Penerapan Neural Style Transfer untuk Pentekstur Ulang Aset Gim

Addin Munawwar Yusuf / 13521085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521085@std.stei.itb.ac.id

***Abstrak*— Neural Style Transfer (NST) merupakan algoritma berbasis *neural network* yang berkembang di ranah pemrosesan citra digital modern. Algoritma NST dapat mengkonstruksi citra berdasarkan citra konten dan citra gaya, sehingga citra konten terlihat memiliki gaya seperti citra (referensi) gaya. Algoritma ini didasarkan pada arsitektur CNN, dengan metode pelatihan *gradient-descent* dan menggunakan *mean squared error* sebagai loss function antara citra kombinasi dengan citra input (citra konten dan citra gaya). Makalah ini melakukan eksplorasi untuk melakukan pentekstur ulang terhadap aset gim.**

Kata Kunci — Neural Style Transfer (NST), Gim, Pentekstur Gradient-Descent

I. Pendahuluan

Di dalam pengembangan gim, pentekstur aset merupakan salah satu proses yang sangat penting. Pentekstur adalah sebuah proses mengaplikasikan citra dua dimensi (yang dikenal sebagai tekstur), ke objek atau model tiga dimensi untuk memberikan karakteristik/penampilan tertentu. Tekstur tersebut dapat mendefinisikan karakteristik dari objek tersebut, seperti warna, material, serta detail dari permukaan tersebut, seperti tingkat kekasaran, tingkat pemantulan cahaya, dan lain-lain. Tanpa pemberian tekstur, objek 3D akan terlihat kurang menarik, karena hanya menampilkan satu warna saja.

Tekstur pada dasarnya adalah sebuah citra dua dimensi. Maka dari itu, teknik-teknik pemrosesan citra digital dapat dilakukan, selayaknya pada citra lainnya. Hal ini mencakup *image enhancement, image restoration, feature extraction,* dan lain sebagainya. Dengan melakukan pemrosesan citra digital terhadap citra-citra tekstur, kita dapat memberikan karakteristik-karakteristik tertentu yang akan muncul di dalam objek 3D di dalam gim. Kita juga dapat memberikan gaya tertentu terhadap tekstur, sehingga tampilan dari gim akan memiliki gaya yang unik, sesuai dengan karakteristik gaya yang kita pilih.

Seiring berkembangnya kecerdasan buatan, metode-metode baru yang mengeksplorasi pemrosesan citra digital muncul. Salah satu perkembangan terbaru di dalam bidang ini adalah Neural Style Transfer (NST) yang dikembangkan oleh Leon et al. pada tahun 2015. NST merupakan teknik pemrosesan citra digital yang memanfaatkan deep learning yang berbasis pada arsitektur Convolutional Neural Network (CNN). NST menggunakan dua citra sebagai masukan, yaitu citra konten dan citra gaya (misalnya lukisan dari pelukis terkenal). Citra konten akan dipadukan dengan citra referensi gaya, sehingga citra konten akan terlihat seperti dilukis dengan gaya dari citra referensi.

Gaya itu sendiri didefinisikan suatu cara dalam melakukan sesuatu, terutama yang berkaitan dengan orang, kelompok, tempat, maupun suatu periode waktu. Di dalam suatu citra atau gambar, gaya umumnya mereferensikan gaya artistik. Hal tersebut mencakup elemen material, warna, goresan, hingga hal-hal yang lebih abstrak seperti objek dan bentuk.

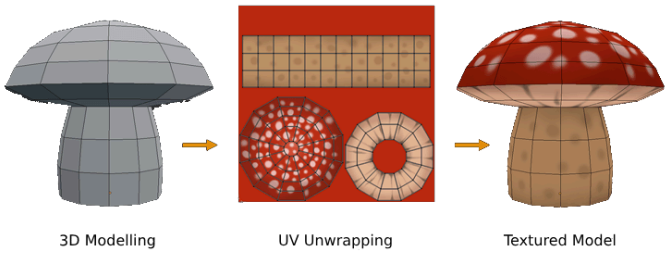
Di dalam pengembangan gim, gaya dari gim merupakan topik yang sangat penting. Penggayaan di dalam gim berkaitan dengan pemilihan desain dan gaya artistik secara sadar oleh pengembang untuk memberikan pengalaman khusus terhadap pemain. Gaya dari suatu gim dapat membangung identitas dari gim, meningkatkan imersi dari pemain, serta mempengaruhi perasaan dan pengalaman dari pemain.

Pada bagian selanjutnya, penulis akan membahas lebih lanjut mengenai pentekstur, landasan dari NST, metode pelatihannya, serta melakukan eksperimen untuk pentekstur ulang aset dari gim dengan menggunakan gaya tertentu.

II. Landasan Teori

A. Pentekstur

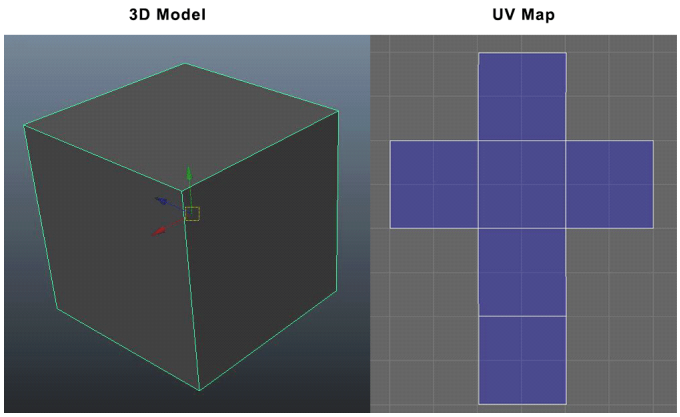
Pentekstur adalah sebuah sebuah proses mengaplikasikan citra dua dimensi (yang dikenal sebagai tekstur), ke objek atau model tiga dimensi. Tekstur di dalam gim memberikan karakteristik penampilan tertentu dari model 3D.



Gambar 1. Proses Pentekstur Model 3D
Sumber: BlenderNation

Penteksturan dapat dilakukan dengan berbagai macam teknik, akan tetapi teknik yang paling umum digunakan adalah dengan menggunakan UV mapping.

UV mapping adalah sebuah proses memetakan vertex dari objek tiga dimensi, ke dalam bidang dua dimensi. Masing-masing vertex pada objek 3D dipetakan ke dalam koordinat yang berada di tekstur 2D. Proses pemetaan ini disebut sebagai proses UV unwrapping.

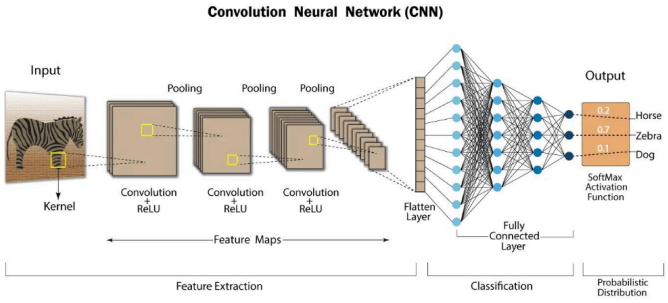


Gambar 2. Contoh UV Mapping pada objek berbentuk kubus
Sumber: TheAppGuruz

Setelah model 3D sudah terpetakan ke dalam UV map (seperti pada Gambar 2), kita dapat memetakan tekstur yang ingin diaplikasikan ke model ke dalam UV map yang sudah terbentuk. Proses ini dapat dilakukan dengan menggunakan menggunakan *software image editing*. Setelah tekstur sudah dibuat (berdasarkan *layout* dari *UV map*), tekstur dapat diaplikasikan ke model 3D.

B. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah jaringan syaraf tiruan yang populer dalam pembelajaran mendalam (*deep learning*). CNN dibuat untuk memproses data yang memiliki topologi seperti grid, contohnya citra. CNN merupakan kombinasi dari Artificial Neural Network (ANN) dan operasi konvolusi. Konvolusi digunakan untuk mengekstraksi fitur-fitur penting dari data input.



Gambar 3. Arsitektur Convolutional Neural Network
Sumber: https://developersbreach.com/convolution-neural-network-deep-learning/

Secara umum, CNN memiliki tiga lapisan utama:

1. *Convolutional Layer*

   Pada lapisan ini, dilakukan operasi konvolusi dengan sebuah operator yang disebut sebagai *filter* atau *kernel*. Hasil dari setiap *filter* ini akan mengekstraksi fitur tertentu dari suatu citra, sehingga hasil tersebut disebut sebagai *feature map*. Citra masukan biasanya akan dilewatkan terhadap sejumlah *filter*, sehingga hanya citra input dibuat lebih kompak dan abstrak, dengan hanya menyimpan fitur-fitur yang penting saja dari citra.

   Hasil dari konvolusi biasanya diikuti oleh lapisan tambahan berupa fungsi aktivasi, misalnya Rectified Linear Unit (ReLU). Lapisan tambahan ini memungkinkan pelatihan yang lebih cepat dan efektif, dengan menambahkan non-linearitas terhadap model. Hal ini memungkinkan jaringan untuk menangkap karakteristik yang kompleks dari data.

2. *Pooling Layer*

   Lapisan selanjutnya dari arsitektur CNN adalah *pooling layer*. *Pooling layer* bermanfaat untuk mengurangi daya komputasi yang diperlukan untuk memproses data. Hal ini dilakukan dengan cara mengurangi dimensi spasial dari matriks fitur dari hasil konvolusi. Hal ini juga bermanfaat untuk mereduksi *overfitting*, yang diakibatkan oleh data input yang terlalu detail. Terdapat beberapa cara untuk melakukan pengurangan dimensi tersebut, akan tetapi dua jenis yang paling populer adalah Max Pooling dan Average Pooling. Max Pooling mengembalikan nilai maksimum dari bagian gambar yang dicakup oleh *kernel*. Sementara itu, Average Pooling mengembalikan nilai rata-rata.

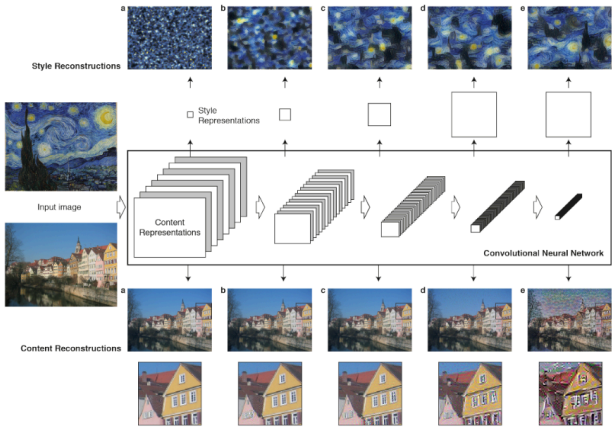
3. *Fully-Connected Layer*

   Lapisan terakhir dari arsitektur CNN adalah *fully-connected layer* (FC). Lapisan ini merupakan lapisan utama yang menyimpan jaringan syaraf seperti pada ANN tradisional. Operasi terakhir dari FC adalah fungsi aktivasi softmax, yang mengklasifikasikan citra input ke label tertentu, berdasarkan probabilitas tertentu.

C. Neural Style Transfer (NST)

Neural Style Transfer (NST) bekerja dengan menggunakan dua citra input, yatu citra konten dan citra gaya. Citra konten merupakan citra yang akan digayakan, sementara citra gaya adalah citra yang menjadi referensi dari gaya yang akan ditransfer.

Neural Style Transfer (NST) merupakan implementasi dari CNN. Secara khusus, NST menggunakan arsitektur jaringan VGG-19, yang merupakan varian dari CNN. VGG tersusun atas 19 layer, dan memiliki model *pretrained* yang telah dilatih dengan satu juta gambar dari basis data ImageNet. Model ini dapat mengklasifikasikan gambar ke dalam 1000 kategori objek, seperti *keyboard*, *mouse*, pensil, dan banyak jenis hewan. Secara umum, arsitektur NST adalah sebagai berikut.



Gambar 4. Arsitektur Neural Style Transfer (NST)
Sumber: Leon et al. (2015)

Pada Gambar 4 di atas, dapat terlihat bahwa pada arsitektur NST, hanya terdapat *convolutional layer* dan *pooling layer*. Pada NST, *fully-connected layer* tidak disertakan, karena tugas pada NST bukanlah melakukan klasifikasi, melainkan merekonstruksi citra konten dengan meniru gaya dari citra gaya.

D. *Gradient-Descent* dan *Backpropagation*

Sebelum melangkah lebih jauh ke dalam algoritma pelatihan NST, kita perlu memahami terlebih dahulu metode pelatihan yang umum dilakukan pada *neural-network*, yaitu *gradient-descent* dan *backpropagation*.

Gradient-descent adalah metode pelatihan yang umum digunakan pada pembelajaran mesin. *Gradient-descent* melakukan pelatihan dengan berusaha meminimumkan *error/loss* antara hasil prediksi dengan hasil yang diharapkan. Hal ini dilakukan dengan menghitung gradien, yang merupakan turunan parsial dari fungsi *loss* terhadap bobot.

$$\nabla E = \partial E / \partial w$$

Formula 1. Gradien error dari prediksi model dengan *expected output*

Setelah gradien ditemukan, bobot disesuaikan dengan menggerakkannya ke arah yang berlawanan dengan gradien (∇E), sehingga bobot lebih terarah menuju bobot yang baik (memiliki *error* yang rendah).

$$w' = w - \eta \nabla E$$

Formula 2. Pembaruan bobot pada *gradient-descent*

η disebut sebagai *learning rate*, parameter yang digunakan untuk mengatur seberapa sensitif pembaruan bobot terhadap error. Proses penyesuaian bobot ini dilakukan dari layer paling akhir hingga paling awal, sehingga proses ini dinamakan sebagai proses *backpropagation.*

E. *Pelatihan NST*

Pelatihan NST didasarkan pada algoritma *gradient-descent* dan *backpropagation.* Terdapat tiga citra yang digunakan dalam pelatihan NST. Citra tersebut adalah citra konten, citra gaya, dan citra kombinasi. Citra kombinasi merupakan hasil penggabungan antara citra konten dengan citra gaya.

Dalam pelatihan NST, citra kombinasi dilewatkan melalui *convolutional* dan *pooling layer*, sehingga menghasilkan abstraksi berupa *feature maps* dari citra kombinasi didapatkan. *Feature maps* ini menyimpan informasi-informasi utama dari citra kombinasi. Setelah itu, sesuai dengan prinsip dari *gradient-descent*, *loss*/*error* dari citra kombinasi dihitung, untuk melakukan penyesuaian bobot dengan *backpropagation*. Loss dari citra kombinasi didefinisikan sebagai jumlah dari *loss* terhadap citra konten dan *loss* terhadap citra gaya.

$$Loss(\text{Kombinasi}) = Loss(\text{Konten}) + \beta \cdot Loss(\text{Gaya})$$

Formula 3. Fungsi Loss dari citra kombinasi

Dalam hal ini, β menjadi parameter yang digunakan untuk mengatur seberapa besar pengaruh dari citra gaya, dibandingkan dengan citra konten.

Dengan mengetahui fungsi *loss* dari citra konten dan fungsi *loss* dari citra gaya, kita dapat mendapatkan fungsi *loss* dari citra kombinasi, menghitung *gradien* dari loss, lalu melakukan *backpropagation* untuk menyesuaikan bobot model hingga konvergen.

1) *Mean Squared Error* (MSE)

Mean Squared Error (MSE) adalah sebuah metrik matematis yang digunakan untuk mengukur error/selisih antara dua data. Dalam konteks citra, maka data yang dimaksud adalah elemen-elemen di dalam 2D-grid. MSE menghitung kuadrat dari selisih antara dua citra. Kuadrat digunakan, untuk mengabaikan nilai positif/negatif pada selisih yang dihasilkan. Dalam citra dua dimensi, formula dari MSE adalah:

$$MSE(I1, I2) = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I1(i,j) - I2(i,j))^2$$

Formula 4. *Mean Squared Error* pada dua citra

M dan N merupakan dimensi dari citra, sementara I(i, j) merupakan nilai piksel citra I pada koordinat (i, j).

2) Fungsi *loss* terhadap citra konten

Ketika menghitung *error* antara citra kombinasi dengan citra konten, kita tidak bisa melakukan perbandingan piksel per piksel. Dengan demikian, perbandingan *error* antara citra kombinasi dengan citra konten dilakukan pada tingkat yang lebih abstrak. Informasi citra yang abstrak ini dapat kita peroleh pada layer-layer akhir pada *convolutional layer*. Dalam konteks NST dengan arsitektur VGG-19, maka perhitungan *error* akan dilakukan pada layer terakhir, yaitu pada layer ke-19, dimana informasi citra sudah memiliki abstraksi yang lebih tinggi. *Error/loss* tersebut dihitung berdasarkan *mean squared error* (MSE) antara kedua citra. Formula untuk error tersebut adalah sebagai berikut.

$$Loss(\text{Konten}) = \text{MSE}(\text{Konten}^{[19]}, \text{Kombinasi}^{[19]})$$

Formula 5. Fungsi loss dari citra konten

Simbol [19] pada Konten$^{[19]}$ menunjukkan bahwa citra yang dibandingkan adalah *feature maps* yang dihasilkan setelah melalui layer ke-19 pada *convolutional layer*.

3) Fungsi *loss* terhadap citra gaya

Menghitung *loss* antara citra kombinasi dengan citra gaya lebih rumit. Berbeda dengan citra konten, informasi dari citra gaya dapat diperoleh dari berbagai macam level abstraksi, dari level rendah hingga level tinggi. Hal ini dikarenakan gaya dapat muncul pada tingkat piksel (misalkan pada *outline*), hingga tingkat tinggi (misalkan bentuk-bentuk pusaran pada lukisan Van Gogh). Maka dari itu, beberapa layer (dari level abstraksi rendah hingga ke abstraksi tinggi) digunakan sebagai sampel untuk mendefinisikan gaya dari citra tersebut.

Di dalam suatu layer, kumpulan *feature maps* yang ada di *layer* tersebut dihitung korelasinya untuk mengetahui seberapa baik hubungan antara pasangan fitur. Hal ini dikalkulasikan dengan menggunakan Gram Matrix.

$$\left|G(v_1,\ldots,v_n)\right| = \begin{vmatrix} \langle v_1,v_1\rangle & \langle v_1,v_2\rangle & \ldots & \langle v_1,v_n\rangle \\ \langle v_2,v_1\rangle & \langle v_2,v_2\rangle & \ldots & \langle v_2,v_n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle v_n,v_1\rangle & \langle v_n,v_2\rangle & \ldots & \langle v_n,v_n\rangle \end{vmatrix}.$$

Gambar 5. Gram Matrix
Sumber: Adaptasi Wikipedia dari Encyclopedia of Mathematics

Gram Matrix adalah sebuah matriks korelasi dari kumpulan vektor. Untuk setiap pasangan vektor <vi, vj>, nilai dot product dari pasangan tersebut menjadi nilai dari Gram Matrix pada koordinat (i,j).

Berkaitan dengan *feature map*, *feature map* perlu di-flatten terlebih dahulu, sehingga bentuknya menjadi vektor satu dimensi. Setelah di-*flatten* kita dapat membentuk Gram Matrix dari feature maps pada layer tersebut. Error/loss dari suatu layer adalah *mean squared error* antara Gram Matrix dari citra kombinasi dengan Gram Matrix citra gaya.

$$Loss(\text{Gaya})^{[i]} = \text{MSE}(\text{GramMatrix}(\text{Konten}^{[i]}), \text{GramMatrix}(\text{Kombinasi}^{[i]}))$$

Formula 6. Fungsi loss dari citra gaya pada layer i

Untuk mendapatkan loss keseluruhan dari citra gaya, maka kita cukup merata-ratakan loss dari n layer sampel yang digunakan.

$$Loss(\text{Gaya}) = \frac{1}{n} \sum_{i \in sample}^{n} Loss(\text{Gaya})^{[i]}$$

Formula 7. Fungsi loss dari keseluruhan citra gaya dari n buah layer sampel

Dengan mensubstitusikan kedua fungsi loss di atas (Formula 5 & 7) ke formula 3, kita dapat menemukan fungsi *loss* dari citra kombinasi, yang dapat digunakan untuk pelatihan dengan *gradient-descent*.

III. Implementasi

A. Kakas Pengembangan

Pada eksperimen yang akan dilakukan, kakas pengembangan gim yang akan digunakan adalah Unity. Versi Unity yang digunakan adalah 6000.0.32f1.

B. Aset yang digunakan

Eksperimen akan menggunakan aset gratis dari Unity Asset Store, yaitu Low Poly Atmospheric Locations Pack. Aset ini dapat diakses melalui tautan berikut.

https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-atmospheric-locations-pack-278928

Aset ini hanya memiliki satu tekstur, yang digunakan pada keseluruhan model dengan *UV mapping* yang berbeda-beda.

C. Implementasi NST untuk Penteksturan Ulang

Untuk melakukan penteksturan ulang terhadap aset gim, pertama-tama kita perlu melakukan *crawling*/penelusuran folder terlebih dahulu untuk menemukan tekstur-tekstur yang akan diproses. Implementasi crawling kurang lebih adalah sebagai berikut.

Penelusuran Citra Tekstur

```
import os
def crawl_textures(root_dir):
    textures = []
    # Telusuri citra di dalam root_dir dan subdirektorinya
    for dirpath, _, filenames in os.walk(root_dir):
        for filename in filenames:
            if filename.endswith('.jpg') or filename.endswith('.png'):
                textures.append(os.path.join(dirpath, filename))
```

```
    return textures
```

Keseluruhan citra tekstur di-*backup* terlebih dahulu, agar hasil dari penteksturan ulang dapat di-*revert*.

Backup Citra Tekstur

```
import os
import shutil
def backup_textures(textures, backup_dir):
    # Buat direktori backup jika belum ada
    if not os.path.exists(backup_dir):
        os.makedirs(backup_dir)

    # Untuk setiap citra, salin ke direktori backup
    for texture in textures:
        filename = os.path.basename(texture)
        backup_path = os.path.join(backup_dir, filename)
        shutil.copy(texture, backup_path)
```

Setelah itu, kita dapat melakukan *neural style transfer* untuk masing-masing citra tekstur.

Implementasi NST akan dilakukan dengan menggunakan bahasa Python. Karena keterbatasan waktu dan sumber daya untuk mengumpulkan dataset yang diperlukan, implementasi NST pada makalah ini akan memanfaatkan model *pretrained*. Model NST yang digunakan pada makalah ini dapat diakses melalui Tensorflow Hub, dengan handle sebagai berikut.

https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2

Secara umum, implementasinya adalah sebagai berikut.

Pengaplikasian NST

```
# Import Library yang dibutuhkan
import os
import tensorflow as tf
import tensorflow_hub as hub

# Load model NST dari Tensorflow Hub
handle =
'https://tfhub.dev/google/magenta/arbitrary-image-stylizatio
n-v1-256/2'
nst_model = hub.load(handle)

# Fungsi untuk menjalankan NST
def stylize(content_img, style_img):
  stylized_image = nst_model(tf.constant(content_img),
tf.constant(style_img))[0]
  return utils.tensor_to_pil_image(stylized_image)
```

Utilities

```
# Import Library yang dibutuhkan
import tensorflow as tf
import numpy as np
import PIL.Image

# Fungsi Load Image, dengan preprocessing agar
panjang/lebar maksimalnya 512
def load_image(image_path):
  # Pembacaan citra
  img = tf.io.read_file(image_path)
  img = tf.image.decode_image(img, channels=3)
  img = tf.image.convert_image_dtype(img, tf.float32)

  # Pre-processing citra
  shape = tf.cast(tf.shape(img)[:-1], tf.float32)
  long_dim = max(shape)
  scale = 512 / long_dim
  new_shape = tf.cast(shape * scale, tf.int32)
  img = tf.image.resize(img, new_shape)
  img = img[tf.newaxis, :]

  return img

# Mengubah format citra dari tensor ke PIL.Image
def tensor_to_pil_image(tensor):
  tensor = tensor * 255
  tensor = np.array(tensor, dtype=np.uint8)
  if np.ndim(tensor) > 3:
    assert tensor.shape[0] == 1
    tensor = tensor[0]
  return PIL.Image.fromarray(tensor)
```

Setelah itu, kita dapat mengaplikasikan NST ke keseluruhan citra tekstur.

Aplikasikan NST ke Seluruh Tekstur

```
def retexture(texture_paths, style_path):
  log('Retexturing...')

  # Load style image
  style_image = load_image(style_path)

  # Stylize masing-masing texture
  for texture_path in texture_paths:
    texture = load_image(texture_path)
    result = stylize(texture, style_image)

    # Overwrite hasil stylize ke texture path
    save_pil_image(result, texture_path)

  log('Retexturing complete!')
```

Untuk mengembalikan tekstur semula, implementasinya adalah sebagai berikut.

Mengembalikan Tekstur Semula

```
def revert_textures(backup_dir):
    log('Reverting textures...')

    # Load backup textures
    backup_textures = crawl_textures(backup_dir)

    # Restore backup textures
    for backup_texture in backup_textures:
        stripped_backup_texture =
os.path.relpath(backup_texture, backup_dir)
        texture_path = os.path.join(UNITY_PROJ_BASE_DIR,
stripped_backup_texture)
        shutil.copy(backup_texture, texture_path)
```

IV. HASIL

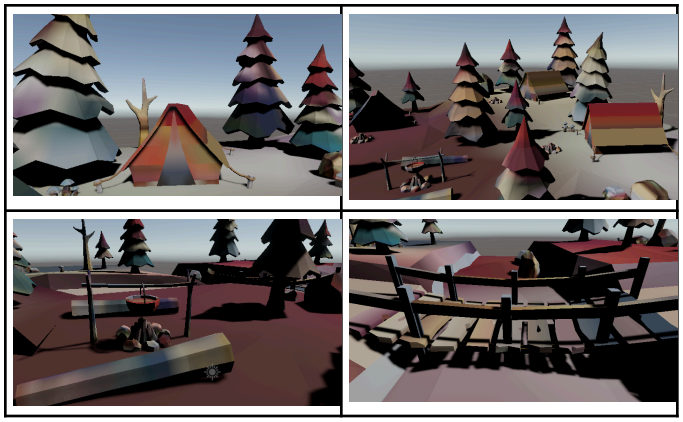
Berikut adalah tampilan aset awal yang akan digunakan di dalam eksperimen.

Gambar 6. Aset 3D awal yang digunakan dalam eksperimen
Sumber: Dokumentasi Penulis

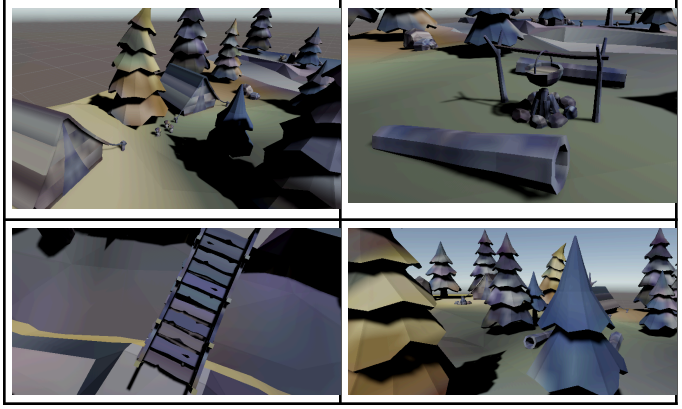
Berikut adalah hasil percobaan terhadap beberapa citra gaya.

Citra Gaya

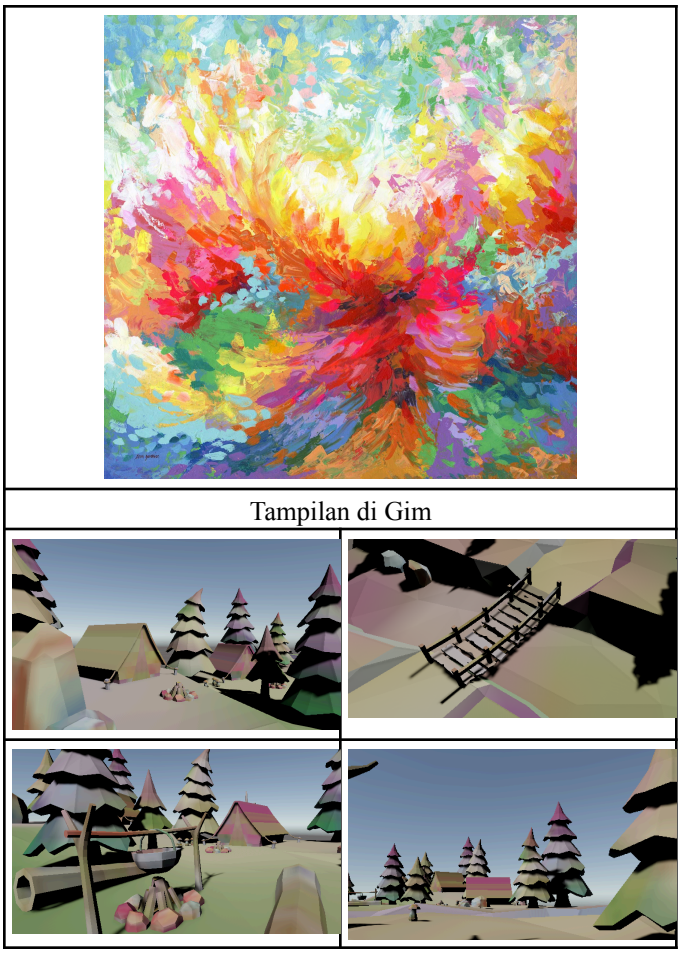
Tampilan di Gim



Citra Gaya

Tampilan di Gim


Citra Gaya



Tampilan di Gim

V. Penutup

A. Kesimpulan

Algoritma NST memungkinkan pemrosesan citra dari citra konten ke dalam bentuk gaya tertentu berdasarkan citra referensi lain. Hal ini didasari pada konsep *convolutional neural network* dengan pelatihan berbasis *gradient-descent*. Dengan menggunakan *pretrained* model yang telah ada, dilakukan pengujian untuk melakukan penteksturan ulang terhadap aset gim.

Berdasarkan hasil pada bagian IV, kita dapat melihat bahwa tekstur yang sudah diproses dengan NST kurang terpetakan dengan baik ke objek tiga dimensi. Hal ini disebabkan karena algoritma NST tidak dapat memahami konteks dari tekstur dan *UV mapping* dari objek tiga dimensi. Hal ini menyebabkan tekstur menjadi acak, dan tidak sesuai dengan konteks aslinya. Selain itu, resolusi dari tekstur yang berubah akibat algoritma NST juga dapat mempengaruhi hasil dari penteksturan ulang.

B. Saran

Penteksturan gaya ulang dari aset gim untuk menghasilkan gaya tertentu merupakan proses yang sangat berguna dalam produksi gim. Akan tetapi, pengembangan untuk automasi hal ini masih sangat jauh dari ideal. Diperlukan sumber daya dan waktu yang lebih banyak untuk merealisasikan hal ini.

Penulis, dalam hal ini percaya bahwa landasan algoritma dari NST dapat dimanfaatkan untuk penteksturan ulang di masa yang akan datang. Model ini mungkin tidak akan hanya mempertimbangkan citra tekstur saja, akan tetapi juga dapat memahami konteks dari objek 3D, misalnya dengan memperhitungkan *UV map* di dalam algoritma pelatihan.

Ucapan Terima Kasih

Pertama-tama, saya ingin mengucapkan terimakasih yang sebesar-besarnya terhadap dosen pengampu mata kuliah Pemrosesan Citra Digital, Bapak Rinaldi Munir yang memberikan tugas pembuatan makalah mengenai pemrosesan citra digital. Saya sangat bersyukur sekali, karena dengan pembuatan makalah ini, saya mendapatkan pemahaman mendalam terhadap *neural style transfer* (NST), topik yang mungkin akan sangat sulit saya dapatkan di luar, dan mungkin tidak akan pernah saya eksplorasi, jika bukan karena tugas ini.

Saya juga ingin berterima kasih keluarga dan teman-teman saya, yang telah memberikan dukungan kepada saya, baik secara konkrit, maupun dukungan secara mental sehingga saya dapat melalui masa-masa sulit dalam semester ini, hingga akhirnya saya bisa menyelesaikan masalah. Secara umum, saya sangat bersyukur atas kesempatan serta keberuntungan yang dilimpahkan kepada saya, sehingga saya dapat menempuh pembelajaran di ITB dan mendapatkan semua pelajaran ini, yang mudah-mudahan dapat bermanfaat bagi saya sampai di masa depan.

Link Github

https://github.com/moonawar/NST-Retexturing-Unity-Script

Referensi

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint*, arXiv:1508.06576, 2015. [Online]. Tersedia: https://arxiv.org/abs/1508.06576

[2] R. Munir, "21-CNN-2024," [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/21-CNN-2024.pdf

[3] MathWorks, "VGG19," [Online]. Tersedia: https://www.mathworks.com/help/deeplearning/ref/vgg19.html

[4] N. Rhodes, "CS 152 NN—15: Neural style transfer: Overview," YouTube, 2021. [Online]. Tersedia: https://www.youtube.com/watch?v=6KGtaXR7yMU

[5] N. Rhodes, "CS 152 NN—15: Neural style transfer: Content loss," YouTube, 2021. [Online]. Tersedia di: https://www.youtube.com/watch?v=c4vuR4vHKd0

[6] N. Rhodes, "CS 152 NN—15: Neural style transfer: Style loss," YouTube, 2021. [Online]. Tersedia: https://www.youtube.com/watch?v=AJOyMJjPDtE

[7] N. Solanki, "What is UV Mapping?" *The App Guruz*. [Online]. Tersedia: https://www.theappguruz.com/blog/uv-mapping.

[8] Teknik Informatika ITB, "Gradient descent," [Online]. Tersedia: https://cdn-edunex.itb.ac.id/storages/files/1710251393858_IF3270-Materi-Gradient-Descent.pdf

[9] W. Muttaqien, "Tutorial: Modeling, UV Unwrapping and Texturing a Mushroom," *BlenderNation.* [Online]. Tersedia: https://www.blendernation.com/2017/04/22/tutorial-modeling-uv-unwrapping-texturing-mushroom/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Januari 2024

Addin Munawwar Yusuf (13521085)