Gram matrices for the appearance loss can be computed on just a single frame. This allows us to effectively animate regions of a static image. The result of this process can be striking and is visualized in Fig. 4.7 (second, third, and bottom rows), where the appearance is taken from a painting and the dynamics from a real world video.

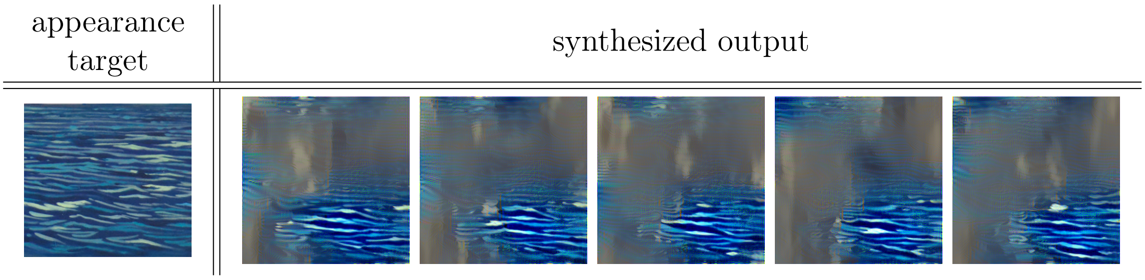


Figure 4.6: Dynamics style transfer with incompatible appearance and dynamics targets. The dynamics from `fireplace_1` are unsuccessfully transferred to a painting of ocean water.

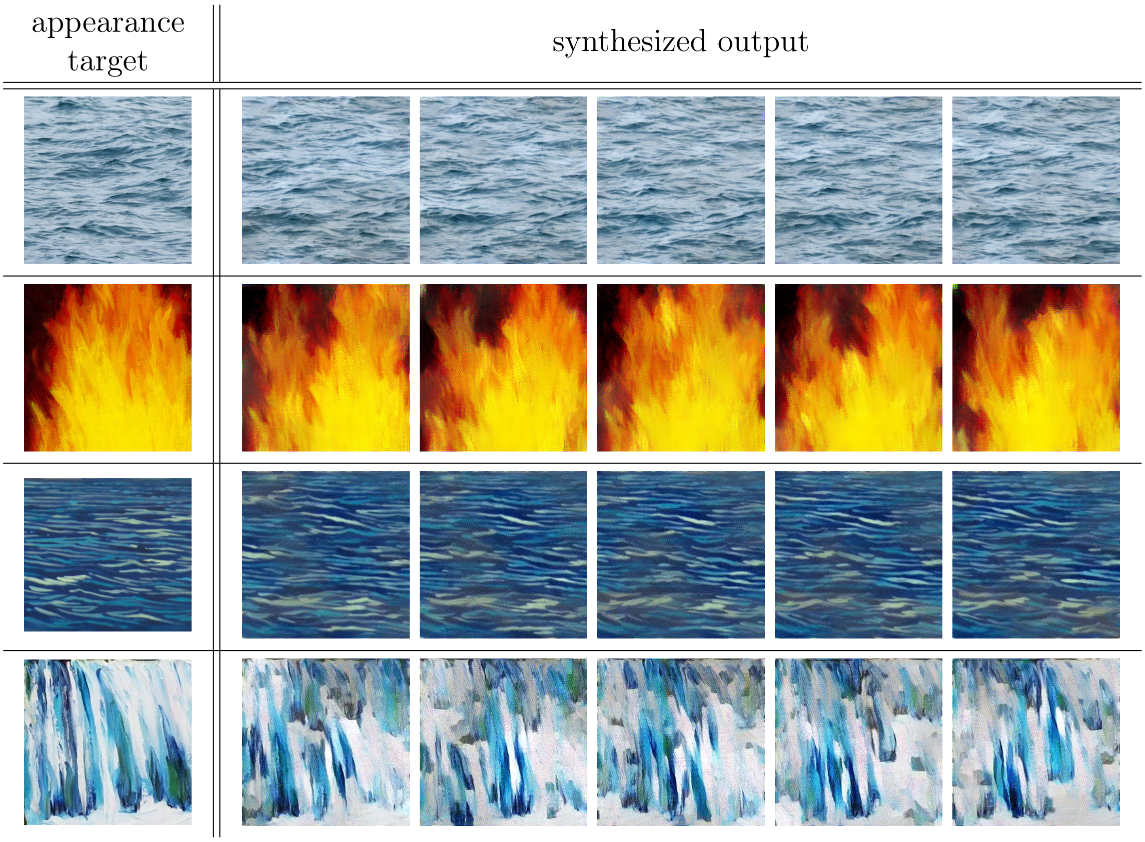


Figure 4.7: Dynamics style transfer. (top row) Appearance of still water was used with the dynamics of a different water dynamic texture (`water_4`). (second row) The appearance of a painting of fire was used with the dynamics of a real fire (`fireplace_1`). (third row) The appearance of a painting of ocean water was used with the dynamics of a water dynamic texture (`water_4`). (bottom row) The appearance of a painting of a waterfall was used with the dynamics of a waterfall dynamic texture (`waterfall`). Animated results and additional examples are available in the supplemental material.

4.2 User study

Quantitative evaluation for (dynamic) texture synthesis is a particularly challenging task as there is no single correct output when synthesizing new samples of a texture. Like in other image generation tasks (*e.g.*, rendering), human perception is ultimately the most important measure. Thus, a user study was performed to evaluate the perceived realism of the synthesized dynamic textures.

Similar to previous image synthesis work (*e.g.*, [8]), a perceptual experiment was conducted with human observers to quantitatively evaluate the synthesis results. A two-way alternative forced-choice (2AFC) evaluation was employed on Amazon Mechanical Turk (AMT) with 200 different users. Each user performed 59 pairwise comparisons between a synthesized dynamic texture and its target. Users were asked to choose which dynamic texture appeared more realistic after viewing the textures independently (with a brief delay between viewing each texture) for an exposure time sampled randomly from discrete intervals between 0.3 and 4.8 seconds. Measures were taken to control the experimental conditions and minimize the possibility of low quality data. Appendix A.1 provides further experimental details of the user study.

For comparison, a baseline was constructed by using the flow decode layer in the dynamics loss of Eq. 3.12. This approach corresponds with attempting to mimic the optical flow statistics of the texture directly. Textures were synthesized with this model and the user study was repeated with an additional 200 users. To differentiate between the models, “Flow decode layer” and “Concat layer” are labelled in the figures to describe the baseline and final model, respectively. An example comparing a dynamic texture synthesized on the baseline and final model

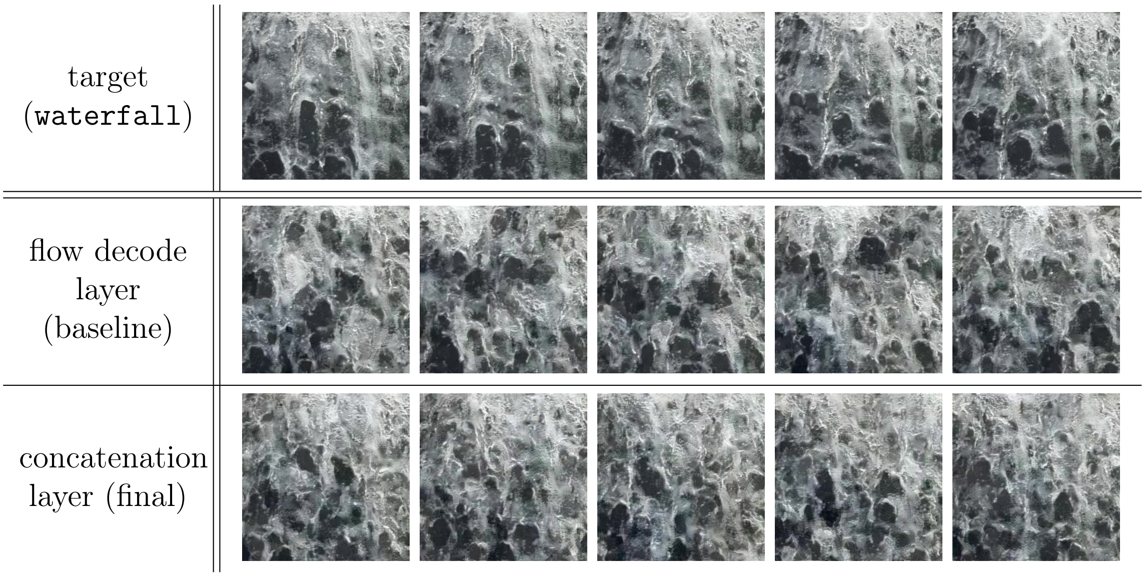


Figure 4.8: Comparison with a dynamic texture synthesized using optical-flow directly. (top row) Target dynamic texture. (middle row) Dynamic texture synthesis when using the "Flow decode layer" on the dynamics stream. This is the baseline model and corresponds to attempting to mimic the optical flow statistics of the texture directly. The dynamics of the waterfall are poorly captured, lacking the downward motion exhibited by the target. (bottom row) Dynamic texture synthesis when using the "Concat layer" on the dynamics stream. This is the final model. The downward motion and overall dynamics of the target are reliably captured.

is shown in Fig. 4.8.

The results of this study are summarized in Fig. 4.9 which shows user accuracy in differentiating real versus synthesized textures as a function of time for both methods. Accuracies are reported with a 95% confidence level, *i.e.*, a margin of error that is between ±1.96 standard deviations from the mean (p-value of 0.05). Overall, users are able to correctly identify the real texture 66.1%±2.5% of the time for brief exposures of 0.3 seconds. This rises to 79.6% ± 1.1% with exposures of 1.2 seconds and higher. Note that "perfect" synthesis results would have an accuracy of 50%, indicating that users were unable to differentiate between the real and synthesized textures and higher accuracy indicating less convincing textures.

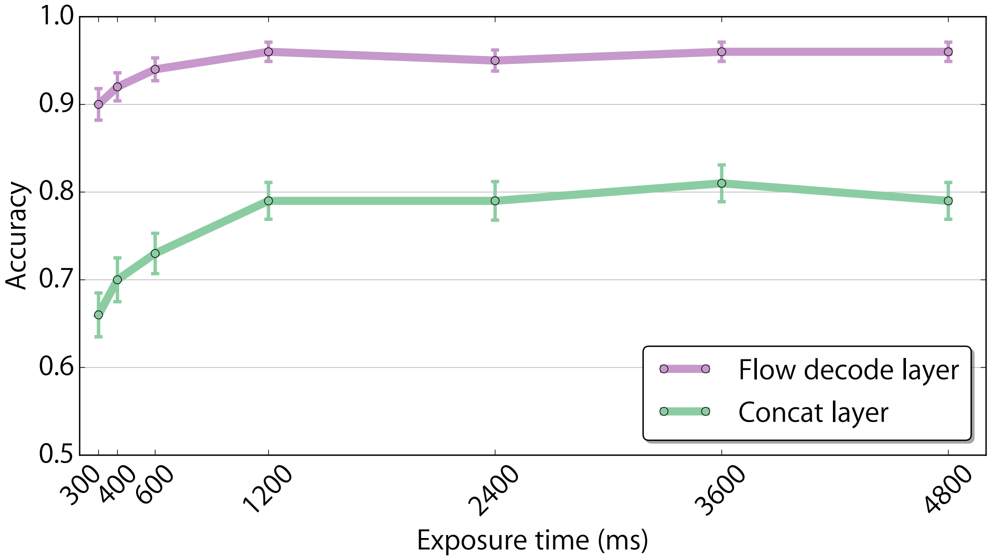


Figure 4.9: Time-limited pairwise comparisons across all textures with 95% statistical confidence intervals.

The results clearly show that the use of the concatenation layer's activations is far more effective than the flow decode layer. This is not surprising, as optical flow alone is known to be unreliable on many textures, particularly those with translucent and/or chaotic dynamics (*e.g.*, water, smoke, flames, etc.). Specifically, its assumptions of a single coherent movement for each pixel and brightness constancy are violated. Also evident in these results is the time-dependant nature of perception for textures from both models. Users' ability to identify the synthesized texture improved as exposure times increased to 1.2 seconds and remained relatively flat for longer exposures.

To better understand the performance of the proposed approach, the results were grouped and analyzed in terms of appearance and dynamics characteristics. For appearance, the taxonomy presented in [41] was used to group textures as either regular/near-regular (*e.g.*, periodic tiling and brick wall), irregular (*e.g.*, a field of flowers), or stochastic/near-stochastic (*e.g.*, tv static or water). For dynamics, the textures were grouped as either spatially-consistent (*e.g.*, closeup of rippling sea water) or spatially-inconsistent (*e.g.*, rippling sea water viewed at an angle, causing inconsistent dynamics due to perspective distortion). The finer-grained dynamics taxonomy presented by Derpanis and Wildes [12] was not used here in its entirety because certain categories of dynamics were under-represented in the database of collected dynamic textures, *e.g.*, unconstrained and underconstrained dynamics. Furthermore, there were dynamic textures that were difficult to categorize under their taxonomy (*i.e.*, dynamic textures with inconsistent dynamics). To maintain statistically-meaningful results, the dominant, multi-dominant, heterogeneous, and isotropic dynamics groups of Derpanis and Wildes [12] were merged into the spatially-consistent group, covering roughly half of the collected dynamic textures, and the spatially-inconsistent group was created to cover the rest. Results based on these groupings can be seen in Fig. 4.10.

A full breakdown of the user study results by dynamic texture and grouping can be found in Appendix A.3. Here some of the overall trends are discussed.

Appearance-based analysis

Based on appearance it is clear that textures with large-scale spatial consistencies (regular, near-regular, and irregular textures) tend to perform poorly. Examples being `flag` and `fountain_2` with user accuracies of 98.9%±1.6% and 90.8%±4.3% averaged across all exposures, respectively. This result is not unexpected and is a fundamental limitation of the local nature of the Gram matrix representation used in the appearance stream which was observed in static texture synthesis [21]. In contrast, stochastic and near-stochastic dynamic textures performed sig-

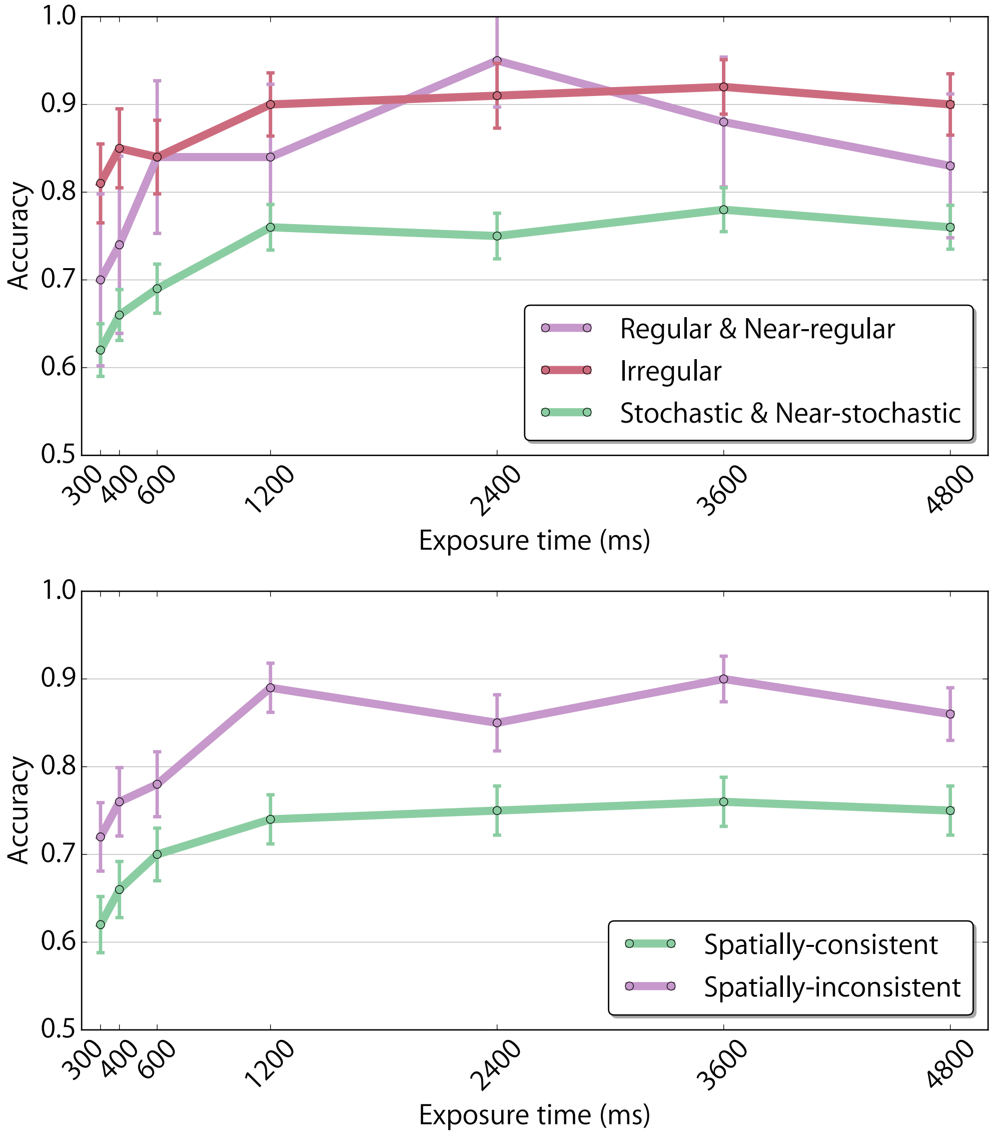


Figure 4.10: Time-limited pairwise comparisons across all textures, grouped by appearance (top) and dynamics (bottom). Shown with 95% statistical confidence intervals.

nificantly better as their smaller-scale local variations are well captured by the appearance stream, for instance `water_1` and `lava` which had average accuracies of 53.8% ± 7.4% and 55.6% ± 7.4%, respectively, making them both statistically indistinguishable from real.

Dynamics-based analysis

In terms of dynamics, the user study showed that textures with spatially-consistent dynamics (*e.g.*, `tv_static`, `water_*`, and `calm_water_*`) perform significantly better than those with spatially-inconsistent dynamics (*e.g.*, `candle_flame`, `fountain_2`, and `snake_*`), where the dynamics drastically differ across spatial locations. For example, `tv_static` and `calm_water_6` have average accuracies of 48.6% ± 7.4% and 63.2% ± 7.2%, respectively, while `candle_flame` and `snake_5` have average accuracies of 92.4% ± 4% and 92.1% ± 4%, respectively. Overall, the two-stream model is capable of reproducing a full spectrum of spatially-consistent dynamics. However, as the appearance shifts from containing small-scale spatial consistencies to containing large-scale spatial consistencies, performance degrades. This pattern was evident in the user study where the best-performing textures typically consisted of a stochastic or near-stochastic appearance with spatially-consistent dynamics. In contrast, the worst-performing textures consisted of regular, near-regular, or irregular appearance with spatially-inconsistent dynamics.

4.3 Qualitative comparisons

In this section, a qualitative comparison with the extant methods of Funke *et al.* [20] and Xie *et al.* [67] is performed. Generally, results from the proposed two-stream model are found to be qualitatively comparable or better than these methods.

To note, Funke *et al.* provided results on only five textures and of those only four are dynamic textures in the sense that their appearance and dynamics are spatiotemporally coherent. Their results on these sequences (`cranberries`, `flames`,

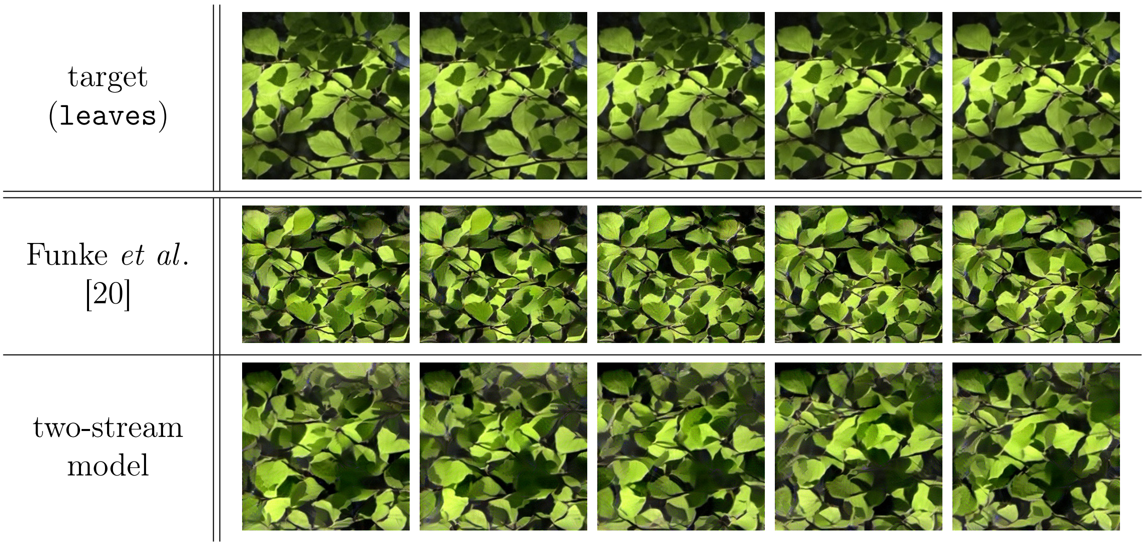


Figure 4.11: Qualitative comparison with Funke *et al.*'s [20] model on one of the sequences used in their work. Note that this sequence does not follow the thesis' assumption of a dynamic texture, in the sense that the appearance and/or dynamics are not spatiotemporally homogeneous. (top row) Target sequence. (middle row) Dynamic texture synthesis when using Funke *et al.*'s model. The model fails to capture the up-right motion of the leaves. (bottom row) Dynamic texture synthesis when using the proposed two-stream model. The up-right motion of the leaves is captured. Results are best viewed in video.

`leaves`, and `water_5`) are included in the folder `funke` under `dynamic_texture_synthesis/comparisons` in the supplementary material. The results from the two-stream model are included as well. An example on the `leaves` sequence is shown in Fig. 4.11.

Results are also compared on nine dynamic textures chosen to cover the full range of the dynamics and appearance groupings introduced in the user study. Publicly available code from Funke *et al.* and Xie *et al.* is used to produce their results with their default parameter settings. For Funke *et al.*'s model, the parameters used are $\Delta t = 4$ and $T = 12$ (recall that target dynamic textures consist of 12 frames). For the spatiotemporal and temporal models from Xie *et al.*, the parameters used are $T = 1200$ and $\tilde{M} = 3$. A comparison between the results from the proposed two-stream model, Funke *et al.*'s model, and Xie *et al.*'s model on the nine dynamic textures are included in the folder `xie_and_funke` under `dynamic_texture_synthesis/comparisons`. An example on the `smoke_plume_1` dynamic texture is shown in Fig. 4.12.

Note for Xie *et al.*, comparisons are made with their spatiotemporal model, labelled "Xie et al. (ST)", designed for dynamic textures with both spatial and temporal homogeneity, and their temporal model, labelled "Xie et al. (FC)", designed for dynamic textures with only temporal homogeneity.

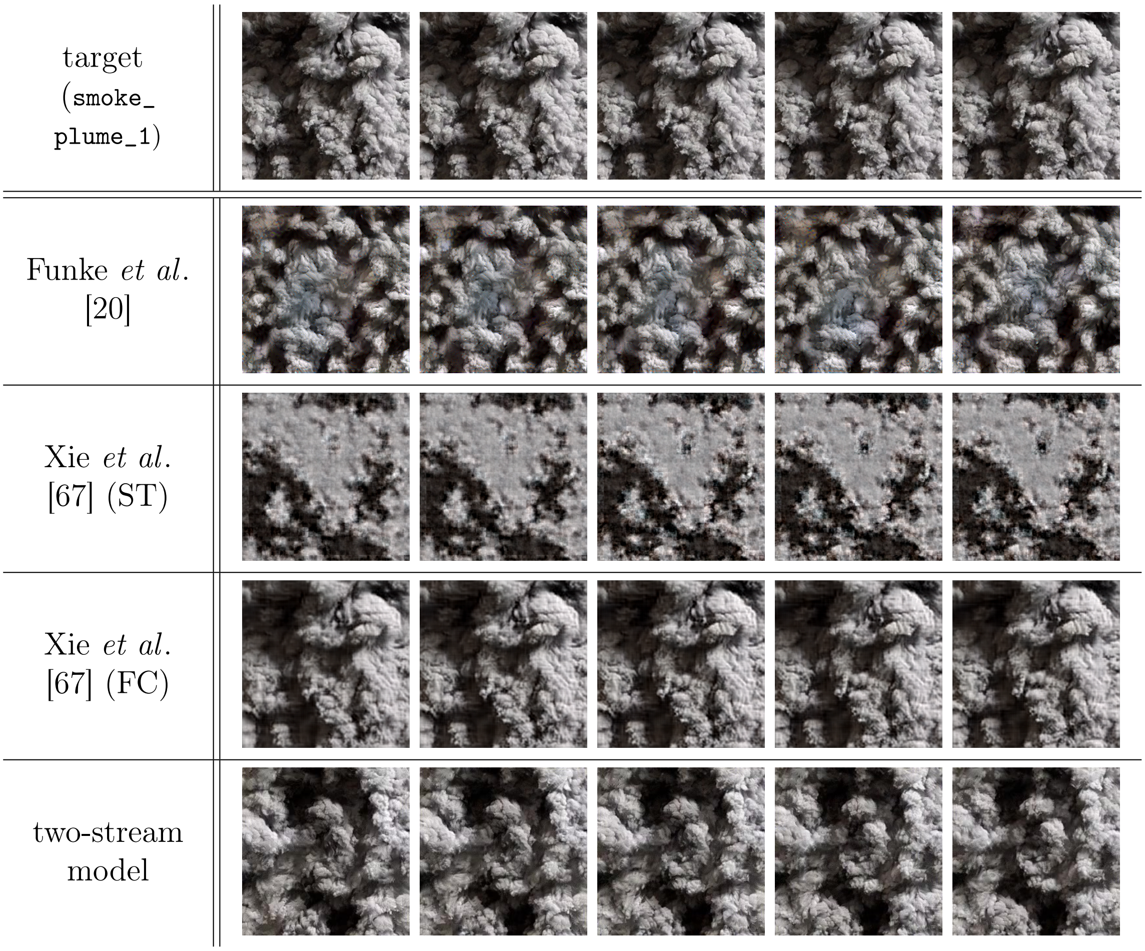


Figure 4.12: Qualitative comparison with Funke *et al.*'s [20] and Xie *et al.*'s [67] models on one of the dynamic textures collected in this thesis. (top row) Target dynamic texture. (second row) Dynamic texture synthesis when using Funke *et al.*'s model. (third row) Dynamic texture synthesis when using Xie *et al.*'s spatiotemporal model. (fourth row) Dynamic texture synthesis when using Xie *et al.*'s temporal model. (bottom row) Dynamic texture synthesis when using the proposed two-stream model.

Overall, the results from the proposed two-stream model appear qualitatively better, showing more temporal coherence and similarity in dynamics as well as fewer spatial and temporal artifacts, such as blur and flicker, respectively. This may be a natural consequence of the limited representation of dynamics in both Funke *et al.*'s and Xie *et al.*'s models. Although the spatiotemporal model of Xie *et al.* [67] is able to synthesize dynamic textures that lack spatial homogeneity (*e.g.*, `bamboo` and `escalator`), note that their method appears to not be able to synthesize novel dynamic textures, *i.e.*, it appears to faithfully reproduce the target texture, reducing the generalizability (*e.g.*, synthesis of textures beyond the spatiotemporal extent of the input) of their approach, and thus its applicability.

As a consequence of jointly modelling appearance and dynamics, both methods [20, 67] are not capable of the novel form of style transfer that was demonstrated above. This capability was enabled by the factored representation of dynamics and appearance. Furthermore, the spatiotemporal extent of the output sequence generated by Xie *et al.*'s [67] method is limited to being equal to the input. The proposed approach does not share this limitation.

4.4 Discussion

This chapter provided a qualitative analysis on a variety of results obtained using the proposed two-stream model for dynamic texture synthesis, incremental texture synthesis, temporally-endless texture synthesis, and dynamics style transfer. For dynamic texture synthesis, it was shown that both the appearance and dynamics streams were required to synthesize dynamic textures that matched both the framewise appearance of the target and its dynamics. For incremental texture synthesis, it was shown that incrementally synthesized dynamic textures exhibited no divergence or degradation when compared to those synthesized using the typical batch process. For temporally-endless texture synthesis, it was shown that the synthesized dynamic textures exhibited no apparent temporal discontinuity between the last and first frames. For dynamics style transfer, it was shown that the dynamics of one dynamic texture can be successfully combined with the appearance of another, under the stipulation that the appearance and dynamics targets came from textures similar in scale and semantics.

Additionally, a large-scale user study was performed to quantitatively evaluate the realism of dynamic textures synthesized by the proposed model by comparing synthesized results with their respective targets. An evaluation on a baseline version of the model was performed as well. Overall, dynamic textures synthesized by the two-stream model were able to fool users $33.9\% \pm 2.5\%$ of the time for brief exposures. In contrast, dynamic textures synthesized by the baseline model were able to fool users $10.2\% \pm 1.8\%$ of the time. Although the final two-stream model was statistically better than the baseline model, users were still able to reliably tell the difference between real and synthesized for longer exposure times. Specifically, $79.6\% \pm 1.1\%$ of the time, they could distinguish real from synthesized when exposure times were 1.2 seconds or higher. It is possible that for longer exposure times, spatial artifacts (*e.g.*, chromatic aberration and noise) of the synthesized texture may have served as cues for distinguishing between real and synthesized. These artifacts suggest room for improvement on the representation of appearance and has been left for future work.

Finally, a qualitative comparison with two extant methods for dynamic texture synthesis was performed. Overall, the results from the proposed two-stream model were shown to be qualitatively better than the compared methods, showing more temporal coherence and similarity in dynamics as well as fewer spatial and temporal artifacts. Furthermore, it is unclear how the other methods can be applied to the novel dynamics style transfer, as can the proposed two-stream model.

Chapter 5

Conclusion

5.1 Thesis summary

This thesis presented a novel, two-stream analysis of dynamic textures using ConvNets to independently represent appearance and dynamics statistics, culminating in four primary contributions to the dynamic texture literature spanning both theory and application.

First, theoretical insight into the characterization of dynamic textures is provided by building a novel factored representation of both appearance and dynamics. Second, for the representation of dynamics, a novel ConvNet based on the "marginalized" spacetime-oriented energy model of Derpanis and Wildes [12] was constructed. It was shown to provide a substantial improvement on the temporal coherence of synthesized dynamic textures when compared to using a dynamics representation based purely on optical flow. Third, a novel form of style transfer was demonstrated, where the appearance and dynamics information from different texture sources are combined to produce a compelling composition of the two.

This capability was shown to be enabled by the factored two-stream representation of appearance and dynamics. Finally, the model was applied to a variety of dynamic texture synthesis tasks and it was shown that, so long as the input textures followed the thesis' assumptions of a dynamic texture, *i.e.*, have spatially invariant statistics and spatiotemporally invariant dynamics, the resulting synthesized textures were compelling. This point was validated both qualitatively and quantitatively through a large user study and comparisons with extant methods for dynamic texture synthesis.

Beyond the theoretical implications of this thesis lie numerous applications in the creative-industry including, but not limited to, computer-generated imagery, digital painting, and image editing. More broadly, the ability to animate static imagery via dynamics style transfer can meaningfully contribute to the emerging artistic medium of computer-generated art.

5.2 Future work

Consequently, a few limitations were revealed; however, in light of the implications of some of the experiments (*e.g.*, dynamics style transfer), potential avenues for artistic exploration have been revealed as well. These have been left as directions for future work. This section will first describe the aforementioned limitations and propose possible solutions, then it will outline some interesting potential artistic applications as well as some extensions to the model to enable these applications.

First, much like has been reported in recent image style transfer work [22], results in this thesis show that high frequency noise and chromatic aberrations are a problem in generation. Another issue that arises is when the model fails to capture textures with spatially-varying appearance, (*e.g.*, `flag` in Fig. 4.3) and spatially-inconsistent dynamics (*e.g.*, `escalator` in Fig. 4.3). By collapsing the local statistics into a Gram matrix, the spatial and temporal pattern organization is lost. Simple post-processing methods, *e.g.*, blurring, may alleviate issues with noise and chromatic aberration, but holistically, these appearance-based issues point to a need for a better representation of appearance. To preserve spatial structure, Berger *et al.* [6] experimented with incorporating long range consistency in ConvNet-based texture synthesis by computing multiple Gram matrices at a layer instead of one. Specifically, rather than computing correlations between activations at a single spatial position, correlations between activations across spatial positions were computed as well. Although this approach seems promising, it is worth noting that it is computationally expensive due to the additional Gram computations.

The multiscale distributed representation of dynamics in the dynamics stream was implicitly learned through the proxy task of optical flow regression. Although optical flow was sufficient to produce spacetime-oriented filters, it is unclear how its limitations in modelling complex dynamics affects the dynamics modelling capacity of the learned filters. The abundance of optical flow groundtruth was a motivating factor for its usage; however, for future work, it would be interesting to compare with other, seemingly more suitable, proxy tasks, such as dynamic texture recognition [12].

The proposed two-stream model used a learned approach for instantiating the filter weights of the MSOE dynamics representation [12]. Although most of the synthesized dynamic textures appeared to be perceptually similar to their targets, it remains to be seen how the learned dynamics representation compares to a handcrafted dynamics representation [12]. An advantage of a handcrafted approach is having analytically-defined oriented filters that do not need to be learned through a possibly noisy proxy task. It is not clear if using handcrafted filters would improve results, however, it is worth comparing against in future work. Likewise, interesting comparisons could be made with using a handcrafted, rather than learned, appearance stream representation via spatially-oriented energy filters, *e.g.*, 2D Gaussian n-th derivative oriented filters.

The user study quantitatively compared the final two-stream model with a baseline model. Although a qualitative comparison was made with previously proposed approaches for dynamic texture synthesis [20, 67], the user study did not include a comparison with these approaches. The addition of a quantitative comparison between the two-stream model and the extant approaches could further illustrate the advantages of the two-stream model that were outlined in the qualitative comparison. Additionally, it would be interesting to quantitatively compare with the handcrafted approach [12] mentioned in the previous paragraph.

Due to GPU memory limitations, the temporal extent of the learned spacetime-oriented filters in the first layer of the dynamics stream was restricted to $T = 2$. In spite of this limitation, the two-stream model still managed to synthesize impressive results. However, a small temporal extent limits the range of temporal frequencies that can be captured, thus limiting the range of dynamics that can be modelled. For future work, it would be interesting to investigate the relationship between the quality of synthesized dynamic textures and the temporal extent of the filters in the first layer of the dynamics stream.

ConvNet-based texture synthesis models use a Euclidean metric (*i.e.*, the Frobenius norm) for measuring the distance between the Gram matrices of the target and synthesized textures. The Gram matrix is a positive semidefinite matrix, existing in a non-Euclidean space. Thus, non-Euclidean metrics may be more suitable to use. For example, the log-Euclidean [4] metric provides many benefits over the Frobenius norm; notably, it is scale-invariant and it defines a minimal geodesic distance on the manifold of positive semidefinite matrices. Minimal geodesic distances are useful in applications for smoothly interpolating between two positive semidefinite matrices, *e.g.*, Gram matrices. For texture synthesis, this may translate to higher quality intermediately-synthesized textures during optimization, and possibly faster convergence.

Beyond addressing these limitations, a natural next step would be to extend the idea of a factorized representation into feed-forward generative networks that have found success in static image synthesis, *e.g.*, [35, 62]. These networks move the computational burden of the optimization process to a learning stage, where given a single example of a texture, a compact "generator" ConvNet is trained to generate multiple samples of the same texture. The same texture-modelling ConvNet used before is kept as the "perceptual" loss, measuring similarity in activation statistics between the synthesized and target textures. These networks have shown to be hundreds of times faster than the traditional approach introduced by Gatys *et al.* [21], on which the two-stream model is based.

Dynamics style transfer is an exciting application of the two-stream model that encourages further exploration on texture-based artistic tools. Like image style transfer [22], however, it is limited in its ability to allow artists granular control over visual aesthetics. Recently, there has been some progress extending image style transfer to include control over spatial location and colour information across spatial scales [23]. An extension to dynamics style transfer could involve incorporating this technique to allow artists to decide which (textured) regions of an image to transfer dynamics to. A metaphorical “motion brush”.

Bibliography

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016. 48

[2] Edward H. Adelson. On seeing stuff: the perception of materials by humans and machines. In *Human Vision and Electronic Imaging VI*, volume 4299, pages 1–13, 2001. 2, 3

[3] Edward H. Adelson and James R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A (JOSA-A)*, 2(2):284–299, 1985. 25, 38

[4] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 56(2):411–421, 2006. 17, 75

[5] Ziv Bar-Joseph, Ran El-Yaniv, Dani Lischinski, and Michael Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics (T-VCG)*, 7(2):120–135, 2001. 1

[6] Guillaume Berger and Roland Memisevic. Incorporating long-range consistency in cnn-based texture generation. *arXiv:1606.01286*, 2016. 73

[7] Alex J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv:1603.01768*, 2016. 45

[8] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 59

[9] James E. Cutting. Blowing in the wind: Perceiving structure in trees and bushes. *Cognition*, 12(1):25 – 44, 1982. 7

[10] Kristin J. Dana, Bram Van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34, 1999. 2, 3

[11] Konstantinos G. Derpanis. *On the Role of Representation in the Analysis of Visual Spacetime*. PhD thesis, 2010. 8, 23, 36

[12] Konstantinos G. Derpanis and Richard P. Wildes. Spacetime texture representation and recognition based on a spatiotemporal orientation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*,

34(6):1193–1205, 2012. 2, 8, 25, 26, 27, 34, 35, 36, 38, 40, 41, 46, 62, 71, 73, 74

[13] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51(2):91–109, 2003. 1, 22

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 24, 34, 39

[15] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, pages 341–346, 2001. 4, 5

[16] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1033–1038, 1999. 15

[17] Manfred Fahle and Tomaso Poggio. Visual hyperacuity: Spatiotemporal interpolation in human vision. *Proceedings of the Royal Society of London B: Biological Sciences*, 213(1193):451–477, 1981. 25

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7, 14

[19] Andrew W. Fitzgibbon. Stochastic rigidity: Image registration for nowhere-static scenes. In *IEEE International Conference on Computer Vision (ICCV)*, pages 662–669, 2001. 22

[20] Christina M. Funke, Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Synthesising dynamic textures using convolutional neural networks. *arXiv:1702.07006*, 2017. x, 22, 64, 65, 67, 68, 74, 91

[21] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, pages 262–270, 2015. 4, 5, 7, 15, 16, 17, 18, 19, 22, 30, 31, 33, 43, 46, 52, 62, 75

[22] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 2, 5, 7, 20, 21, 45, 72, 75

[23] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 45, 75

[24] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013. 24

[25] James J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, Oxford, England, 1950. 2

[26] Melvyn A. Goodale and A. David. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, 1992. 7, 14

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`. 10, 14

[28] Isma Hadji and Richard P. Wildes. A spatiotemporal oriented energy network for dynamic texture recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 26

[29] Isma Hadji and Richard P. Wildes. What do we understand about convolutional networks? *arXiv:1803.08834*, 2018. 10

[30] David J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision (IJCV)*, 1(4):279–302, 1988. 25

[31] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, pages 229–238, 1995. 2, 3, 7, 15

[32] David J. Heeger and Alex P. Pentland. Seeing structure through chaos. In *IEEE Motion Workshop: Representation and Analysis*, pages 131–136, 1986. 1

[33] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence (A.I.)*, 17:185–203, 1981. 23

[34] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 24, 34

[35] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 694–711, 2016. 75

[36] Béla Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962. 15

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014. 40

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 19

[39] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *SIGGRAPH*, pages 277–286, 2003. 15

[40] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Conference on Neural Information Processing Systems (NIPS) Workshops*, pages 9–50, 1998. 35

[41] Wen-Chieh Lin, James Hays, Chenyu Wu, Yanxi Liu, and Vivek Kwatra. Quantitative evaluation of near regular texture synthesis algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 427–434, 2006. 61

[42] Dong C. Liu and Jorge Nocedal. On the limited memory method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989. 43

[43] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981. 23

[44] Anthony J. Movshon and Eero P. Simoncelli. Representation of naturalistic image structure in the primate visual cortex. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 79, pages 115–122, 2014. 20

[45] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010. 11

[46] Shree K. Nayar, Katsushi Ikeuchi, and Takeo Kanade. Surface reflection: physical and geometrical perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (7):611–634, 1991. 3

[47] Randal C. Nelson and Ramprasad Polana. Qualitative recognition of motion using temporal texture. *International Conference on Visualization, Graphics and Image Processing (CVGIP)*, 56(1), 1992. 1

[48] Shinji Nishimoto and Jack L. Gallant. A three-dimensional spatiotemporal receptive field model explains responses of area MT neurons to naturalistic movies. *Journal of Neuroscience*, 31(41):14551–14564, 2011. 25

[49] Renaud Péteri, Sándor Fazekas, and Mark J. Huiskes. DynTex: A Comprehensive Database of Dynamic Textures. *Pattern Recognition Letters (PRL)*, 31(12), 2010. 49

[50] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision (IJCV)*, 40(1):49–70, 2000. 4, 7, 15

[51] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 24

[52] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1172, 2015. 23, 39

[53] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition (GCPR)*, pages 26–36, 2016. 21

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 14, 15

[55] Arno Schödl, Richard Szeliski, David Salesin, and Irfan A. Essa. Video textures. In *SIGGRAPH*, pages 489–498, 2000. 15

[56] Eero P. Simoncelli and David J. Heeger. A model of neuronal responses in visual area MT. *Vision Research*, 38(5):743 – 761, 1998. 25, 34, 38, 46

[57] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Conference on Neural Information Processing Systems (NIPS)*, pages 568–576, 2014. 7, 14

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 15, 18, 19, 30, 31

[59] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012. 39

[60] Martin Szummer and Rosalind W. Picard. Temporal texture modeling. In *IEEE International Conference on Image Processing (ICIP)*, pages 823–826, 1996. 22

[61] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Deep end2end voxel2voxel prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 402–409, 2016. 39

[62] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning (ICML)*, pages 1349–1357, 2016. 75

[63] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4105–4113, 2017. 35

[64] Yizhou Wang and Song Chun Zhu. Modeling textured motion: Particle, wave and sketch. In *IEEE International Conference on Computer Vision (ICCV)*, pages 213–220, 2003. 1, 22

[65] Andrew B. Watson and Albert J. Ahumada. A look at motion in the frequency domain. In *Motion workshop: Perception and representation*, pages 1–10, 1983. 25

[66] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, pages 479–488, 2000. 15

[67] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. x, 22, 64, 67, 68, 74, 91

[68] Jason J. Yu, Adam W. Harley, and Konstantinos G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision (ECCV) Workshops*, 2016. 24

Appendix

A.1 Experimental procedure

Provided here are the experimental details of the user study using Amazon Mechanical Turk (AMT). Experimental trials were grouped into batches of Human Intelligence Tasks (HITs) for users to complete. Each HIT consisted of 59 pairwise comparisons between a synthesized dynamic texture and its target. Users were asked to choose which texture appeared more realistic after viewing each texture independently for an exposure time (in seconds) sampled randomly from the set $\{0.3, 0.4, 0.6, 1.2, 2.4, 3.6, 4.8\}$. Note that 12 frames of the dynamic texture corresponds to 1.2 seconds, *i.e.*, 10 frames per second. Since the dynamic textures were collected from various sources with varying framerates, a canonical framerate was chosen for all input and synthesized dynamic textures. 10 frames per second was chosen to allow sufficient viewing time (1.2 seconds) while maintaining easily distinguishable dynamics. Before viewing a dynamic texture, a centred dot is flashed twice to indicate to the user where to look (left or right). To prepare users for the task, the first three comparisons were used for warm-up, exposing them to the shortest (0.3s), median (1.2s), and longest (4.8s) durations. To prevent spamming and bias, the experiment was constrained as follows:

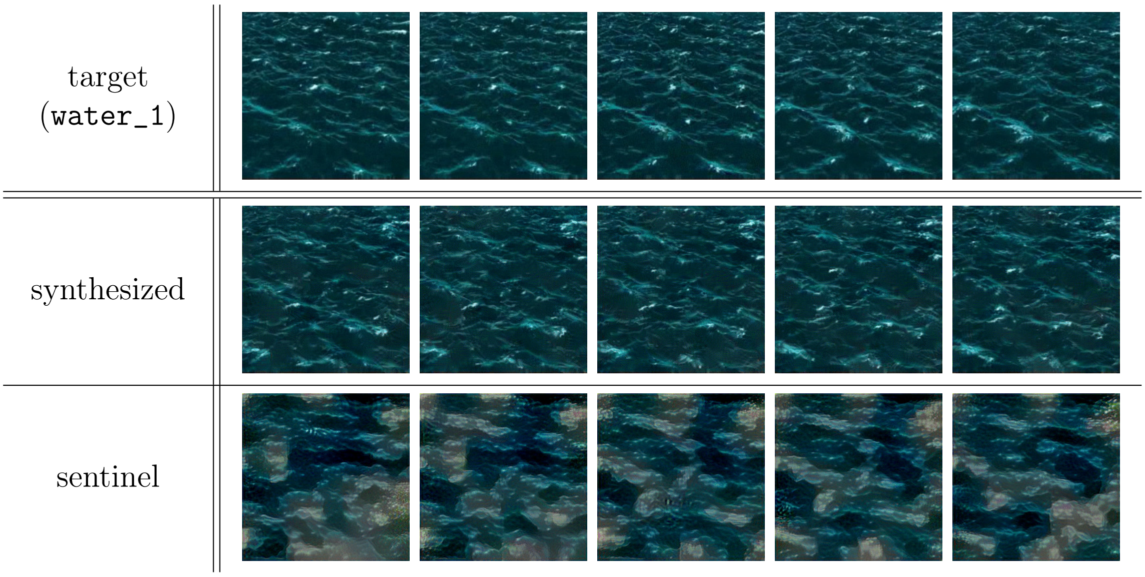


Figure A.1: Example of a sentinel dynamic texture for the user study. (top row) Target dynamic texture. (middle row) An obviously-unrealistic synthesized dynamic texture (a sentinel example). Achieved by terminating synthesis early (100 optimization iterations). (bottom row) Dynamic texture synthesized after $6,000$ optimization iterations.

1. Users could make a choice only after both dynamic textures were shown;
2. The next texture comparison could only be made after a decision was made for the current comparison;
3. A choice could not be changed after the next pair of dynamic textures were shown;
4. Users were each restricted to a single HIT.

Obvious unrealistic dynamic textures were synthesized by terminating synthesis early (100 iterations) and were used as sentinel tests. An example is shown in Fig. A.1.

Three of the 59 pairwise comparisons were sentinels and results from users who gave incorrect answers on any of the sentinel comparisons were not used. The left-right order of textures within a pair, display order within a pair, and order of pairs within a HIT, were randomized. An example of a HIT is shown in a video included with the supplemental material: `HIT_example.mp4`.

Users were paid $2 USD per HIT, and were required to have at least a 98% HIT approval rating, greater than or equal to 5000 HITs approved, and to be residing in the US. Results were collected from 200 unique users to evaluate the final model (which uses the "Concat layer") and another 200 to evaluate the baseline model (which uses the "Flow decode layer").

A.2 Qualitative results

Provided in the supplemental material are videos showcasing the qualitative results of the two-stream model, including the experiments mentioned in the main manuscript. The videos are in MP4 format (H.264 codec) and are best viewed in a loop. They are enclosed in the following folders in the supplemental:

- `target_textures`: This folder contains the 59 dynamic textures used as targets for synthesis.

- `dynamic_texture_synthesis`: This folder contains synthesized dynamic textures where the appearance and dynamics targets are the same. Specifically, it contains the folders `comparisons`, `using_concatenation_layer`, and `using_flow_decode_layer`.

- `using_concatenation_layer`: This folder contains synthesized dynamic textures where the concatenation layer was used for computing the Gram matrices on the dynamics stream. These are the results from the final model. It contains the folders `appearance_stream_only`, `dynamics_stream_only`, `full_synthesis`, `incrementally_synthesized`, and `temporally_endless`.

- `using_flow_decode_layer`: This folder contains synthesized dynamic textures where the predicted flow output is used for computing the Gram matrices on the dynamics stream. These are the results from the baseline. It contains the folder `full_synthesis`.

- `full_synthesis`: This folder contains regularly-synthesized dynamic textures, *i.e.*, not incrementally-generated, nor temporally-endless, etc.

- `appearance_stream_only`: This folder contains dynamic textures synthesized using only the appearance stream of the two-stream model. The dynamics stream is not used.

- `dynamics_stream_only`: This folder contains a dynamic texture synthesized using only the dynamics stream of the two-stream model. The appearance stream is not used.

- `incrementally_synthesized`: This folder contains dynamic textures synthesized using the incremental process outlined in Sec. 3.3.1 in the main manuscript.

- `temporally_endless`: This folder contains a synthesized dynamic texture (`smoke_plume_1`) where there is no discernible temporal seam between the last and first frames. Played as a loop, it appears to be temporally endless, thus, it is presented in animated GIF format.

- `dynamics_style_transfer`: This folder contains synthesized dynamic textures where the appearance and dynamics targets are different. Also included are videos where the synthesized dynamic texture is “pasted” back onto the original image it was cropped from, showing a proof-of-concept of dynamics style transfer as an artistic tool.

- `comparisons/funke`: This folder contains four dynamic texture synthesis comparisons between the two-stream model and a recent (unpublished) approach [20]. The dynamic textures chosen are those reported by Funke *et al.* [20] which exhibit spatiotemporal homogeneity. For ease of comparison, the results from both models have been concatenated with their corresponding targets.

- `comparisons/xie_and_funke`: This folder contains nine dynamic texture synthesis comparisons between the two-stream model, Funke *et al.*’s [20], and Xie *et al.*’s [67]. The dynamic textures chosen cover the full range of the appearance and dynamics groupings listed in Sec. 4.2. For ease of comparison, the results from all models have been concatenated with their corresponding targets.

A.3 Full user study results

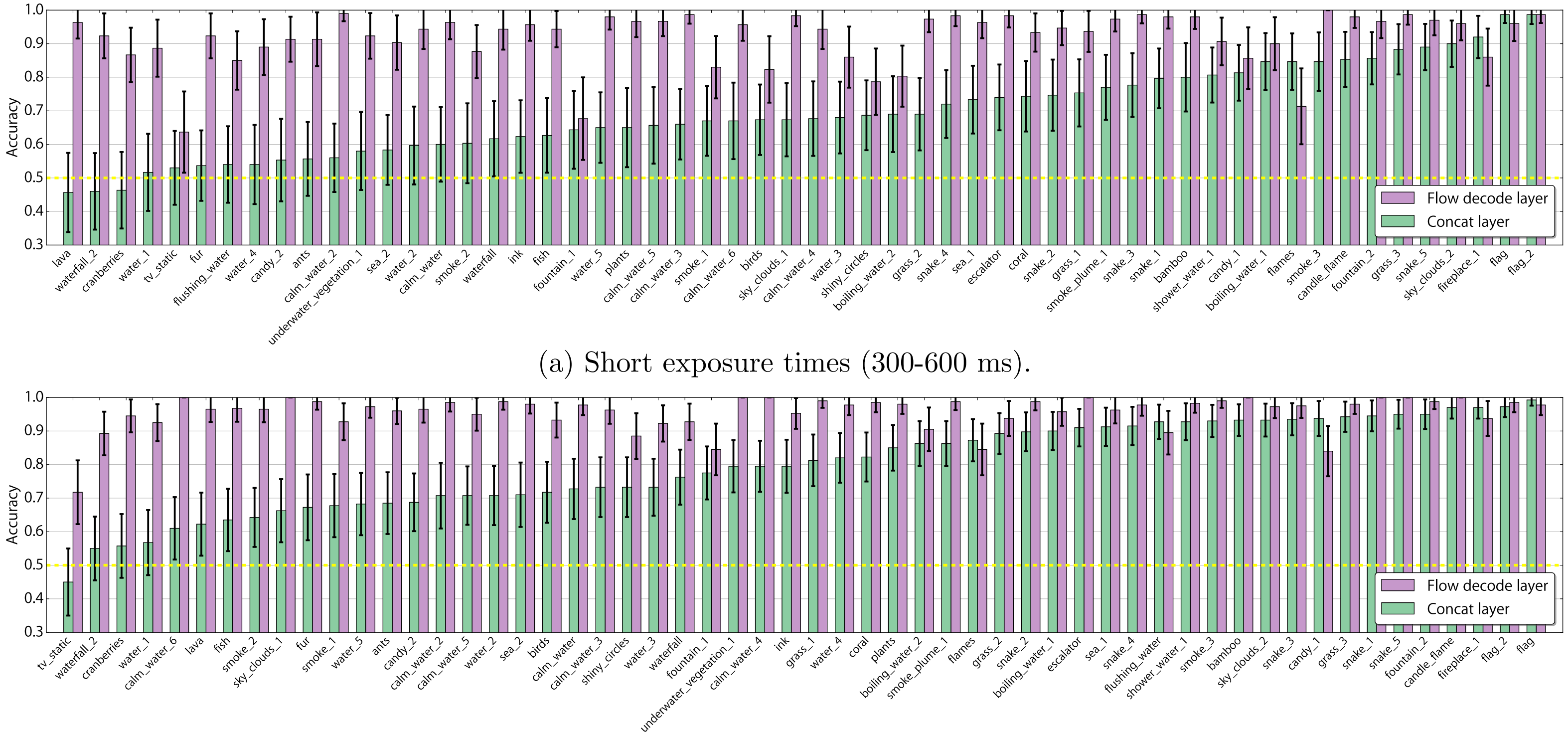
Figures A.2a and A.2b show histograms of the average user accuracy on each texture, averaged over a range of exposure times. The histogram bars are ordered from lowest to highest accuracy, based on the results when using the final model.

Tables A.1 and A.2 show the average user accuracy on each texture when using the final model. The results are averaged over exposure times. Similarly, Tables A.3 and A.4 show the results when using the baseline.

Tables A.5 and A.6 show the average user accuracy on texture appearance groups when using the final model. The results are averaged over exposure times. Similarly, Tables A.7 and A.8 show the results when using the baseline.

Tables A.9 and A.10 show the average user accuracy on texture dynamics groups when using the final model. The results are averaged over exposure times. Similarly, Tables A.11 and A.12 show the results when using the baseline.

Tables A.13 and A.14 show the average user accuracy over all textures when using the final model. The results are averaged over exposure times. Similarly, Tables A.15 and A.16 show the results when using the baseline.



(a) Short exposure times (300-600 ms).

(b) Long exposure times (1200-4800 ms).

Figure A.2: Per-texture accuracies averaged over exposure times. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamic texture	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
ants	0.625±0.194	0.333±0.161	0.714±0.193	0.536±0.185	0.636±0.201	0.857±0.15	0.704±0.172
bamboo	0.769±0.162	0.786±0.215	0.842±0.164	0.906±0.101	0.95±0.096	0.938±0.084	0.926±0.099
birds	0.609±0.199	0.786±0.152	0.615±0.187	0.542±0.199	0.867±0.122	0.682±0.195	0.778±0.192
boiling_water_1	0.806±0.139	0.88±0.127	0.846±0.196	0.714±0.193	0.97±0.058	0.96±0.077	0.963±0.071
boiling_water_2	0.533±0.252	0.842±0.164	0.7±0.164	0.87±0.138	0.731±0.17	0.852±0.134	1.0±0.0
calm_water	0.607±0.181	0.571±0.212	0.615±0.187	0.636±0.164	0.75±0.19	0.762±0.182	0.762±0.182
calm_water_2	0.44±0.195	0.621±0.177	0.622±0.156	0.7±0.201	0.652±0.195	0.773±0.175	0.706±0.217
calm_water_3	0.813±0.135	0.5±0.245	0.667±0.169	0.7±0.201	0.824±0.181	0.63±0.182	0.781±0.143
calm_water_4	0.727±0.186	0.654±0.183	0.65±0.209	0.767±0.151	0.875±0.132	0.848±0.122	0.682±0.195
calm_water_5	0.609±0.199	0.773±0.175	0.591±0.205	0.609±0.199	0.708±0.182	0.724±0.163	0.786±0.152
calm_water_6	0.6±0.248	0.773±0.175	0.643±0.177	0.5±0.2	0.519±0.188	0.765±0.202	0.658±0.151
candle_flame	0.806±0.139	0.75±0.212	1.0±0.0	0.909±0.12	1.0±0.0	1.0±0.0	0.968±0.062
candy_1	0.81±0.168	0.839±0.129	0.788±0.139	0.9±0.131	0.938±0.119	0.963±0.071	0.952±0.091
candy_2	0.5±0.219	0.429±0.212	0.727±0.186	0.636±0.164	0.652±0.195	0.724±0.163	0.741±0.165
coral	0.591±0.205	0.81±0.168	0.826±0.155	0.815±0.147	0.773±0.175	0.885±0.123	0.828±0.137
cranberries	0.48±0.196	0.318±0.195	0.593±0.185	0.64±0.188	0.548±0.175	0.519±0.188	0.524±0.214
escalator	0.792±0.162	0.733±0.158	0.696±0.188	0.967±0.064	0.933±0.126	0.926±0.099	0.815±0.147
fireplace_1	0.909±0.12	0.952±0.091	0.897±0.111	0.917±0.111	1.0±0.0	0.962±0.074	1.0±0.0
fish	0.571±0.212	0.65±0.209	0.656±0.165	0.652±0.195	0.696±0.188	0.692±0.177	0.5±0.179
flag	1.0±0.0	1.0±0.0	0.964±0.069	0.968±0.062	1.0±0.0	1.0±0.0	1.0±0.0
flag_2	0.964±0.069	1.0±0.0	1.0±0.0	0.923±0.102	1.0±0.0	1.0±0.0	0.966±0.066
flames	0.72±0.176	0.909±0.12	0.913±0.115	0.889±0.119	0.889±0.119	0.875±0.132	0.833±0.133
flushing_water	0.5±0.209	0.565±0.203	0.552±0.181	0.871±0.118	0.92±0.106	0.917±0.111	1.0±0.0
fountain_1	0.435±0.203	0.688±0.227	0.808±0.151	0.833±0.149	0.788±0.139	0.667±0.189	0.808±0.151
fountain_2	0.929±0.095	0.826±0.155	0.815±0.147	1.0±0.0	0.905±0.126	0.967±0.064	0.933±0.089
fur	0.452±0.175	0.538±0.192	0.621±0.177	0.75±0.15	0.737±0.198	0.526±0.225	0.667±0.218
grass_1	0.813±0.135	0.778±0.192	0.667±0.202	0.792±0.162	0.735±0.148	0.895±0.138	0.826±0.155
grass_2	0.632±0.217	0.667±0.202	0.767±0.151	0.88±0.127	1.0±0.0	0.88±0.127	0.813±0.135
grass_3	0.8±0.175	0.903±0.104	0.95±0.096	0.958±0.08	1.0±0.0	0.92±0.106	0.889±0.119
ink	0.476±0.214	0.714±0.167	0.679±0.173	0.724±0.163	0.808±0.151	0.783±0.169	0.87±0.138
lava	0.458±0.199	0.346±0.183	0.556±0.23	0.733±0.158	0.593±0.185	0.522±0.204	0.652±0.195
plants	0.632±0.217	0.667±0.202	0.652±0.195	0.767±0.151	0.806±0.139	0.857±0.15	0.96±0.077
sea_1	0.6±0.192	0.769±0.162	0.826±0.155	0.955±0.087	0.857±0.15	0.964±0.069	0.88±0.127
sea_2	0.542±0.199	0.625±0.168	0.581±0.174	0.75±0.173	0.75±0.19	0.533±0.252	0.808±0.151
shiny_circles	0.517±0.182	0.741±0.165	0.8±0.175	0.609±0.199	0.9±0.131	0.767±0.151	0.652±0.195
shower_water_1	0.767±0.151	0.903±0.104	0.75±0.16	1.0±0.0	0.952±0.091	0.87±0.138	0.889±0.145
sky_clouds_1	0.667±0.202	0.737±0.198	0.613±0.171	0.72±0.176	0.652±0.195	0.571±0.259	0.714±0.15
sky_clouds_2	0.792±0.162	0.938±0.119	0.97±0.058	0.957±0.083	0.92±0.106	0.889±0.119	0.962±0.074
smoke_1	0.538±0.192	0.731±0.17	0.741±0.165	0.471±0.237	0.895±0.138	0.76±0.167	0.588±0.165
smoke_2	0.478±0.204	0.727±0.186	0.6±0.215	0.72±0.176	0.5±0.173	0.724±0.163	0.63±0.182
smoke_3	0.769±0.162	0.833±0.149	0.938±0.119	0.821±0.142	0.931±0.092	0.968±0.062	1.0±0.0
smoke_plume_1	0.724±0.163	0.783±0.169	0.81±0.168	0.963±0.071	0.84±0.144	0.778±0.157	0.87±0.138
snake_1	0.862±0.126	0.704±0.172	0.826±0.155	0.88±0.127	0.905±0.126	1.0±0.0	1.0±0.0
snake_2	0.72±0.176	0.708±0.182	0.813±0.191	0.958±0.08	0.852±0.134	0.9±0.107	0.88±0.127
snake_3	0.643±0.177	0.773±0.175	0.917±0.111	0.87±0.138	0.913±0.115	1.0±0.0	0.964±0.069
snake_4	0.643±0.177	0.815±0.147	0.714±0.193	1.0±0.0	0.917±0.111	0.889±0.119	0.852±0.134
snake_5	0.826±0.155	0.947±0.1	0.889±0.103	0.875±0.132	0.923±0.102	1.0±0.0	1.0±0.0
tv_static	0.538±0.192	0.63±0.182	0.423±0.19	0.615±0.187	0.227±0.175	0.619±0.208	0.333±0.178
underwater_vegetation_1	0.656±0.165	0.5±0.231	0.579±0.222	0.821±0.142	0.813±0.191	0.733±0.158	0.821±0.142
water_1	0.556±0.23	0.32±0.183	0.667±0.169	0.727±0.186	0.571±0.212	0.583±0.197	0.394±0.167
water_4	0.375±0.237	0.586±0.179	0.652±0.195	0.826±0.155	0.706±0.153	0.818±0.161	0.917±0.111
water_2	0.632±0.217	0.64±0.188	0.52±0.196	0.739±0.179	0.667±0.202	0.724±0.163	0.7±0.164
water_3	0.545±0.208	0.741±0.165	0.75±0.173	0.833±0.149	0.771±0.139	0.652±0.195	0.682±0.195
water_5	0.688±0.161	0.667±0.218	0.586±0.179	0.759±0.156	0.65±0.209	0.652±0.195	0.667±0.189
waterfall	0.571±0.183	0.586±0.179	0.688±0.227	0.792±0.162	0.696±0.188	0.731±0.17	0.833±0.133
waterfall_2	0.444±0.187	0.364±0.201	0.583±0.197	0.75±0.16	0.37±0.182	0.632±0.217	0.452±0.175

Table A.1: Per-texture accuracies using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamic texture	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
ants	0.526±0.111	0.673±0.093	0.608±0.072
bamboo	0.797±0.103	0.928±0.048	0.882±0.048
birds	0.675±0.105	0.723±0.09	0.702±0.069
boiling_water_1	0.841±0.086	0.915±0.053	0.886±0.047
boiling_water_2	0.703±0.112	0.864±0.066	0.802±0.06
calm_water	0.6±0.111	0.716±0.091	0.665±0.071
calm_water_2	0.571±0.102	0.707±0.098	0.636±0.072
calm_water_3	0.692±0.102	0.729±0.089	0.713±0.067
calm_water_4	0.676±0.111	0.798±0.075	0.751±0.064
calm_water_5	0.657±0.114	0.712±0.087	0.69±0.069
calm_water_6	0.677±0.114	0.604±0.093	0.632±0.072
candle_flame	0.861±0.08	0.97±0.033	0.924±0.04
candy_1	0.812±0.083	0.94±0.051	0.876±0.05
candy_2	0.556±0.123	0.688±0.086	0.64±0.071
coral	0.742±0.106	0.827±0.073	0.794±0.061
cranberries	0.473±0.114	0.558±0.095	0.522±0.073
escalator	0.74±0.098	0.909±0.057	0.835±0.055
fireplace_1	0.917±0.064	0.971±0.032	0.949±0.033
fish	0.63±0.111	0.627±0.094	0.629±0.072
flag	0.987±0.025	0.99±0.02	0.989±0.016
flag_2	0.985±0.03	0.971±0.032	0.976±0.023
flames	0.843±0.085	0.87±0.063	0.86±0.051
flushing_water	0.541±0.114	0.918±0.054	0.756±0.064
fountain_1	0.646±0.116	0.776±0.079	0.727±0.067
fountain_2	0.859±0.077	0.947±0.045	0.908±0.043
fur	0.535±0.105	0.682±0.097	0.609±0.073
grass_1	0.761±0.099	0.8±0.078	0.784±0.062
grass_2	0.7±0.107	0.88±0.064	0.806±0.059
grass_3	0.887±0.074	0.941±0.046	0.919±0.041
ink	0.636±0.107	0.792±0.079	0.725±0.066
lava	0.441±0.118	0.631±0.093	0.556±0.074
plants	0.651±0.118	0.841±0.069	0.771±0.063
sea_1	0.73±0.101	0.917±0.055	0.835±0.056
sea_2	0.586±0.103	0.729±0.094	0.657±0.071
shiny_circles	0.671±0.106	0.729±0.089	0.703±0.068
shower_water_1	0.809±0.082	0.93±0.054	0.869±0.05
sky_clouds_1	0.662±0.11	0.68±0.093	0.673±0.071
sky_clouds_2	0.904±0.068	0.931±0.05	0.92±0.04
smoke_1	0.671±0.104	0.674±0.094	0.672±0.07
smoke_2	0.6±0.119	0.637±0.089	0.624±0.071
smoke_3	0.833±0.09	0.927±0.049	0.892±0.046
smoke_plume_1	0.767±0.097	0.863±0.067	0.823±0.057
snake_1	0.797±0.089	0.947±0.045	0.879±0.049
snake_2	0.738±0.107	0.896±0.058	0.836±0.055
snake_3	0.77±0.096	0.94±0.047	0.868±0.05
snake_4	0.724±0.101	0.903±0.06	0.822±0.058
snake_5	0.885±0.071	0.95±0.043	0.921±0.04
tv_static	0.532±0.11	0.448±0.099	0.486±0.074
underwater_vegetation_1	0.594±0.116	0.794±0.078	0.713±0.068
water_1	0.521±0.115	0.55±0.098	0.538±0.074
water_4	0.559±0.118	0.806±0.076	0.708±0.068
water_2	0.594±0.116	0.709±0.088	0.663±0.071
water_3	0.685±0.107	0.74±0.084	0.718±0.066
water_5	0.646±0.105	0.688±0.093	0.669±0.07
waterfall	0.603±0.112	0.767±0.082	0.699±0.068
waterfall_2	0.466±0.114	0.543±0.095	0.511±0.073

Table A.2: Per-texture accuracies averaged over a range of exposure times, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamic texture	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
ants	0.933±0.126	0.9±0.186	0.913±0.115	0.963±0.071	1.0±0.0	1.0±0.0	0.885±0.123
bamboo	1.0±0.0	0.944±0.106	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
birds	0.895±0.138	0.652±0.195	0.933±0.126	0.947±0.1	0.9±0.131	0.913±0.115	0.966±0.066
boiling_water_1	0.846±0.196	0.895±0.138	0.957±0.083	0.96±0.077	0.92±0.106	0.952±0.091	1.0±0.0
boiling_water_2	0.808±0.151	0.889±0.119	0.714±0.193	0.95±0.096	0.857±0.183	0.889±0.145	0.92±0.106
calm_water	0.929±0.135	0.963±0.071	1.0±0.0	0.962±0.074	0.952±0.091	1.0±0.0	1.0±0.0
calm_water_2	1.0±0.0	1.0±0.0	0.966±0.066	1.0±0.0	1.0±0.0	1.0±0.0	0.941±0.112
calm_water_3	1.0±0.0	1.0±0.0	0.957±0.083	0.941±0.112	0.955±0.087	0.96±0.077	1.0±0.0
calm_water_4	0.875±0.162	0.947±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
calm_water_5	1.0±0.0	0.897±0.111	1.0±0.0	0.857±0.15	1.0±0.0	0.944±0.106	1.0±0.0
calm_water_6	0.913±0.115	1.0±0.0	0.958±0.08	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
candle_flame	0.944±0.106	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
candy_1	0.765±0.202	0.87±0.138	0.938±0.119	0.905±0.126	0.846±0.139	0.81±0.168	0.8±0.157
candy_2	0.864±0.143	0.875±0.132	1.0±0.0	1.0±0.0	0.96±0.077	0.952±0.091	0.95±0.096
coral	0.84±0.144	0.957±0.083	1.0±0.0	1.0±0.0	0.941±0.112	1.0±0.0	1.0±0.0
cranberries	0.75±0.212	0.917±0.111	0.926±0.099	0.958±0.08	0.867±0.172	0.95±0.096	1.0±0.0
escalator	0.947±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
fireplace_1	0.905±0.126	0.765±0.202	0.923±0.102	0.867±0.172	0.929±0.095	0.947±0.1	1.0±0.0
fish	0.933±0.089	0.957±0.083	0.944±0.106	0.87±0.138	1.0±0.0	1.0±0.0	1.0±0.0
flag	0.875±0.162	1.0±0.0	1.0±0.0	0.958±0.08	1.0±0.0	1.0±0.0	0.947±0.1
flag_2	0.958±0.08	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	0.938±0.119
flames	0.667±0.189	0.75±0.19	0.722±0.207	0.789±0.183	0.826±0.155	0.917±0.111	0.842±0.164
flushing_water	0.941±0.112	0.88±0.127	0.727±0.186	1.0±0.0	0.8±0.157	0.906±0.101	0.867±0.172
fountain_1	0.609±0.199	0.65±0.209	0.769±0.229	0.913±0.115	0.762±0.182	0.818±0.161	0.895±0.138
fountain_2	0.95±0.096	1.0±0.0	0.947±0.1	0.952±0.091	1.0±0.0	1.0±0.0	1.0±0.0
fur	0.818±0.161	0.95±0.096	1.0±0.0	0.955±0.087	1.0±0.0	1.0±0.0	1.0±0.0
grass_1	0.952±0.091	0.938±0.119	0.917±0.111	1.0±0.0	1.0±0.0	0.958±0.08	1.0±0.0
grass_2	1.0±0.0	0.92±0.106	1.0±0.0	0.913±0.115	0.95±0.096	1.0±0.0	0.895±0.138
grass_3	1.0±0.0	1.0±0.0	0.958±0.08	0.923±0.145	1.0±0.0	1.0±0.0	1.0±0.0
ink	0.947±0.1	0.962±0.074	0.96±0.077	1.0±0.0	1.0±0.0	1.0±0.0	0.813±0.191
lava	0.952±0.091	1.0±0.0	0.941±0.112	1.0±0.0	0.906±0.101	1.0±0.0	0.95±0.096
plants	0.9±0.131	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	0.958±0.08	0.958±0.08
sea_1	0.889±0.145	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	0.958±0.08	0.889±0.145
sea_2	0.85±0.156	0.857±0.183	1.0±0.0	0.955±0.087	1.0±0.0	0.968±0.062	1.0±0.0
shiny_circles	0.808±0.151	0.8±0.175	0.75±0.19	0.88±0.127	0.8±0.202	0.96±0.077	0.9±0.131
shower_water_1	0.941±0.112	0.857±0.15	0.923±0.102	0.929±0.135	1.0±0.0	1.0±0.0	1.0±0.0
sky_clouds_1	1.0±0.0	1.0±0.0	0.947±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
sky_clouds_2	0.941±0.112	1.0±0.0	0.941±0.112	1.0±0.0	0.933±0.126	0.96±0.077	1.0±0.0
smoke_1	0.867±0.172	0.773±0.175	0.846±0.139	0.889±0.145	0.944±0.106	0.929±0.095	0.95±0.096
smoke_2	0.667±0.239	0.957±0.083	1.0±0.0	1.0±0.0	0.947±0.1	1.0±0.0	0.909±0.12
smoke_3	1.0±0.0	1.0±0.0	1.0±0.0	0.96±0.077	1.0±0.0	1.0±0.0	1.0±0.0
smoke_plume_1	1.0±0.0	0.958±0.08	0.964±0.069	1.0±0.0	0.955±0.087	1.0±0.0	1.0±0.0
snake_1	0.941±0.112	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
snake_2	0.958±0.08	0.917±0.111	0.962±0.074	1.0±0.0	1.0±0.0	1.0±0.0	0.955±0.087
snake_3	0.957±0.083	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	0.905±0.126	1.0±0.0
snake_4	1.0±0.0	0.947±0.1	1.0±0.0	0.957±0.083	0.95±0.096	1.0±0.0	1.0±0.0
snake_5	0.909±0.12	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
tv_static	0.684±0.209	0.588±0.234	0.64±0.188	0.778±0.192	0.55±0.218	0.76±0.167	0.783±0.169
underwater_vegetation_1	0.857±0.183	0.958±0.08	0.952±0.091	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
water_1	0.929±0.135	0.778±0.192	0.952±0.091	0.929±0.095	0.889±0.145	1.0±0.0	0.88±0.127
water_4	0.778±0.272	1.0±0.0	0.889±0.119	1.0±0.0	1.0±0.0	0.909±0.12	1.0±0.0
water_2	0.867±0.172	1.0±0.0	0.962±0.074	1.0±0.0	1.0±0.0	0.955±0.087	1.0±0.0
water_3	0.737±0.198	0.905±0.126	0.938±0.119	0.897±0.111	1.0±0.0	0.875±0.132	0.909±0.12
water_5	1.0±0.0	0.944±0.106	1.0±0.0	1.0±0.0	1.0±0.0	0.933±0.089	0.962±0.074
waterfall	0.947±0.1	0.933±0.126	0.952±0.091	0.85±0.156	0.929±0.095	0.926±0.099	1.0±0.0
waterfall_2	0.941±0.112	0.88±0.127	0.947±0.1	1.0±0.0	0.773±0.175	0.905±0.126	0.9±0.131

Table A.3: Per-texture accuracies using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamic texture	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
ants	0.917±0.078	0.959±0.039	0.945±0.037
bamboo	0.983±0.034	1.0±0.0	0.993±0.013
birds	0.807±0.102	0.934±0.051	0.885±0.051
boiling_water_1	0.909±0.076	0.955±0.044	0.937±0.04
boiling_water_2	0.811±0.089	0.909±0.064	0.861±0.055
calm_water	0.963±0.05	0.979±0.028	0.974±0.026
calm_water_2	0.986±0.027	0.988±0.024	0.987±0.018
calm_water_3	0.985±0.029	0.964±0.04	0.974±0.026
calm_water_4	0.95±0.055	1.0±0.0	0.979±0.023
calm_water_5	0.954±0.051	0.948±0.05	0.951±0.036
calm_water_6	0.956±0.049	1.0±0.0	0.98±0.023
candle_flame	0.986±0.028	1.0±0.0	0.993±0.014
candy_1	0.857±0.092	0.839±0.075	0.846±0.058
candy_2	0.912±0.067	0.963±0.042	0.939±0.039
coral	0.932±0.057	0.985±0.029	0.957±0.033
cranberries	0.881±0.078	0.952±0.046	0.92±0.043
escalator	0.98±0.038	1.0±0.0	0.993±0.014
fireplace_1	0.875±0.081	0.94±0.051	0.912±0.046
fish	0.944±0.054	0.961±0.043	0.953±0.034
flag	0.963±0.05	0.979±0.029	0.973±0.026
flag_2	0.987±0.025	0.985±0.029	0.986±0.019
flames	0.71±0.113	0.847±0.077	0.789±0.066
flushing_water	0.844±0.089	0.884±0.068	0.867±0.054
fountain_1	0.661±0.124	0.847±0.077	0.773±0.069
fountain_2	0.96±0.054	0.989±0.021	0.979±0.023
fur	0.917±0.07	0.988±0.023	0.958±0.033
grass_1	0.934±0.062	0.988±0.023	0.966±0.03
grass_2	0.969±0.042	0.939±0.052	0.952±0.034
grass_3	0.983±0.034	0.989±0.022	0.986±0.019
ink	0.957±0.047	0.963±0.041	0.96±0.031
lava	0.966±0.046	0.956±0.043	0.96±0.032
plants	0.964±0.049	0.978±0.03	0.972±0.027
sea_1	0.968±0.044	0.965±0.039	0.966±0.029
sea_2	0.902±0.082	0.979±0.029	0.952±0.035
shiny_circles	0.788±0.099	0.894±0.065	0.848±0.057
shower_water_1	0.906±0.071	0.988±0.023	0.952±0.034
sky_clouds_1	0.985±0.029	1.0±0.0	0.993±0.014
sky_clouds_2	0.966±0.046	0.978±0.031	0.973±0.026
smoke_1	0.825±0.094	0.929±0.055	0.884±0.052
smoke_2	0.91±0.068	0.964±0.04	0.94±0.038
smoke_3	1.0±0.0	0.989±0.022	0.993±0.013
smoke_plume_1	0.972±0.038	0.986±0.026	0.979±0.023
snake_1	0.983±0.032	1.0±0.0	0.993±0.013
snake_2	0.946±0.052	0.986±0.027	0.966±0.03
snake_3	0.986±0.026	0.973±0.037	0.98±0.023
snake_4	0.984±0.03	0.975±0.034	0.979±0.023
snake_5	0.964±0.049	1.0±0.0	0.986±0.019
tv_static	0.639±0.121	0.721±0.095	0.687±0.075
underwater_vegetation_1	0.932±0.064	1.0±0.0	0.972±0.027
water_1	0.887±0.085	0.921±0.056	0.908±0.047
water_4	0.907±0.077	0.978±0.03	0.952±0.035
water_2	0.95±0.055	0.988±0.023	0.972±0.027
water_3	0.857±0.092	0.914±0.057	0.893±0.05
water_5	0.981±0.037	0.969±0.035	0.973±0.026
waterfall	0.945±0.06	0.921±0.056	0.931±0.042
waterfall_2	0.918±0.069	0.897±0.064	0.905±0.047

Table A.4: Per-texture accuracies averaged over a range of exposure times, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Appearance group	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
Regular & Near-regular	0.702±0.098	0.74±0.101	0.838±0.088	0.84±0.083	0.954±0.051	0.878±0.074	0.827±0.082
Irregular	0.806±0.046	0.853±0.044	0.837±0.043	0.903±0.036	0.909±0.037	0.919±0.031	0.902±0.035
Stochastic & Near-stochastic	0.616±0.03	0.658±0.029	0.687±0.028	0.76±0.026	0.751±0.026	0.776±0.026	0.762±0.025

Table A.5: Accuracies of textures grouped by appearances, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Appearance group	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
Regular & Near-regular	0.756±0.056	0.871±0.038	0.821±0.033
Irregular	0.831±0.026	0.908±0.017	0.875±0.015
Stochastic & Near-stochastic	0.654±0.017	0.762±0.013	0.717±0.01

Table A.6: Accuracies of textures grouped by appearances, averaged over a range of exposure times, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Appearance group	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
Regular & Near-regular	0.889±0.078	0.933±0.063	0.921±0.067	0.961±0.043	0.948±0.057	0.984±0.031	0.964±0.049
Irregular	0.89±0.041	0.942±0.031	0.957±0.026	0.953±0.028	0.96±0.025	0.968±0.022	0.947±0.029
Stochastic & Near-stochastic	0.901±0.021	0.916±0.018	0.937±0.016	0.957±0.014	0.945±0.015	0.955±0.013	0.96±0.013

Table A.7: Accuracies of textures grouped by appearances, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Appearance group	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
Regular & Near-regular	0.914±0.04	0.964±0.023	0.943±0.022
Irregular	0.93±0.019	0.957±0.013	0.946±0.011
Stochastic & Near-stochastic	0.919±0.011	0.954±0.007	0.939±0.006

Table A.8: Accuracies of textures grouped by appearances, averaged over a range of exposure times, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamics group	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
Spatially-consistent	0.625±0.032	0.664±0.032	0.698±0.03	0.741±0.028	0.753±0.028	0.762±0.028	0.755±0.028
Spatially-inconsistent	0.721±0.039	0.763±0.039	0.777±0.037	0.885±0.028	0.854±0.032	0.902±0.026	0.861±0.029

Table A.9: Accuracies of textures grouped by dynamics, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamics group	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
Spatially-consistent	0.663±0.018	0.753±0.014	0.715±0.011
Spatially-inconsistent	0.753±0.022	0.876±0.015	0.823±0.013

Table A.10: Accuracies of textures grouped by dynamics, averaged over a range of exposure times, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamics group	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
Spatially-consistent	0.886±0.024	0.911±0.02	0.934±0.018	0.947±0.016	0.945±0.016	0.955±0.014	0.954±0.015
Spatially-inconsistent	0.92±0.027	0.942±0.023	0.949±0.021	0.974±0.016	0.954±0.02	0.966±0.017	0.964±0.018

Table A.11: Accuracies of textures grouped by dynamics, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Dynamics group	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
Spatially-consistent	0.911±0.012	0.95±0.008	0.934±0.007
Spatially-inconsistent	0.937±0.013	0.964±0.009	0.953±0.008

Table A.12: Accuracies of textures grouped by dynamics, averaged over a range of exposure times, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Group	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
All textures	0.661±0.025	0.699±0.025	0.726±0.023	0.791±0.021	0.788±0.022	0.812±0.021	0.793±0.021

Table A.13: Average accuracy over all textures, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Group	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
All textures	0.695±0.014	0.796±0.011	0.754±0.009

Table A.14: Average accuracy over all textures, averaged over a range of exposure times, using the concatenation layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Group	300 ms.	400 ms.	600 ms.	1200 ms.	2400 ms.	3600 ms.	4800 ms.
All textures	0.898±0.018	0.922±0.015	0.94±0.013	0.956±0.012	0.948±0.013	0.959±0.011	0.957±0.012

Table A.15: Average accuracy over all textures, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.

Group	Short (300-600 ms.)	Long (1200-4800 ms.)	All (300-4800 ms.)
All textures	0.921±0.009	0.955±0.006	0.941±0.005

Table A.16: Average accuracy over all textures, averaged over a range of exposure times, using the flow decode layer. Each texture accuracy includes a margin of error with a 95% statistical confidence.