**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 → \( f_1, f_2, \ldots, f_k \)
  - Feed random noise and reference image into model and compute Gram matrices \( G_i(x) = \langle f_i(x), f_i(x) \rangle \)
  - 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 \( \mathbf{z}_i \in \mathbb{R}^{H \times W} \) 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 by \( 2^k \) 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**

**Discussion**

- In Gram-matrix-based methods, extracting different layers produces different structures in synthesised textures; more layers extracted will result in better texture representation.
- Normal maps: trained a multilayer-perceptron to predict normal maps from generated RGB textures.
  - Generative Adversarial Networks (GAN)
  - Spatial Generative Adversarial Networks

**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**