

MASTER'S THESIS 2026

Animated Neural Billboarding for Real-Time Rendering

Edwin Gustafsson

Elektroteknik
Datateknik

ISSN 1650-2884

LU-CS-EX: 2026-13

DEPARTMENT OF COMPUTER SCIENCE

LTH | LUND UNIVERSITY



EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2026-13

**Animated Neural Billboarding for
Real-Time Rendering**

Neural ljussättning av animerade billboards
för datorgrafik i realtid

Edwin Gustafsson

Animated Neural Billboarding for Real-Time Rendering

Edwin Gustafsson
ed3047gu-s@student.lu.se

May 7, 2026

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Rikard Olajos, rikard.olajos@cs.lth.se

Examiner: Michael Doggett, michael.doggett@cs.lth.se

Abstract

In real-time computer graphics, billboards are a frequently used way to save on computation time by simplifying an object down to a flat image. Billboarding for video game rendering has been used since the first 3D games and continues its popularity today, with a common use case being trees. Often these billboards are animated to show movement in the object, like a tree swaying in wind. Billboards come with drawbacks, such as not being able to realistically relight the object with self shadowing and complex light interaction. Previous work has proposed neural shading as a solution, running a small neural network in the fragment shader to encode light interaction, though only on a static billboard. In this work, we continue the proposed neural shading solution by adding the ability to animate the billboard texture. We present an ablative study of four model architectures along with image evaluation and performance metrics. Our results show a promising potential for this application, though more work is needed to bring a similar performance to the non-animated version.

Keywords: MSc, neural, rendering, shading, billboards, computer graphics, real-time

Acknowledgements

I would like to thank Rikard Olajos for the help and support provided throughout this project.

I would like to thank LUNARC: The Centre for Scientific and Technical Computing at Lund University, for greatly accelerating the neural network training process.

I would also like to thank Michael Doggett for the inspiring computer graphics courses hosted at Lund University.

Contents

1	Introduction	7
1.1	Background	7
1.2	Research Questions	8
1.3	Scope	8
2	Theory	9
2.1	The Rendering Equation	9
2.2	Path-Tracing	10
2.2.1	Subsurface Scattering	10
2.3	Phong Shading	11
2.4	Artificial Neural Networks	11
2.4.1	Datasets	12
2.4.2	Generalization	13
2.5	Neural Rendering	13
2.5.1	Cooperative Vectors	13
2.6	Frequency Encoding	14
2.7	Image Evaluation	14
2.7.1	Peak Signal-to-Noise Ratio	14
2.7.2	FLIP	15
3	Method	17
3.1	Model Architecture	17
3.1.1	Frequency Encoding	18
3.2	Datasets Structure	19
3.2.1	Training	19
3.3	Real-Time Inference	20
4	Results	21
4.1	Datasets	21
4.2	Inference	22

4.3	Image Evaluation	22
5	Discussion	33
5.1	Interpolation	34
5.2	Answers to Research Questions	34
5.3	Usability in Artist Workflows	34
5.4	Further Work	34
5.4.1	Motion Vectors	35
5.4.2	Dataset Streaming	35
5.4.3	Encoding Schemes	35
6	Conclusion	37
	References	39

Chapter 1

Introduction

1.1 Background

In the application of computer graphics for real-time rendering, there is a limited amount of mesh geometry that can be rendered at once before rendering times become unacceptably long. To reduce the amount of mesh geometry that an object is comprised of, different level-of-detail (LOD) models are used when the object is far away from the camera or not the primary focus of the scene. The lowest level of detail possible is a single face element with an image of the object baked onto it, commonly referred to as billboards. This simple representation of the object introduces limitations, particularly to the ability to relight the object, a lack of self-shadowing, and vertex based animation, since the shape of the object is no longer defined by its vertices.

Neural rendering is a technique developed primarily for real-time rendering to encode some characteristic of a scene in a neural network representation for fast approximation of otherwise too costly rendering techniques. These neural networks are trained offline on pre-rendered ground truth information and inferred in real time, accelerated by the suitable GPU architecture. In recent years, neural rendering has been shown applicable in a wide range of computer graphics techniques, such as high ratio texture compression [11], fast rendering of complex materials [12], encoding of full 3D assets along with their light interaction [7], among others.

In the paper Neural Billboarding for Real-Time Rendering [8], Olajos and Doggett introduce a solution to making billboards relightable by using neural rendering, approximating a path-traced ground truth with global illumination. However, the solution did not explore the ability to animate these billboards, which can commonly and easily be done with traditional billboards using texture animation, for example, to show a tree swaying in the wind or a plume of smoke billowing.

This paper aims to expand on the work done by Olajos, to add the ability to encode an arbitrary animation into the neural network. This would increase the use cases of neural

billboards and bring it closer to being a potential application as a general purpose rendering technique for retaining lighting detail in billboards. The primary application example of focus for this paper is trees swaying in the wind, illuminated by a dynamic sun position.

1.2 Research Questions

The questions we aim to answer in this paper are the following:

- Is it possible for us to animate a neural shaded billboard while retaining comparable visual fidelity to the static version?
- Is it possible to encode the increased amount of information while keeping the neural network to a feasible size for real-time application?
- Can an encoding scheme be used on the time variable to increase the performance of the network?

1.3 Scope

To fit the scope of a master’s project, this study can not be exhaustive, while there may be several promising strategies for the problem of encoding animation information in a neural system, and several ways of preprocessing the time dependent information, we will only present the first explorations of solving this problem, additional ideas for more advanced solutions to explore are presented in the Further Work section.

Chapter 2

Theory

In this chapter we discuss the relevant theory behind the techniques used in this paper. We introduce the rendering equation and how path-tracing renderers utilize it for realistic rendering, the traditional real-time technique in contrast, using phong shading. How artificial neural networks are constructed and specialized for use in rendering along with feature pre-processing in the form of frequency encoding, and lastly covering the image evaluation methods that will be used for result analysis. The theory introduced here will be used in the implementation to construct an artificial neural network, processing phong shaded data to approximate a path-traced image.

2.1 The Rendering Equation

In 1986 James T. Kajiya presented an equation to generally describe the light intensity at a point as a function of the sum of light scattered toward x from all other surface points [5].

$$I(x, x') = g(x, x') \left[e(x, x') + \int_S \rho(x, x', x'') I(x', x'') dx'' \right] \quad (2.1)$$

where $I(x, x')$ is the light intensity that is transferred from point x' to point x .

$g(x, x')$ is a coefficient denoting the geometric relation of two points, a completely blocked path between the points would result in $g(x, x')$ equaling 0. If the points are not relatively occluded then $g(x, x') = 1/r^2$, in accordance with the inverse square law of radiance intensity over the distance r .

If the point x' is a light source, the emittance term, $e(x, x')$, will contain information about the intensity of that light in the direction of x .

The most complex part of the equation is to determine the amount of light that is reflected on the point x' from other scene points. The integral in the second term covers all other points in the scene for their contribution to the reflected light, with the reflectance coefficient $\rho(x, x', x'')$, and is notably recursive as it contains the function $I(x', x'')$, meaning

that to determine the intensity of the light at point x , the intensity of the light at all other points must be evaluated.

This formulation of the rendering equation does not cover all aspects of light interaction, such as diffraction and variable refraction, but provides a good foundation for the most common scenarios of rendering a scene.

If one attempts to implement this equation in a computer program, they immediately run into obstacles, the continuous integral must be approximated discretely, and the recursive evaluation must be stopped at a certain depth. Even with sacrifices made to the surface resolution and recursion depth it becomes clear that this approach to rendering is not suitable for real-time applications without heavy modifications due to the large amount of processing necessary.

2.2 Path-Tracing

The modern technique for realistic rendering, path-tracing, is closely based on the rendering equation. It aims to numerically solve for an approximation of the rendering equation by recursively evaluating light intensity at many points in the scene using a Monte Carlo method [5].

The process of path-tracing starts at a pixel in the camera, from which a ray traverses the scene until encountering a surface point where a decision must be made as to where to send the next recursive ray. The naive solution picks a direction at random, but not all surfaces contribute an equal amount of light and so there exists surface points that are more advantageous to sample around for rapid convergence to an acceptable approximation of the image of the scene. Most path-tracing implementations therefore implement a scheme for efficient sampling known as importance sampling [5], which can on-the-fly generate a probability distribution over sampling paths that are more likely to provide valuable information in the intensity of the light scattered to the point under investigation.

Path-Tracing produces images that are true to life, but is very computationally expensive, and so is often presented as a ground-truth ideal that real-time techniques strive to emulate.

2.2.1 Subsurface Scattering

Subsurface Scattering (SSS) is a rendering technique for capturing the complex transfer of light through semi-translucent surfaces, where a soft glow can be seen around the spot where light enters and exits an object due to the scattering of photons occurring within the material. This effect is most commonly seen in soft organic materials, such as skin and leaves, but also in inorganic materials such as marble and jade.

SSS is relatively simple to implement in a path tracer as transferred light rays are already simulated, and implementing SSS becomes a question of perturbing these transferred light rays in an appropriate way, although this approach is natural, it is slow to converge and therefore some approximations are used in modern path tracers [4]. The modern solution bases the color on the distance between points where the light enters and leaves the material, using some probability distribution function for sampling the two points [2]. Modern real-time renderers commonly use the same color approximation but a more computationally efficient

sampling scheme, with some predetermined importance based screen space sampling pattern [3].

2.3 Phong Shading

Real-time rendering requires images to be generated in a specified amount of milliseconds to create a video feed smooth enough for intuitive interaction, which calls for sacrifices to be made in light interaction and material properties realism, and it is required to use fast, approximating algorithms to be able to generate the usual minimum requirement of 30 frames per second. In 1975, Bui Tuong Phong presented a simple model for shading an object with a diffuse light and specular light component [9] that is still in use today, but has been largely antiquated by the use of Physical Based Rendering (PBR) shaders.

The Phong Shading Model consists of both an expression to generate a smooth normal direction transition between mesh faces and the light intensity at a point.

The smooth surface normal is generated by linearly interpolating the vertex normals of the vertices that define the surface.

The light intensity consists of two parts: a diffuse term and a specular term. The diffuse light intensity is defined by the following expression.

$$D = \max(\mathbf{N} \cdot \mathbf{L}, k) \quad (2.2)$$

where $\mathbf{N}$ is the Normal vector, $\mathbf{L}$ is the direction vector to the point light source, and k is the ambient light intensity.

The specular component is defined by the following expression.

$$S = C \cdot \max(\mathbf{R} \cdot \mathbf{V}, 0)^n \quad (2.3)$$

where C is the specular coefficient, $\mathbf{R}$ is the inverted reflected light direction, $\mathbf{V}$ is the view vector, and n is the specular exponent, an arbitrary constant that controls the sharpness of the specular reflection.

2.4 Artificial Neural Networks

The general objective of machine learning is to encode and recreate a relationship between some input and output values. Many models and algorithms exist to solve this problem each with their own advantages and disadvantages. Artificial Neural Networks (ANN) have shown to be generally applicable to complex non-linear problems, combined with their high customizability makes them well suited for many machine learning applications.

An artificial neural network model consists of a sequence of layers of nodes called neurons. A layer is connected to the next by directed edges from each neuron to each neuron in the subsequent layer. These edges have a variable weight coefficient which are the variables that are adjusted during the training process. These weights are multiplied with the value from the tail neuron to produce, in addition to the rest of the neurons in the previous layer and its bias, the value for the head neuron. Then the produced value is passed through an activation function whose common primary purpose is to prevent large negative values which

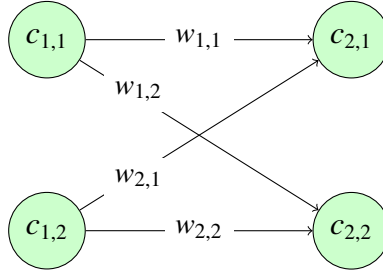


Figure 2.1: The weights between two layers with two neurons.

can cause instability in the model. This process is repeated for each layer until the values for the output neurons are evaluated. The value of a neuron is expressed as follows:

$$c_{a,b} = f_a\left(\sum_{i=0}^n c_{a-1,i} \cdot w_{i,b} + b_{a,b}\right) \quad (2.4)$$

Where $c_{x,x}$ is the value of a neuron, $w_{x,x}$ is the weight of the edge between neurons, $b_{x,x}$ is a constant bias term for the neuron, and f_a is the activation function, which in this work is ReLU:

$$f_a(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (2.5)$$

The specific case of two successive layers with two neurons each is shown in Figure 2.1, where

$$c_{2,1} = f_a(c_{1,1}w_{1,1} + c_{1,2}w_{2,1} + b_{2,1}) \quad (2.6)$$

and

$$c_{2,2} = f_a(c_{1,1}w_{1,2} + c_{1,2}w_{2,2} + b_{2,2}) \quad (2.7)$$

There are many variations of the ANN model, in this paper we use a Multi-Layer Perceptron (MLP) which is characterized by the layers being fully connected, in contrast to convolutional neural networks for example in which only some of the neurons of a previous layer is connected to a neuron in the subsequent layer. The MLP is a commonly used general purpose ANN design capable of encoding complex non-linear patterns in the relationship between input and output. An example of a small MLP can be seen in Figure 2.2

The architecture of an MLP model is defined by the amount of hidden layers, the amount of neurons in the layers, as well as the activation function.

2.4.1 Datasets

A model is trained on some data in which input features and expected behavior are well defined. A dataset is needed where the inputs and desired output are paired so that the model can adjust the internal weight parameters to approximate the given output values. The dataset should be as large as reasonably possible so that complex patterns can be reinforced over many training iterations.

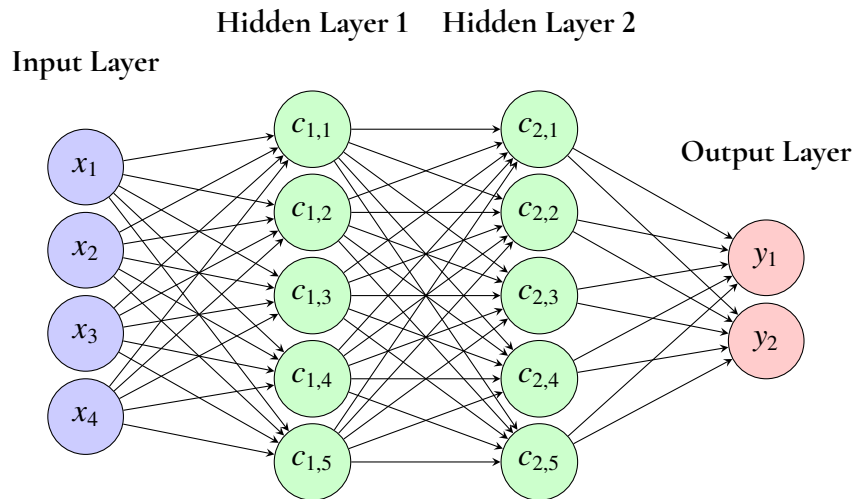


Figure 2.2: Example of a small neural network with 4 input variables, 2 hidden layers of 5 neurons each, and 2 output variables.

2.4.2 Generalization

In machine learning, generalization refers to the ability of a trained model to apply the encoded information to new problems, a well trained model will produce a similar quality of results for inputs that are used in the training dataset and inputs that are never before seen by the model. A model that produces good results for the inputs present in training data but poor results for new inputs are considered to be overfitted. Overfitted models are commonly produced in neural networks when the model has a large amount of neurons and a smaller than ideal dataset.

2.5 Neural Rendering

Neural Rendering refers to a wide field of graphics techniques that utilize artificial neural networks to make rendering images more efficient and is a very active topic of research. It has been widely used in modern video game production, most commonly in NVidia's Deep Learning Super Sampling (DLSS) and AMD's FSR technologies for real time predictive up-scaling and frame interpolation.

The subfield of neural rendering that is explored in this paper is the specific application of ANN's in the shading pipeline, this is called Neural Shading. By running a small neural network in one of the shader stages it becomes possible to approximate detail which would otherwise be too costly to recreate using traditional computation.

2.5.1 Cooperative Vectors

Cooperative vectors are a datatype that help accelerate the vector-matrix multiplications that are performed when inferring a neural network [6]. The idea is to separate a vector-matrix multiplication operation into chunks to facilitate SIMD parallel computation, then add the separate results to complete the operation. It is relevant to note that if the vector is

very large, cooperative vector can give diminishing results as the chunks no longer fit in the limited amount of registers for SIMD operations.

Cooperative vectors give a significant increase in performance specifically for neural network acceleration and is a large part of making neural shading viable for real-time applications.

2.6 Frequency Encoding

Neural networks show a large problematic difference in the training time required to converge on low and high-frequency patterns leading to what is known as spectral bias where a model struggles to recreate high-frequency detail present in the training dataset, leading to a blurry image in the visual media application. To assist in the efficient encoding of high-frequency detail in a neural network, frequency encoding, also known as fourier features, as proposed by Tancik et al. [10] has proven effective by expanding the feature space with one or multiple frequency sine and cosine features:

$$\boldsymbol{\gamma}(\mathbf{v}) = [\sin(2^0\pi\mathbf{v}), \cos(2^0\pi\mathbf{v}), \dots, \sin(2^{L-1}\pi\mathbf{v}), \cos(2^{L-1}\pi\mathbf{v})] \quad (2.8)$$

Where the vector $\mathbf{v}$ contains the features to be encoded using L frequencies.

This equation is slightly modified to make sure the first order elements produce a full sine period by replacing π with 2π .

$$\boldsymbol{\gamma}(\mathbf{v}) = [\sin(2^1\pi\mathbf{v}), \cos(2^1\pi\mathbf{v}), \dots, \sin(2^L\pi\mathbf{v}), \cos(2^L\pi\mathbf{v})] \quad (2.9)$$

The encoded features are passed to the MLP in combination with the unprocessed feature.

$$\hat{\boldsymbol{\gamma}}(\mathbf{v}) = [\boldsymbol{\gamma}(\mathbf{v}), \mathbf{v}] \quad (2.10)$$

This gives the network more opportunities to converge on patterns of a frequency that appear in the encoded feature space.

2.7 Image Evaluation

To evaluate and compare the images we create to be able to draw conclusions about the performance about a rendering technique we need an algorithm to generate a score of some kind, preferably with a perception that resembles the opinion a human would have when observing the image, since the human eye is what the field of computer graphics is generally optimizing for.

The image evaluators presented in this section base the score on the difference between a ground truth image and the image to be evaluated.

2.7.1 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) is a purely signal processing approach to image evaluation. The image is treated as a continuous signal, the ground truth image is considered a pure

signal, and the image to be evaluated is the pure signal with some added noise. PSNR provides the strength of the pure signal as compared to the noise to give a relative score denoted in decibels, where a higher score is better.

2.7.2 FLIP

FLIP is an image difference evaluator that aims to produce a heatmap image based on how a human would perceive the difference between two images, developed specifically to judge image differences in computer graphics [1]. The algorithm contains two pipelines considering both perceptually compensated color differences and edge detection to compare shapes in the image. The combination of these two pipelines creates an error score for each pixel in the image to form a heatmap, where a higher value refers to a larger visually distinct difference between the ground truth and the image to be evaluated. The per pixel error values can also be averaged for the whole image to create a metric that can be compared to other images.

Chapter 3

Method

In this chapter we present the implementation of the proposed solution. We present the model architecture utilized, along with the system design and feature pre-processing including frequency encoding. We outline the training dataset structure and method for generation as well as the procedure for real-time inference.

3.1 Model Architecture

The MLP architecture used is based on the foliage model architecture in static neural billboards [8], here expanded with a time dependent variable to facilitate animation.

The MLP architecture was chosen over other types of ANNs for its highly efficient inference when kept to a small amount of neurons, which is necessary in this real-time application.

The output of the model are the red, green, and blue channels for a single pixel. The activation functions used between layers are rectified linear units (ReLU). The inputs to the model consist of UV coordinates of the relevant pixel, the normalized direction vector from the light source, the RGB values from a diffuse phong shading calculation, and the normalized time variable denoting the current position in the animation cycle.

The inputs use only the diffuse term of phong shading and discards the specular term as it produces highlights that are not congruent with the ground truth, as shown experimentally in static neural billboards [8].

To facilitate animation the input space must include a time dependent variable, here represented by a floating point number in the interval 0 to 1, where 0 is the first frame in the animation and 1 is the last. A floating point representation for the time position naturally allows for the neural network to interpolate frames, making it possible to infer a smoother, higher frame-rate, animation than what was used during training of the model.

We present 4 variations of the MLP model for an ablative study of the relative effects of two independent modifications:

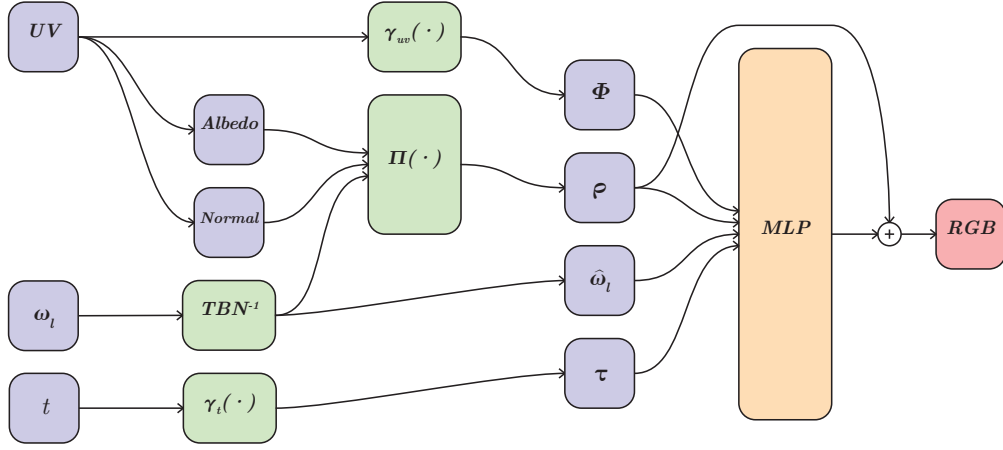


Figure 3.1: The model system containing the frequency encoded time τ . The UV coordinates are used by the albedo and normal texture lookups for the diffuse phong shading feature ρ , and is passed by itself through frequency encoding to be used as a feature in the MLP as Φ . The light direction vector ω_l is gathered from the scene and transformed to object space before being used for the diffuse phong shading ρ and as the MLP feature $\hat{\omega}_l$. The diffuse phong shading feature is added to the residual output of the MLP to produce the final fragment color.

- the amount of neurons in the model, with a small version of 3 hidden layers with 32 neurons each, as well as a large version of 5 hidden layers with 64 neurons each.
- The encoding scheme for the time variable, the version without any encoding; a single floating point variable representing the time position in the animation cycle, and the version with first order frequency encoding with the floating point time variable as well as the sine and cosine of the time variable.

The small model has the same internal architecture as in static neural billboards to be able to compare the quality of the produced images between the static and animated versions. The larger model is presented to investigate how the increased encoding space could allow the network to converge along the added time dimension and the non-linear relationship that stems from the changes in expected behavior between animation frames. The model is residual, which means that a part of the input is combined with the output before loss is calculated and backpropagation is performed, in this case the diffuse phong shading is added to the MLP output, this helps with stable and consistent color recreation, as the non-residual version of the model experimentally showed a tendency to over-represent the dominant color of the image in pixels that should not contain that color.

A system diagram of the MLP and its feature pre-processing is shown in Figure 3.1.

3.1.1 Frequency Encoding

We applied frequency encoding to two features: UV coordinates and the time variable. The dimensions of the UV coordinate encoding was 20, as inherited from static neural billboards [8],

expanding the 2 floating point UV features to 42 features by using Equation 2.9 with a dimensionality of 20:

$$\xi = \begin{pmatrix} u \\ v \end{pmatrix} \quad (3.1)$$

$$\gamma(\xi) = [\sin(2^1\pi\xi), \cos(2^1\pi\xi), \dots, \sin(2^{20}\pi\xi), \cos(2^{20}\pi\xi)] \quad (3.2)$$

For the time variable high-frequency detail is not as much of a priority, and frequency encoding was as much implemented with consideration to the feature domain expansion as to the periodicity of the animation; to avoid a harsh discontinuous appearance when looping the animation from the final frame to the first, since the frequency encoding generates periodic time variable features from the aperiodic original representation. A high-frequency encoding of the time variable could theoretically also lead to unwanted high-frequency fluctuations in the output appearing as flickering and coarse frame interpolation. The frequency dimension used is therefore very low as compared to the UV encoding, with a dimensionality of 1, resulting in the single time feature being expanded to 3.

$$\gamma(t) = [\sin(2\pi t), \cos(2\pi t)] \quad (3.3)$$

3.2 Datasets Structure

In this paper, we use the Cycles path-tracing renderer in Blender for producing ground truth images for the training and evaluation datasets.

We produced two datasets used to generate the results presented in the Results chapter:

- a synthetic dataset showing a cube with a protruding cylinder on the front face to clearly display the capability of self-shadowing, the animation shows the object swiveling around the z-axis.
- A foliage application example showing a tree, displaying self-shadowing from its branches and subsurface scattering in the leaves. The animation shows the tree moving in strong winds.

The tree 3D model was sourced online under a CC0 license, and we produced the animation specifically for this application using Blender.

A training dataset consists of 500 images; 50 animation frames of 10 images each with different light directions with a resolution of 1000 by 1000 pixels. These images are then decomposed to form the individual training set for the per-pixel model into 500 000 unique pixels.

3.2.1 Training

Training was performed with a 90% training, 10% validation dataset split. The Adam optimizer was used during training along with a step learning rate schedule that multiplies the learning rate by 0.5 every 50 epochs to allow for finer convergence.

3.3 Real-Time Inference

A real-time Vulkan 1.4 testbed for viewing the billboards was inherited from static neural billboards [8] and modified for animation. Textures (albedo and normal) were sampled from a texture atlas containing a 10 by 10 grid of animation frames of 1000 by 1000 pixel resolution.

The procedure for inference of our model starts with gathering information from the scene: time position in the animation cycle, albedo and normal texture loading, MLP binary loading, and light direction.

This data is then passed to the fragment shader where per-pixel computation is performed; albedo and normal textures are sampled, the albedo alpha channel is checked to determine if the fragment should be discarded to avoid unnecessary MLP calculations, the light direction is transformed to object space using the inverse TBN matrix, diffuse phong shading is calculated as per Equation (2.2), frequency encoding of UV coordinates and time value are produced as per Equation (2.9), and inputs are then passed to the MLP. The MLP calculations consists of one cooperative vector-matrix multiplication per layer of the model, with ReLU activation functions in between. The output of the MLP is added with the phong diffuse shading to produce the final fragment color.

The cooperative vectors use 16 bit floating point numbers as data type for the neuron values, chosen over 8 bit floating point as it was shown to produce better looking results in static neural billboards.

Chapter 4

Results

In this chapter, results are presented as visual and metric based evaluations of the inferred images as compared to the path-traced ground truth. FLIP heatmap images are provided alongside the comparisons to highlight problem areas.

All models were trained for 250 epochs in the LUNARC high-performance computing environment, using an NVidia A100 GPU. The full 250 epoch training took around 1 hour for each model.

The real-time inference frame times were generated on a computer with an RTX 4060 Ti GPU with 16 GB VRAM, a 13th Gen Intel Core i7-13700KF CPU, and 32 GB RAM.

For easy referencing, the four model architectures will be referred to by the following naming scheme:

Alias	Number of Hidden Layers	Neurons Per Layer	Time Frequency Encoding
(3×32)	3	32	
$(3 \times 32_F)$	3	32	X
(5×64)	5	64	
$(5 \times 64_F)$	5	64	X

4.1 Datasets

We present results generated from two training datasets, the tree and the synthetic dataset.

Two types of evaluation datasets are considered, both with 100 images; one for each animation frame, notably this is twice as many as included in the training dataset to investigate the ability of frame interpolation, and so every other frame is an interpolated frame. One of the evaluation datasets contains light directions that are randomized for each image, and the other has a constant light direction, to eliminate the variance in results from different

light directions when investigating the performance difference due to animation frames and interpolation.

4.2 Inference

The real-time inference frame times are presented in Table 4.1.

Model	Frame Time (ms)
Baseline	0.2
(3×32)	0.7
$(3 \times 32_F)$	0.7
(5×64)	2.6
$(5 \times 64_F)$	2.6

Table 4.1: Frame time of the models running once for every pixel of a 1920 by 1080 pixel monitor. Baseline is the billboard with only phong shading, no model inference.

4.3 Image Evaluation

In this section, $\mathcal{F}$ LIP error scores and PSNR scores are presented. These metrics do not sufficiently represent the visual fidelity of the images in isolation and should be considered alongside the images they represent. The metrics become useful when they are considered relative to each other as a lower $\mathcal{F}$ LIP error and higher PSNR values generally imply better performance.

A demonstration of the textures and phong shading used for the MLP inference is shown in Figure 4.1.

A comparison of the model inference of the third frame of the synthetic dataset is presented in Figure 4.2. A similar comparison for the fourth frame of the tree dataset is presented in Figure 4.4.

The static neural billboard version for the synthetic and tree dataset is presented in the same conditions in Figure 4.3 and Figure 4.5 respectively, with $\mathcal{F}$ LIP error and PSNR metrics shown in Figure 4.8 and Figure 4.9.

The $\mathcal{F}$ LIP error and PSNR scores are plotted in Figure 4.6 and Figure 4.7 for the synthetic and tree dataset respectively.

A small section of the first animation frame of the tree dataset with static lighting conditions is shown in Figure 4.10 to highlight the detailed shadows that are not recreated by the MLP model.

In Figure 4.11 and Figure 4.12, the $\mathcal{F}$ LIP error scores for each frame in the full synthetic and tree animation cycles are shown. The evaluation is done on a dataset with 100 animation frames with consistent lighting, double the amount of animation frames used in training, meaning that every other frame is interpolated. Both the (5×64) and $(5 \times 64_F)$ error scores are presented.

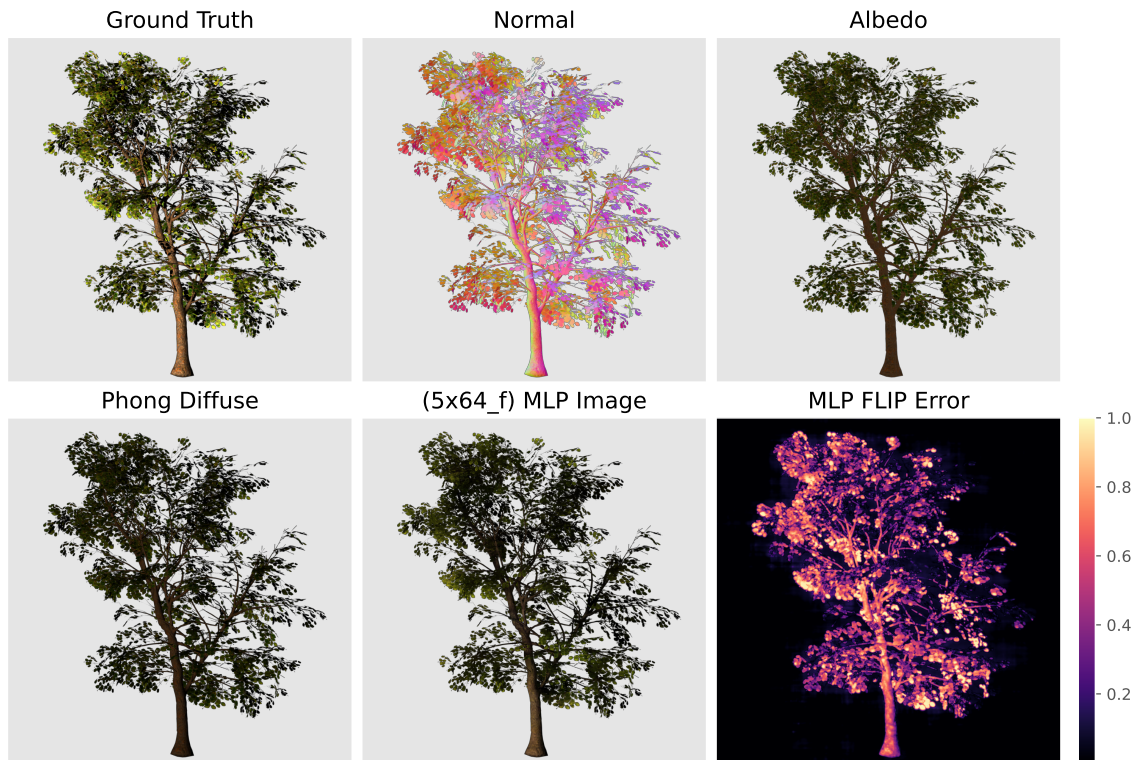


Figure 4.1: The ground truth image, the normal and albedo textures that are used in the phong diffuse shading, which is used as input to the MLP. The FLIP error shows a comparison between the MLP image and ground truth.

Figure 4.13 shows the FLIP error scores of the non-interpolated frames alongside the scores of the interpolated frames.

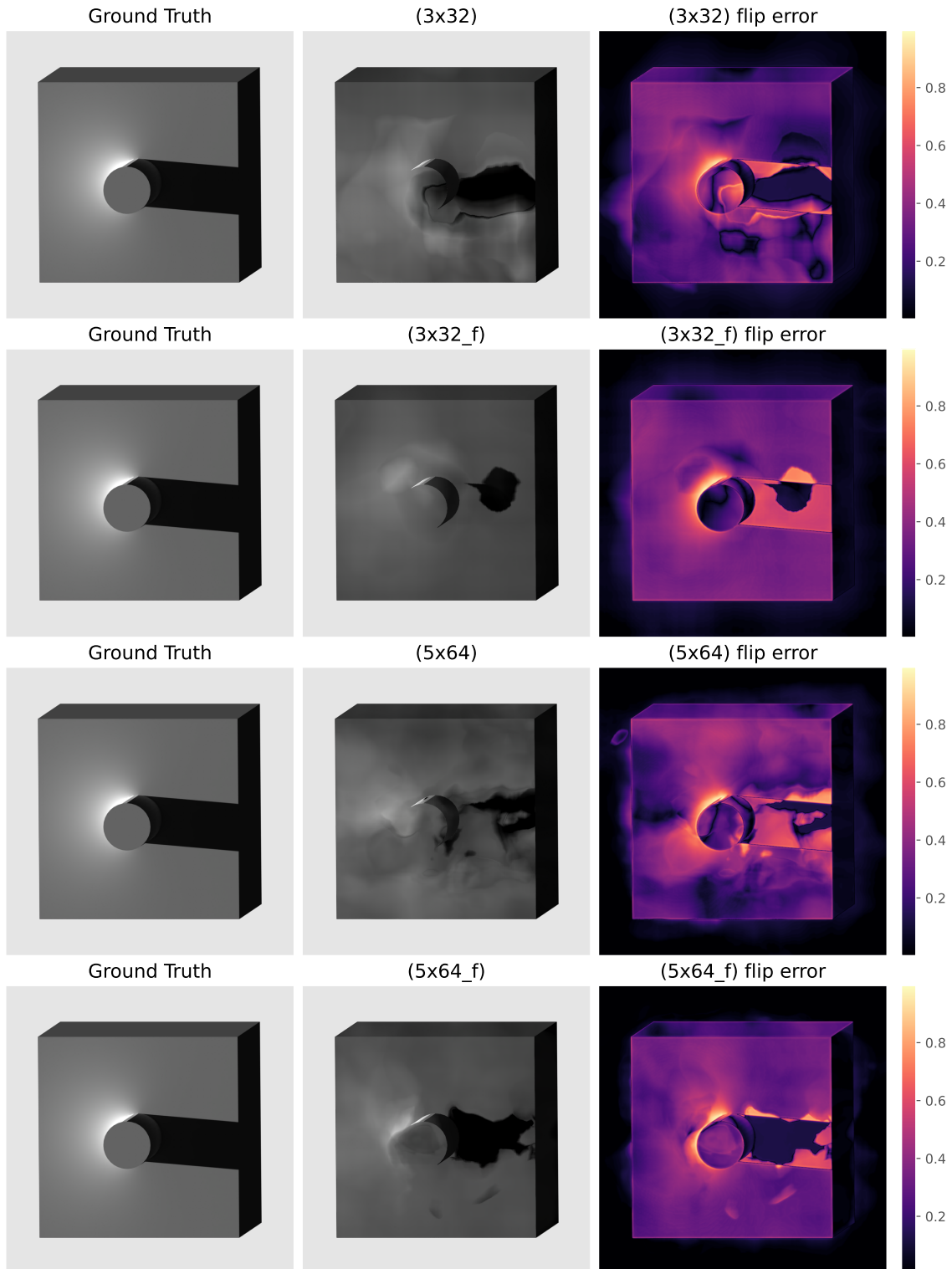


Figure 4.2: Comparison of the 4 models on the same evaluation synthetic evaluation image, with FLIP error heatmaps.

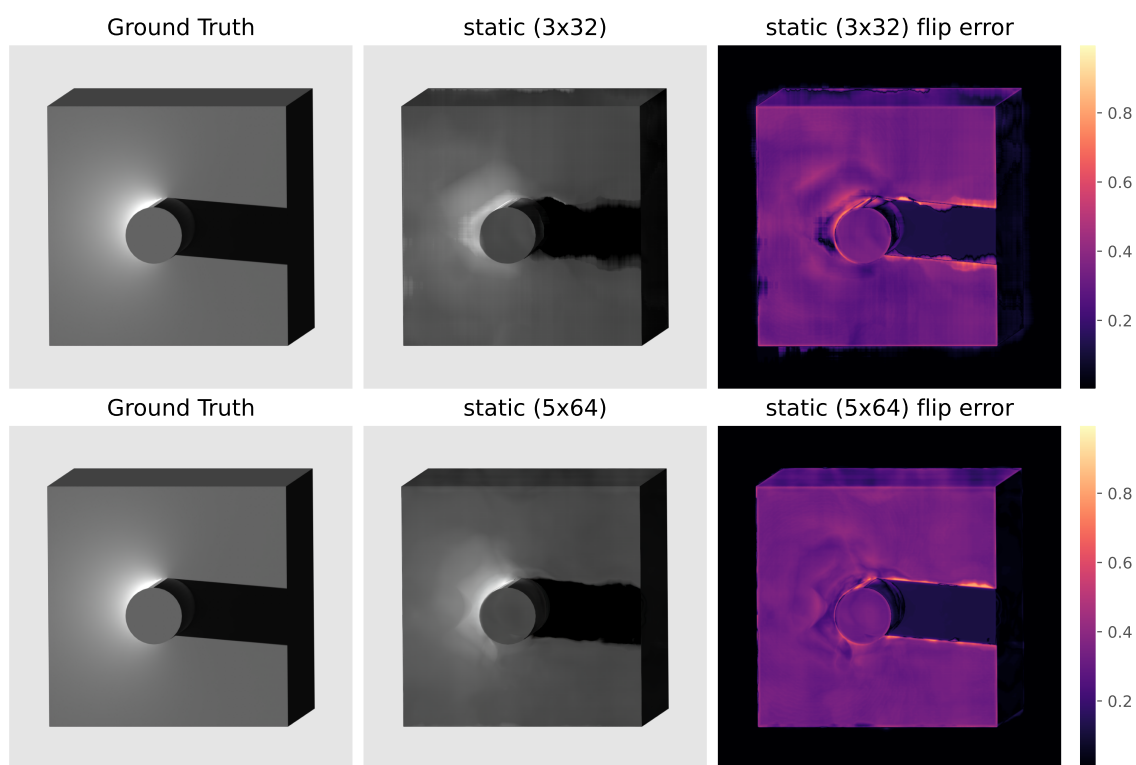


Figure 4.3: Comparison of the static version of the model, two sizes reflecting the sizes of the animated version.

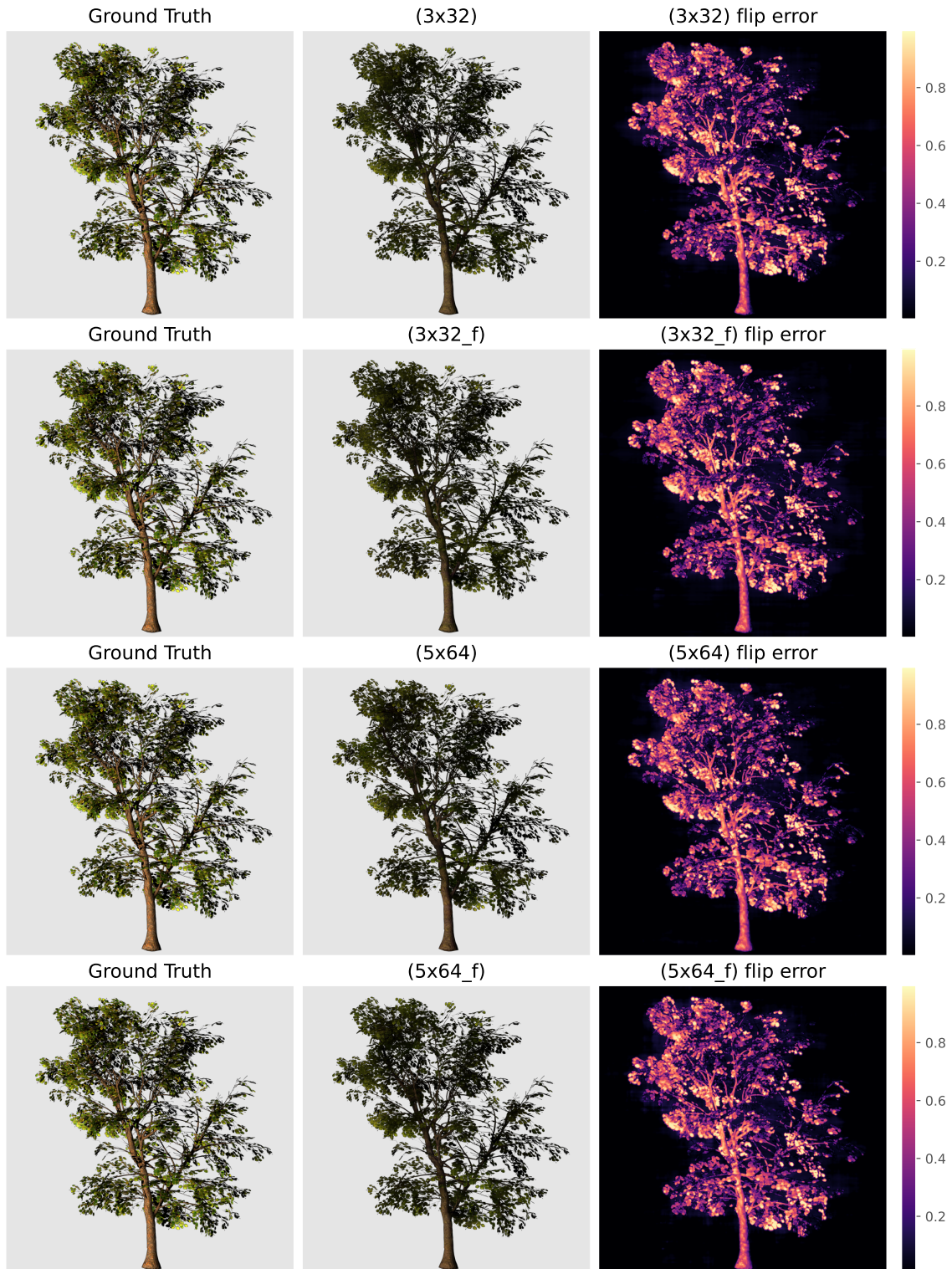


Figure 4.4: Comparison of the 4 models on the same tree evaluation image, with FLIP error heatmaps.

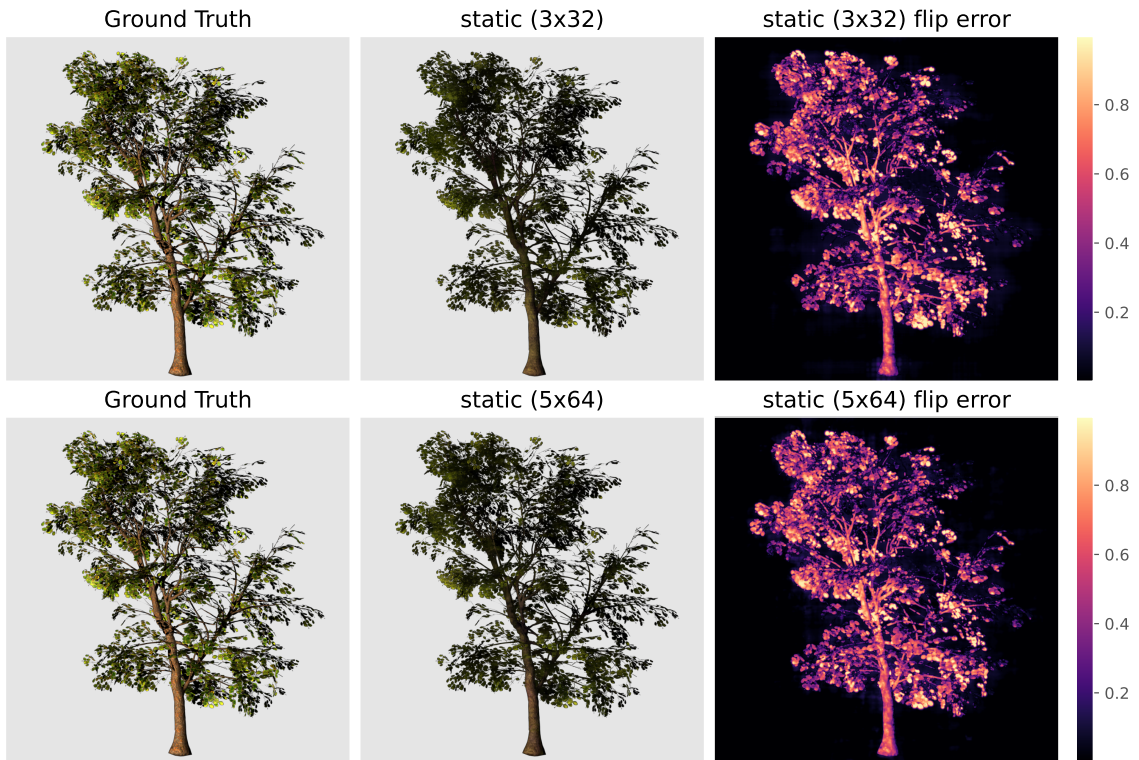
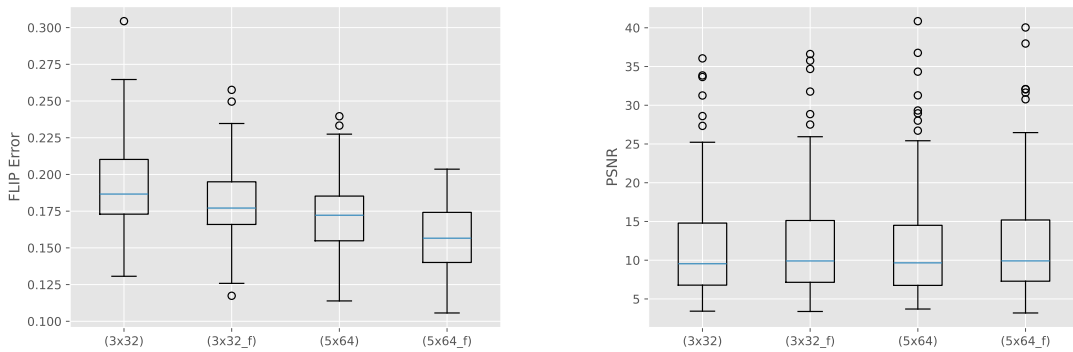


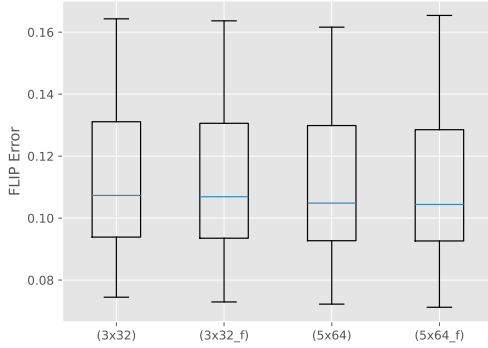
Figure 4.5: Comparison of the static version of the model, two sizes reflecting the sizes of the animated version.



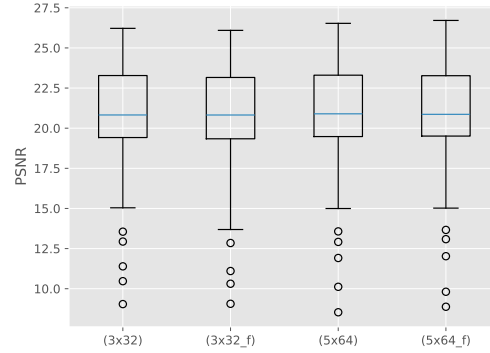
(a) FLIP error scores over the synthetic evaluation dataset.

(b) PSNR scores over the synthetic evaluation dataset.

Figure 4.6: Performance of the four models on the synthetic evaluation dataset with 100 animation frames with random lighting directions.

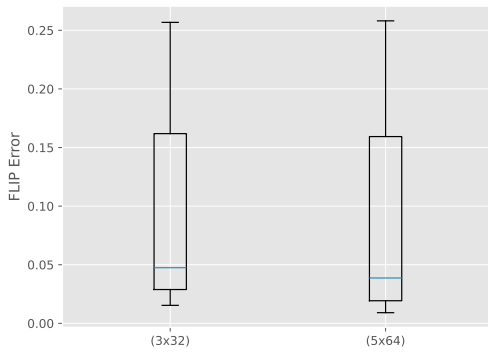


(a) FLIP error scores over the tree evaluation dataset.

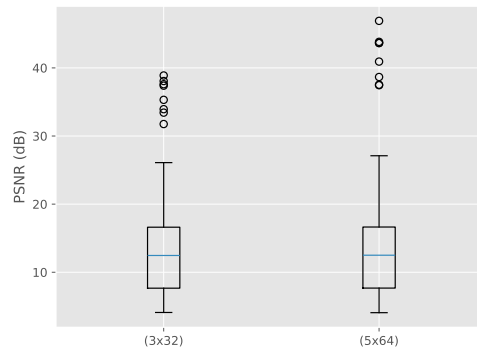


(b) PSNR scores over the tree evaluation dataset.

Figure 4.7: Performance of the four models on the tree evaluation dataset with 100 animation frames with random lighting directions.

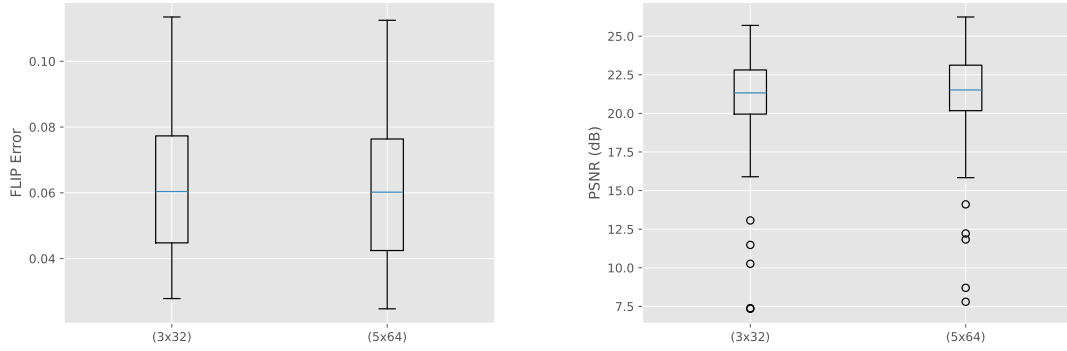


(a) FLIP error scores over the synthetic evaluation dataset.



(b) PSNR scores over the synthetic evaluation dataset.

Figure 4.8: Performance of the two static models on the static synthetic evaluation dataset



(a) FLIP error scores over the tree evaluation dataset.

(b) PSNR scores over the tree evaluation dataset.

Figure 4.9: Performance of the two static models on the static tree evaluation dataset



Figure 4.10: Section of the ground truth image (left) and the $(5 \times 64_F)$ image (right) of the tree dataset. Notice the absence of sharp shadows in the MLP generated image.

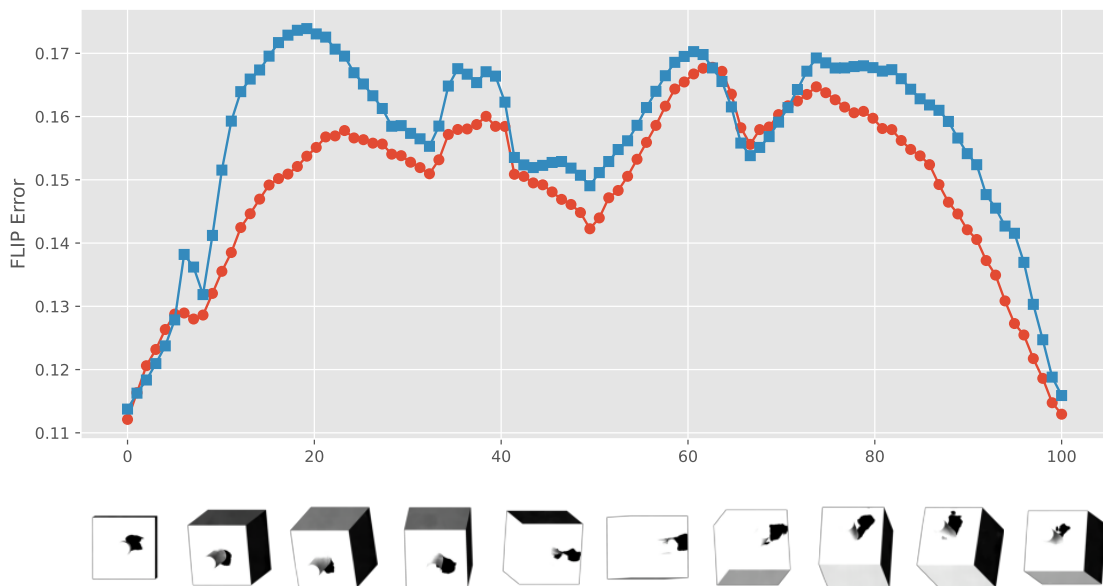


Figure 4.11: FLIP Errors over the full animation cycle of the synthetic evaluation dataset under static light conditions. Every second frame is interpolated and was not part of training data. The red curve with circle markers denotes $(5 \times 64_F)$, and the blue curve with square markers denotes (5×64) .

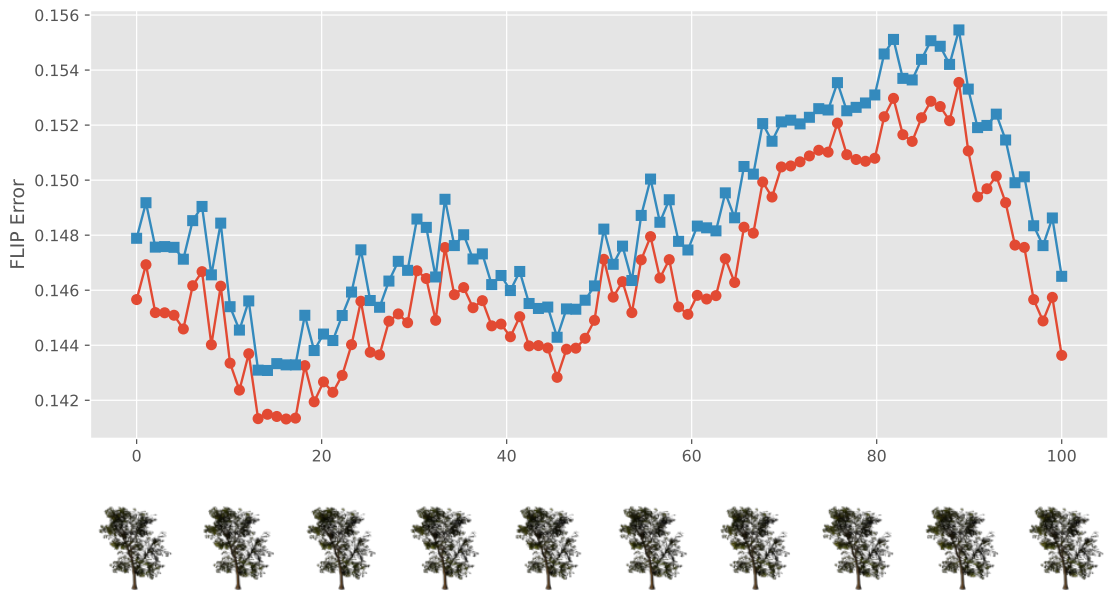


Figure 4.12: FLIP Errors over the full animation cycle of the tree evaluation dataset under static light conditions. Every second frame is interpolated and was not part of training data. The red curve with circle markers denotes $(5 \times 64_F)$, and the blue curve with square markers denotes (5×64) .

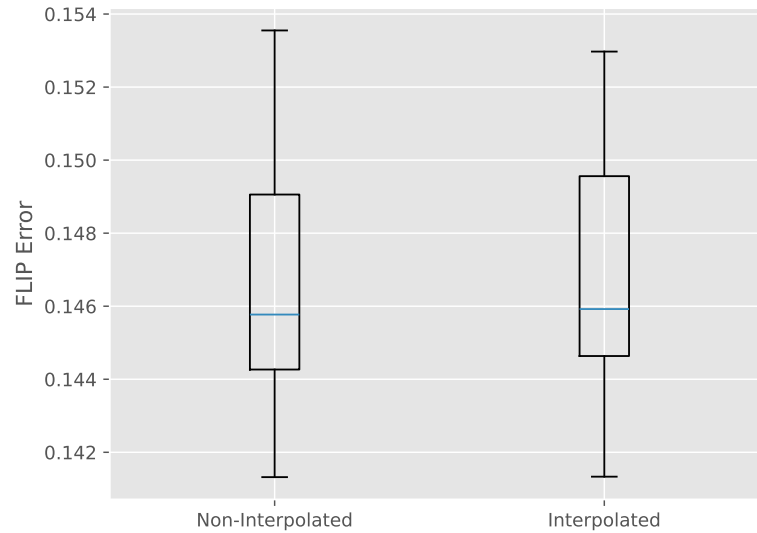


Figure 4.13: Comparison in FLIP error between the tree evaluation animation frames that are non-interpolated i.e. present in dataset, and interpolated i.e. not present in dataset. The evaluation dataset has static lighting conditions for this comparison.

Chapter 5

Discussion

In our results we have shown a solution for neurally relighting animated billboard elements, with four variations in model architecture. There is a large increase in frame time when using the large model of around 3.86x the frame time of the small model (Table 4.1), for only a relatively modest decrease in $\mathfrak{F}$ LIP error scores (Figure 4.6 and 4.7), though it is debatable if the increase in frame time is acceptable or not depending on the application, as it is still well within the time window for real-time rendering.

On the synthetic dataset especially, there is a visible increase in accuracy of the self shadowing and indirect light when increasing the model size (Figure 4.2), however, on the more visually busy tree dataset, this difference is negligible (Figure 4.4). Adding the frequency encoding to the time variable is found to be highly favorable, as there is no increase in frame time (Table 4.1), but a comparable $\mathfrak{F}$ LIP error decrease to the more expensive increase in model size (Figure 4.6). Figures 4.11 and 4.12 show a clear improvement in $\mathfrak{F}$ LIP error scores when frequency encoding for the time variable is used in a frame by frame comparison.

The self-shadowing of the tree is highly chaotic due to its thin branches, and therefore the MLP model struggles to faithfully recreate the fine detail and highly light- and time-dependent nature of the shadows as seen in Figure 4.10. We speculate that a performance increase for the tree dataset would require a much larger dataset with more lighting conditions. The synthetic dataset would also benefit from a larger dataset, though the performance metrics show a limitation from model architecture in the synthetic dataset application (Figure 4.6), whereas almost no such implication is shown by the tree evaluation performance metrics (Figure 4.7).

The static neural billboard version shows near zero difference in performance between the two model sizes (Figure 4.8 and Figure 4.9), implying that its performance is not limited by the model size. This strongly suggests that the amount of information present in the animated datasets is significantly larger than the static datasets, and requires a more complex model architecture for convergence.

5.1 Interpolation

As seen in Figure 4.13, there is only a very small difference in FLIP error between the non-interpolated and interpolated frames and so the model shows a good ability to interpolate the animation frames, showing that space can be saved in the training dataset by only using half of the frames in the target animation.

5.2 Answers to Research Questions

- We were not able to achieve similar performance as the static neural billboards, and we theorize the need for a larger dataset and possibly a larger model to capture the increase in information between the versions.
- The performance is shown to benefit from an increase in model size, with a moderate frame time increase, though the limit of frame time budget to afford the neural billboards in actual application remains ambiguous, especially as the billboards would realistically not occupy a majority of rendered fragments.
- The results show that the MLP model performance is greatly aided by an encoding scheme on the time axis with no drawbacks.

5.3 Usability in Artist Workflows

With proper tooling development, one could imagine a program that could allow a 3D artist to generate datasets for their 3D-models and train ANN-models without much manual labor or technical proficiency, ANN hyperparameters and view angle would need to be configured for the needs of the specific application. A potential limiting factor for usability in the workflow of asset production comes from the computation time needed to train the model, though with optimizations done to the dataset handling by the training program and advanced hardware it could be within realistic possibility to use the type of ANN solution presented in this paper.

5.4 Further Work

There are a wide assortment of possibilities where improvement in this type of MLP model is concerned. Much progress has been made in the field of Neural Video Encoding which could be relevant for this application, though there is a fundamental difference that could make many of the techniques used in Neural Video Encoding impossible or impractical: the need for this MLP system to be run on a per-pixel basis for compatibility with the fragment shader and the acceleration associated with it, whereas Neural Video Encoding models are generally constructed for entire frame inference.

5.4.1 Motion Vectors

It could be advantageous to encode motion vectors for the animation movement. The usage of motion vectors could expose some valuable features for the training of the network and could let the model reuse information from previous frames where a point of interest has the same appearance but has moved to another point in UV coordinate space. This could in turn reduce the required size of the dataset for adequate convergence.

5.4.2 Dataset Streaming

The dataset size and therefore the accuracy and generalization of the model was limited by the amount of hardware memory available on our machine. We theorize that the quality of results from the model could be greatly improved by training on a larger dataset that includes more examples of incident light angles.

5.4.3 Encoding Schemes

In this work, only one type of encoding scheme was attempted on the time axis in one dimension. It is possible that another encoding scheme or frequency dimensionality gives stronger results and is worth experimenting with, although we speculate that greater gains are to be had in a more fundamental rework of the model, such as the use of motion vectors.

Chapter 6

Conclusion

In this work, we have presented a neural model system for relighting an animated billboard element. Four model architectures were compared, demonstrating the effectiveness of encoding the time feature, and showing a need for a larger model with a longer inference time while remaining in the window for potential real-time application. We show that animation frames can be interpolated during inference and therefore the required size of the training dataset can be reduced by a notable amount. We theorize the need for a larger dataset and possibly a larger model architecture to capture the increase in information that comes with the animation and bring the visual fidelity closer to the static billboard version. With further work, we see potential for animated neural billboards in real-time applications as an efficient technique for increased visual fidelity without much added manual labor.

References

- [1] P. Andersson, J. Nilsson, T. Akenine-Möller, M. Oskarsson, K. Åström, and M. D. Fairchild. FLIP: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.*, 2020.
- [2] B. Burley. Extending the disney brdf to a bsdf with integrated subsurface scattering. *ACM SIGGRAPH 2015*, 2015.
- [3] E. Golubev. Efficient screen-space subsurface scattering using burley’s normalized diffusion in real-time. *ACM SIGGRAPH Talks*, 2018.
- [4] R. Habel, P.H. Christensen, and W. Jarosz. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. *Eurographics Symposium on Rendering*, 2013.
- [5] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’86, pages 143–150, New York, NY, USA, 1986. ACM. doi:10.1145/15922.15902.
- [6] Khronos Group and J. Bolz. VK_NV_cooperative_vector, 2024. Vulkan Documentation Project, last modified 2024-05-23, accessed 2026-03-05. URL: https://docs.vulkan.org/features/latest/features/proposals/VK_NV_cooperative_vector.html.
- [7] K. Mullia, F. Luan, X. Sun, and M. Hašan. Rna: Relightable neural assets. *ACM Trans. Graph.*, 2023.
- [8] R. Olajos and M. Doggett. Neural billboard for real-time rendering. 2026.
- [9] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975. doi:10.1145/360825.360839.
- [10] M. Tancik, P. P. Srinivasan, Mildenhall B., Fridovich-Keil S., N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. URL: <https://arxiv.org/abs/2006.10739>, arXiv:2006.10739.

- [11] K. Vaidyanathan, M. Salvi, B. Wronski, T. Akenine-Möller, P. Ebelin, and A. Lefohn. Random-access neural compression of material textures. *ACM Trans. Graph.*, 42(4):1–25, July 2023. URL: <http://dx.doi.org/10.1145/3592407>, doi:10.1145/3592407.
- [12] T Zeltner, F. Rousselle, A. Weidlich, P. Clarberg, J Novák, B. Bitterli, A. Evans, T. Davidovič, S. Kallweit, and A. Lefohn. Real-time neural appearance models. *ACM Trans. Graph.*, 2023.

EXAMENSARBETE Animated Neural Billboards for Real-Time Rendering**STUDENT** Edwin Gustafsson**HANDLEDARE** Rikard Olajos (LTH)**EXAMINATOR** Michael Doggett (LTH)

Neural ljussättning av animerade billboards för datorgrafik i realtid

POPULÄRVETENSKAPLIG SAMMANFATTNING Edwin Gustafsson

Att rita många detaljrika objekt, såsom träd, i datorspel kan bli krävande för datorn. Objekt kan då förenklas till platta bilder, så kallade billboards, dessa är enklare för datorn att hantera, men kan inte interagera med ljus realistiskt. Detta arbete presenterar en neural modell för att göra realistisk ljussättning möjlig på animerade billboards.

För rendering av datorgrafik i realtid är det viktigt att hålla nere antalet komplexa objekt i en scen. Billboards har använts sedan de första 3D-renderade spelen och fortsätter vara populära idag, där komplexa objekt som inte är menade att synas i fokus reduceras till en platt bild, detta är speciellt vanligt för träd långt från kameran, ofta är dessa billboards även animerade, till exempel för att visa träd som svajar i vinden. Billboards kommer med kompromisser, det blir omöjligt att låta det avbildade objektet interagera med ljus som det vanligtvis hade gjort. I mitt arbete är målet att tillåta realistisk ljussättning av dessa animerade billboards medan effektiviteten behålls.

Ett neuralt nätverk är ett stort system av variabler som anpassas under en inlärningsprocess för att kunna återskapa mönster som uppstår i en samling in- och utdata. I denna applikation matas nätverket med realistiska renderingar av ett objekt i de olika delarna av dess animationscykel och med många olika positioner av ljuskällan. Nätverket producerar färgen för en pixel åt gången, så att systemet sömlöst passar in i den vanliga renderingsprocessen. Nätverket tar information om scenen och färgen av den traditionella billboarden för att producera den nya färgen för pixeln som visas på skärmen.



Figur 1: Övre raden, från vänster: detaljrik bild av ett komplext objekt, traditionell billboard, den neurala ljussättningen. Nedre raden: neuralt ljussatt syntetiskt exempel över sin animationscykel.

I Figur 1 syns tre bilder av ett träd och ett exempel av en animation. Den neuralt ljussatta bilden visar ett utseende närmre den detaljerade referensbilden än den traditionella billboarden, och kan producera mer komplicerade ljusinteraktioner, som att ljus skinner igenom löven och att objekt skugglägger sig själv.

Denna teknik visar potential för realtidsrendering, men den skulle gynnas av vidare utveckling för mer precis återskapning av referensbilderna.