Material Synthesis Techniques from a Data-driven Perspective

Can Liu, Yinchen Xu, Qile Zhi

canliu, yxu72, qilezhi2@stanford.edu

Introduction

In computer graphics, materials are represented by RGB texture images, normal maps, and some other attributes. A generative model that automatically synthesizes paired textures and normals will be helpful for enriching the material assets. To tackle this problem, we need to

1. **Synthesize high quality, diverse, seamless textures of arbitrary size in real time;**
2. **Generate paired normal maps**

Experiments

- **Texture synthesis**: explored three different methods for texture synthesis: Gatys' Convolutional Neural Networks (CNN), Texture Networks, and Spatial Generative Adversarial Networks (SGAN)
- **Normal map**: trained a multilayer-perceptron to predict normal maps from generated RGB textures.

Dataset

- Paired RGB textures and normal maps of size 1024 × 1024, collected from graphics asset websites.
- 3 categories: woods, stones, walls
- Each image is sliced into 15K+ overlapping patches of size 256 × 256 for data augmentation

Features

- We use pre-trained CNNs (e.g. VGG19) to **extract features** at different levels
- We compute **feature correlation** (i.e. inner product between extracted feature maps) to capture non-localized texture information
- We adopt **hypercolumn representation** to combine features from different layers, which are used as inputs for normal map prediction

Methods

- **Neural Style Transfer [1]**
  - Pre-trained VGG19 extracts features from convolutional layers $\rightarrow F^1, F^2, \cdots, F^L$
  - Feed random noise and reference image into model and compute Gram matrices $G^l_{i,j}(\vec{x}) = < F^l_i(\vec{x}), F^l_j(\vec{x}) >$
  - Optimize over the pixels of generated images to minimize the Mean Square Error (MSE) between Gram matrices
- **Texture Networks [2]**
  - Generator takes multiple scales of noises $\vec{z}_i \in \mathbb{R}^{\frac{M}{2^i}\times\frac{M}{2^i}}$ where M is the size of the reference texture
  - Processes with a sequence of convolutional and activation layers, upsampling, and concatenation
  - Pre-trained VGG19 compares Gram matrices and updates the generator parameters
- **Spatial Generative Adversarial Networks [3]**
  - Adopt the architecture of the Deep Convolutional Neural Network (DCGAN)
  - Drop all fully connected layers to allow scalability: a spatial noise of height H and width W will be scaled 2^k larger after k (stride 2) deconvolutional layers
  - Optimize the generator with non-saturating loss to stabilize training
- **Normal Map Prediction**
  - Concatenate extracted features into hypercolumn representation for randomly sampled pixels
  - Train a multilayer-perceptron to predict normal for each pixel

Results

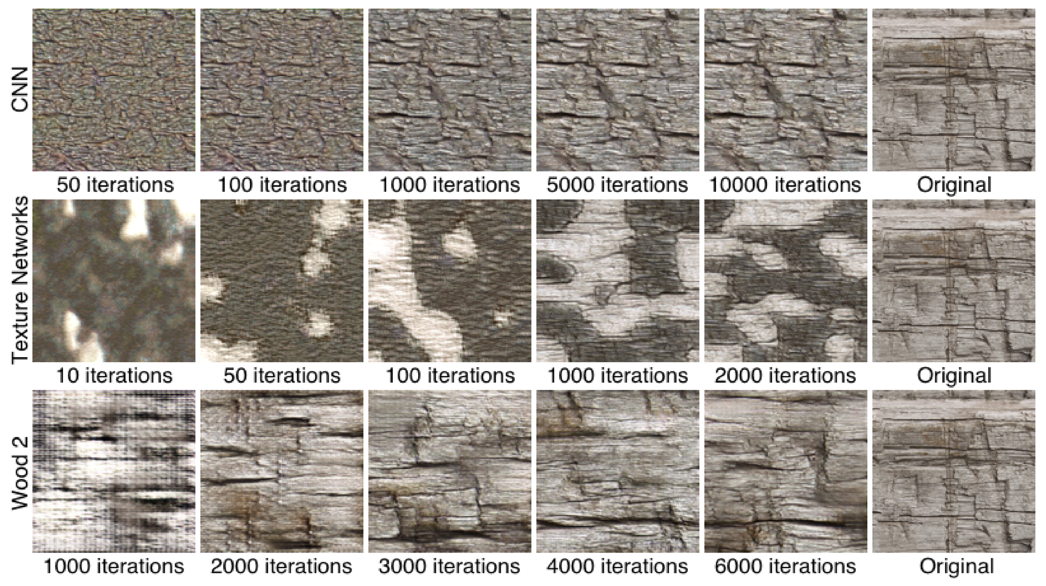

Fig 1 Training for different models

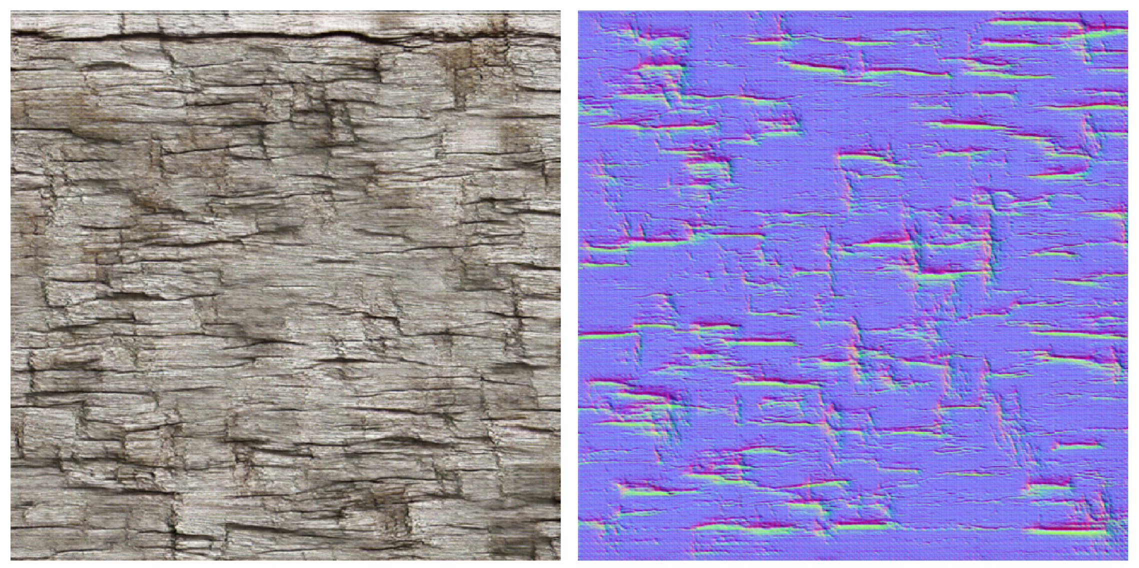

Fig 2 Generated image of size 1024*1024

	CNN + Duplication	TextureNet	SGAN
Wood Texture	12.1077	11.3018	12.5532
Stone Texture	13.7882	11.8861	N/A

Fig 3 Peak signal-to-noise ratio (PSNR)

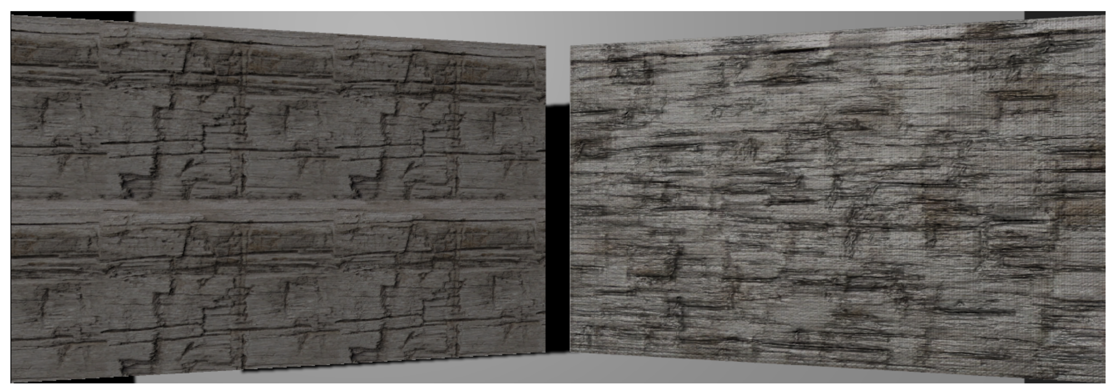

Fig 4 Original and Generated Textures

All involved models successfully generated result textures. CNN model cannot form flexible size texture and thus in the evaluation stage we duplicate the result in grid form. SGAN and TextureNet accept various size as input.

Discussion

- In Gram-matrix-based methods, extracting different layers produces different structures in synthesised textures; more layers extracted will result in better texture representation.
- Texture is a relatively *local* concept, so gram matrices of the feature maps, though losing spatial information, are able to capture its characteristic. The scaling mechanism in SGAN also leverages the "locality" property.
- In terms of efficiency, both Texture Networks and SGAN achieve real-time synthesis once the model is trained, while Neural Style Transfer Method requires optimization for each sample
- Using SGAN, we generated diverse and arbitrary size of normals with reasonable scales.

Future Work

- Experiment with different architectures and losses of GAN
- Explore unsupervised image-to-image translation to jointly generate paired RGB textures and normal maps.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks, 2015.

[2] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images, 2016.

[3] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks, 2016.