Methodologies and Mathematical Concepts in Real-time 3D-aware Portrait Video Relighting

Om Shrivastava - 210685

1 Introduction

The paper introduces a real-time 3D-aware portrait video relighting method using Neural Radiance Fields (NeRF). The proposed method relights and adjusts views of talking faces in videos, achieving a balance between quality and efficiency. The approach uses dual feed-forward encoders to capture albedo and shading information and leverages a temporal consistency network to ensure smooth transitions and reduce flickering.

2 Mathematical Formulation

2.1 Tri-plane Representation

The method decomposes a 3D scene into *albedo* and *shading* components, stored as tri-planes. Each frame F_i in a video is embedded into an albedo tri-plane T_A^i and a shading tri-plane T_S^i, given by:

$$T_A^i = f_{\text{albedo-encoder}}(F_i) \quad \text{and} \quad T_S^i = f_{\text{shading-encoder}}(T_A^i, L)$$

where L represents lighting conditions modeled by spherical harmonics (SH) coefficients.

2.2 Spherical Harmonics (SH) for Lighting

The SH lighting model $L = [L_0, L_1, \ldots, L_n]$ provides a mathematical basis for representing directional lighting effects. The shading tri-plane is conditioned on SH coefficients up to the second order:

$$T_S = \sum_{l=0}^{2} \sum_{m=-l}^{l} L_{lm} Y_{lm}(\theta, \phi)$$

where Y_{lm} are the spherical harmonics functions, and (θ, ϕ) denote lighting directions. This formulation allows efficient illumination estimation across frames.

2.3 Albedo Loss

The albedo loss measures the dissimilarity between the predicted albedo image $\hat{A}$ and the ground truth albedo A:

$$L_{\text{albedo}} = ||\hat{A} - A||_1 + ||\hat{A}_r - A_r||_1 + L_{\text{lpips}}(\hat{A}, A) + \lambda_g ||T_g - T_{\hat{g}}||_1,$$

where L_{lpips} is a perceptual loss, and T_g represents the albedo tri-plane.Here we fine-tune the parameter from 1 to 0.01 to get an optimal result.

2.4 Shading Loss

The shading loss ensures that the predicted shading maps and tri-planes match the ground truth:

$$L_{\text{shading}} = ||\hat{S} - S||_1 + \lambda_s ||T_S - \hat{T}_S||_1.$$

2.5 RGB Loss

The RGB loss compares the predicted RGB images $\hat{I}$ (in both raw and super-resolution domains) with the ground truth images I, incorporating perceptual and identity losses:

$$L_{\text{rgb}} = ||\hat{I} - I||_1 + ||\hat{I}_r - I_r||_1 + L_{\text{lpips}}(\hat{I}, I) + L_{\text{lpips}}(\hat{I}_r, I_r) + \lambda_f ||\hat{I}_f - I_f||_1 + L_{\text{id}}(\hat{I}, I).$$

2.6 Adversarial Loss

An adversarial loss is used to improve the realism of generated images by training a discriminator D to distinguish between real and synthesized images:

$$L_{\text{adv}} = -\left(E[\log D(I)] + E[\log D(I_r)] + E[\log(1 - D(\hat{I}))] + E[\log(1 - D(\hat{I}_r))] \right).$$

Our final loss function for training the dual encoder is the weighted sum of all the losses.

$$L = \lambda_{\text{albedo}} L_{\text{albedo}} + \lambda_{\text{shading}} L_{\text{shading}} + \lambda_{\text{rgb}} L_{\text{rgb}} + \lambda_{\text{adv}} L_{\text{adv}}$$

initially, we set lambda of albedo, shading, and RGB to 1 and for adv to 0. After 16M iteration, we set it back λ_{adv} $to 1$.

2.7 Temporal Consistency Loss

The temporal consistency loss is designed to enforce coherence between consecutive video frames. It includes a short-term temporal loss:

$$L_{\text{short}} = M_s \sum_{\omega \in \{I, I_r, A, A_r S\}} L_{\text{lpips}}(\omega^i - \tilde{\omega}^{i-1}),$$

where M_s is a mask mitigating errors in the warping process, and ω refers to various image modalities (RGB, albedo, shading).

A long-term consistency loss is similarly defined, but the current frame is compared with the first frame to ensure long-term stability.

3 Idea

In implementing this method, we can start by processing each video frame independently to simplify the initial testing phase then focus on separating each frame into two representations: albedo (base texture) and shading (lighting effects). we could use a basic neural network to predict each of these components separately—albedo for consistent facial textures and shading to simulate changes in lighting.

To achieve realistic lighting effects, I will try to go with spherical harmonics (SH) to control the direction and intensity of light on the face. To ensure temporal consistency across frames, we could use a basic RNN, to smooth the transitions and reduce any flickering.

For loss functions, I think to use albedo loss to ensure texture consistency, shading loss for accurate lighting, and temporal consistency loss to avoid abrupt shifts between frames. Initially, I will focus on frame-by-frame relighting and observing the quality of albedo and shading predictions. Gradually, will try to refine the model for smoother transitions and more realistic results by enhancing the temporal network and fine-tuning the lighting model.

4 Implementation

I got the following GitHub repo https://github.com/GhostCai/PortraitRelighting. which I run as given in the instructions. It lacked many directories, pre-trained models, and other requirements. I debugged the following thing and then I used the cc GPU server to run the example.py. Here with other GPU servers, I found an issue that there are some very low-level build-related issues, finally, it was able to run using CSE dept. GPU server. I generated the parameter and log file for the file. The final output was equivalent to the one given in the example.Rest implementation is being explained in readme file.

5 Dataset

The example file present in repo were used to test the implemented code.