جامعة ابن طفيل
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉ ⵎⵓⵃⴰⵎⵎⴷ
Ibn Tofaïl University
Faculté des Sciences

Université Ibn Tofail
Faculté des Sciences, Kénitra

Mémoire de Projet de Fin d'Etudes

Master Intelligence Artificielle et Réalité Virtuelle

3D Mesh Saliency Prediction Using Deep Learning Methods

Établissement d'accueil : FS KENITRA -UIT

Elaboré par : Hicham EN-NASIRY et Zakariae LAGOUADER

Encadré par : Mr. Anass NOURI (FS KENITRA -UIT)

Mme. Souad EDDAROUICH (Centre Régional des Métiers de l'Education et de la Formation de Rabat)

Soutenu le ***24/09/2024, devant le jury composé de :***

- Mme Raja TOUAHNI (FS KENITRA – UIT)
- Mr Rochdi MESSOUSSI (FS KENITRA – UIT)
- Mr Anass NOURI (FS KENITRA – UIT)
- Mme Souad EDDAROUICH (CRMEF Rabat)

Année universitaire 2023/2024

Abstract

Immersive applications and virtual environments are becoming more common in everyday life. These technologies create computer-generated worlds where users feel fully immersed, interacting with 3D scenes and objects as if they were real. In these environments, users control what they see and how they interact, which means everyone experiences the same scene differently. This makes it harder for content creators to guide the user's attention to specific areas. As a result, predicting where users will look in virtual environments remains a challenge, but it is important for tasks like improving graphics performance and designing better content.

Researchers have studied human visual attention to understand which parts of a scene attract the most attention. Eye-tracking data is often used to identify these important areas, called salient regions. In 3D, attention is generally drawn to the same regions of a scene or object when viewed for the first time, as the human visual system is similar across individuals. This consistency in attention makes it possible to predict saliency in 3D environments, especially during initial exposures to a scene.

This thesis presents a deep learning model to predict saliency maps on 3D meshes. Unlike older methods that focus on point clouds or geometric properties like the shape of the object, our model uses convolutional neural networks (CNNs) and real gaze data to predict where users are likely to look. By using this data and adapting an existing CNN model, we were able to predict the most important areas on 3D objects. The results show that our approach can successfully highlight the most attention-grabbing parts of a 3D mesh, offering a promising solution for saliency prediction in virtual environments.

Acknowledgments

We would like to extend our sincere thanks to our professors, Anass Nouri and Souad EDDAROUICH, for their invaluable guidance, patience, and expertise throughout this project. Their support has been crucial to the success of our research.

We are also deeply grateful to the teaching staff of Ibn Tofail University for providing us with the resources, knowledge, and encouragement that helped us grow throughout our studies.

We would like to acknowledge our friends and fellow students, whose collaboration and support made this experience even more enriching. Their advice and encouragement kept us motivated during the challenging phases of our work.

Lastly, we are profoundly thankful to our families for their constant support and belief in us. Their encouragement has been our foundation during this journey, and we couldn't have achieved this without them.

Contents

List of Figures

List of Tables

Introduction

Project context

In computer graphics, predicting visual attention in 3D environments has become increasingly important. As immersive applications and virtual environments grow, understanding and modeling human visual attention plays an important role in areas like rendering compression, 3D content design, face recognition. Over the years, the challenge of predicting saliency in 3D meshes has been explored extensively. Early approaches relied on statistical algorithms [8, 9, 15, 18], while more recent methods use neural networks trained on classification tasks [16, 11]. These methods all aim to capture the fact that certain regions of a 3D surface are more visually important based on human perception. However, many saliency prediction models still rely on handcrafted features and descriptors, which often fail to generalize effectively. For instance, Fig. 1 shows a saliency map created using real gaze data from an experiment by Lavoué et al. [7], compared to predictions from statistical saliency methods. These methods tend to incorrectly highlight areas with sharp curves and fine details, rather than the regions that truly capture human attention.

Figure 1: Results of 3D mesh saliency prediction using statistical algorithms.

Goal and scope of the Project

The motivation of this Master's thesis is to face the saliency prediction problem on 3D meshes by making use of deep learning and utilising real gaze data as ground

truth to train the model. The project proposes several goals:

- Reviewing the latest research on how humans focus their attention when looking at 2D images and 3D objects, how saliency (important areas) is predicted for both 2D and 3D content, and exploring deep learning methods for 3D meshes (Section 1).
- Adaptation and implementation of a deep learning-based model for saliency prediction on 3D meshes (Section 2).
- Evaluation of the performance of the model and comparison with other state-of-the-art works that predict saliency on 3D meshes (Section 3).

Chapter 1

Related Work

1.1 What are 3D Meshes?

A 3D mesh is a representation used to model the surface of three-dimensional objects in computer graphics and related fields. It consists of interconnected vertices, edges, and faces that together define the shape of an object.

Vertices are the basic units, representing points in 3D space, each defined by its x, y, and z coordinates. These vertices are connected by edges, which are straight lines that form the skeleton of the mesh. Faces are the surfaces enclosed by these edges, with the most common face types being triangular and quadrilateral. Triangular meshes, in particular, are widely used due to their simplicity and ability to approximate complex surfaces with a high degree of accuracy.

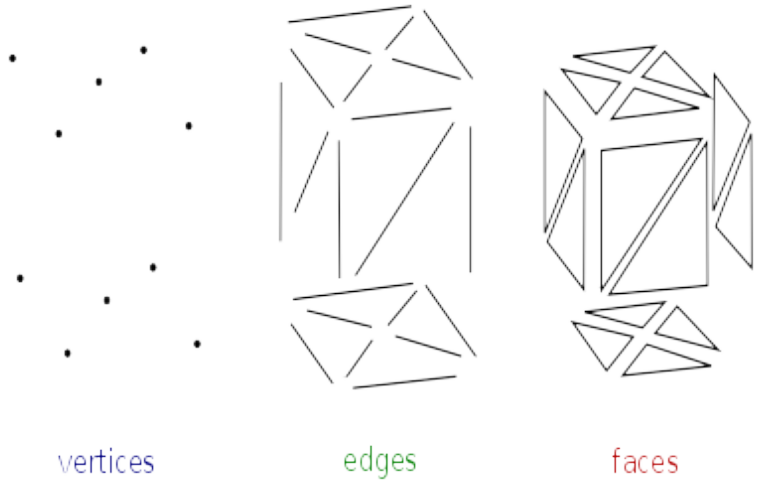

Figure 1.1: The components of a 3d mesh: vertices, edges and faces

3D meshes play a critical role across a range of industries. In gaming and virtual reality, they are used to create lifelike characters, environments, and objects that enhance user immersion. In medical imaging, 3D meshes help in visualising and analysing anatomical structures, providing valuable insights for diagnosis and treatment planning. In computer-aided design (CAD), they are essential for modeling and simulating real-world objects, allowing for precise design and engineering.

3D meshes are typically stored in specialized file formats that efficiently encode the geometric and sometimes material information of the object. These file formats vary in their capabilities, compression methods, and industry adoption. Some formats are optimized for specific applications, while others are more general-purpose. Below is a table listing five popular file formats used for storing 3D meshes:

File Format	**Extension(s)**	**Description**
Wavefront	.obj	Text-based, stores geometry and materials.
Stereolithography	.stl	Geometry-only, widely used in 3D printing.
Filmbox	.fbx	Supports animations, textures, and geometry.
COLLADA	.dae	XML-based, for 3D data exchange.
GLTF	.gltf, .glb	Web-friendly, supports geometry and animations.

Table 1.1: Popular file formats for storing 3D meshes.

Each of these formats has its strengths and is used in different contexts. For instance, OBJ is widely supported and simple, STL is common in 3D printing, FBX is often used in animation and game development, COLLADA is designed for exchanging digital assets, and GLTF is optimized for efficient transmission and loading of 3D scenes and models.

1.2 2D Saliency Prediction

Visual saliency in the context of 2D images refers to the perceptual quality that makes certain regions of an image stand out and attract more attention than others. This concept is rooted in the idea that the human visual system prioritizes certain areas within a scene, often driven by factors like contrast, color, and texture. Identifying these salient regions is crucial for various applications, from image compression to guiding attention in user interfaces.

Traditional approaches to 2D saliency prediction largely relied on heuristic-based methods and early computer vision techniques. One of the proposed was by Itti et al. [5], which drew inspiration from the human visual system. This model used

a bottom-up approach, combining features such as color, intensity, and orientation to create a saliency map. Other early methods employed various techniques, such as center-surround differences, frequency domain analysis, and graph-based models, each aiming to mimic aspects of human visual attention.

The advent of deep learning has significantly advanced the field of 2D saliency prediction, offering more accurate and robust models. Convolutional Neural Networks (CNNs) have become the cornerstone of modern saliency prediction techniques, thanks to their ability to automatically learn hierarchical features from large datasets. Models like those proposed by Cornia et al. [2] and Judd et al. [6] have set new benchmarks by leveraging deep architectures to predict saliency with high precision. These networks typically involve multiple layers of convolutional and pooling operations, allowing them to capture both low-level and high-level visual features that contribute to saliency.

For example, Cornia et al. [2] introduced an architecture that combines features extracted at different levels of a CNN, significantly improving the accuracy of saliency prediction. Likewise, Judd et al. [6] proposed a model that integrates low-level visual features with high-level semantic information, such as the presence of faces or text, to enhance the prediction of eye fixations.

1.3 3D Mesh Saliency Prediction

3D mesh saliency prediction, much like its 2D counterpart, aims to identify regions of a 3D mesh that are most likely to attract human attention. The primary goal is to reveal the perceptually significant areas of a 3D object, taking into account the additional complexity introduced by the third dimension. This capability is particularly valuable in various applications, such as mesh compression, where understanding saliency allows for the reduction of polygon counts in less critical areas while maintaining high detail in regions that are more likely to capture attention. Additionally, in user interaction scenarios, predicting where users are likely to focus enables the design of more sophisticated and responsive interactions within 3D environments.

While the prediction of saliency in 2D images has been widely studied, the extension of these concepts to 3D objects presents unique challenges and has received less attention. Early approaches to 3D saliency often relied on geometric and statistical methods to estimate which areas of a mesh would be considered salient. For instance, Lee et al. [8] introduced a method that defined mesh saliency in a scale-

dependent manner, employing a center-surround operator on Gaussian-weighted mean curvatures. This approach calculated local mesh saliency by measuring the absolute difference between the Gaussian-weighted mean curvatures at different scales, thereby highlighting areas of interest based on their geometric properties. Similarly, Leifman et al. [9] developed an algorithm that identifies regions of interest on surfaces by detecting areas that are distinct both locally and globally, while also considering the distance to potential foci of attention. Another significant contribution was made by Song et al. [15], who analyzed the log-Laplacian spectrum of a mesh to capture saliency in the frequency domain. By identifying frequencies that deviated from expected behavior and localising this information spatially across multiple scales, they were able to pinpoint salient features on the mesh.

The advent of deep learning has brought about a significant shift in the approach to 3D saliency prediction, enabling more sophisticated and accurate models. For example, Song et al. [16] proposed the Classification-for-Saliency CNN (CfS-CNN), a deep learning network designed to predict 3D mesh saliency in a weakly supervised manner. This model, which leverages mesh class labels instead of detailed vertex-level annotations, utilizes a multi-view setup with a two-channel structure to handle both classification and saliency prediction tasks. By incorporating transfer learning from 3D object classification and introducing a novel saliency pooling layer, the CfS-CNN effectively enhances the accuracy of saliency prediction.

Another notable deep learning approach is presented by Martin et al. [10], who developed SAL3D, a model based on the PointNet++ [12] architecture. This model is designed to predict visual saliency on 3D meshes using real-world gaze data collected from virtual reality environments, capturing natural human viewing behavior. SAL3D extracts features from local regions of the mesh and propagates them to predict saliency, offering a significant improvement over previous methods that relied on handcrafted features or data gathered in less natural settings.

In addition, Nousias et al. [11] introduced a CNN-based approach that addresses the challenges of detecting saliency in large and dense 3D meshes. Their architecture combines spectral and geometric features to create a robust training set, enabling the network to learn and predict saliency maps directly from the 3D mesh data. By employing a patch-based descriptor and using a Hilbert curve for data arrangement, the model achieves rotation invariance and maintains accuracy even with smaller training datasets. This approach not only improves the speed and scalability of saliency detection but also proves effective in applications such as mesh simplification and compression.

1.4 Machine and Deep Learning for Saliency Prediction of 3D Meshes

The application of deep learning to 3D mesh data represents a significant advancement in computer vision and graphics, leveraging the power of neural networks to address the complexities of three-dimensional data. Deep learning techniques have evolved to handle various forms of 3D data, including point clouds, voxels, and meshes. Each representation offers unique advantages and challenges, particularly with respect to how neural networks process and interpret 3D information.

Point clouds, composed of sets of points in 3D space, are often used in deep learning for tasks such as object recognition and segmentation. Voxels, which represent 3D data in a grid-like structure, are another common representation, enabling the application of 3D convolutional neural networks (CNNs) to process volumetric data. However, meshes, which are defined by vertices, edges, and faces, present additional challenges due to their irregular and non-uniform structure. This complexity necessitates the development of specialized deep learning architectures that can effectively handle the geometric intricacies of mesh data.

One of the approaches in applying CNNs to 3D meshes is MeshCNN [3], which adapts traditional CNNs to work directly with mesh data. MeshCNN operates by applying convolutional operations on the edges of a mesh, which enables it to capture local and global features more effectively than voxel-based approaches. This method is particularly useful for tasks such as mesh segmentation and classification, where understanding the relationships between adjacent faces and edges is crucial.

In addition to these architectures, graph neural networks (GNNs) have emerged as a powerful tool for processing 3D mesh data. GNNs are designed to handle data represented as graphs, where nodes represent points or vertices and edges represent the connections between them. This approach allows for the modeling of complex relationships in mesh data, capturing both local and global geometric information. some papers used this architecture like [14, 13]

This Master's thesis introduces an approach to 3D mesh saliency prediction, building upon the method developed by Nousias et al. [11] Unlike their approach, which utilized synthetic data for training, this work employs real gaze data as the ground truth for saliency prediction. Additionally, with several modifications that will be detailed in Section 2.

The main contributions of this research are: (1) the development of a deep learning

model specifically for predicting 3D mesh saliency, (2) the training of this model using real saliency data, (3) a comparison of the model's predicted saliency maps with the real ground truth, as well as a comparison with other existing methods for 3D mesh saliency prediction.

Chapter 2

Model for 3D mesh saliency prediction

As previously mentioned at the end of Section 1.4, the deep learning model used in this project for predicting saliency on 3D meshes is based on the architecture proposed by Nousias et al. [11]. The key concept of their model is as follows: for each triangle (face) in the 3D mesh, a 3D descriptor, referred to as a "Patch," is generated. These patches are then fed into a CNN model to train on them and predict saliency values for each triangle (face) in the mesh.

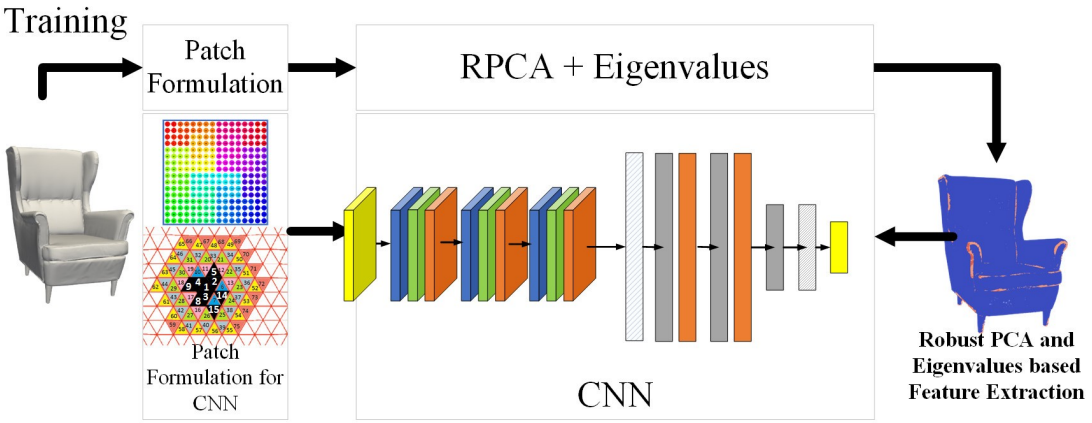

Figure 2.1: CNN-based saliency map extraction pipeline. Adapted from Nousias et al. [11].

As shown in Figure 2.1, an additional step in the training process, labeled "RPCA + Eigenvalues," was included in the original study. This step was used to generate baseline saliency maps for the 3D meshes to train the CNN. The method relies on Robust Principal Component Analysis (RPCA) and Eigenvalue decomposition to extract saliency features, capturing both the spectral and geometric properties of each

patch. However, we have chosen to omit this step in our work, as we use real gaze data for training instead.

2.1 Preliminaries and Basic Assumptions

Before explaining the patch formulation process, it is essential to outline several key definitions regarding 3D mesh geometry.

A triangular 3D mesh $\mathcal{M}$ is defined by n_v vertices $\mathbf{v}$ and n_f faces $\mathbf{f}$. Each vertex $\mathbf{v}_i$ is defined as:

$$\mathbf{v}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \quad \forall i = 1, 2, \ldots, n_v \tag{2.1}$$

Each face $\mathbf{f}_j$ is a triangle formed by three vertices:

$$\mathbf{f}_j = \{\mathbf{v}_{j1}, \mathbf{v}_{j2}, \mathbf{v}_{j3}\}, \quad \forall j = 1, 2, \ldots, n_f \tag{2.2}$$

The centroid $\mathbf{c}_j$ of each face is calculated as:

$$\mathbf{c}_j = \frac{\mathbf{v}_{j1} + \mathbf{v}_{j2} + \mathbf{v}_{j3}}{3} \tag{2.3}$$

The outward unit normal $\mathbf{n}_{ci}$ of a face, which is a key feature in saliency extraction, is computed as:

$$\mathbf{n}_{ci} = \frac{(\mathbf{v}_{j2} - \mathbf{v}_{j1}) \times (\mathbf{v}_{j3} - \mathbf{v}_{j1})}{\|(\mathbf{v}_{j2} - \mathbf{v}_{j1}) \times (\mathbf{v}_{j3} - \mathbf{v}_{j1})\|} \tag{2.4}$$

2.2 Patch Formulation

The sliding 3D saliency descriptor (referred to as a "Patch") takes as input a list of normal vectors of the face in question, along with the normals of $N-1$ neighboring faces. This descriptor is then used as input for the deep neural network. Let S_i represent the set of face indices neighboring face f_i, and let $k = |S|$ be the cardinality of the neighborhood. We choose k to be a power of 2, i.e., $k = w^2$, where $w \in [4, 8, 16, 32, 64, 128, \ldots]$. The patch is then represented as:

$$S = [\mathbf{n}_{c1}, \mathbf{n}_{c2}, \mathbf{n}_{c3}, \ldots, \mathbf{n}_{ck}] \tag{2.5}$$

where $\mathbf{n}_c$ are the outward normal vectors from each face in the patch.

Currently, the patch is in a 1D format. To use it as input for the CNN, it must be transformed into a 2D representation. This can be done via a simple reshaping function or by using a space-filling curve like the Hilbert curve, as proposed by Nousias et al. [11]

The Hilbert curve is a continuous fractal that maps 1D data into 2D while preserving locality, meaning nearby points in the 1D space remain close in the 2D grid. This property ensures that the spatial relationships between neighboring face normals are preserved, making the grid more representative of the mesh's geometry. This structured conversion is crucial for CNNs to effectively capture local geometric features, which may be lost with a basic reshaping approach.

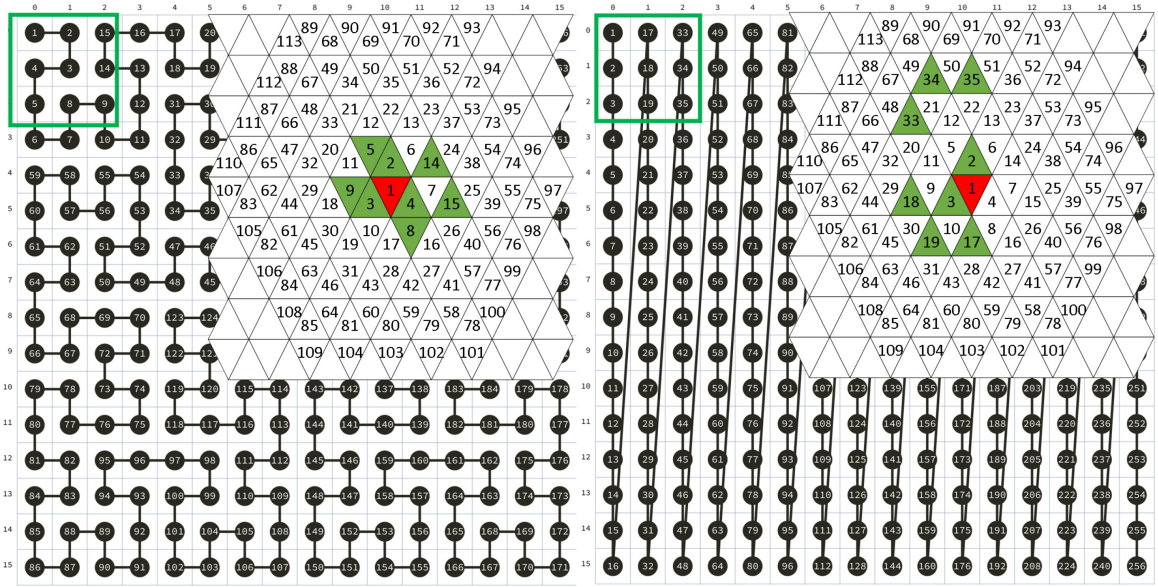

Figure 2.2: Comparison between using the Hilbert curve on the left and using simple reshape function on the right

To ensure rotation invariance, i.e., the descriptor remains consistent regardless of the mesh's size or orientation, each patch is rotated to align with a predefined target vector. This alignment allows the CNN to learn the saliency features without being affected by arbitrary rotations of the object.

2.3 CNN Architecture

The CNN architecture proposed by the paper is designed to take the processed patches as input and predict their corresponding saliency values. The architecture

consists of three convolutional layers, each followed by a max-pooling operation and a ReLU activation layer.

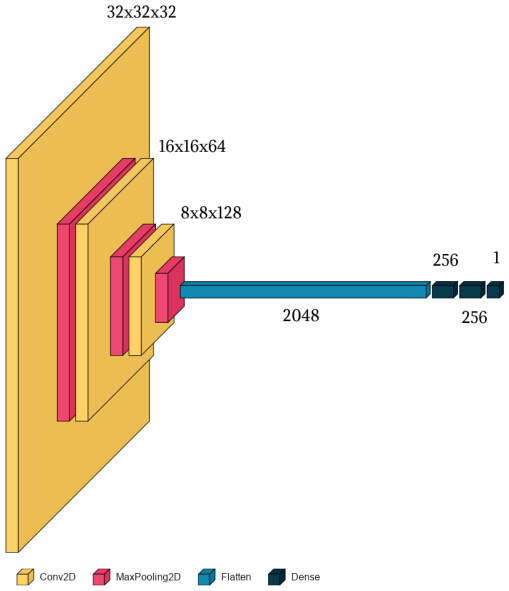

Figure 2.3: Nousias et al CNN architecture

The CNN architecture starts by processing $32 \times 32 \times 3$ patches, where the three channels correspond to the XYZ components of the face normals. The first convolutional layer has 32 filters of size 3×3, with ReLU activation. This is followed by a second convolutional layer with 64 filters, also using ReLU activation. The third convolutional layer increases the number of filters to 128, enabling the network to capture more detailed geometric features.

After the convolutional layers, the output is flattened into a 1D vector and passed through two fully connected layers, each with 256 units and ReLU activation. The final output layer consists of a single neuron with a sigmoid activation function, providing a saliency value normalized between 0 and 1 for each patch.

2.4 Model Modification

The first significant modification we made from the model proposed by Nousias et al. is replacing the synthetic dataset generated based on spectral and geometric properties with real gaze data. This change alone is expected to improve saliency prediction significantly, as real gaze data more accurately reflects human visual attention on 3D meshes. The figure below illustrates the comparison between the generated

dataset and the real gaze dataset.

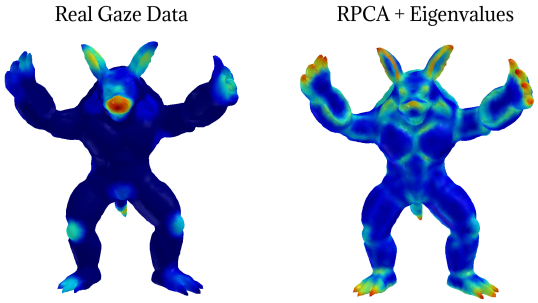

Figure 2.4: Comparison between generated dataset and real gaze dataset.

Another crucial modification was made to the architecture of the CNN model. We added several layers to enhance the model's ability to generalize and capture finer details from the input patches. These additional layers help the CNN better understand complex geometric patterns and improve its robustness to overfitting.

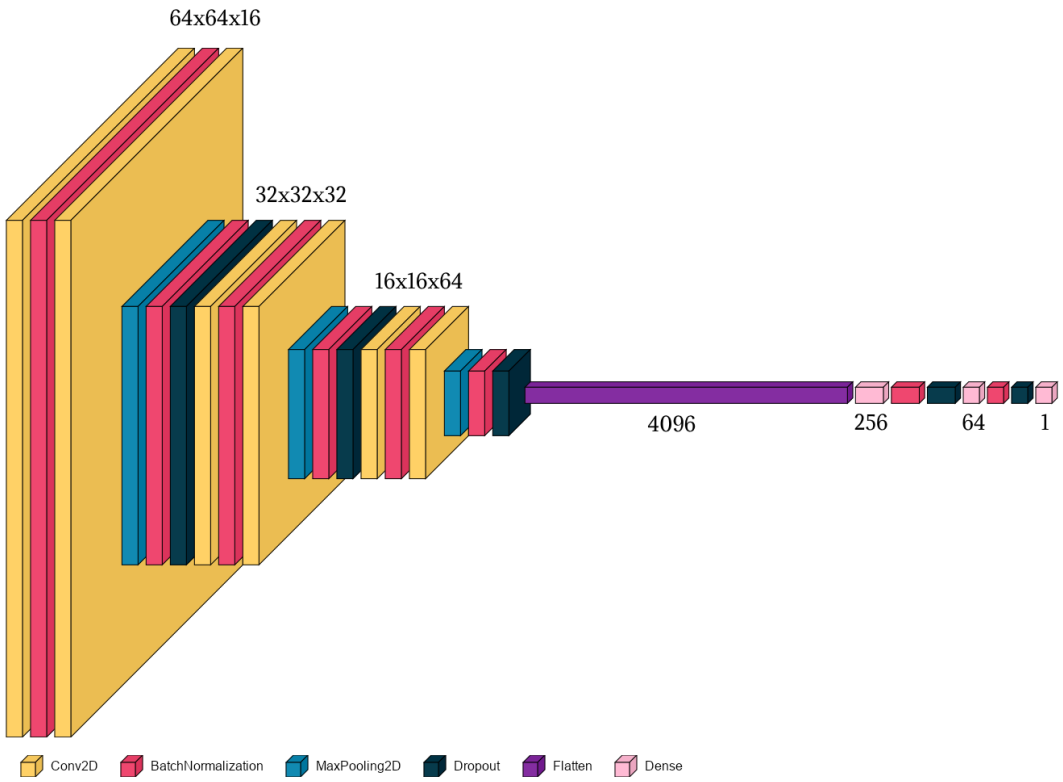

Figure 2.5: Our modified CNN architecture.

The key layers added to the architecture include:

- **Dropout:** Dropout is a regularization technique used to prevent overfitting by randomly "dropping out" a fraction of the neurons during training. This forces

the network to learn more robust features, as it cannot rely on any specific neuron during training. In our model, a dropout rate of 0.25 was applied after each convolutional block, and 0.5 after the fully connected layers, improving the model's generalization.

- **Batch Normalization:** Batch normalization is applied to normalize the activations of each layer. This helps in stabilising and accelerating training by reducing internal covariate shifts, allowing higher learning rates and improving convergence. Batch normalization also serves as a regularizer, slightly reducing the need for dropout.

Additionally, we reduced the number of filters in each convolutional layer compared to the original model, starting with 16 filters and gradually increasing them to 64. This adjustment strikes a balance between capturing important details and maintaining computational efficiency, particularly when combined with the added dropout and batch normalization layers.

Nousias et al. [11] used the Mean Squared Error (MSE) as their loss function. MSE measures the average squared difference between the predicted and actual values, effectively penalising larger errors. It is defined as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where y_i represents the actual saliency value, $\hat{y}_i$ represents the predicted saliency value, and n is the number of data points.

In contrast, we replaced MSE with the **Log-Cosh** loss function in our model. The Log-Cosh loss function is a smooth approximation of MSE, offering the advantage of being less sensitive to large errors (outliers). Log-Cosh calculates the hyperbolic cosine of the difference between the true and predicted values and is defined as:

$$\text{Log-Cosh} = \sum_{i=1}^{n}\log(\cosh(\hat{y}_i - y_i))$$

Log-Cosh behaves like MSE for small differences but grows more slowly for larger differences, making it less harsh on outliers. This property allows the model to be more robust to noisy or extreme data points, which may be particularly beneficial when working with real-world gaze data.

The primary difference between MSE and Log-Cosh is their sensitivity to large errors. MSE penalizes larger errors quadratically, which can cause the model to focus excessively on outliers. On the other hand, Log-Cosh provides a softer penalty for these errors, making it more robust and often leading to better performance in practice when dealing with noisy data.

We will further justify our choice of the Log-Cosh loss function in the Ablation Study section.

2.5 Training Details

For training our CNN model, we split the dataset of 49 3D meshes into training, validation, and test sets. 6 meshes were reserved as permanent test samples for evaluating the model's performance and generating visualizations. The remaining 43 meshes were split into 80% (35 meshes) for training and 20% (8 meshes) for validation.

To reduce training time, we separated the patch formulation step from the training process. We extracted patches from the 3D meshes beforehand and saved them in numpy file format. This allowed us to avoid repeating the patch extraction step in every training iteration, thereby speeding up the training pipeline.

Hyperparameters: We optimized several key hyperparameters using Keras Tuner, an open-source library for hyperparameter optimization. The following parameters were fine-tuned:

- **Learning rate:** The Adam optimizer was initialized with a learning rate of $1e^{-4}$. Additionally, we employed a learning rate reduction callback from TensorFlow, which dynamically lowered the learning rate if the training progress stagnated.
- **Layer parameters:** The number of filters in each convolutional layer and the units in the dense layers were determined using Keras Tuner.
- **Batch size:** The optimal batch size was also selected based on experiments conducted through Keras Tuner, allowing us to find the most suitable balance between memory consumption and training speed.
- **Dropout rate:** We experimented with different dropout rates to regularize the model and prevent overfitting. The selected dropout rates were also determined using Keras Tuner.

- **Epochs:** The model was trained for 150 epochs. However, to avoid overfitting, early stopping was employed, which halted training when validation loss showed no further improvement.

Hardware: The training process was executed on a Kaggle GPU instance, leveraging two NVIDIA Tesla T4 GPUs, each with 15GB of VRAM, along with 30GB of system RAM and a dual-core CPU. This hardware setup was crucial in accelerating the training process, particularly for handling the large 3D mesh patches.

The training took approximately 2 hours. Early stopping and learning rate reduction techniques helped minimize overfitting and ensured an efficient convergence process.

Chapter 3

Experiments and Results

This section takls about the data preparation in section 3.1, then in Section 3.2, the metrics utilised for measuring the performance of the model are detailed. Next, the results obtained with the proposed model are shown in Section 3.3. Then, a comparison with previous state-of-the-art works is presented in Section 3.4. Finally, Section 3.5 offers an ablation study that validates the decisions made for the implementation of the model.

3.1 Data Preparation

3.1.1 Data Sources

For training our saliency prediction model, we used the dataset provided by XI WANG et al. [19]. This dataset is the first large collection of human fixations on physical 3D objects under varying viewing conditions. It allows for mapping pupil positions to 3D coordinates and models fixated positions as a probability distribution, highlighting how salient features depend on viewing direction.

The dataset consists of 49 3D meshes, each stored in `.ply` format. These meshes represent a diverse range of objects, with each mesh containing approximately 50,000 vertices. For every 3D mesh, the corresponding saliency maps—representing vertex-level saliency—are provided in `.txt` files. These saliency values have already been normalized to a range between 0 and 1, allowing for direct use in training the model. The saliency maps serve as the ground truth for predicting the visually significant regions of the 3D meshes.

Figure 3.1: A section the 3D meshes used as a dataset

3.1.2 Data Preprocessing

Once the dataset was acquired, several preprocessing steps were necessary to make the data more manageable for our deep learning model. The primary challenge we faced was the high complexity of the 3D meshes, with each mesh consisting of approximately 50,000 vertices. This level of detail is computationally expensive and impractical for our model, necessitating a reduction in complexity through mesh simplification.

We determined that reducing each mesh to around 20,000 triangles would be a good balance between maintaining important geometric details and reducing computational demands. To achieve this, we developed a script using the C++ library libigl[1], which offers robust tools for mesh processing. Specifically, we employed a mesh decimation technique, a process that simplifies a mesh by systematically reducing its number of triangles while preserving its overall shape and structure. The mesh decimation method we used is based on Hugues Hoppe's work [4]. The technique involves constructing a sequence of increasingly simplified meshes from the original high-resolution mesh M_0 down to a much lower-resolution mesh M_n by collapsing edges iteratively.

$$M_0 \underset{\text{edge collapse}}{\longrightarrow} M_1 \underset{\text{edge collapse}}{\longrightarrow} \dots \underset{\text{edge collapse}}{\longrightarrow} M_{n-1} \underset{\text{edge collapse}}{\longrightarrow} M_n$$

[1] `https://libigl.github.io/`

During this process, each edge collapse reduces the number of triangles in the mesh, gradually simplifying it until the desired resolution is achieved. This method ensures that the most critical features of the mesh are preserved, even as the overall complexity is reduced.

Another important aspect of preprocessing involved transferring the saliency information from the original high-resolution mesh to the simplified mesh. To do this, we computed the point-to-mesh squared distance, which allowed us to accurately map the saliency values from the original vertices to the new, decimated vertices. This approach ensures that the saliency map remains consistent, even after the mesh has been simplified.

The following comparison image illustrates the difference between the original and the decimated meshes, highlighting how the simplification preserves key details while significantly reducing the number of triangles.

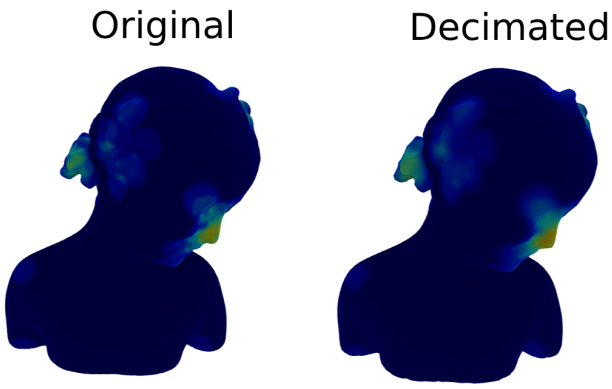

Figure 3.2: Comparison between original mesh and decimated mesh. The left image is the original mesh and the right image is the decimated mesh

3.2 Metrics

In order to evaluate the performance of the developed model and provide a meaningful comparison with other state-of-the-art works, three different metrics commonly used in regression problems were chosen. To compute these metrics, both the ground truth and the predicted saliency maps have been normalized between 0 and 1. The metrics used to measure the difference between the predicted saliency map P and its ground truth G are the following:

Mean Squared Error (MSE): This metric measures the average squared difference between the predicted values and the actual values. In the context of saliency

prediction, it quantifies the error between the ground truth saliency maps and those predicted by the model. It is computed as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.1}$$

where y_i is the ground truth value for face of the mesh i, $\hat{y}_i$ is the predicted value, and N is the total number of faces in the mesh.

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual values. Unlike MSE, MAE gives equal weight to all individual errors, making it less sensitive to outliers. The formula is shown below:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.2}$$

Here, y_i represents the ground truth value, $\hat{y}_i$ is the predicted value, and N is the number of faces.

Root Mean Squared Error (RMSE): RMSE is the square root of the MSE. It provides a measure of the average magnitude of the error in the same units as the original values, offering an interpretable metric for the magnitude of the prediction errors. RMSE is computed as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{3.3}$$

Like MSE and MAE, y_i denotes the actual value, $\hat{y}_i$ the predicted value, and N the number of faces.

3.3 Results

After training the CNN model, we evaluated its performance on a test set consisting of six 3D meshes: *skull*, *ant*, *vase*, *Isis*, *teddy*, and *teapot*. For each mesh,

we generated the predicted saliency maps and compared them to the ground truth saliency maps.

Figure 3.3 presents a visual comparison between the predicted and ground truth saliency maps, where warmer colors represent regions likely to attract more attention (indicating higher saliency), and cooler colors denote less salient areas.

For instance, in the case of the *Skull* model, the prediction aligns well with the ground truth, capturing the most salient features such as the eye sockets and nose. Similarly, for the *Vase*, the model accurately detects the handles as the most attention-grabbing elements. However, for the *Ant* model, while the model recognizes the horns as salient, it is less confident in this prediction, as indicated by the greenish color, which suggests uncertainty.

In the case of the *Teddy*, the model correctly identifies the ears as the most appealing feature but struggles to predict the saliency of other parts like the legs. A similar pattern can be observed in the *Isis* and *Teapot* objects. Although the model captures the most significant regions, it overlooks smaller details like the handle of the teapot.

Our model demonstrated consistent performance across all test meshes, effectively capturing the geometric characteristics that draw human attention. This alignment with the ground truth saliency maps confirms that our approach can generalize well to unseen 3D data, but there are instances where it could improve in capturing finer details.

We will discuss the quantitative metrics and the numerical results in more detail in the following Ablation Study section, which includes an in-depth comparison with previous models.

3.4 Comparaison with state-of-the-art

The approach proposed in this project has been compared with other state-of-the-art works that also predicted saliency on 3D meshes. On the one hand, the selected works that predict saliency using statistical methods have been Tao et al. [17], Song et al. [15], and Tasse et al. [18]. On the other hand, the selected work that predict saliency using deep learning have been Song et al. [16]. All these works used the ground truth dataset provided by Chen et al. [1] to compare their results. The ground truth provided by Chen et al. has been created doing an experiment in

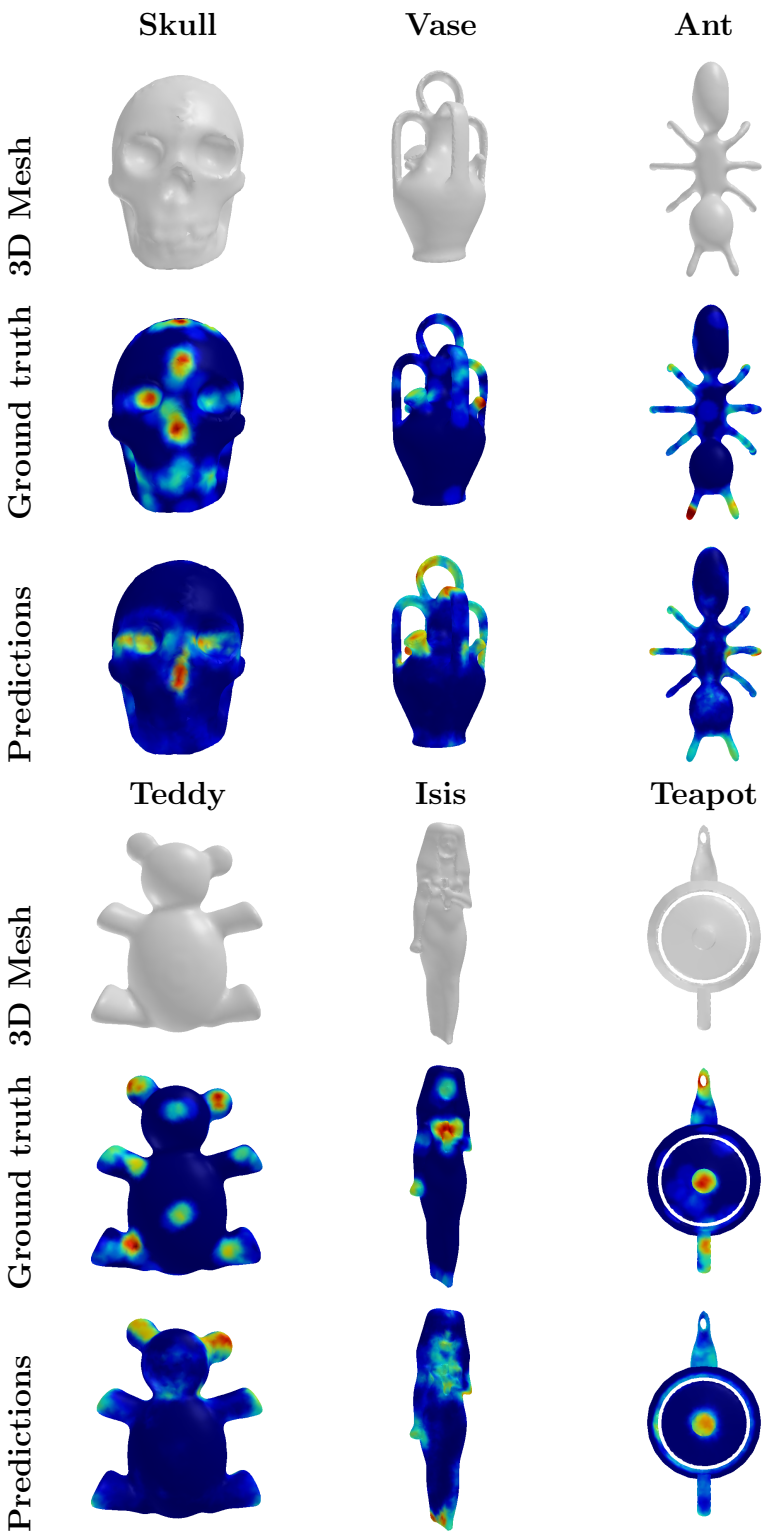

Figure 3.3: Saliency prediction results. The first three rows show the original mesh, the saliency of the ground truth, and the saliency predicted by our model for the Skull, Vase and the Ant object, respectively. The last three rows show the original mesh, the saliency of the ground truth, and the saliency predicted by our model for the Teddyn Isis and the Teapot object, respectively. Saliency is represented as a color map blended with the 3D meshes, where warmer colours correspond to more salient areas.

which the authors asked people to manually select points of the meshes that are most likely to be selected by the other participants as interesting. After the experiment, they performed a post-processing to create the saliency maps (more details in their work [1]). So, in order to compare our results with theirs, saliency predictions on the meshes provided by Chen et al. have been made.

Figure 3.4: Comparison of saliency prediction of our approach with statistical methods. Rows from top to bottom: our approach, Tasse et al. [18], Tao et al. [17], Song et al. [15], ground truth from Chen et al. [1]. Source of the image: Tasse et al. [18]. Warmer colours correspond to more salient areas.

First, the results are compared qualitatively with the works that predict saliency using statistical methods [17, 15, 18] along with the ground truth. These results are shown in Fig. 3.4. It can be seen that our approach tends to generalize reasonably well. The clearest examples are the chair at the second column, the bacterium, the vase at the fourth column, the pig, the bear and the hand. In these cases our approach is able to detect the parts marked as salient in the ground truth, while the other approaches tend to fail due to the assumption that the parts with steep curvatures and corners are salient. The cases of the human standing and the vase are the least close to the ground truth. In the human, the rest of the works tend to predict the man's head and limbs as salient since they are the parts with strongest curvatures, what in this case matches the ground truth, while our approach gets to detect the man's hands and the

beginning of the legs, but fails to detect the head. The cases of the vase in the first column it did manage to detect the rim of the vase and the upper part of the handle but it failed at predicting the base of the vase unlike the statistical methods that predicted it correctly with the result of Tao et al. beeing the closest to the ground truth. Finally, the poorest prediction for all approaches has been the glasses. In this case, as the glasses are a difficult mesh to make predictions, none of the approaches matches the ground truth, but they all match in the rims.

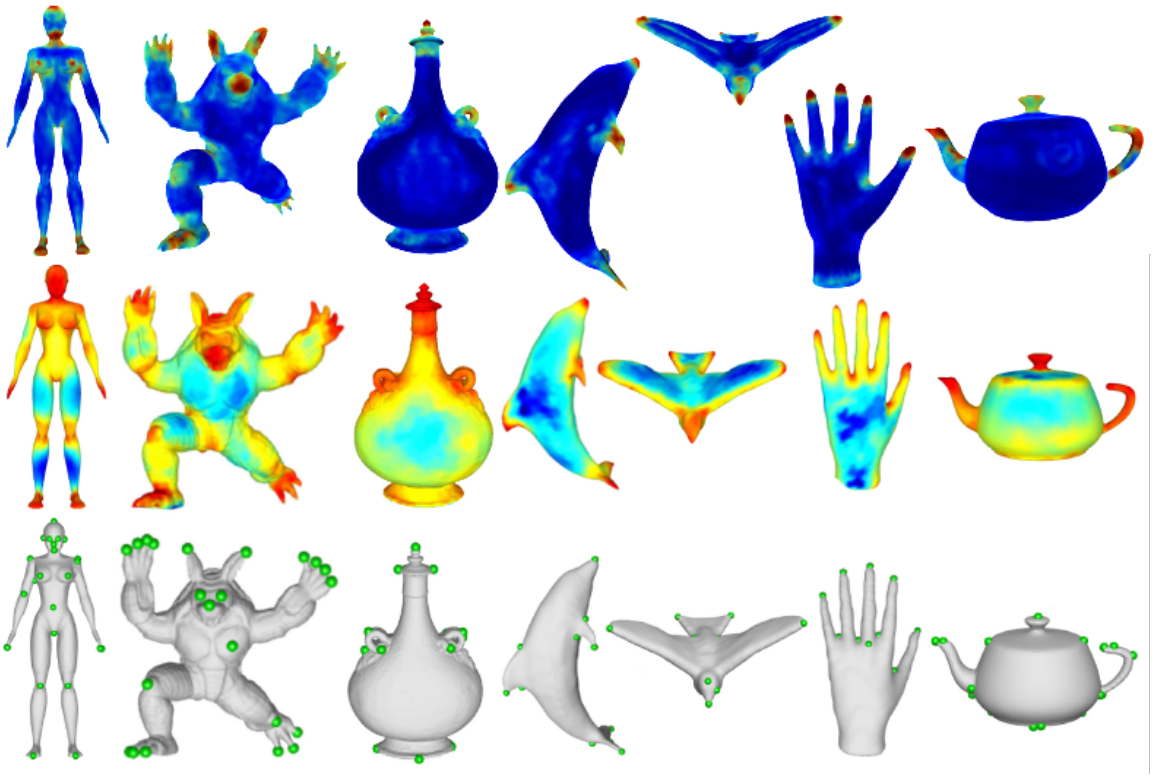

Figure 3.5: Comparison of saliency prediction with the approach of Song et al. [16]. Rows from top to bottom: our approach, Song et al. [16] and the ground truth from Chen et al. [1]. Warmer colours correspond to more salient areas. Source of the image: Song et al. [16]

Next, our approach is compared qualitatively with the works that utilised deep learning to make saliency predictions on meshes [16]. As mentioned before, this approach takes advantage of the features extracted within the hidden layers of existing networks designed for classification and assumes that if those features are different, that part of the mesh is salient. The expected results for this work are the same than in the previous ones, that is, areas with high-frequency details and strong curvatures will tend to be predicted as salient, while flat areas will not.

In Fig. 3.5 the comparison between our results and the results from Song et al. [16], along with the ground truth are compared. It is possible to see that our approach again tends to predict reasonably well the most interesting parts of each

mesh. In the case of the girl, both our approach and Song's are able to detect that the most salient areas are the head, the chest, the hands and the feet. However, comparing the two approaches with the ground truth, our approach fails to detect the face and the knees, while Song's approach detects all body as salient, which does not match the ground truth.For the armadillo, our approach is able to detect that the most salient areas are the face, the ears, the front knee and the front foot fingertips and the fingers of the hand, which match the ground truth correctly. On the other hand, Song's approach, thanks to the feature vectors assumption (different feature vectors correspond to salient parts), detects reasonably well the salient parts of the armadillo.The predictions on the vase are similar for both approaches, they match on the handles and on the top part, but our approach ignored the base the vase as non salient.The rest of the cases, the dolphin, the bird, the hand and the teapot are also very similar, both approaches detect reasonably well the salient parts. Nevertheless, in the hand and the teapot, because of the feature vectors assumption, Song's approach predicts as salient all fingers and the body of the teapot, which does not match the ground truth.

The authors of these works do not provide the files with their predicted saliency maps, and since we tried to reproduce the code provided by the different authors but it did not work, the comparison of our approach with these works has been made qualitatively.

3.5 Ablation Study

In order to justify the decisions made regarding the architecture of the model, some ablation studies have been carried out, which are presented below. These studies analyze the influence of the input data resolution (i.e.the Patch Size), the loss function used and some variations in the architecture of the model.

3.5.1 input resolution

This first ablation study analyzes the influence of the input resolution, specifically the Patch Size, on the model's performance in predicting accurate saliency maps. To evaluate this, the proposed model, detailed in Section 4, was trained with the default CNN parameters using three different resolutions: 16x16, 32x32, and 64x64.

From the table 3.1, it is evident that the error metrics (MSE, MAE, and RMSE)

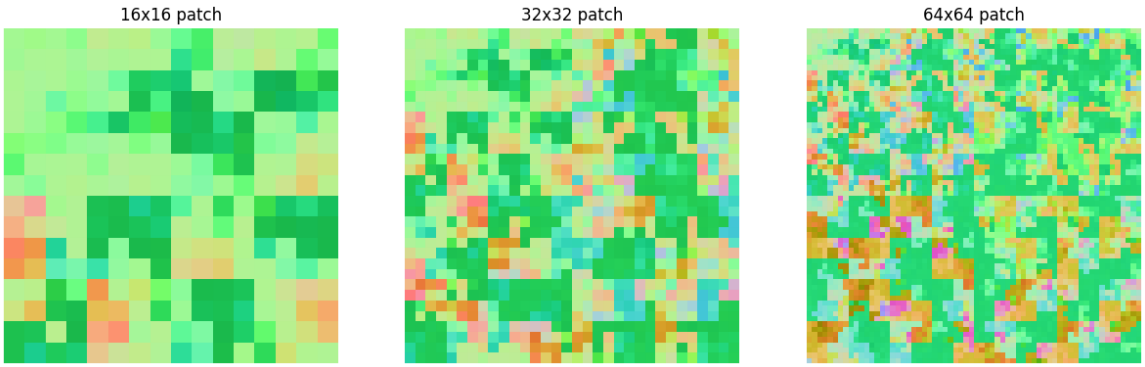

Figure 3.6: Side by side plot of the different patch sizes used.

vary only slightly across the different patch sizes, suggesting that the model maintains a consistent performance regardless of input resolution. Specifically, the 64x64 patch size yields the lowest Mean Squared Error (MSE) of 1.54, indicating that it slightly outperforms the other patch sizes in minimizing the error between predicted and ground truth saliency maps. However, the training time is significantly longer due to the increased computational cost.

Patch Size	**MSE** ($\times 10^{-2}$)	**MAE** ($\times 10^{-2}$)	**RMSE** ($\times 10^{-2}$)	**Time to Run**
16×16	1.677	7.410	12.952	105s
32×32	1.906	7.718	13.808	352s
64×64	1.540	7.049	12.410	1334s

Table 3.1: Numerical results for different input patch sizes. The time to run is for a single mesh

Furthermore, in Fig. 3.7 and 3.8, it can be observed that although the computed errors for the different resolutions differ, the model remains consistent in its predictions. It consistently highlights the same salient regions of the 3D mesh, regardless of the patch size. However, the 64x64 patch size is the most accurate among the three, likely due to the larger context it provides. Since the 64x64 patch size covers 4096 neighboring points, it allows the CNN model to gather more information, leading to better saliency predictions.

3.5.2 Alternative Loss Functions

In this ablation study, we evaluate the impact of different loss functions on the performance of the model. Since the patch size of 64x64 provided the best results in the previous experiment, we kept this patch size and the same CNN architecture, but varied the loss functions. Four loss functions were tested, defined as follows:

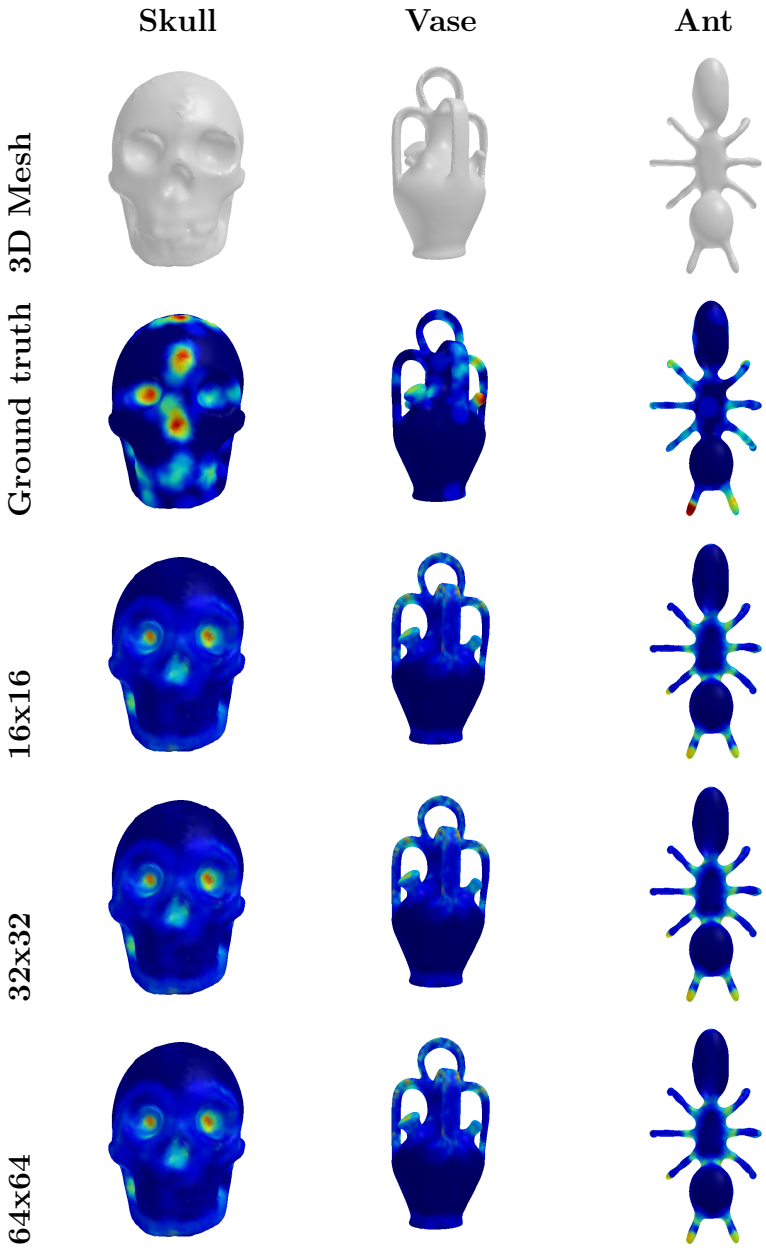

Figure 3.7: Comparison of saliency predictions for different patch sizes.

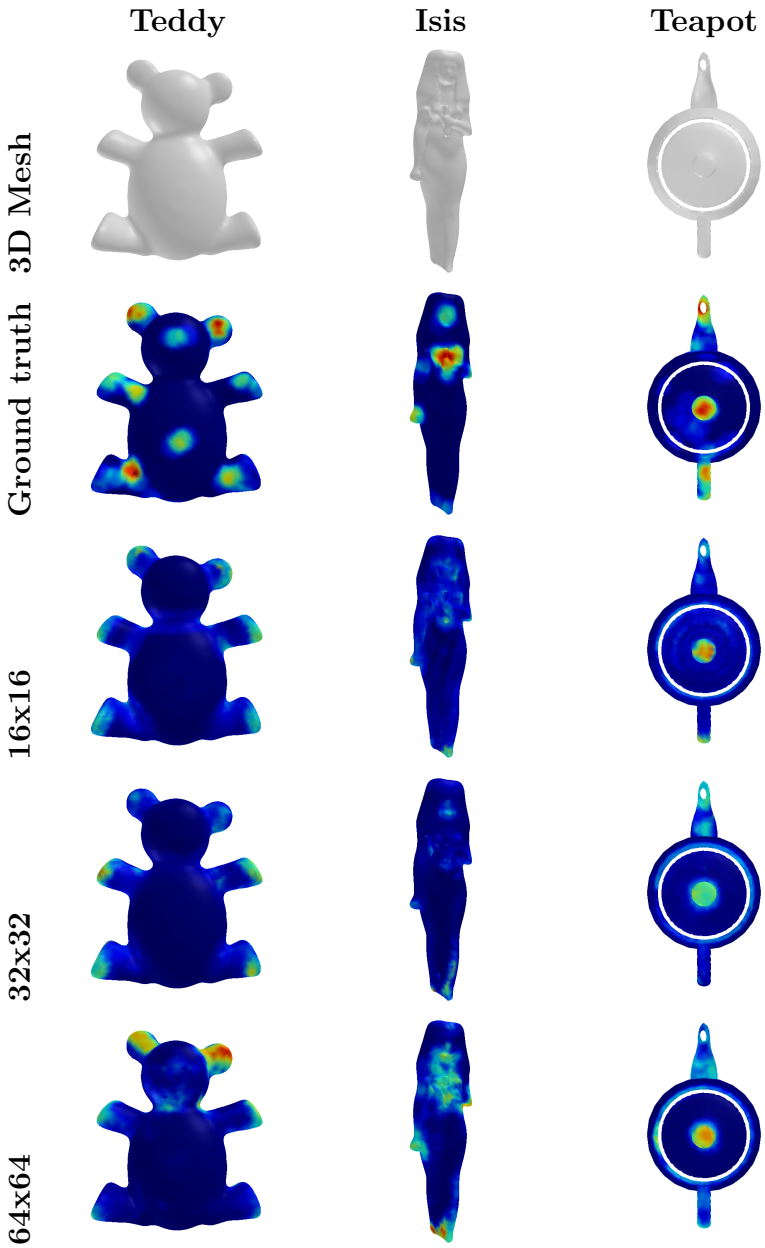

Figure 3.8: Comparison of saliency predictions for different patch sizes.

Mean Squared Error (MSE): This loss function computes the average squared difference between the predicted values and the actual ground truth values. It is defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Huber Loss: This loss function is less sensitive to outliers in the data than the MSE. It is defined as:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

where $a = y_i - \hat{y}_i$ is the residual error and δ is a threshold parameter.

Mean Logarithmic Error (MLE): This loss function computes the logarithmic difference between the true and predicted values, providing a penalty on underestimation or overestimation:

$$\text{MLE} = \frac{1}{N}\sum_{i=1}^{N}(\log(1 + y_i) - \log(1 + \hat{y}_i))^2$$

Logcosh Loss: This is the logarithm of the hyperbolic cosine of the prediction error, which is smoother than MSE and less affected by outliers:

$$\text{Logcosh} = \frac{1}{N}\sum_{i=1}^{N}\log(\cosh(\hat{y}_i - y_i))$$

The following table presents the results for the different loss functions with respect to the error metrics:

Loss Function	**MSE** ($\times 10^{-2}$)	**MAE** ($\times 10^{-2}$)	**RMSE** ($\times 10^{-2}$)
MSE	1.56	7.07	12.50
Huber	1.72	7.35	13.11
Mean Logarithmic Error	1.74	7.43	13.19
Logcosh	1.54	7.05	12.41

Table 3.2: Comparison of different loss functions and their respective error metrics.

The results in Table 3.2 show that the **Logcosh** and **MSE** loss functions yield the lowest errors overall, with Logcosh slightly outperforming MSE in terms of both MSE and RMSE. The **Huber** and **Mean Logarithmic Error** loss functions, while still producing reasonable results, show slightly higher error metrics across all categories.

From these results, we can conclude that the **Logcosh** loss function provides the best balance between sensitivity to small errors and robustness to outliers, making it a strong choice for our model. This performance might be due to the fact that Logcosh behaves like MSE for smaller errors but is more tolerant of larger errors, which can help in situations where some parts of the 3D mesh have more significant variation.

Figures 3.9 and 3.10 illustrate the predicted saliency maps generated by the model using the different loss functions, alongside the smoothed ground truth. From these figures, we can deduce that while all loss functions provide visually similar saliency predictions, Logcosh and MSE exhibit slightly sharper and more defined salient regions, aligning more closely with the ground truth.

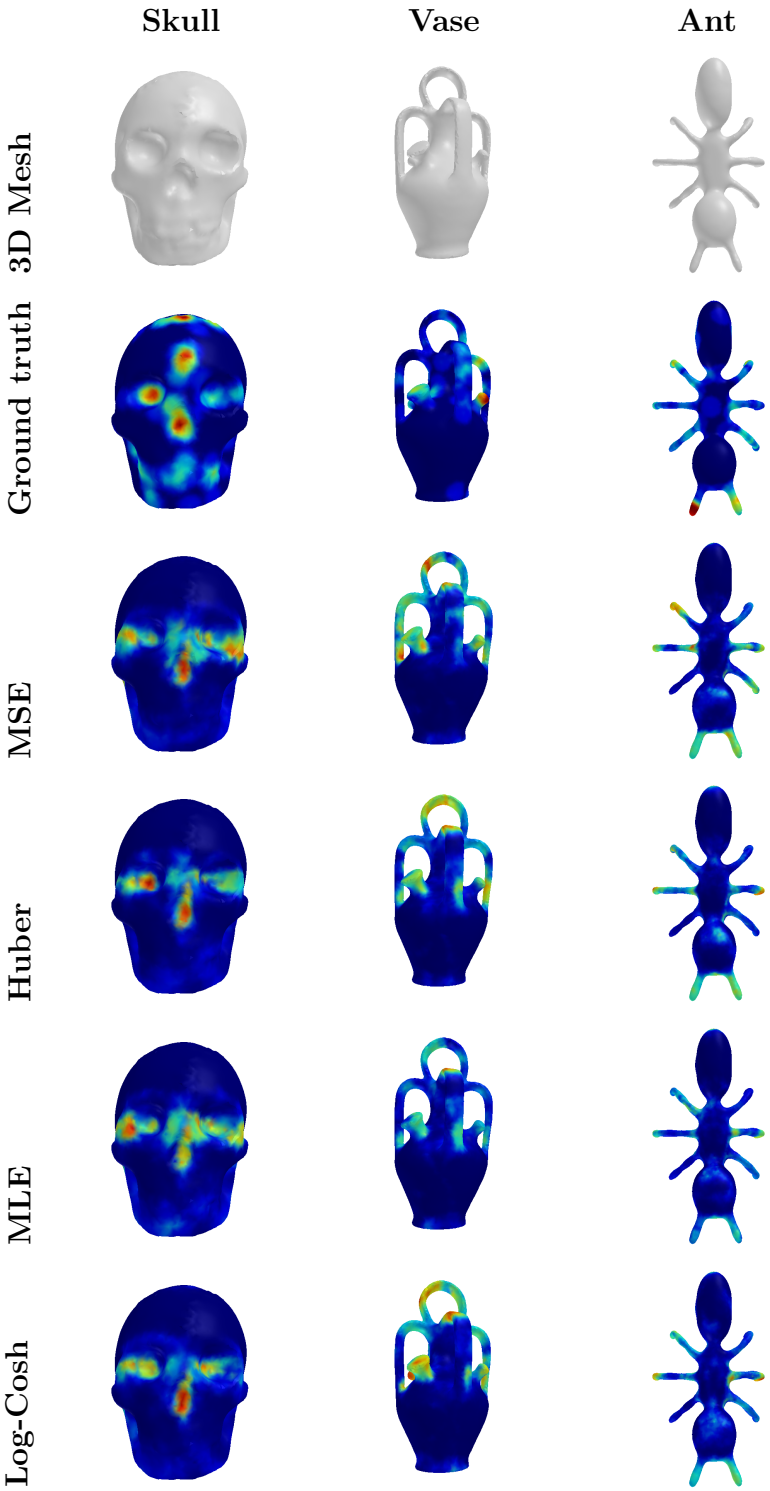

Figure 3.9: Comparison of saliency predictions for different loss functions.

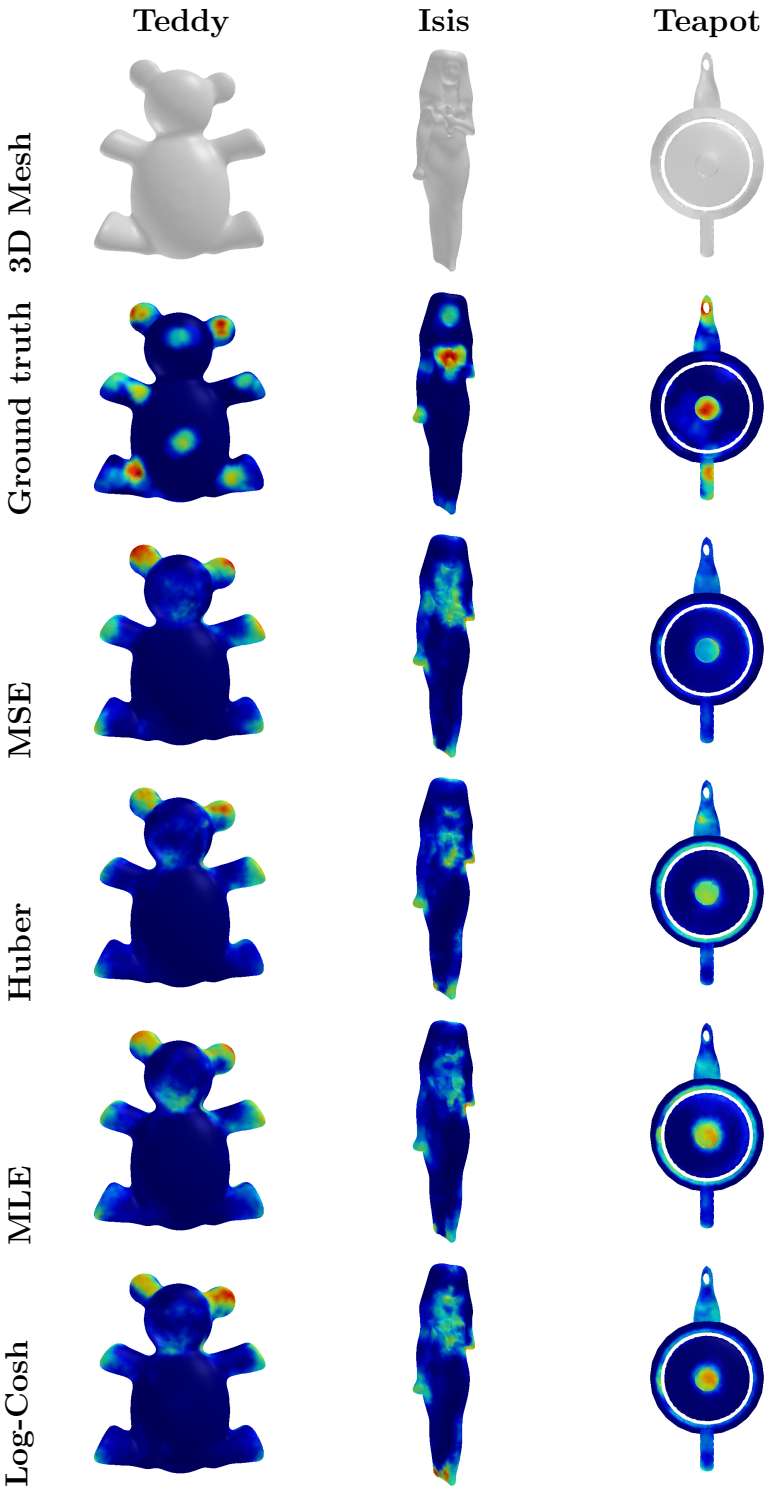

Figure 3.10: Comparison of saliency predictions for different loss functions.

Conclusion, Limits and Future Works

This Master's thesis has presented a deep learning-based approach to model human visual attention behavior when viewing 3D objects through saliency prediction. The proposed saliency prediction model has been adapted from Nousias et al work, taking into account real gaze data for training. Also, it has been demonstrated that our approach has been able to generalize the areas of interest of 3D objects reasonably well and obtains accurate saliency predictions in a wide variety of objects.

In addition, according to the comparison of our work with other works, it has been shown that our approach is capable of obtaining saliency prediction results that are close to the obtained ones by other approaches, which is a great achievement in the field since our work has been carried out with real gaze data as ground truth without making any assumption regarding the feature vectors extracted by the network in its hidden layers or the steep curvatures and corners of the meshes.

One limitation of this model is the computational cost, particularly during the patch formulation step. As the patch size increases, the time required for this step grows significantly. As shown in Table 3.1, the time to process a patch size of 16×16 is 105 seconds, whereas a patch size of 64×64 takes 1334 seconds, making it nearly 10 times slower. This increased time complexity can hinder the model's scalability for larger datasets or real-time applications.

Another limitation of this architecture is that the model captures only local features of the 3D mesh. The patch is constructed from the neighboring triangles of the face being predicted, which limits the model's ability to understand the global structure of the mesh. Incorporating global features alongside local ones could improve the model's performance by providing a more comprehensive view of the 3D object.

Bibliography

[1] Xiaobai Chen, Abulhair Saparov, Bill Pang, and Thomas Funkhouser. Schelling points on 3d surface meshes. *ACM Transactions on Graphics (TOG)*, 31(4):1–12, 2012.

[2] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. A deep multi-level network for saliency prediction. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3488–3493. IEEE, 2016.

[3] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4):1–12, 2019.

[4] Hugues Hoppe. Progressive meshes. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 111–120. 2023.

[5] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[6] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *2009 IEEE 12th international conference on computer vision*, pages 2106–2113. IEEE, 2009.

[7] Guillaume Lavoué, Frédéric Cordier, Hyewon Seo, and Mohamed-Chaker Larabi. Visual attention for rendered 3d shapes. In *Computer Graphics Forum*, volume 37, pages 191–203. Wiley Online Library, 2018.

[8] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, pages 659–666. 2005.

[9] George Leifman, Elizabeth Shtrom, and Ayellet Tal. Surface regions of interest for viewpoint selection. *IEEE transactions on pattern analysis and machine intelligence*, 38(12):2544–2556, 2016.

[10] Daniel Martin, Andres Fandos, Belen Masia, and Ana Serrano. Sal3d: a model for saliency prediction in 3d meshes. *The Visual Computer*, pages 1–11, 2024.

[11] Stavros Nousias, Gerasimos Arvanitis, Aris S Lalos, and Konstantinos Moustakas. Mesh saliency detection using convolutional neural networks. In *2020 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2020.

[12] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[13] Shwetha Salimath, Francesca Bugiotti, and Frédéric Magoules. A hybrid gnn approach for predicting node data for 3d meshes. In *European Conference on Advances in Databases and Information Systems*, pages 130–139. Springer, 2023.

[14] Bahareh Shakibajahromi, Edward Kim, and David E Breen. Rimeshgnn: A rotation-invariant graph neural network for mesh classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3150–3160, 2024.

[15] Ran Song, Yonghuai Liu, Ralph R Martin, and Paul L Rosin. Mesh saliency via spectral processing. *ACM Transactions On Graphics (TOG)*, 33(1):1–17, 2014.

[16] Ran Song, Yonghuai Liu, and Paul L Rosin. Mesh saliency via weakly supervised classification-for-saliency cnn. *IEEE transactions on visualization and computer graphics*, 27(1):151–164, 2019.

[17] Pingping Tao, Junjie Cao, Shuhua Li, Xiuping Liu, and Ligang Liu. Mesh saliency via ranking unsalient patches in a descriptor space. *Computers & Graphics*, 46:264–274, 2015.

[18] Flora Ponjou Tasse, Jiri Kosinka, and Neil Dodgson. Cluster-based point set saliency. In *Proceedings of the IEEE international conference on computer vision*, pages 163–171, 2015.

[19] Xi Wang, Sebastian Koch, Kenneth Holmqvist, and Marc Alexa. Tracking the gaze on objects in 3d: How do people really look at the bunny? *ACM Transactions on Graphics (TOG)*, 37(6):1–18, 2018.