Two-Stream Convolutional Networks for Dynamic Texture Synthesis

BY

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 prediction. Given an input dynamic texture, statistics of filter responses from the object recognition ConvNet encapsulate the per-frame appearance of the input texture, while statistics of filter responses from the optical flow ConvNet model its dynamics. To generate a novel texture, a randomly initialized input sequence is optimized to match the feature statistics from each stream of an example texture. Inspired by recent work on image style transfer and enabled by the two-stream model, the synthesis approach is applied to combine the texture appearance from one texture with the dynamics of another to generate entirely novel dynamic textures. The proposed approach generates novel, 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 thorough user study.

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

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 spatial 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 [25] (for extracting optical flow from fluids undergoing irregular fluctuations), temporal textures [37] (for motion recognition of moving patterns such as windblown trees or rippling water), time-varying textures [3] (for synthesizing stochastically-moving patterns), dynamic textures [9] (for modelling and synthesizing stochastically-moving patterns), textured motion [55] (for modelling and synthesizing patterns undergoing stochastic or consistent motion), and spacetime textures [8] (for classifying moving patterns). In this thesis, the term "dynamic

Figure 1.1: 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 looks roughly the same.

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 [20], texture synthesis [24], and more recently, image style transfer [17].

Shape from texture involves recovering the three-dimensional (3D) shape of an object from a 2D image by using texture as a cue. Gibson [20] proposed the *texture gradient* as the primary basis of surface perception by humans. He revealed that neighbouring areas on a textured surface are perceived differently only due to differences in surface orientation and distance from the observer.

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 [24] took advantage of the fact that two

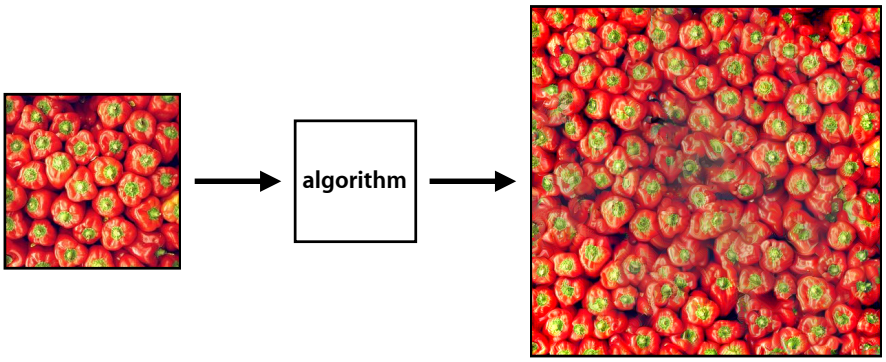


Figure 1.2: (Static) texture synthesis is the process of algorithmically constructing a texture (right) that matches or extends a given source texture (left) by taking advantage of its structural content.

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 steerable pyramids to deconstruct a given texture and synthesized a new texture by matching the distributions of responses from each pyramid level. More recently, Gatys *et al.* [16] 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 extension of texture synthesis where the goal is to synthesize a texture from a source image while constraining the process in order to preserve the semantic content of another image. This can be considered as a texture transfer problem, as previously demonstrated by Efros *et al.* [11] 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. As a follow-up to [16], Gatys *et al.* modified

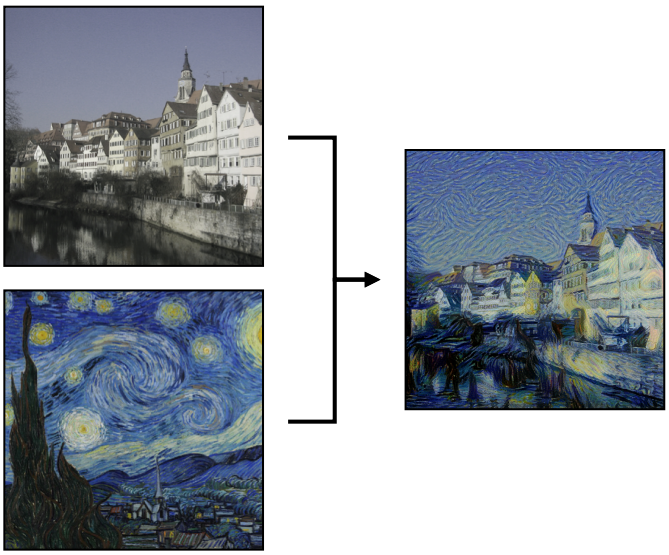


Figure 1.3: Image style transfer. The goal is to synthesize a texture (right) from a source image (bottom-left) while constraining the process in order to preserve the semantic content of another image (top-left).

their previous ConvNet to support texture transfer by including an additional objective that enforced the synthesized texture to match the semantic content of a given image, naming the process *image style transfer* [17]. Unlike the patch-based method of Efros *et al.* [11], Gatys *et al.*'s [17] approach of using a ConvNet was more robust to textures with long-range consistencies.

Motivated by the ConvNet model of Gatys *et al.* [17] for texture synthesis, the focus of this work is on the synthesis of dynamic texture samples based on a single exemplar through the use of ConvNets.

1.1 Summary of thesis

Matthew provide chapter-by-chapter outline (**??**)

Many common dynamic textures are naturally described by the ensemble of

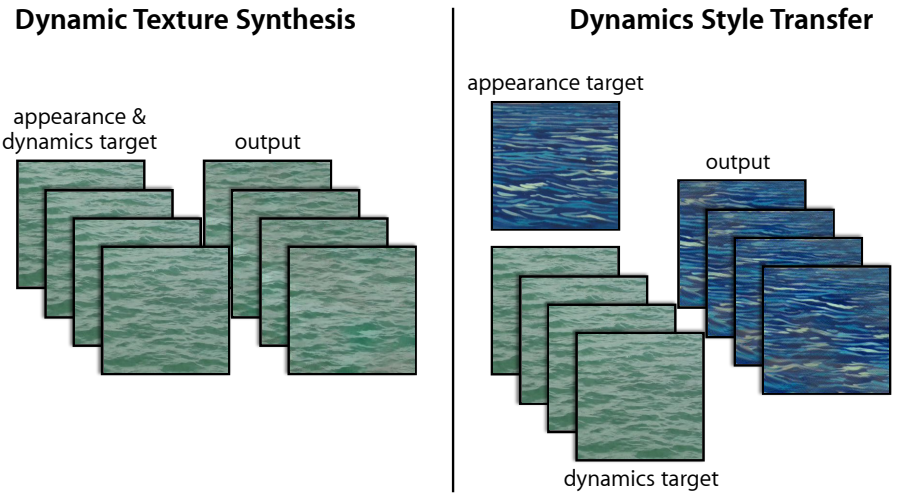


Figure 1.4: Dynamic texture synthesis. (left) Given an input dynamic texture as the target, the two-stream model synthesizes a novel dynamic texture that preserves the target's appearance and dynamics characteristics. (right) The two-stream approach enables synthesis that combines the texture appearance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

appearance and dynamics (*i.e.*, spatial and temporal pattern variation) of their constituent elements. In this work, a factored analysis of dynamic textures in terms of their appearance and dynamics is proposed. This factorization is then used to enable dynamic texture synthesis which, based on example dynamic texture inputs, will generate a novel dynamic texture instance. It shall also enable 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.

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 prediction, respectively. Similar to previous work on spatial textures [16, 24, 40], 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 [6] 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 [21] 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 [48, 14].

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*. [17]. 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.

1.2 Contributions

Matthew split contributions into a numbered list (**??**)

The contributions of this work span 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 is constructed and trained for optical flow prediction. Third, through a qualitative evaluation, it is shown that the two-stream representation is effective in generating visually compelling, novel instances of a wide range of dynamic textures. Fourth, 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. 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 will show which sequences work better than others and why. This may point to future work to address limitations of the proposed model. 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. More broadly, the ability to animate static imagery via dynamics style transfer can meaningfully contribute to the emerging artistic medium of computer-generated art.

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

A convolutional network (ConvNet) is a feed-forward computational graph of processing nodes, commonly used in analyzing visual imagery. It is a class of artificial neural networks (ANNs), which are computational systems vaguely inspired by the biological neural networks that constitute animal brains. ConvNets perform tasks (*e.g.*, object classification) 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 spatial dimensions and C_* represents the chan-

nel dimension. 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 called *neurons*. At each spatial-channel location of the input to a layer, a neuron computes local non-linear transformations, *e.g.*, $\sigma(\mathbf{x}) = \max(0, \phi(\mathbf{x}))$ (known as the *rectified linear unit* or ReLU [36]). 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 an *activation* or *feature*, $\sigma(\mathbf{x}) \in \mathbb{R}$. The set of activations produced by a neuron at every location of the input is known as an *activation map* or *feature map*, $\sigma \in \mathbb{R}^{H_\sigma \times W_\sigma \times C_\sigma}$. 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 neighbouring neurons at the previous layer, $l-1$:

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

where $\mathbf{w}^l$ are the *weights* (or *filter*) of the neuron applied to the input $\sigma^{l-1}(\mathbf{x})$, 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. Convolution is typically done in a sliding window fashion across the entire input. At the base of the ConvNet, inputs are typically images (*e.g.*, RGB image $\mathbf{X} \in \mathbb{R}^{H_x \times W_x \times 3}$), while inputs at intermediate layers are activation maps (*e.g.*, $\sigma \in \mathbb{R}^{H_\sigma \times W_\sigma \times C_\sigma}$).

ConvNets "learn" to perform tasks through an iterative process called *training*. At each iteration, an input is fed through the network to produce an output which

is subsequently evaluated against the "true" 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 error). Starting from the loss, the network *backpropagates* adjustments to its weights and biases at each layer via the gradient of the loss with respect to the weights and biases at that layer. After a suitable amount of training iterations, this *gradient descent* process will have adjusted the weights and biases of the network such that it is able to achieve its objective in minimizing the loss.

2.2 Parametric texture synthesis

To review, 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 [12, 32, 45, 57], 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 thesis will focus on these parametric approaches.

The statistical characterization of visual textures was introduced in the seminal work of Julesz [29]. 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 [24, 40]. 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.* [16] demonstrated impressive results by replacing the linear filter bank with the VGG-19 [49] ConvNet pre-trained on the ImageNet [44] dataset for the task of object recognition. This ConvNet, in effect, served as a proxy for the ventral visual processing stream. 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 ConvNet

Since the two-stream approach to dynamic texture synthesis proposed by this thesis is an extension of the above 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 VGG-19, where N_l and M_l denote the number of activation maps and the number of spatial locations, respectively. 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:

$$G^l_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} A^l_{ik} A^l_{jk} \ , \tag{2.2}$$

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:

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

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:

$$\mathcal{L} = \frac{1}{L} \sum_l \|\mathbf{G}^l - \hat{\mathbf{G}}^l\|^2_F \ , \tag{2.4}$$

where L is the number of VGG-19 layers used when computing Gram matrices and $\|\cdot\|_F$ is the Frobenius norm. Gram matrices are computed on layers *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4*.

Before synthesizing a texture, an initial forward pass through VGG-19 is performed with the target texture as input. The target texture's Gram matrices across various layers in the network are computed and saved as objectives for the synthesis process. Then the synthesized texture is initialized with IID Gaussian noise. Finally, with the weights of VGG-19 kept fixed, the final objective (Eq. 2.4) is minimized with respect to the synthesized texture. With each iteration of the optimization process, the synthesized texture is updated to appear increasingly perceptually similar to the target texture. An overview of this process is presented in Fig. 2.1.

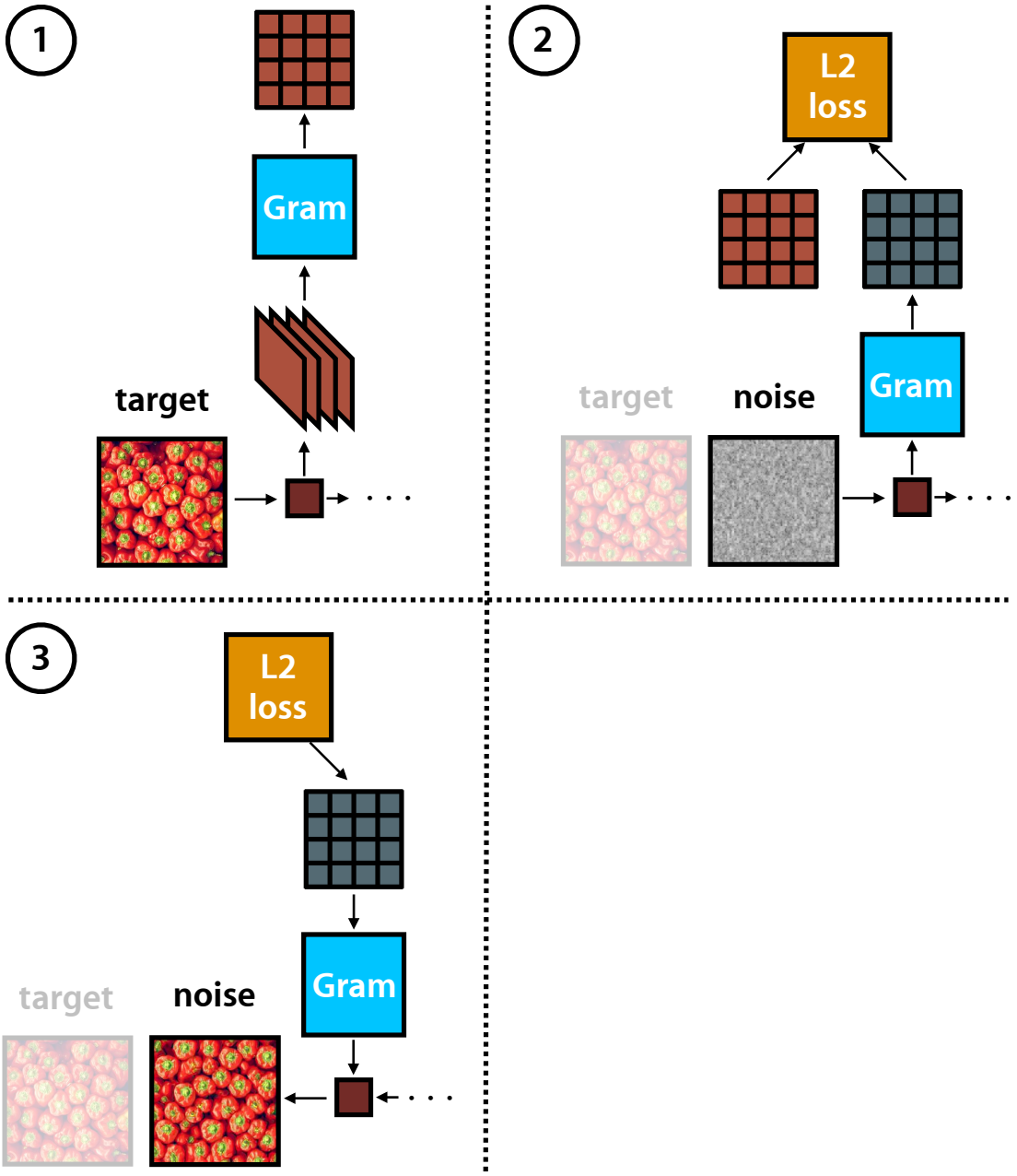


Figure 2.1: Gatys *et al.*'s [16] approach to static texture synthesis with the VGG-19 [49] convolutional network. Only the first layer of VGG-19 is shown. (1) An initial forward pass is performed with the target texture. Its Gram matrices across various layers are computed and saved. (2) The total L2 loss between the synthesized texture's Gram matrices and the target's is computed. (3) The loss is optimized with respect to the synthesized texture (with the weights of VGG-19 fixed), updating it to appear perceptually similar to the target.

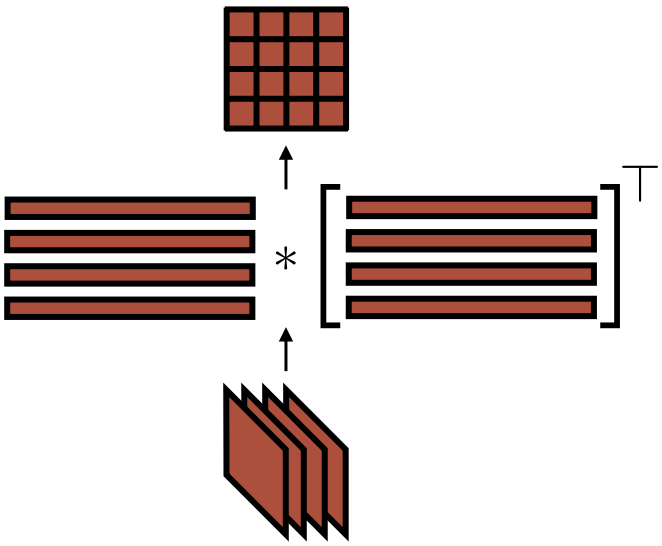


Figure 2.2: Computing the Gram matrix from the activation maps of a layer. Activation maps are first reshaped to row vectors, then the Gram matrix is realized by computing the inner product between the row-vectorized activation maps and their transpose.

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. Simply, the Gram matrix of a set of vectors $v_1, \ldots, v_n$ in an inner product space is the Hermitian matrix of inner products, whose entries are given by $G_{ij} = \langle v_i, v_j \rangle$ (a Hermitian matrix is a complex square matrix, A, that is equal to its own conjugate transpose, *i.e.*, $A = \overline{A^\top}$). Since the inner product space of activation maps is over the real number field, the Gram matrix is also a symmetric matrix. 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 texture synthesis with a ConvNet, the set of vectors used to compute the Gram matrix are row-vectorized activation maps (Fig. 2.2).

In texture synthesis, the Gram matrix measures the amount that co-located

features tend to activate together. 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 texture model was used in image style transfer [17], 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. 2.4 that enforced the synthesized texture to match the semantic content of the given content image. Specifically,

$$\mathcal{L} = \frac{1}{L_{style}} \sum_{l} \|\mathbf{G}^l - \hat{\mathbf{G}}^l\|_F^2 \; + \frac{1}{L_{content}} \sum_{l} \|\mathbf{A}^l - \hat{\mathbf{A}}^l\|_F^2 \; , \tag{2.5}$$

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.

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. Somewhat related to dynamic texture synthesis, Ruder *et al.* [43] extended the image style transfer model 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 texture on a per-frame basis. This is not to be confused with dynamic texture synthesis, which requires an analysis of *dynamic* textures across space *and time*.

Variants of linear autoregressive models have been studied [51, 9] 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.* [58] 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 approach proposed by this thesis, which assumes pre-trained fixed networks, this approach requires the ConvNet weights to be trained using the input texture prior to synthesis.

A recent preprint from Funke *et al.* [15] described preliminary results extending the framework of Gatys *et al.* [16] 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

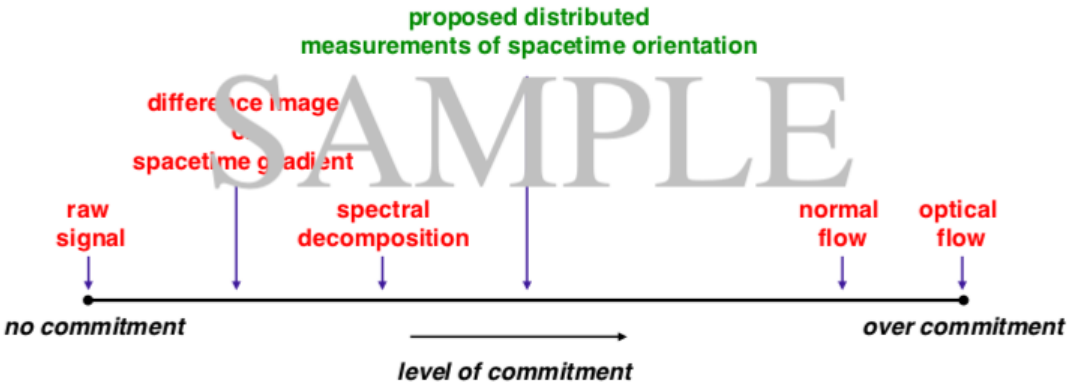


Figure 2.3: Dynamics representation spectrum (adapted from Derpanis [7]). Common abstractions of dynamics of temporal imagery and their respective level of commitment to an underlying model.

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. Adapted from Derpanis' dissertation on the role of representation in the analysis of visual spacetime [7], Fig. 2.3 illustrates an organization of several extant representations of temporal imagery dynamics. At one extreme, no commitment to an abstraction is made, the raw signal is used directly (*e.g.*, pointwise intensity and colour). This representation fails to leverage the rich underlying structure in the data. The remaining

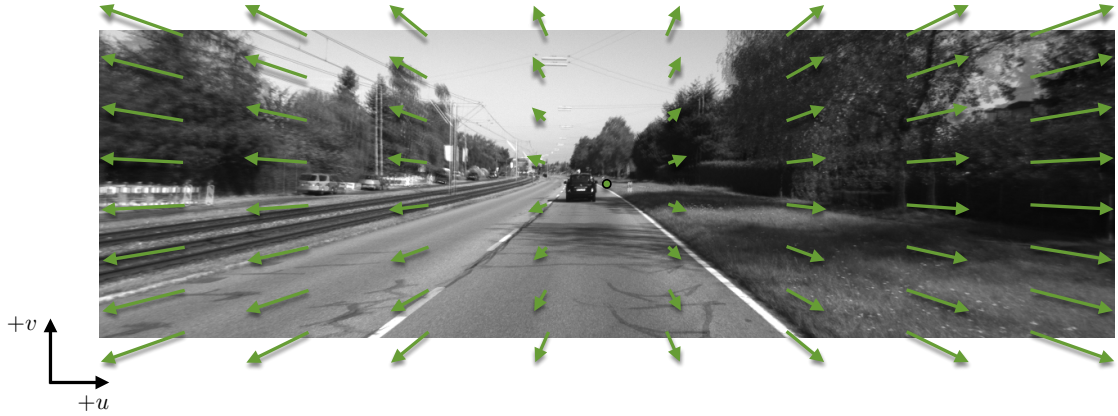


Figure 2.4: Visualizing optical flow. Optical flow is a 2D displacement vector field used to represent the apparent motion of image pixels between two consecutive frames, caused by the movement of objects or the camera. Pictured are two, super-imposed, consecutive frames taken from the KITTI dataset [19].

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 is a displacement vector consisting of a horizontal and vertical component, describing the movement of pixels from the first frame to the second. Fig. 2.4 provides a useful 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.*, [26, 35, 42]. Recently, ConvNet approaches have been demonstrated as viable alternatives [10, 27, 41, 59].

A limitation of optical flow is its over commitment to local translational motion, which only partially describes the dynamics one may encounter in the real world. Examples of dynamics optical flow would fail to capture include flickering, semi-transparent motion, and stochastic dynamics. These are dynamics typically existent in dynamic textures. One of the main causes of its failure is its dependency on assumptions like brightness constancy and local smoothness, which are generally difficult to justify for stochastic dynamics. Therefore, optical flow is not an optimal measure for representing the dynamics existent in dynamic textures.

2.4.2 Marginalized spacetime oriented energies

At the midpoint 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 which 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 visual spacetime that manifest themselves as motion or non-motion (*e.g.* flickering, stochastic dynamics, etc.). This is visualized in Fig. 2.5 alongside their frequency-space counterparts.

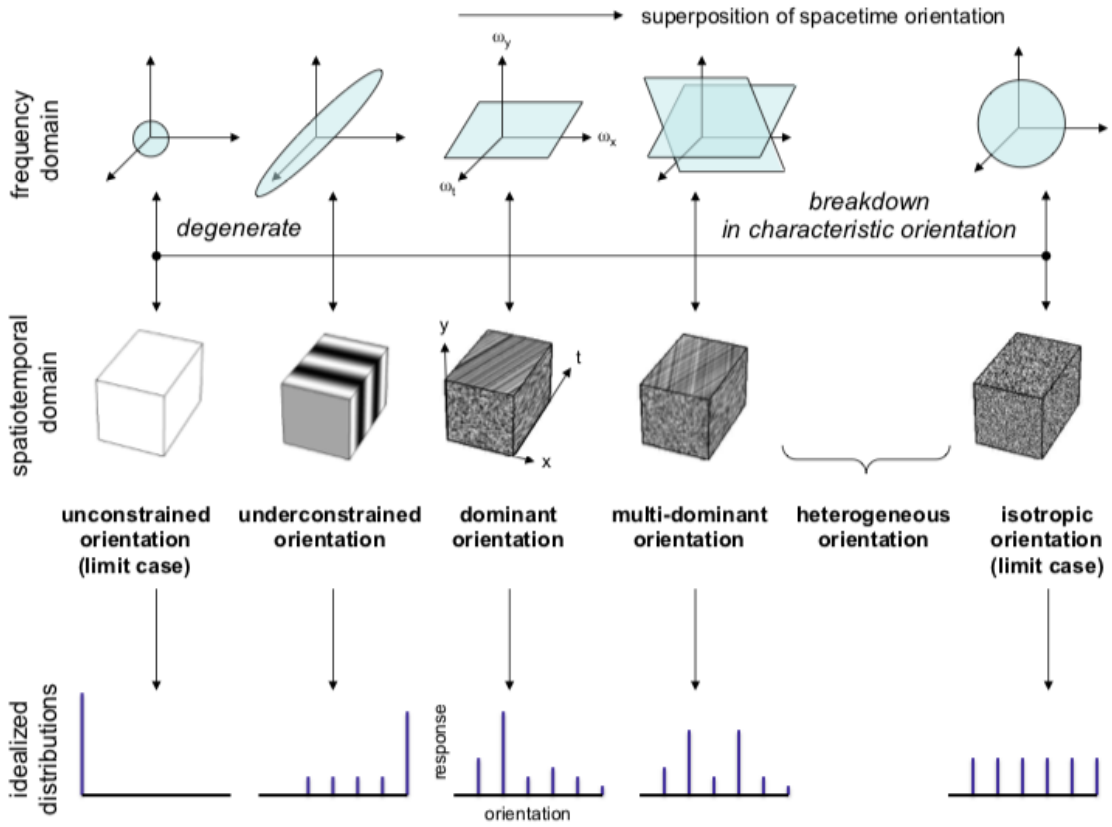


Figure 2.5: Dynamic texture spectrum (adapted from Derpanis [8]). The top and middle rows depict prototypical dynamic textures in the frequency and spatiotemporal domains, respectively. From left-to-right, an increasing amount of spacetime structures are superimposed in a texture. The bottom row depicts 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.

Specifically, this thesis adopts the Marginalized Spacetime Oriented Energy (MSOE) approach of Derpanis *et al.* [7] in representing the observed dynamics of a dynamic texture. This representation was successfully used for the task of dynamic texture recognition [8]; here it is used for the task of dynamic texture synthesis.

Most closely related to this approach are energy models of visual motion [2, 23, 47, 38, 8, 31] that have been motivated and studied in a variety of contexts, including computer vision, visual neuroscience, and visual psychology. 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. Here, an MSOE model motivates the representation of observed dynamics which will then be encoded as a ConvNet. Significantly, a completely analytically-defined oriented energy ConvNet model provides the current state-of-the-art for the related task of dynamic texture recognition [22].

In motion energy models, the velocity of image content (*i.e.*, motion) is interpreted as a 3D orientation in the x-y-t spatiotemporal domain [2, 13, 23, 47, 56], previously defined as a spacetime structure. In the frequency domain, the signal energy of a translating pattern can be shown to lie on a plane through the origin where the slant of the plane is defined by the velocity of the pattern (Fig. 2.5). Thus, motion energy models attempt to identify this orientation-plane (and hence the pattern's velocity) via a set of image filtering operations. More generally, 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 [8]. This observation suggests that motion energy models may form an ideal basis for the dynamics stream of the proposed dynamic texture synthesis ConvNet. As stated

previously, the spacetime-oriented energy model (MSOE) proposed by Derpanis *et al.* [7, 8] is used to motivate the network architecture. The MSOE model is reviewed here.

Matthew clean up redundant parts of MSOE review section in technical approach (**??**) Given input temporal imagery, $\mathbf{I} \in \mathbb{R}^{T \times H \times W}$ (time $\times$ height $\times$ 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:

$$E_{\hat{\theta}} = G_{3_{\hat{\theta}}} * \mathbf{I} \ , \tag{2.6}$$

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:

$$\bar{E}_{\hat{\theta}} = \sum_{(x,y,t) \in \Omega} E_{\hat{\theta}}(x, y, t)^2 \ . \tag{2.7}$$

At this point, each oriented energy measurement is confounded with spatial orientation. Consequently, in cases where the spatial image structure varies wildly about an otherwise coherent dynamic region, the responses of the bank of oriented filters will reflect this behaviour and thereby become dependent on spatial appearance; whereas, 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:

$$E_{\hat{\mathbf{n}}} = \sum_{i=1}^{N} E_{\hat{\theta}_i} \ , \tag{2.8}$$

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. For example, in the case where a spacetime structure is defined by the image velocity $(u, v)^\top$ (*i.e.*, optical flow), the unit normal is given by $\hat{\mathbf{n}} = (u, v, 1)^\top / ||(u, v, 1)^\top||$. 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:

$$\hat{E}_{\hat{\mathbf{n}}_i} = \frac{E_{\hat{\mathbf{n}}_i}}{\sum_{j=1}^{M} E_{\hat{\mathbf{n}}_j} + \epsilon} \ , \tag{2.9}$$

where $\hat{E}_{\hat{\mathbf{n}}_i}$ denotes an oriented filter response from Eq. 2.8 corresponding to a plane in frequency-space with unit normal $\hat{\mathbf{n}}_i$.

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 these activation statistics. 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 spatial texture model introduced by Gatys *et al*. [16] which was reviewed in the previous chapter **Matthew** talk about Gatys style transfer in related work (**??**). **Matthew** talk about texture synthesis in related work (**??**) To briefly review, the key idea is that feature auto-correlations (*i.e*., normalized *Gram* matrices) in a ConvNet trained

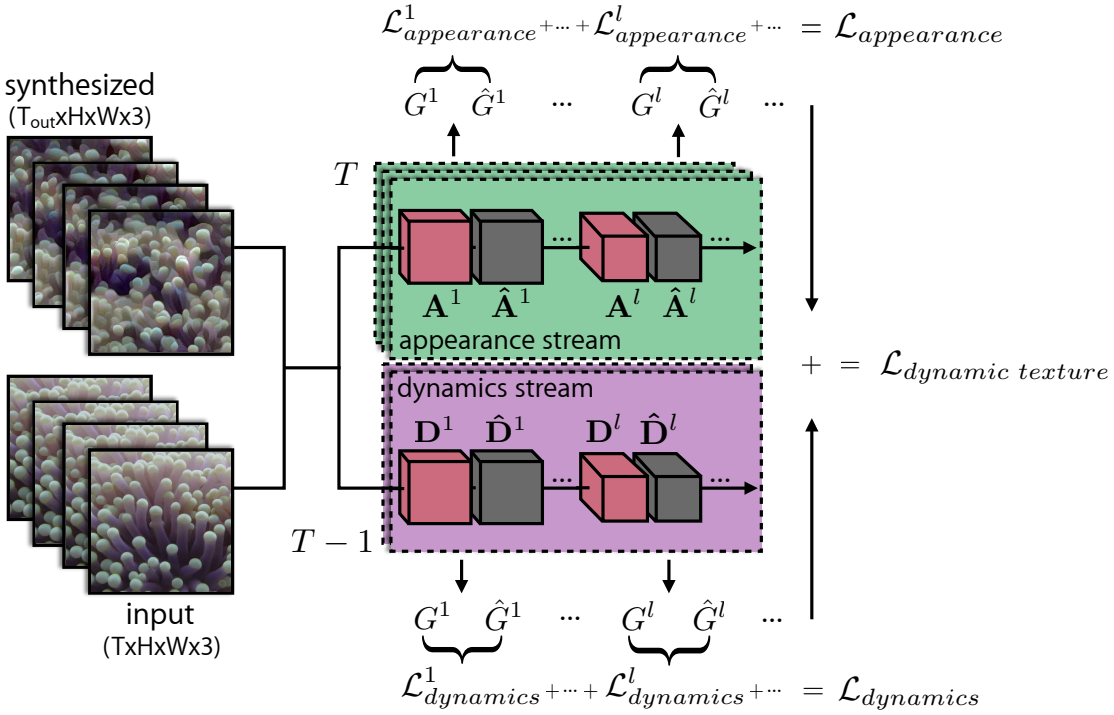


Figure 3.1: Two-stream dynamic texture generation. Two sets of Gram matrices represent a dynamic texture's appearance and dynamics. Matching these statistics allows for the generation of novel textures as well as style transfer between textures. Here, G^l and $\hat{G}^l$ are the Gram matrices of activations 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.

for object recognition (*e.g.*, VGG-19 [49]) capture texture appearance. The same publicly available normalized VGG-19 ConvNet [49] used by Gatys *et al.* [16] is used here.

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:

$$G_{ij}^l = \frac{1}{TN_lM_l} \sum_{t=1}^{T} \sum_{k=1}^{M_l} A_{ik}^{lt} A_{jk}^{lt} \ , \tag{3.1}$$

where T denotes the number of input frames and A_{ik}^{lt} 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:

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

where $\hat{A}_{ik}^{lt}$ 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 then 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 [16], Gram matrices are computed on the following layers: *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4*.

3.2 Texture model: Dynamics stream

There are three primary goals in designing the dynamics stream ConvNet.

1. The activations of the ConvNet must 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 must 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 **Matthew** talk about optical flow in related work (**??**)

(*e.g.*, [10, 27]) 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 [8, 47]. **Matthew** talk about SOE for texture recognition in related work? (**??**)

3.2.1 Review: Marginalized spacetime oriented energies

In motion energy models, the velocity of image content (*i.e.*, motion) is interpreted as a 3D orientation in the x-y-t spatiotemporal domain [2, 13, 23, 47, 56]. **Matthew** include figure of motion in xyt (**??**) In the frequency domain, the signal energy of a translating pattern can be shown to lie on a plane through the origin where the slant of the plane is defined by the velocity of the pattern. **Matthew** include figure of motion in frequency space (**??**) Thus, motion energy models attempt to identify this orientation-plane (and hence the pattern's velocity) via a set of image filtering operations. More generally, 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 [8]. This observation suggests that motion energy models may form an ideal basis for the dynamics stream. Specifically, the spacetime-oriented energy models proposed by Derpanis *et al.* [8] and Simoncelli *et al.* [47] are used to motivate the network architecture, which is briefly reviewed here; see Chapter 2 **Matthew** talk about Derpanis' MSOE in related work (**??**) for a more in-depth description.

Given an input video, a bank of oriented 3D filters, which are sensitive to a range of spatiotemporal orientations, are applied. These filter activations are then rectified (squared) and pooled over local 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. At this point, each oriented energy measurement is confounded with spatial orientation. Consequently, in cases where the spatial image structure varies wildly about an otherwise coherent dynamic region, the responses of the bank of oriented filters will reflect this behaviour and thereby become dependent on spatial appearance; whereas, 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 activations consistent with the same temporal orientation (not necessarily the same spatial orientation) are summed. **Matthew** clarify with Kosta (**??**) These responses provide a pixelwise distributed measure of which orientations (frequency domain planes) 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 orientation or simply due to high image contrast. To address this ambiguity, an L_1 normalization is applied across orientation responses which results in a representation that is robust to local appearance variations but highly selective to spacetime orientation.

3.2.2 ConvNet architecture

Using this model as the basis, the following fully convolutional network [46] **Matthew** talk about this in relate 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 $\times$ height $\times$ width $\times$ 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" [54]), as follows:

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

where μ is the average pixel value of the input pair, σ is the standard deviation of the input pair, and ϵ is a small value to prevent dividing by zero. This step provides a level of invariance **Matthew** Rick: Given that you're using normalized, bandpass filters, is this preprocessing necessary? (**??**) to overall brightness and contrast, *i.e.*, global additive and multiplicative signal variations, easing the training process of the ConvNet. **Matthew** maybe talk about how it relates to batchnorm? (**??**) 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$\times$height$\times$width), resulting in an output of spacetime oriented energy measurements, **Matthew** Rick: Really only 2 taps in time? (**??**)

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

where E_F denotes the response of filter F (of size $2 \times 11 \times 11$) after a convolution, $*$, centered about $\mathbf{x} \equiv (t, x, y)$. Next, 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 orientation:

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

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: **Matthew** explain why this is not redundant due to contrast norm on input (**??**)

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

where $\| \cdot \|_1$ is the L_1 norm computed over the filter responses of all filters.

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:

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

where $(\cdot)_{\downarrow \times n}$ and $(\cdot)_{\uparrow \times n}$ denote an n-times downsample and upsample, respectively, 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.*, [2, 8, 47]) 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

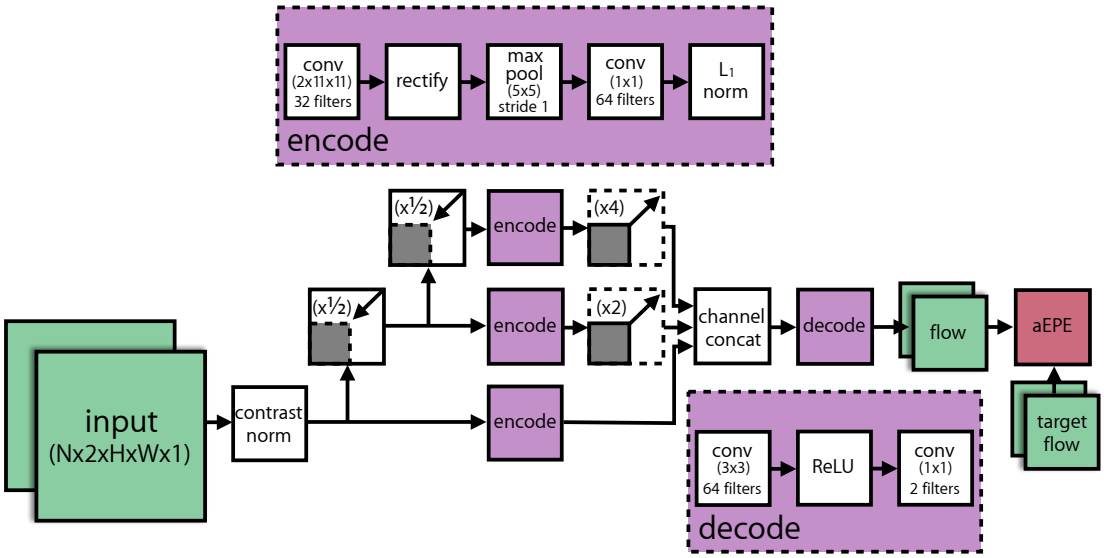


Figure 3.2: Dynamics stream ConvNet. The ConvNet is based on a spacetime-oriented energy model [8, 47] and is trained for optical flow prediction. Three scales are shown for illustration; in practice five scales are used.

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. **Matthew** introduce optical flow in related work as background (??) The proposed architecture is illustrated in Fig. 3.2.

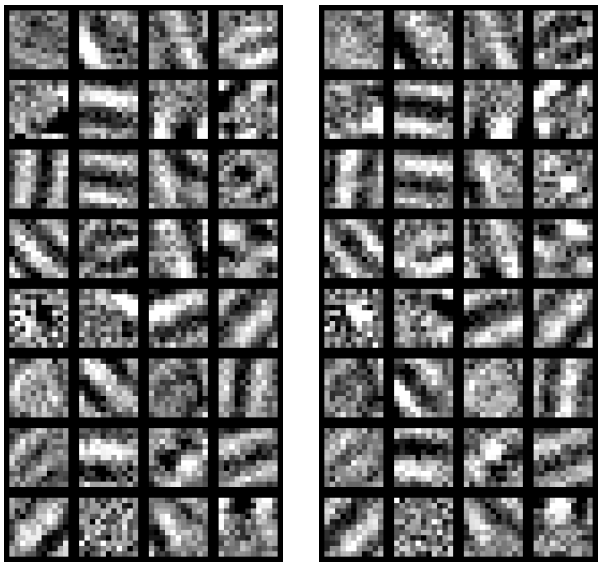
For training, the standard average endpoint error (aEPE) flow metric (*i.e.*, L_2 norm) is used between the predicted flow and the ground truth flow as the loss. Since no large-scale optical flow dataset exists that captures natural imagery with groundtruth flow, a new dataset had to be made. Videos from an unlabeled video dataset are fed through an existing flow estimator to produce optical flow "semi-groundtruths" for training, *cf*. [52]. For the unlabeled video dataset, the UCF101 dataset for action recognition [50] is used as it contains complex movements of natural imagery at desirably slow speeds **Matthew** histogram of flow magnitudes (??), similar to the expected pattern velocity of the dynamic textures used in this work. The

synthetic Flying Chairs dataset [10] 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 less complex and faster movement of the rigid objects in Flying Chairs, which is undesirable for estimating motion of dynamic textures. For producing the optical flow semi-groundtruths, a ConvNet-based model, EpicFlow [42], is used for its superior performance over traditional handcrafted approaches and tendency to produce smooth predictions—ideal training data for the dynamics stream. **Matthew** talk about why EpicFlow and not FlowNet? (**??**)

The distribution of movement directions **Matthew** radial histogram of flow for ucf101 (**??**) 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 **Matthew** provide geo augmentations (**??**) similar to those used by FlowNet [10] are used to equalize the distribution of movement directions in the generated dataset. Additionally, photometric data augmentations **Matthew** provide photo augmentations (**??**) similar to those used by FlowNet [10] are used here as well. The aEPE loss is optimized using Adam [30]. 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 [8]. This is illustrated in Fig. 3.3.

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



(a) Frame 1 (b) Frame 2

Figure 3.3: 32 learned spacetime-oriented filters of size $2 \times 11 \times 11$. (a) and (b) each depict a temporal slice of the learned filters, 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 [8] (*e.g.*, row 3, col 1 captures rightward movement and row 8, col 1 captures down-right movement).

encapsulated by a Gram matrix, $\mathbf{G}^l \in \mathbb{R}^{N_l \times N_l}$, whose entries are given by:

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

where D^{lt}_{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, $\hat{\mathbf{G}}^{lt} \in \mathbb{R}^{N_l \times N_l}$, with entries:

$$\hat{G}^{lt}_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} \hat{D}^{lt}_{ik} \hat{D}^{lt}_{jk} \ , \tag{3.11}$$

where $\hat{D}^{lt}_{ik}$ 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:

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

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 α and β 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. **Matthew** provide explicit math for this (**??**) Variations in the resulting texture are found by initializing the optimization process using IID Gaussian noise. Consistent with previous work [16], L-BFGS [34] is used for optimization. Dynamic texture synthesis results are provided in Chapter 4.

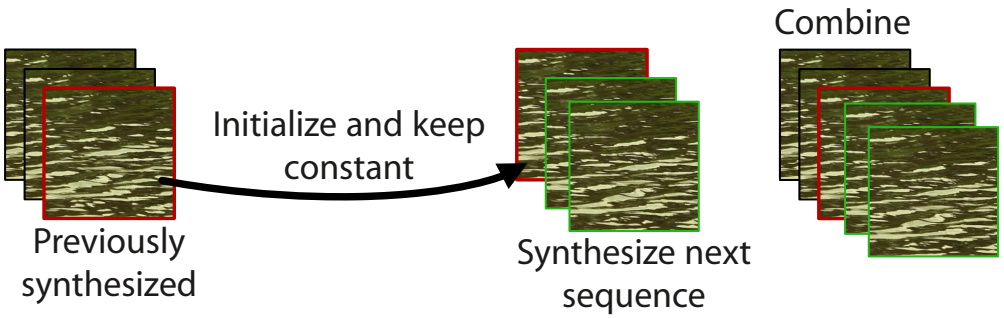


Figure 3.4: Incremental texture synthesis. Long sequences can be incrementally generated by separating the sequence into subsequences and optimizing them sequentially. The last frame (red) of a previously synthesized subsequence (left) is used to initialize the first frame of the next subsequence (middle)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

3.3.1 Incremental texture synthesis

Naive application of the outlined approach will consume increasing amounts of memory as the temporal extent of the dynamic texture grows; this makes it impractical to generate longer sequences. Instead, long sequences can be incrementally generated by separating the sequence into subsequences and optimizing them sequentially. This 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 ensures temporal consistency across synthesized subsequences and can be viewed as a form of coordinate descent for the full sequence objective as well as a form of autoregression. 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. Fig. 3.4 shows a visualization of the incremental texture synthesis process.

3.3.2 Temporally-endless texture synthesis

An interesting extension that was explored are 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 adding an additional loss to the dynamics stream that ties the last frame to the first. **Matthew** provide math (**??**)

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 has been explored previously for image style transfer [4, 18] with static imagery. This 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. **Matthew** provide figure showing how it works (**??**)

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 competing methods and a user study to quantitatively evaluate the realism of the results. Given their temporal nature, the results are best viewed as videos. The two-stream architecture was implemented using TensorFlow [1], an open source machine learning framework. Results were generated using an NVIDIA Titan X (Pascal) GPU and synthesis times ranged between one to three hours to generate 12 frames with an image resolution of 256×256. For the full synthesis results and source code, please refer to the supplemental material. **Matthew** link website? **(??)**

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 which were selected from the DynTex [39] database and others that were collected in the wild. 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 but readers are encouraged to view the videos to fully appreciate the results.

Fig. 4.1 shows some example success cases with the two-stream method where appearance and dynamics characteristics from the target are reliably preserved in the synthesized result. The upward fiery motion and flickering appearance of `fireplace_1`, the outward motion of the explosive splash of `lava`, the wispy fluid dynamics of `smoke_1`, the translating god rays and flowing vegetation in `underwater_vegetation_1`, and the downward rippling flow of `water_3`.

Failure modes

Example failure modes of the two-stream method are presented in Fig. 4.2. 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

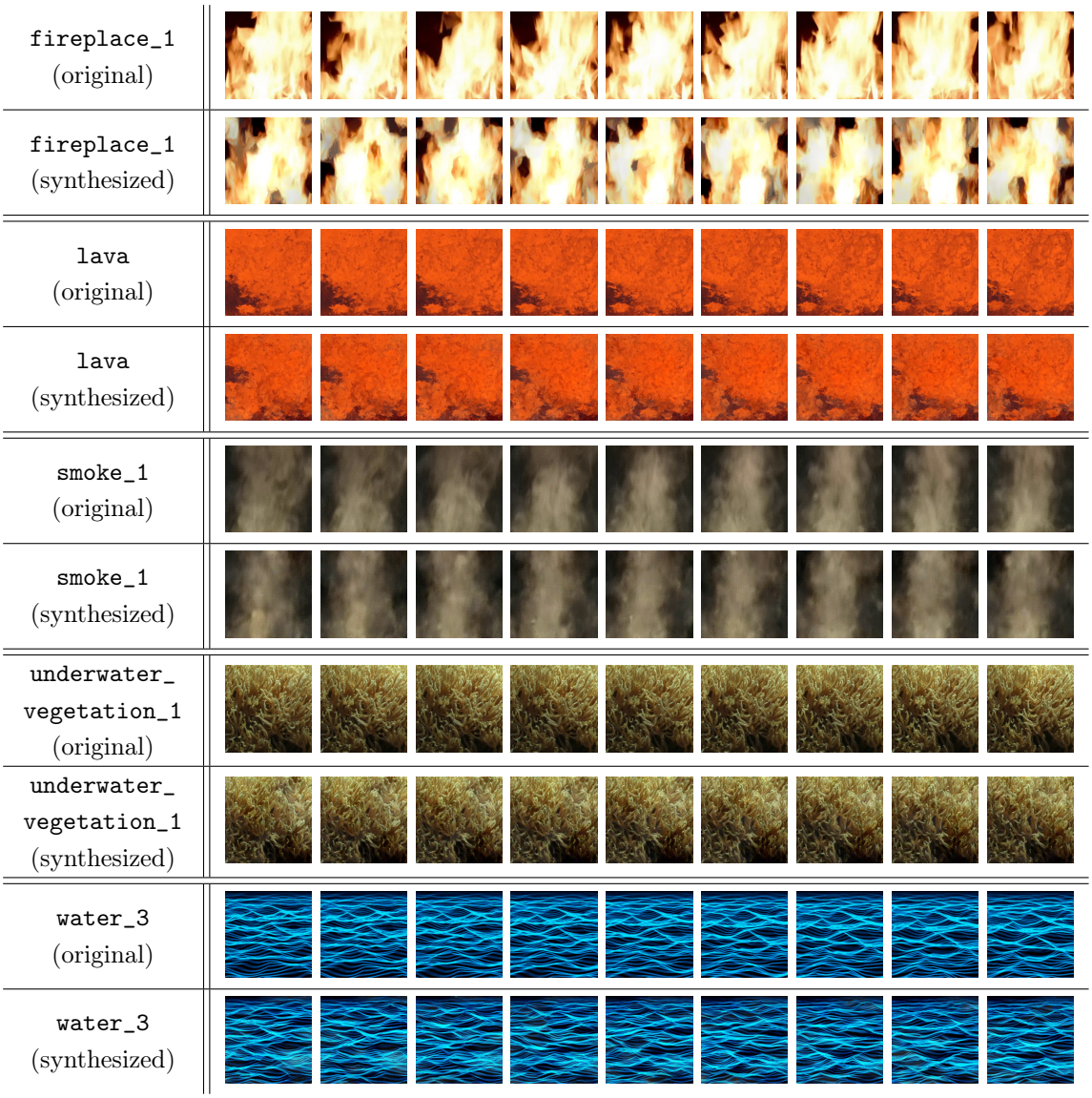


Figure 4.1: Dynamic texture synthesis success examples. Names correspond to files in the supplemental material.

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 [16] and suggests it is a limitation of the local nature of the Gram

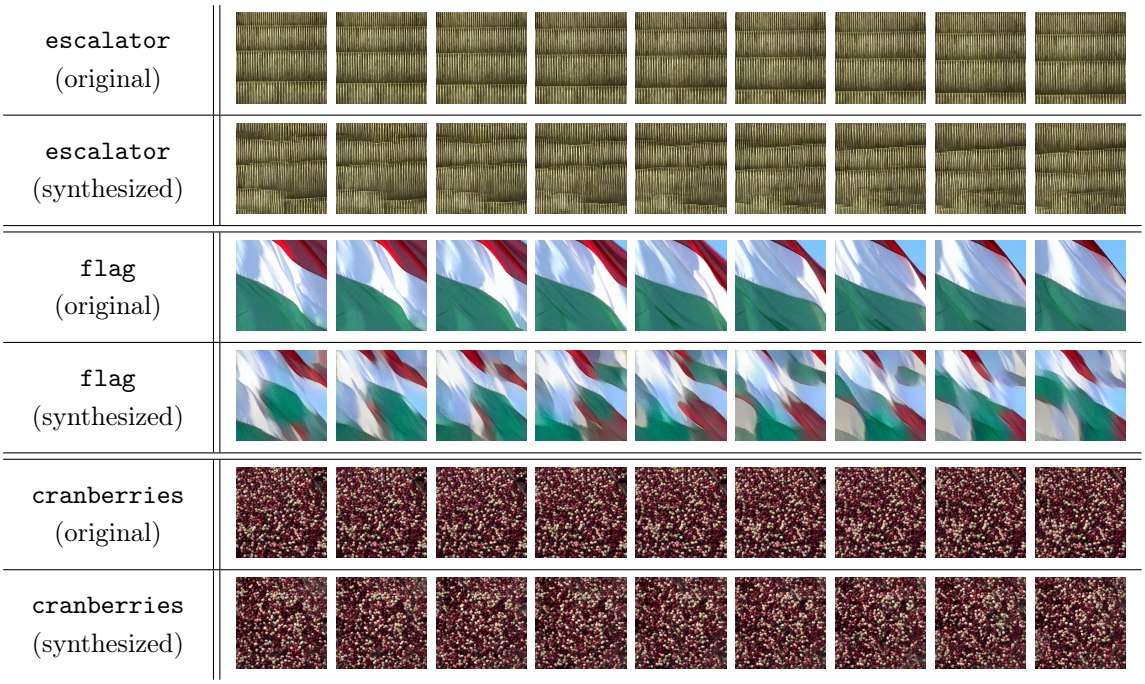


Figure 4.2: Dynamic texture synthesis failure examples. In these cases, the failures are attributed to either the appearance or the dynamics not being homogeneous.

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 generated texture does not faithfully reproduce the appearance; however, it does exhibit plausible rippling dynamics.

Also shown in Fig. 4.2 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.

Appearance vs. dynamics streams

Two experiments were held 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 generated 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. Results for a sequence containing a school of fish are shown in Fig. 4.3; 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 $[0, 255]$. **Matthew** maybe include? (**??**) This indicates a general invariance to appearance and suggests that the two-stream dynamic texture representation has factored appearance and dynamics, as desired.

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 batch process. **Matthew** elaborate? (**??**)

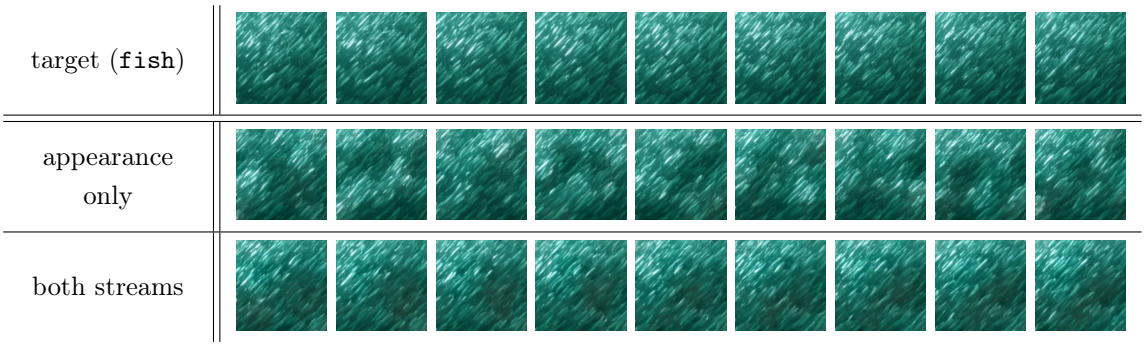


Figure 4.3: Dynamic texture synthesis versus texture synthesis. (top row) Target texture. (middle) Texture synthesis without dynamics constraints shows consistent per-frame appearance but no temporal coherence. (bottom) Including both streams induces consistent appearance and dynamics.

4.1.3 Temporally-endless texture synthesis

An example of a synthesized temporally-endless dynamic texture is shown in Fig. 4.4. As described in Sec. 3.3.2, the dynamic texture appears temporally endless, *i.e.*, there is no explicit beginning or end.

4.1.4 Dynamics style transfer

A dynamics style transfer result is shown in Fig. 4.5 (top), using two real videos as the appearance and dynamics target, respectively. Additional examples are available in the supplemental material. **Matthew** maybe show all examples in thesis? (**??**) When performing dynamics style transfer it is important that the appearance structure of both targets to be similar in scale and semantics, otherwise, the generated dynamic textures will look unnatural. For instance, transferring the dynamics of a flame onto a water scene will generally produce implausible results. **Matthew** show example of this failure case? (**??**)

The dynamics of a texture can also be applied to a static input image, as the

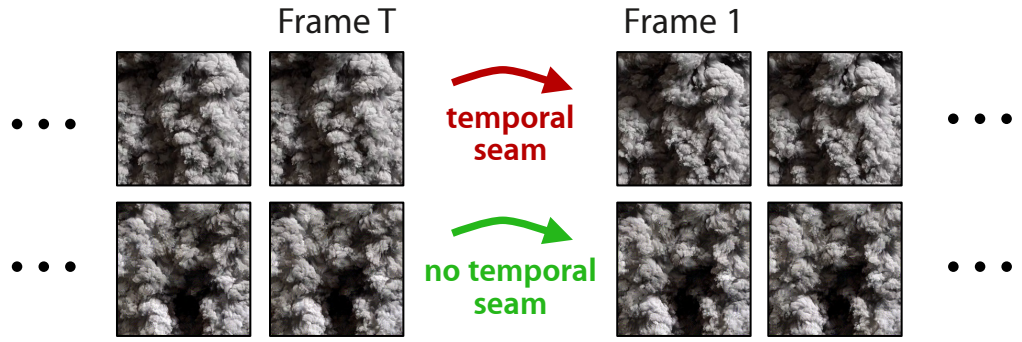


Figure 4.4: Temporally-endless texture synthesis. (top row) Target texture. (bottom row) Synthesized texture. By adding an additional loss to the dynamics stream that ties the last frame to the first, the synthesized dynamic texture appears to be temporally endless. Note the lack of an abrupt appearance change (*i.e.*, temporal seam) between the last frame and the first frame of the synthesized dynamic texture.

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. 4.5 (bottom), where the appearance is taken from a painting and the dynamics from a real world video.

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.*, [5]), 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

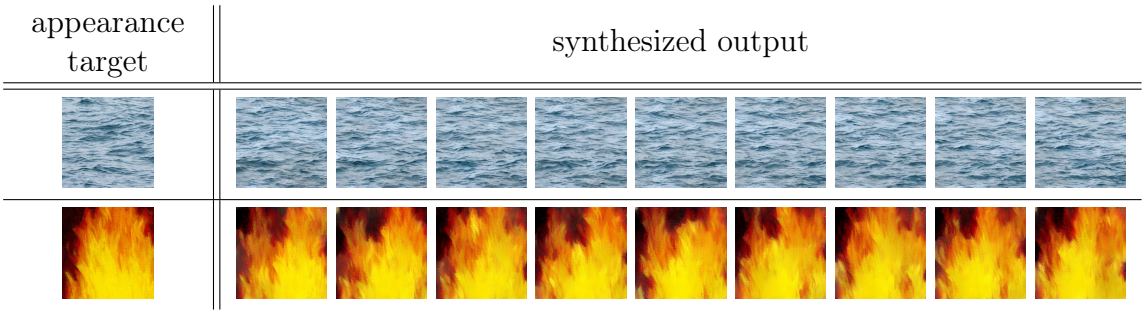


Figure 4.5: Dynamics style transfer. (top row) Appearance of still water was used with the dynamics of a different water dynamic texture (`water_4`). (bottom row) The appearance of a painting of fire was used with the dynamics of a real fire (`fireplace_1`). Animated results and additional examples are available in the supplemental material.

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 appeared more realistic after viewing the textures 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. See the appendix **Matthew** reference Appendix (??) for 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 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.

The results of this study are summarized in Fig. 4.6 which shows user accuracy in differentiating real versus generated textures as a function of time for both methods. Overall, users are able to correctly identify the real texture $66.1\% \pm 2.5\%$

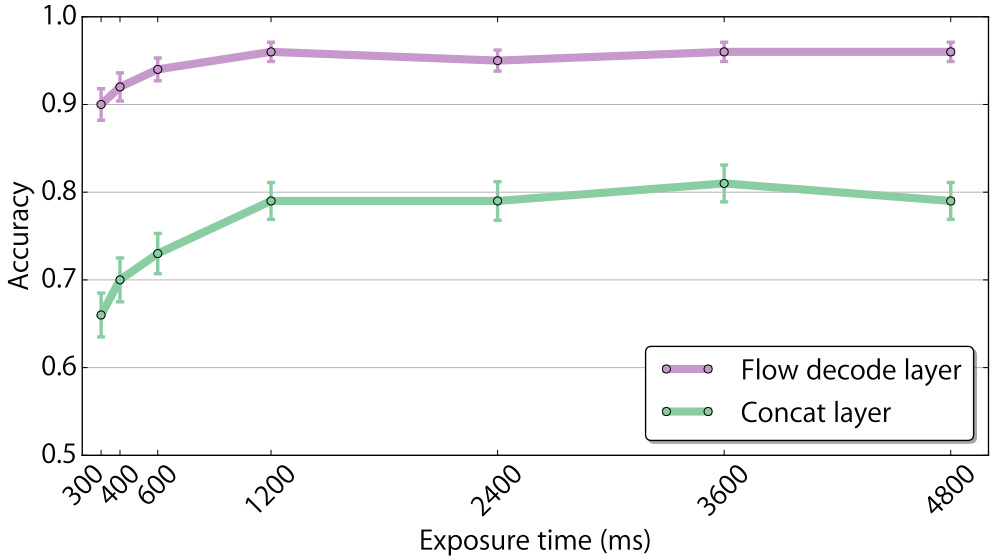


Figure 4.6: Time-limited pairwise comparisons across all textures with 95% statistical confidence intervals.

of the time for brief exposures of 0.3 seconds. This rises to $79.6\% \pm 1.1\%$ 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 generated textures and higher accuracy indicating less convincing textures.

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, **Matthew** provide reference? **(??)** particularly those with translucent and/or chaotic motion (*e.g.*, water, smoke, flames, etc.). Also evident in these results is the time-dependant nature of perception for textures from both models. Users' ability to identify the generated 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 [33] 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 juxtaposed with translating clouds in the sky). Results based on these groupings can be seen in Fig. 4.7.

A full breakdown of the user study results by dynamic texture and grouping can be found in the appendix. 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 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 [16]. In contrast, stochastic and near-stochastic dynamic textures performed significantly 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 $53.8\% \pm 7.4\%$ and $55.6\% \pm 7.4\%$, respectively, making them both statistically indistinguishable from real.

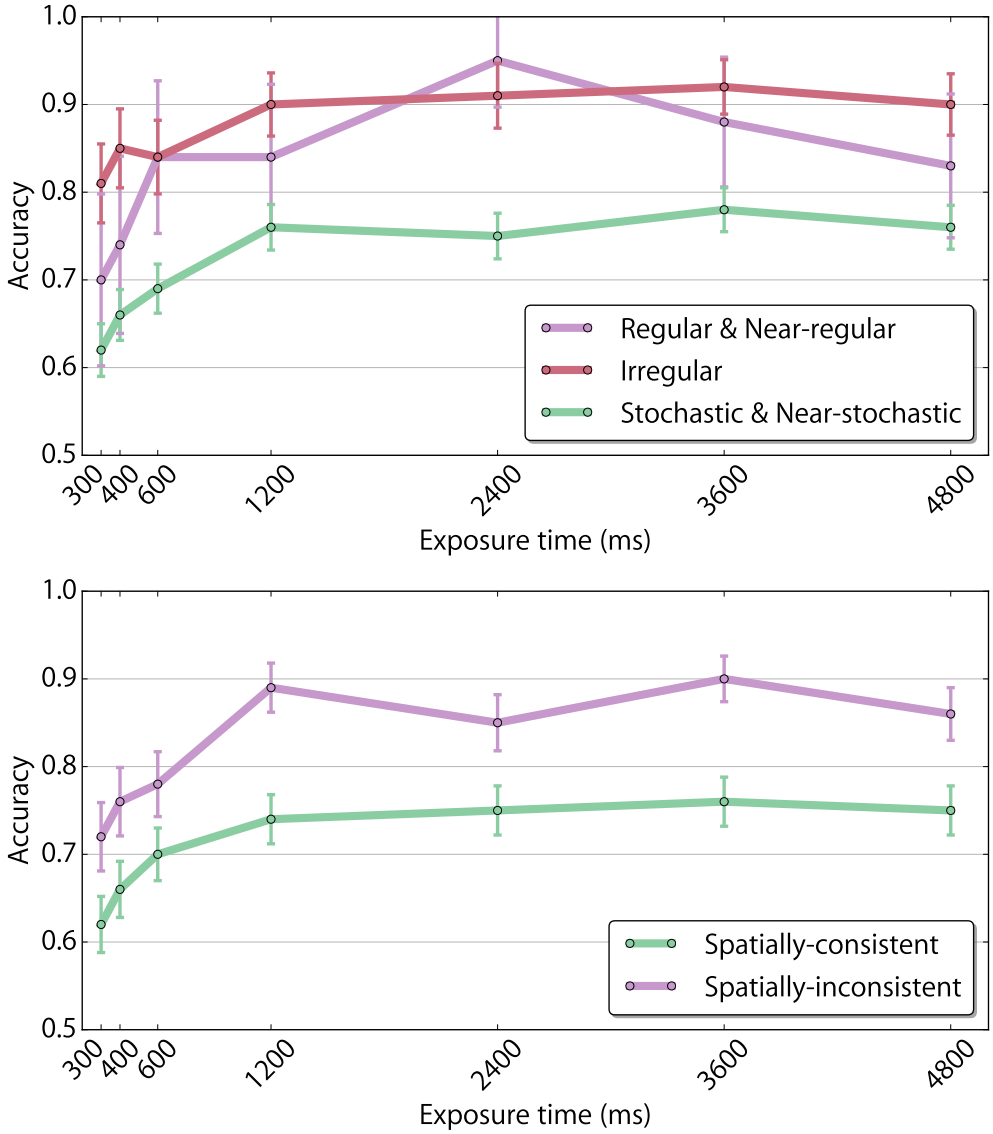


Figure 4.7: Time-limited pairwise comparisons across all textures, grouped by appearance (top) and dynamics (bottom). Shown with 95% statistical confidence intervals.

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 $48.6\% \pm 7.4\%$ and $63.2\% \pm 7.2\%$, respectively, while `candle_flame` and `snake_5` have average accuracies of $92.4\% \pm 4\%$ and $92.1\% \pm 4\%$, 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 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 competing methods of Funke *et al.* [15] and Xie *et al.* [58] is performed. **Matthew** explain their methods in related work (**??**) Generally, results from the proposed two-stream model are found to be qualitatively comparable or better than these methods.

To note, Funke *et al.* [15] 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`, `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.

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 and the same parameters used in their experiments are used here. For Funke *et al.*'s model [15], the parameters used are $\Delta t = 4$ and $T = 12$ **Matthew** explain these params (**??**) (recall that target dynamic textures consist of 12 frames). For the spatiotemporal and temporal models from Xie *et al.* [58], the parameters used are $T = 1200$ and $\tilde{M} = 3$. **Matthew** explain these params (**??**) A comparison between the results from the proposed two-stream model, Funke *et al.*'s [15] model, and Xie *et al.*'s [58] model on the nine dynamic textures are included in the folder `xie_and_funke` under `dynamic_texture_synthesis/comparisons`. **Matthew** also include figure (**??**) **Matthew** maybe leave folder descript Note for Xie *et al.* [58], comparisons are made with their spatiotemporal model (labeled "Xie et al. (ST)") designed for dynamic textures with both spatial and temporal homogeneity, and their temporal model (labeled "Xie et al. (FC)") designed for dynamic textures with only temporal homogeneity.

Overall, the results from the proposed two-stream model appear qualitatively better, showing more temporal coherence and similarity in dynamics and fewer artifacts, *e.g.*, blur and flicker. 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.* [58] is able to synthesize dynamic textures that lack spatial homogeneity (*e.g.*, `bamboo` and `escalator`), note that their method can not synthesize novel dynamic textures, *i.e.*, it appears to faithfully reproduce the target texture, reducing the applicability of their approach.

As a consequence of jointly modelling appearance and dynamics, the methods of [15, 58] are not capable of the novel form of style transfer that was demonstrated above. This was enabled by the factored representation of dynamics and appear-

ance. Furthermore, the spatiotemporal extent of the output sequence generated by Xie *et al.*'s [58] method is limited to being equal to the input. The proposed approach does not share this limitation.

Chapter 5

Conclusion

Matthew haven't touched yet (??)

In this paper, we presented a novel, two-stream model of dynamic textures using ConvNets to represent the appearance and dynamics. We applied this model to a variety of dynamic texture synthesis tasks and showed that, so long as the input textures are generally true dynamic textures, *i.e.*, have spatially invariant statistics and spatiotemporally invariant dynamics, the resulting synthesized textures are compelling. This was validated both qualitatively and quantitatively through a large user study. Further, we showed that the two-stream model enabled dynamics style transfer, where the appearance and dynamics information from different sources can be combined to generate a novel texture.

We have explored this model thoroughly and found a few limitations which we leave as directions for future work. First, much like has been reported in recent image style transfer work [17], we have found that high frequency noise and chromatic aberrations are a problem in generation. Another issue that arises is

the model fails to capture textures with spatially-variant appearance, (*e.g.*, `flag` in Fig. 4.2) and spatially-inconsistent dynamics (*e.g.*, `escalator` in Fig. 4.2). By collapsing the local statistics into a Gram matrix, the spatial and temporal organization is lost. Simple post-processing methods may alleviate some of these issues but we believe that they also point to a need for a better representation. 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.*, [28, 53].

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. 39

[2] 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. 21, 28, 31

[3] 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

[4] Alex J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv:1603.01768*, 2016. 38

[5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 45

[6] James E. Cutting. Blowing in the wind: Perceiving structure in trees and bushes. *Cognition*, 12(1):25 – 44, 1982. 6

[7] Konstantinos G. Derpanis. *On the Role of Representation in the Analysis of Visual Spacetime*. York University, 2010. 17, 21, 22

[8] 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. 1, 20, 21, 22, 28, 31, 32, 33, 34

[9] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51(2):91–109, 2003. 1, 16

[10] 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. 18, 28, 33

[11] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, pages 341–346, 2001. 3, 4

[12] 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. 10

[13] M. Fahle and T. Poggio. Visual hyperacuity: Spatiotemporal interpolation in human vision. *Proceedings of the Royal Society of London B: Biological Sciences*, 213(1193):451–477, 1981. 21, 28

[14] 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. 6

[15] Christina M. Funke, Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Synthesising dynamic textures using convolutional neural networks. *arXiv:1702.07006*, 2017. 16, 50, 51, 66

[16] 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. 3, 5, 11, 13, 16, 24, 25, 27, 36, 41, 48

[17] 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, 4, 6, 15, 53

[18] 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. 38

[19] 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. 18

[20] James J. Gibson. The perception of the visual world. 1950. 2

[21] Melvyn A. Goodale and A. David. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, 1992. 6

[22] Isma Hadji and Richard P. Wildes. A spatiotemporal oriented energy network for dynamic texture recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 21

[23] David J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision (IJCV)*, 1(4):279–302, 1988. 21, 28

[24] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, pages 229–238, 1995. 2, 5, 10

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

[26] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence (A.I.)*, 17:185–203, 1981. 18

[27] 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. 18, 28

[28] 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. 54

[29] Béla Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962. 10

[30] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014. 33

[31] Kishore Konda, Roland Memisevic, and Vincent Michalski. Learning to encode motion using spatio-temporal synchrony international conference on learning representation. In *International Conference on Learning Representations (ICLR)*, 2014. 21

[32] 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. 10

[33] Wen-Chieh Lin, James Hays, Chenyu Wu, Yanxi Liu, and Vivek Kwatra. Quantitative evaluation of near regular texture synthesis algorithms. In *CVPR*, volume 1, pages 427–434, 2006. 48

[34] Dong C. Liu and Jorge Nocedal. On the limited memory method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989. 36

[35] 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. 18

[36] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010. 9

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

[38] 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. 21

[39] Renaud Péteri, Sándor Fazekas, and Mark J. Huiskes. DynTex: A Comprehensive Database of Dynamic Textures. *Pattern Recognition Letters (PRL)*, 31(12), 2010. 40

[40] 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. 5, 10

[41] A. Ranjan and M. J. Black. Optical Flow Estimation using a Spatial Pyramid Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 18

[42] 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. 18, 33

[43] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition (GCPR)*, pages 26–36, 2016. 16

[44] 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. 11

[45] Arno Schödl, Richard Szeliski, David Salesin, and Irfan A. Essa. Video textures. In *SIGGRAPH*, pages 489–498, 2000. 10

[46] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(4):640–651, 2017. 29

[47] Eero P. Simoncelli and David J. Heeger. A model of neuronal responses in visual area MT. *Vision Research*, 38(5):743 – 761, 1998. 21, 28, 31, 32

[48] 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. 6

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 11, 13, 25

[50] 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. 32

[51] Martin Szummer and Rosalind W. Picard. Temporal texture modeling. In *IEEE International Conference on Image Processing (ICIP)*, pages 823–826, 1996. 16

[52] 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. 32

[53] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357, 2016. 54

[54] 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. 30

[55] 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

[56] 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. 21, 28

[57] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, pages 479–488, 2000. 10

[58] 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. 16, 50, 51, 52, 66

[59] J. J. Yu, A. W. Harley, and K. 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. 18

.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. 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:

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. **Matthew** maybe provide some examples? (**??**) Three of the 59 pairwise comparisons were sentinels and results from users which 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").

.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:

- `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.
- `using_concatenation_layer`: This folder contains synthesized dynamic textures where the concatenation layer was used for computing the Gramian on the dynamics stream. These are the results from the final model.
- `using_flow_decode_layer`: This folder contains synthesized dynamic textures where the predicted flow output is used for computing the Gramian on the dynamics stream. These are the results from the baseline.
- `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.
- `incrementally_generated`: 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 [15]. The dynamic textures chosen are those reported by Funke *et al.* [15] 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 [15], and Xie *et al.*'s [58]. 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.

.3 Full user study results

Figures 1a and 1b show histograms of the average user accuracy on each texture, averaged over a range of exposure times. The histogram bars are ordered from

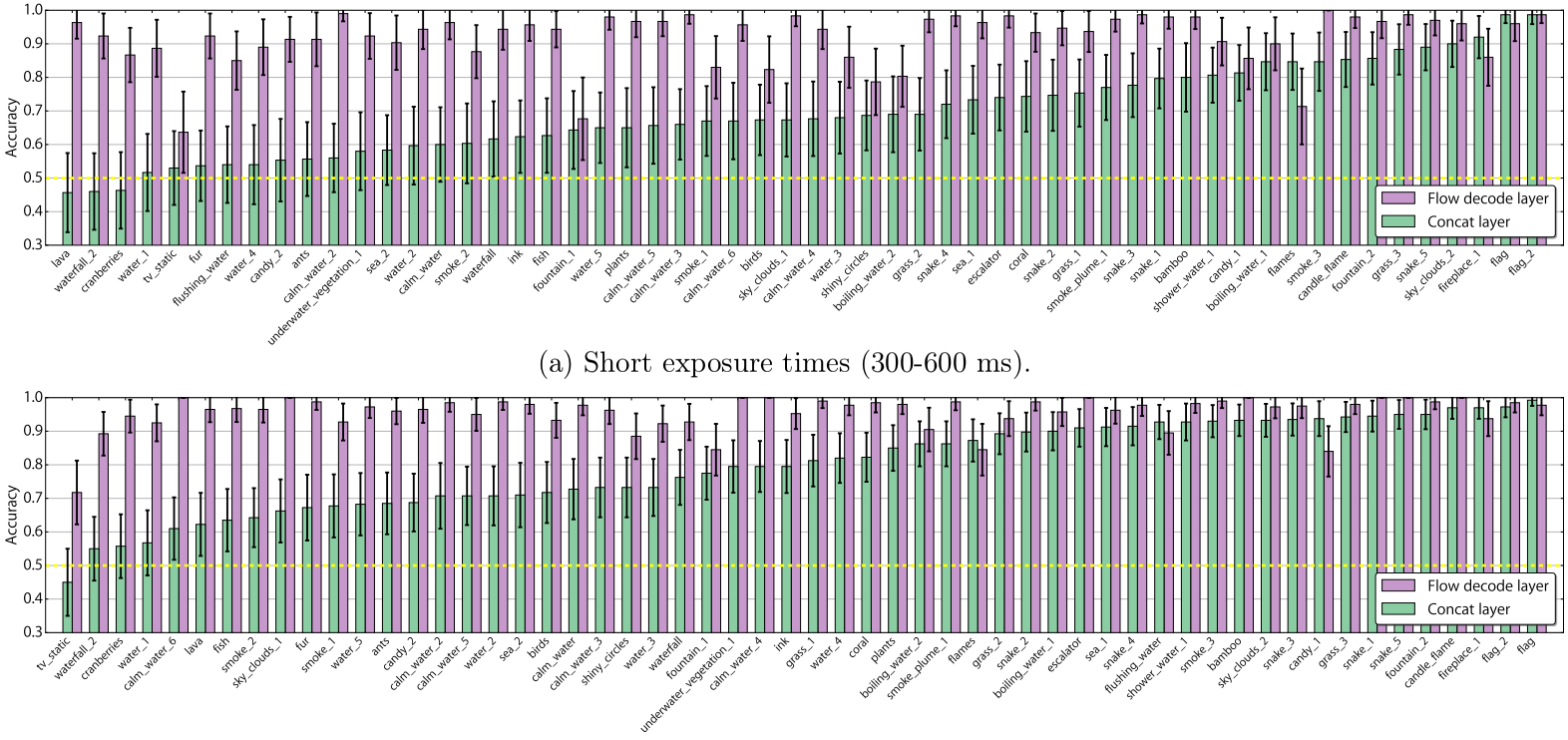
lowest to highest accuracy, based on the results when using the final model.

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

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

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

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



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

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

Figure 1: 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 1: Per-texture accuracies averaged over exposure times, 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 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 3: Per-texture accuracies averaged over exposure times, 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 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 5: Accuracies of textures grouped by appearances, averaged over exposure times, 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 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 7: Accuracies of textures grouped by appearances, averaged over exposure times, 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 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 9: Accuracies of textures grouped by dynamics, averaged over exposure times, 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 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 11: Accuracies of textures grouped by dynamics, averaged over exposure times, 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 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 13: Average accuracy over all textures, averaged over exposure times, 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 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 15: Average accuracy over all textures, averaged over exposure times, 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 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.