Traditional and Neural Order-Independent Transparency

G. Tsopouridis[1] and C. Georgiou-Mousses[1] and I. Fudos[1] and D. Corrigan[2] and T. A. Franke[2]

[1]Department of Computer Science and Engineering, University of Ioannina, Ioannina, Greece
[2]Huawei Ireland Research Center, Dublin, Ireland

Abstract

Order independent transparency (OIT) is a technique in computer graphics that allows for accurate rendering of transparent objects without the need to sort them in a specific order based on their depth. Traditional transparency methods often suffer from artifacts and inaccuracies due to this sorting process, especially in complex scenes with many overlapping transparent surfaces. OIT is important because it provides a more visually correct representation of transparent materials, ensuring that colors mix accurately and that all elements are rendered consistently, regardless of their draw order. This enhances realism in applications such as video games, simulations, and visual effects in films. The tutorial will provide an overview of traditional (exact, approximate and hybrid) and deep learning approaches to OIT and examine their scope, performance and accuracy.

CCS Concepts

• ***Computing methodologies*** → *Neural networks; Rasterization; Visibility;*

1. Topic and Format

The tutorial explores "Traditional and Neural Order-Independent Transparency" (OIT). It presents exact, approximate, hybrid, and deep learning approaches to OIT, examining their performance, accuracy, and applications.

We propose a 180-minute half-day course. Following the conference, the course materials will be made accessible to a broader audience through a GitHub page. This tutorial was presented at Eurographics 2025, London.

Keywords: order-independent transparency, rendering, deep learning, visibility determination

2. Motivation and Definition of Scope

Order Independent Transparency (OIT) is an essential technique in computer graphics for rendering transparent objects in scenes while accurately maintaining the light transmittance process. OIT addresses the challenge of ordering transparent fragment samples correctly, even when objects intersect.

OIT is crucial in real-time applications such as computer games and VR applications. By leveraging recent deep learning techniques, efficient methods can be developed to achieve high accuracy on modern desktop and mobile platforms. To explore this, we propose an introductory course on OIT, covering both traditional exact and approximate methods as well as modern deep learning approaches.

This tutorial aims to provide a thorough understanding of OIT foundations and demonstrate the enhancement of transparency rendering through deep learning. A background in computer graphics, linear algebra, and neural networks is recommended, as these are aligned with the typical mathematical foundations of a computer science curriculum.

The course is designed to be accessible to a wide audience, including beginner PhD students and practitioners new to OIT. It will gradually advance to more complex topics, such as using deep learning to improve classic transparency methods, creating learned approximations of transparency, and integrating these into existing rendering pipelines. We will achieve this by exploring a blend of traditional and contemporary OIT techniques enhanced by deep learning.

3. Overview

Order Independent Transparency (OIT) is essential for accurately rendering transparent objects, ensuring the correct blending of overlapping transparent geometry without requiring explicit geometry sorting. Exact methods, such as depth-peeling [Eve01; BM08], A-buffer techniques[Car84; YHGT10; VF12], and K-buffer[BCLCS07; VF14], achieve high accuracy by storing and sorting per-pixel fragment data.

However, these techniques are often computationally intensive and memory-demanding, making them unsuitable for real-time applications, especially in complex scenes with high depth complexity [MCTB11].

To address these limitations, several approximate OIT methods have been proposed. Blended techniques, such as Weighted Average Transparency (WAVG)[BM08] and Weighted Blended OIT (WBOIT)[MB13], offer efficient alternatives by blending fragment properties through approximate transparency operators. While these methods are highly performant and memory-efficient, they often sacrifice visual quality, particularly in scenes with high opacity or depth complexity. Moment-Based Transparency (MBOIT)[MKKP18] improves blended methods by leveraging statistical moments to capture and reconstruct the transmittance of each fragment with greater accuracy, resulting in increased computational complexity. Hybrid approaches, such as Hybrid Transparency (HT)[MCTB13; TFV22] and Multi-Layer Alpha Blending [SV14], combine the precision of exact techniques (k-Buffer) for the nearest fragments with the efficiency of approximate methods for deeper layers, allowing a user-defined balance between performance and quality.

Recent advances in neural rendering have introduced novel approaches for addressing complex rendering problems, but their application to transparency rendering remains relatively limited. While neural techniques such as Neural Shadow Mapping[DNSD22] have demonstrated the potential of deep learning in tackling occlusion and visibility problems, transparency poses unique challenges due to the need for accurate blending of multiple overlapping fragments. However, several methods featuring lightweight neural networks have emerged, either improving or replacing traditional Order Independent Transparency (OIT) techniques. Deep Hybrid OIT[TFV22] enhances Variable k-Buffer methods by employing a neural network to predict the most visually important fragments in the scene. This approach achieves higher rendering quality while maintaining the same memory budget, effectively balancing performance and accuracy. Neural Moment Transparency [TVF24a] builds on Moment-Based Transparency by integrating a neural network to predict fragment transmittance more precisely, using the statistical moment representation as input features. Deep and Fast Approximate OIT (DFAOIT) [TVF24b] introduces a neural network that directly infers the OIT color of each pixel. By relying primarily on average per-pixel color and opacity as input features, DFAOIT offers a highly efficient and scalable alternative for real-time applications.

4. Tutorial Outline

Duration: Half-day tutorial (3 hours).

This tutorial is structured to provide a comprehensive understanding of Order-Independent Transparency (OIT), covering foundational concepts, technical methods, and cutting-edge advancements. Section 1 begins with an Introduction & Motivation, outlining the challenges and importance of transparency in real-time rendering (rasterization), followed by the metrics used to evaluate transparency methods. Section 2 then explores traditional transparency methods, categorizing techniques into Exact and Approximate approaches, with detailed discussions of numerous methods, including A-Buffer, k-Buffer, Weighted Blended Transparency, and Moment-Based methods, along with hybrid strategies that balance accuracy and performance by combining both approaches. Section 3 focuses on Deep Learning for OIT, introducing how neural networks are improving transparency rendering through approaches like Deep Hybrid OIT and Neural Moment Transparency, as well as exploring their potential to enhance or replace parts of the traditional transparency pipelines. Finally, Section 4 highlights practical applications of OIT across industries such as gaming, film, medical visualization, data analytics, and novel view synthesis. The tutorial combines theoretical insights and comparisons to guide participants through the evolving landscape of transparency rendering.

5. Introduction

5.1. Background

The tutorial begins by explaining the concept of **rasterization**, which is the process of converting vector graphics into a raster image (pixels). A key element in this process is the fragment, which represents the projected part of a triangle onto a pixel. Each fragment has an RGB color determined by a lighting model and a transparency value (alpha).

Multifragment rendering involves blending the colors of all fragments that contribute to a single pixel.

A significant challenge arises with transparency. Rasterization can generate multiple fragments per pixel, and these fragments may be out of order relative to their depth. Simply applying alpha blending without considering depth order leads to incorrect results. Correct transparency composition requires blending fragments in depth order using the overcompositing operator [PD84], a concept known as **Order Independent Transparency (OIT)**.

One approach to handling transparency is object-sorted transparency, where objects are sorted by distance from the camera and then blended. However, this method is incorrect when dealing with self-intersecting or very large objects.

Order Independent Transparency (OIT) aims to resolve transparency with a consistent method that does not depend on the order the fragments are processed. OIT methods can be classified as either **exact (buffer-based)** or **approximate**. Exact methods store some or all fragment information, while approximate methods are faster but less accurate, often modifying the compositing operator.

5.2. Quality Metrics and Benchmarking Tools

The tutorial then discusses how to evaluate the quality of transparency rendering methods. The **ground truth** is considered to be OIT with storing and sorting all fragments per pixel (A-Buffer). The goal is to achieve results as close as possible to this ground truth, avoiding temporal artifacts and approximating human-perceived differences.

Several metrics are presented for evaluating rendering quality:

MSE (Mean Squared Error): A measure of the average squared difference between the approximated and original images.

RMSE (Root Mean Squared Error): The square root of the MSE, providing a distance metric.

PSNR (Peak Signal-to-Noise Ratio): A measure of the ratio between the maximum possible power of a signal and the power of corrupting noise.

SSIM (Structural Similarity Index Measure): A method for predicting the perceived quality of an image by comparing its luminance, contrast, and structure to a ground truth image. For more information on SSIM see the paper by Wang et al. [WBSS04]. The source code is available[†].

FLIP: A measure that approximates the difference perceived by humans taking into consideration several features. For more information on FLIP refer to the paper by Andersson et al. [ANAOÅF20]. The source code is available[‡].

Several tools are available for GPU profiling and benchmarking, including Pix (D3D, Windows), Renderdoc, Frame Profiler (Open Harmony OS), Xcode (iOS), Arm Performance Studio (ARM GPUs), and Snapdragon Profiler (Qualcomm).

6. Traditional Transparency Methods

Transparency is a fundamental challenge in computer graphics, especially when multiple semi-transparent surfaces overlap. Unlike opaque objects, which can rely on depth buffering for correct visibility, transparent surfaces require careful consideration of order-dependent blending. Each layer must be composited to produce accurate results. However, achieving this efficiently is non-trivial, as the correct sorting of fragments becomes computationally expensive in complex scenes.

6.1. Exact Methods Overview

Exact transparency techniques aim to render overlapping transparent surfaces accurately by resolving visibility and blending in the correct order. In this section, techniques such as the A-Buffer, depth peeling, k-Buffer, and their variants will be discussed.

6.1.1. A-Buffer and Variants

The A-Buffer [Car84] uses linked lists to store all fragments within a pixel. By storing all fragments, a variety of per-pixel effects can be implemented, including transparency. Due to the order-dependent nature of transparency and color blending, the fragments inside the linked list must be sorted either in a front-to-back or a back-to-front manner. Finally, after sorting, the list inside the pixel is traversed and composited using the appropriate operator (**over** or **under**).

The A-Buffer method is high demanding in memory resources especially in scenes with complex, self-intersecting geometry, or with multiple geometries intersecting each other. Every pixel has to store information about every fragment that lies inside it, making the method restricting on applications where memory is either limited or restricted. Fragment sorting, being a computationally intensive operation, makes this method highly inefficient for real-time applications. Besides the aforementioned drawbacks, the resulting image is correct as geometry is correctly blended, making this perfect for off-line rendering.

Several adaptations of the A-Buffer that try to mitigate either the memory requirements or the data structure used to store the fragments have been proposed. The S-Buffer [VF12] addresses the inefficiency of A-Buffer memory by dynamically allocating only the exact per-pixel memory required, eliminating the unbounded memory requirement of linked lists. This is achieved by computing the exact fragment count per pixel in a separate pass, ensuring optimal memory usage. Furthermore, the F-Buffer [MP01] and R-Buffer [Wit01] replace the linked list with a FIFO structure while Z^3 [JC99] limits the fragment count per pixel to three.

[†] source code available at https://github.com/andyj1/ssim_index.
[‡] source code available at https://github.com/NVlabs/flip/tree/main

6.1.2. Depth Peeling

Depth peeling [Eve01] uses two z-buffers to peel the geometry layers one by one. The first z-Buffer acts as a depth barrier by discarding fragments that are in front of the N^{th} peeled layer. The second one acts as usual by finding the closest fragment seen. With this method the n^{th} depth layer is effectively peeled. By extracting the layers one by one and then blending them, the resulting image correctly implements the transparency effect.

The depth peeling method, although not memory demanding, is computationally intensive. To correctly peel n consecutive layers, n geometry passes are required. In addition to the A-buffer method, depth peeling also produces correct image results. Due to the peeling nature of the method, no ordering or sorting of fragments is required, making it truly order-independent.

6.1.3. Dual Depth Peeling

Bavoil and Myers introduced dual depth peeling [BM08] that refines the depth peeling method by peeling two layers in each geometry pass. The first (closest to camera) and last (furthest from camera) layers are peeled and blended each time, cutting the required number of layers by half.

6.1.4. k-Buffer

The k-Buffer method [BCLCS07] is a generalization of the z-buffer. Instead of storing a single value in each pixel, it uses the framebuffer memory as an RMW (Read, Modify, Write) pool of k entries. It is a semi-exact method as only the k stored fragments are correctly composited. For each incoming fragment inside a pixel, the k-Buffer method reads the currently stored elements from memory, modifies the entries using the incoming fragment, and finally writes the elements back to memory. In this way, k fragments are effectively captured in one geometry pass.

To correctly capture k elements, techniques such as dynamic fragment scheduling or atomic operations must be deployed to ensure RMW hazard avoidance. Due to the limited number of fragments stored, visual artifacts might arise in complex scenes. This can be mediated by increasing the number of k. However, by increasing k, the allocated memory can be underutilized by pixels that have a significantly smaller number of fragments than k. The memory requirements for this method are bound by the maximum number of fragments taken into account for each pixel. The computational complexity is moderate, making it a good compromise between speed and accuracy.

6.1.5. k^+-buffer

k^+-buffer [VF14] deploys a max-array, max-heap strategy to ensure the k foremost fragments are captured. RMW hazards are aleviated by using a spin-lock mechanism and high memory demands are bound to the exact per-pixel number of fragments by dynamically allocating k using histogram analysis.

6.1.6. Variable k-Buffer

Variable k-Buffer [VVPM17] effectively utilizes k-Buffer memory by estimating a per-pixel k. Estimation is carried out by calculating an importance map based on three heuristics.

- **Depth Complexity**: Pixels with a larger number of fragments need a larger k.
- **Periphery**: Pixels near the center of the image are perceptually more important.
- **Fresnel**: Near the grazing viewing angle of a transparent surface, reflected light prevails, and thus transparency effects are dimmer.

6.1.7. Comparative Evaluation of Exact Methods

Method	Memory	Geometry Passes	Complexity
A-Buffer	High	1	Simple
Depth Peeling	Low	N	Simple
Dual Depth Peeling	Low	$\frac{N}{2}$	Simple
k-Buffer	Medium	1	Moderate
k^+-Buffer	Medium	1	Moderate
Variable k-Buffer	Medium	2	Moderate

Table 1: *Comparative evaluation of exact transparency methods.*

In table 1 the memory demands, number of passes, and complexity of the exact methods are compared. As mentioned, the A-buffer uses unbound memory, but a small number of passes are needed to implement the transparency effect. However, depth peeling and its variants use very little memory, but require a large number of passes to achieve the same effect. k-Buffer and its variants provide a good middle ground with respect to memory restriction, complexity, and number of passes.

6.2. Approximate Methods

Exact transparency methods, such as the ones discussed above, guarantee correct blending order but come with high computational costs, making them impractical for real-time applications. Approximate methods trade perfect image quality for performance while still maintaining visually plausible results. Most of the methods discussed in this section define a new order-independent blending operator, making them extremely efficient.

6.2.1. Weighted Sum

Weighted Sum [Mes07] formulates a commuting blending operator by rearranging and taking the commutative part of the classic non-commuting **over** operator. When the alpha values are relatively low, this operator provides good results as the order-dependent part of the **over** operator contribute little to the final result of the formula.

6.2.2. Weighted Average

Weighted Average [BM08] builds on the observation that if all fragments were identical, the result then becomes order-independent. Instead, multiple colors per pixel are replaced by a uniform color per pixel. To handle non-uniform alpha values, this method uses an alpha-weighted average of the colors for every pixel. Despite being an improvement over Weighted Sum, invisible or near-invisible surfaces override the color of surfaces with high coverage, producing inaccurate results.

6.2.3. Weighted Blended OIT

Weighted Blended OIT [MB13] improves on Weighted Average by introducing a weight function that takes into account alpha and depth. The weight function acts as an occlusion estimator by giving greater importance to fragments that are closer to the viewer and dimming fragments that are farther away from the viewer. WBOIT gives better results, but the weight function is found by trial and error and is highly scene dependent.

6.2.4. Moment Transparency

Moment-Based OIT [MKKP18] and Moment Transparency [Sha18] build upon WBOIT and replace the empirically selected weight function with a transmittance estimator using statistical moments. Although both methods were developed independently and provide identical solutions, Moment-Based OIT provides a generalized moment-based framework for transparency and provides 4, 6, 8 power and trigonometric moments to reconstruct fragment transmittance. Due to the exponential nature of its decay, the transmittance is converted to logarithmic scale (absorbance) to enable additive accumulation. Then, a moment-generating function encodes absorbance into a cumulative distribution function which is then reconstructed and sampled to produce the final transparency effect on a pixel. Despite enabling WBOIT's method to drop empirical weights and estimate them procedurally, it requires an additional geometry pass to construct and calculate the moments.

6.2.5. Adaptive Transparency

Adaptive Transparency [SML11] uses the principle of the k-Buffer by storing a limited number of fragments per pixel. For each incoming fragment, a visibility function is calculated and then the fragment is inserted into the pool of candidate fragments. When a fragment is inserted into the pool and does not fit the memory budget, the stored fragment with the smallest error to the integration of the visibility function is removed. In a separate pass, the chosen fragments are composited.

6.2.6. Wavelet Transparency

Wavelet Transparency [ASM21] deploys a visibility function approximation similar to MBOIT but with the use of Haar wavelets. Wavelets are families of basis functions that can approximate other functions. Given a number of coefficients (wavelet rank), one can reconstruct the visibility function by the linear combination of the wavelet coefficients and their respective basis function. This method produces high-quality results beacause of the nature of Haar wavelets being able to better approximate monotonically non-increasing functions. Even though Wavelet Transparency produces better results, it introduces greater algorithmic complexity and temporal artifacts due to the Haar wavelets lack of shift invariance.

6.2.7. Comparative Evaluation of Approximate Methods

In table 2 approximate methods are compared. Weighted Sum, Weighted Average, and WBOIT prioritize simplicity and low memory overhead, requiring a single geometry pass. However, because of their heuristic nature, they introduce visual artifacts in scenes with high depth complexity. In contrast, Moment Transparency and Adaptive Transparency improve image quality significantly by increasing computational complexity and memory demands to a moderate level. Wavelet Transparency further refines accuracy by requiring a higher algorithmic overhead to tackle intricate transparency cases.

Method	Memory	Geometry Passes	Complexity
Weighted Sum	Low	1	Simple
Weighted Average	Low	1	Simple
WBOIT	Low	1	Simple
Moment Transparency	Low	2	Moderate
Adaptive Transparency	Medium	2	Moderate
Wavelet Transparency	Medium	3	Moderate

Table 2: *Comparative evaluation of approximate transparency methods.*

6.3. Hybrid Methods

Hybrid methods try to incorporate the best of both exact and approximate methods by calculating the final pixel color as a combination of an accurate blending of k layers and an approximate blending of the rest. Hybrid Transparency and Multi Layer Alpha Blending are analyzed.

6.3.1. Hybrid Transparency

Hybrid Transparency [MCTB13] is based on the observation that when looking through multiple transparent layers, after a threshold, the visibility of each layer has little contribution to the final color of the pixel. For that reason, the pixel's color is split in half. The first half, called the core, is calculated by computing OIT on the first k layers. The second half, called the tail, is calculated by the weighted average of all remaining layers. In this way, an accurate method is combined with an approximate but very fast method giving real-time frame rates without sacrificing significant image quality.

To extract the k foremost (core) fragments, a truncated A-Buffer is used (k-TAB). As with k-Buffer and its variants, atomic operations are used to correctly keep the k-TAB updated. The k-TAB is updated by a single-pass bubble sort. The tail, which consists of the remaining layers, is calculated by computing the sums needed and then computing their weighted average.

6.3.2. Multi Layer Alpha Blending

Multi Layer Alpha Blending [SV14] works similarly to Hybrid Transparency but uses a k-entry blending array to compress the core and the tail. The blending array's entries consist of the per-pixel accumulated color, depth, and transmittance. Instead of storing k elements for the core and having one element for the tail, the blending array has k+1 entries, where the first k fragments are stored in the first k entries of the array, and the tail is stored in the last entry.

For every incoming fragment, an ordered insertion of the fragment is made into the array and then the last two rows are merged. By merging the last two rows of the array, the pair with the smallest transmittance is merged because it is a monotonically decreasing function. Due to this merging step, MLAB is not fully order-independent, and artifacts can arise when rendering complex transparent objects overlapping in image space.

6.3.3. Comparative Evaluation of Hybrid Methods

Method	Memory	Geometry Passes	Complexity
Hybrid Transparency	Medium	1	Moderate
MLAB	Medium	1	Moderate

Table 3: *Comparative evaluation of hybrid transparency methods.*

In table 3 the memory demands, number of passes, and complexity of the hybrid methods are compared. These methods have bound memory requirements dictated mainly by the screen's resolution and not by depth complexity. The two methods achieve similar performance and image quality. MLAB achieves the same results with half the memory, but can produce order-dependent artifacts because of its compression step. As discussed, these methods provide a cheap and moderately easy way to having good image quality and real-time performance.

7. Deep Learning for Order Independent Transparency

7.1. Introduction & Challenges

Order Independent Transparency (OIT) is essential for correctly rendering transparent objects, offering a plethora of methods that achieve that without requiring costly sorting operations. However, traditional OIT methods may face significant challenges, including high memory usage and performance bottlenecks due to sorting layers that usually appear in especially complex scenes. As real-time rendering demands continue to grow, deep learning has emerged as a promising solution to these challenges. Deep learning is already well-established in

computer graphics, playing a key role in tasks such as particle and fluid simulation, Bidirectional Reflectance Distribution Function (BRDF) modeling for complex materials, Neural Radiance Fields (NeRFs) for novel view synthesis, Generative Adversarial Networks (GANs) for mesh generation, and deep learning-driven animation synthesis for realistic motion and character animation.

By leveraging neural networks, we can approximate sorting and compositing operations, or enhance existing OIT approaches to further improve visual quality or performance. This section explores how deep learning techniques are beginning to appear in real-time transparency rendering, offering both speed and quality improvements.

7.2. Deep learning for real time-rendering

Deep learning has made significant strides in real-time rendering, offering powerful solutions to problems traditionally solved by computationally expensive heuristics. One such application is Neural Network Ambient Occlusion (NNAO) [HSK16], a machine learning-based approach that improves Screen Space Ambient Occlusion (SSAO). NNAO trains a neural network to predict ambient occlusion using depth and normal data from a scene. It utilizes a compact four-layer network that learns from a dataset of rendered scenes and is later converted into an optimized shader, enabling real-time performance with high accuracy. This approach reduces the artifacts and performance costs associated with traditional SSAO methods, making it well suited for modern rendering pipelines. Datta et al. [DNSD22] introduce Neural Shadow Mapping, a deep learning-based approach that enhances traditional shadow mapping by using a compact neural network to generate high-quality hard and soft shadows in real-time. The network takes rasterized screen-space g-buffers, including depth, normals, and a shadow mapping pass as inputs, and learns to approximate ray-traced shadows. By using a problem-specific neural network architecture, and a novel temporal stability loss, NSM achieves visually accurate, temporally stable shadows without requiring costly denoising or temporal anti-aliasing. Real-Time Neural Appearance Models [ZRWCNBEDKL23] introduce a neural rendering approach that brings film-quality materials to real-time applications. The model consists of an encoder and two decoders, with a neural (latent) texture in between. The encoder maps BRDF parameters into a latent space, converting traditional per-layer textures like albedo and normal maps into a compact multi-channel representation. This latent texture is then decoded using two networks: an evaluation network that predicts BRDF values for a given pair of directions and a sampling network that generates outgoing directions from random inputs. By efficiently integrating neural networks into rendering code and leveraging tensor operations, the approach enables fast inference across various shader stages, including ray tracing and fragment shaders, making it highly suitable for real-time graphics and game engines.

These advancements in deep learning for rendering highlight the potential for transforming Order Independent Transparency (OIT) as well. Neural networks can be applied to transparency rendering by predicting the transmittance of transparent layers or replacing the blending equations entirely, reducing the need for costly sorting, storing and compositing operations.

7.3. Enhancing the OIT Pipeline

Transparency rendering is challenging to render using traditional OIT algorithims due to complex issues such as sorting, and memory constraints. To alleviate some of these issues, traditional methods can enhanced with deep learning. To achieve this, the method must maintain fast performance, a small memory footprint. The neural networks must be compact as they were implemented using fragment shaders, to ensure real-time execution. Deep Hybrid Order-Independent Transparency (DHOIT) [TFV22] builds on the variable k-Buffer approach by incorporating a deep learning prediction mechanism to assess pixel importance. This importance prediction enables an adaptive k-value, where memory is allocated more effectively, prioritizing significant areas of the scene. A fast geometry pass extracts relevant information for the neural network, which then determines per-pixel importance. To predice each pixels importance, both image-based and per-pixel features are extracted. At the image level, memory allocation is determined by the ratio of allocated fragments to total scene fragments, along with depth range information. At the per-pixel level, properties such as nearest and farthest fragment depths, diffuse color values, and the number of pixel fragments contribute to the importance calculation. A synchronized k-Buffer is then used to store and sort core fragments, while remaining fragments are accumulated efficiently using a Weighted Average approximation, similar to Hybrid Transparency [MCTB13].

Neural Moment Transparency (NMT) [TVF24a] builds upon Moment-Based Transparency [MKKP18], which can introduce transmittance reconstruction errors depending on the number of moments used for transmittance compression. By storing absorbance using four power moments, NMT allows a neural network to infer per-fragment transmittance with greater accuracy. This method requires two geometry passes and a neural network, making slower compared to other neural OIT methods such as DFA [TVF24b]. The first geometry pass extracts scene inputs, followed by a second pass where the network computes the per-fragment transmittance. A full-screen pass then determines the final color, blending it with the background. The entire neural network is implemented using fragment shaders. The network uses both global (per-pixel) and local (fragment) information to assess the transmittance value of each fragment. Per-pixel inputs include the number of fragments per-pixel and absorbance values stored using four power moments, while per-fragment depth values are calculated relative to the minimum and maximum depths in the scene and used as a tool to estimate each fragments significance. By leveraging deep learning for transmittance approximation, this method significantly improves visual quality while retaining a real-time performance.

7.4. Replacing the OIT Pipeline

Rather than enhancing existing methods, deep learning also enables entirely new approaches for Order-Independent Transparency (OIT), much like its application in approximating other complex effects using neural networks [HSK16; DNSD22]. This section introduces Deep and Fast Approximate OIT (DFA) [TVF24b], a deep learning-based solution that replaces traditional compositing methods (blending functions) with a neural network.

DFA operates in two passes. The first is a geometry pass that extracts 10 per-pixel input features, including the exact OIT RGB color of the two front fragments, the average opacity and RGB excluding these, and the accumulated premultiplied alpha RGB of all fragments. These features aim to provide color information throughout the whole pixel space, allowing the neural network to identify a correct color blending solution. The second pass is a full-screen pass that utilizes a multilayer perceptron (MLP) with two hidden layers (32 and 16 neurons with ReLU activation) and an output layer with sigmoid activation, directly predicting transparent pixel colors. Trained in a data set of five million samples from eight transparent scenes, with ground-truth transparency results [Car84] as targets, DFA demonstrates robust performance in various scenarios with varying depth complexity, opacity, and models. By eliminating sorting and fragment storage, DFA delivers a performance advantage over traditional OIT methods while maintaining high-quality results, making it particularly suitable for real-time applications, game engines, and VR environments.

7.5. Future Research Directions

In the future, we can aim to further enhance the rendering pipeline by integrating screen-space deep learning to approximate other complex visual effects more efficiently. Deep learning can be leveraged to precompute and bake effects for specific scenes, such as transparency, reducing computational costs while maintaining high-quality rendering. Additionally, modern hardware architectures can be exploited to optimize neural rendering, making it more practical for real-time applications.

Another promising direction is utilizing multifragment rendering to provide detailed spatial information that can serve as valuable input features for neural networks. Currently, two-layer deep G-buffers [MMNL16] are used for global illumination, ambient occlusion, and indirect lighting. Enhancing these techniques with neural networks could improve the accuracy and efficiency of these effects, leading to more realistic and visually rich scenes. Finally, neural networks can be employed to approximate other screen-space effects, such as ambient occlusion, reflections, and shadows. By using depth buffers combined with spatial information as inputs, neural models could dynamically predict and refine these effects, reducing computational overhead while preserving visual fidelity.

8. Applications and Challenges

The tutorial highlights the extensive use of transparency in various applications, particularly focusing on gaming, medical visualization, cultural heritage, autonomous vehicles, CAD, and novel view synthesis.

Gaming: Transparency is crucial for creating realistic effects involving translucent materials (glass, water), particle effects (smoke, fog), transparent cloth, card-based hair models, special effects, and user interfaces. While game engines often default to object sorting and alpha blending for transparency, there's increasing adoption of OIT algorithms on PC and consoles. Mobile gaming, historically limited by resource constraints, is now seeing increased use of transparency due to rising consumer expectations for realism and microtransaction-based business models involving elaborate character clothing with transparent elements. However, power consumption and render times remain significant challenges for OIT on mobile devices. Game engines provide excellent platforms for prototyping OIT algorithms, offering customizable render pipelines, multi-platform support, open-source scenes, and debugging/profiling tools.

CGI Movies: In computer-generated imagery for movies, global illumination algorithms (path tracing, ray tracing) can be used for rendering since real-time performance is not a primary concern. Transparency rendering, particularly using A-Buffer, can help produce faster, impressive pseudo-photorealistic frames.

Visualizing Diagnostic Data: OIT is valuable for rendering MRI scans and specialized diagnostic methods like functional MRI, DTI, and tractography [OPvdWC23]. This aids medical professionals in understanding the relative positioning of nerves, blood vessels, tissues, and organs, helping to avoid misdiagnosis. OramaVR [Ora] utilizes multiple tissue transparency techniques for medical XR training, where OIT is critical for trainees to perceive the relative spatial positioning of tissues and organs.

Cultural Heritage: Data extracted from phasmatoscopy and CT scans can be visualized using transparency to reveal inner layer details. This allows scientists and the public to better understand how findings/exhibits have been crafted. XR technology is also used to visualize archaeological sites by overlaying their historical appearance onto the current setting with OIT [MADTF18].

Autonomous Vehicles: XR technology with transparency is used for user perception in cooperative autonomous vehicles (CAV), utilizing real-time information from side road Lidar units or Lidar units of other cars. Small latency is of critical importance in these applications.

Computer Aided Design: OIT is crucial in real-time rendering of CAD parts for inspecting parts and detecting flaws, where naive approximate methods are often insufficient.

Novel View Synthesis: In Gaussian splat rendering, fast transparency is essential due to the millions of ellipsoids involved. Transparency alpha values encode information for correct view-dependent rendering. Approximate methods like WBOIT may struggle with large depth complexity. Solutions include per-pixel or smaller tile sorting/rendering with hybrid transparency and the use of more efficient or neural approximate OIT (DFAOIT). Future directions involve enhancing the rendering pipeline using screen-space deep learning to approximate other effects, using deep learning to bake effects for specific scenes (e.g., transparency), and exploiting modern hardware architectures for neural rendering.

9. Conclusions

The tutorial concludes with Tables 4 and 5 that summarize the characteristics of various OIT methods, including memory needs, time/power dissipation, accuracy, suitability for different scenes, and support for various applications.

	memory need	*time / power dissipation*	*accuracy / temporal artifacts*	*works for large scenes*	*works for large depth complexity*	*mobile platforms*	*support for CAD*	*support for games*	*support for VR/XR*	*support for gaussian splats*	*support for volume rendering*
A-buffer	high	high	perfect / no	yes	no (out of memory)	no	no (slow)	desktop	no (slow)	no (slow, use tiles)	yes
k-buffer	medium	medium	medium / low depends on k	yes	depends on k	small k	yes small k	desktop	no (slow)	no (require large k)	yes
k+-buffer	medium	medium	medium / very low depends on k	yes	depends on k	no	yes small k	desktop	no (slow)	no (require large k)	yes
variable k-buffer	medium	high	high / no	yes	yes	no	no	desktop	no (slow)	maybe (depends on heuristic)	yes
depth peeling	low	high	perfect / no	yes	yes	no	no	no	no (slow)	no (slow)	no (slow)
weighted average	low	low	low / medium	yes	yes	yes	no	medium	yes	no (quality)	no (quality)
weighted sum	low	low	low / high	yes	yes	yes	no	no (quality)	low (quality)	no (quality)	no (quality)
WBOIT	low	low	medium / low function needs customization	yes	yes	yes	medium	yes (per scene function)	yes	medium (quality)	medium (quality)

Table 4: *Table summarizing the characteristics of OIT methods, part 1 of 2*

	memory need	*time / power dissipation*	*accuracy / temporal artifacts*	*works for large scenes*	*works for large depth complexity*	*mobile platforms*	*support for CAD*	*support for games*	*support for VR/XR*	*support for gaussian splats*	*support for volume rendering*
MBOIT	low	medium	medium / no	yes	yes	medium / performance	yes	desktop	medium (2 passes)	maybe/needs adaptation	yes
adaptive	medium	low	high/no	yes	yes	small values of k	yes	desktop	yes	medium (depend on k)	yes
multi layer alpha blending	medium	medium	high / no	yes	yes	small values of k	yes	desktop	medium (2 passes)	medium (depend on k)	yes
wavelets	low	medium	high / yes	yes	yes	medium	yes	desktop	medium (2 passes)	maybe/needs adaptation	yes
hybrid	medium	medium	high / no	yes	yes	small values of k	yes small k	desktop	no (slow)	medium (depend on k)	yes
deep hybrid	medium	high	high / no	yes	yes	no / small values of k	yes small k	desktop	no (slow)	maybe/needs adaptation	yes
DFAOIT	low	low	medium / no	yes	yes	yes	yes	yes	yes	maybe/needs adaptation	yes
NMT	low	medium	medium / no	yes	yes	medium	yes	desktop	medium (2 passes)	maybe/needs adaptation	yes

Table 5: *Table summarizing the characteristics of OIT methods, part 2 of 2*

10. Presenters Information

10.1. Ioannis Fudos

- **Name:** Ioannis Fudos,
- **Affiliation:** Department of Computer Science & Engineering, University of Ioannina
- **email:** fudos@uoi.gr
- **URL:** google scholar link

Ioannis Fudos is a Professor at the Department of Computer Science and Engineering of the University of Ioannina. He received a diploma in computer engineering and informatics from University of Patras, Greece, in 1990 and an MSc and PhD in Computer Science both from Purdue University, USA in 1993 and 1995, respectively. His research interests include animation, rendering, morphing, CAD systems, reverse engineering, geometry compilers, solid modeling, and image retrieval. He has published in well-established journals and conferences (ICDE, Eurographics, Siggraph, I3D, SPM, SMI, GRAPP, CAD, JCAD, IEEE TVCG, ACM TOG, CAGC, CGI) and has served as program committee member and reviewer in various conferences and journals. Prof. Fudos has obtained funding from the EC and national sources (more than 19 projects and 2,5 MEuro). Ioannis Fudos has served as Department Chair of the Dept of Computer Science and Engineering of the University of Ioannina from September 2014 to August 2018. Prof. Fudos has long research experience in CAD, rendering, animation, mesh parameterization and surface representation of solids. Ioannis has also worked on graphics, visualization and other areas of computer science such as sensors, social networks and VLSI design. His work on graphics, image retrieval, data management and CAD has received more than 1900 citations with h-index 22 and g-index 43 (source: Harzing's Publish or Perish).

10.2. Grigorios Tsopouridis

- **Name:** Grigorios Tsopouridis
- **Affiliation:** Department of Computer Science & Engineering, University of Ioannina
- **email:** g.tsopouridis@uoi.gr
- **URL:** webpage link

Grigoris Tsopouridis is a Ph.D. candidate at the University of Ioannina, specializing in deep learning for rendering. His research focuses on integrating modern deep learning techniques with real-time rendering methods, with a specific emphasis in order-independent transparency. His work has been published in venues such as the Computer Graphics Forum, Visual Computer, and Eurographics.

10.3. Christos Georgiou-Mousses

- **Name:** Christos Georgiou-Mousses
- **Affiliation:** Department of Computer Science & Engineering, University of Ioannina
- **email:** ch.georgioumousses@uoi.gr
- **URL:** webpage link

Christos Georgiou-Mousses is a PhD Candidate at the University of Ioannina, specializing in deep learning for Gaussian splats and illumination. His success in various programming and game development competitions sparked a strong interest in the field of computer graphics, which continues to shape his professional pursuits. In addition to his academic research, Christos is passionate about the practical applications of computer graphics, machine learning, game development, and data mining.

10.4. Tobias Alexander Franke

- **Name:** Tobias Alexander Franke
- **Affiliation:** Huawei Ireland Research Center
- **email:** tobias.alexander.franke@huawei.com
- **URL:** webpage link

Tobias Alexander Franke is a Principal Game Engine Architect and the lead of the Game Rendering Lab at the Huawei Ireland Research Centre. Under the supervision of Prof. Dieter W. Fellner, Tobias received a PhD from TU-Darmstadt in Germany covering novel volumetric real-time relighting algorithms for augmented reality. In parallel, he worked at Fraunhofer IGD in the Visual Computing System Technologies department, advancing the state of web-based 3D rendering and streaming for large automobile models. In 2016, Tobias joined Unity in Copenhagen, Denmark, where he was embedded in the Lighting Team, shipping both the CPU and GPU Lightmapper, and eventually researching and developing the hyper-scalable Adaptive Probe Volume algorithm to render pre-baked volumetric global illumination for large 3D game scenes. Since 2021, he has been leading the Game Rendering Lab, focusing on the development of a broad range of SDKs to help developers ship mobile games with higher power-efficiency and scalable rendering technology.

10.5. David Corrigan

- **Name:** David Corrigan
- **Affiliation:** Huawei Ireland Research Center
- **email:** david.corrigan@huawei.com
- **URL:** webpage link

David Corrigan is a Principal Engineer in the Game Rendering Lab at the Huawei Ireland Research Centre. He studied for his doctoral degree in Image and Video Processing in Trinity College Dublin under the supervision of Prof. Anil Kokaram, graduating in 2008. The topic of his dissertation was motion estimation reliability and its application to the restoration of degraded archived film. He spent 9 months working as an Algorithm Engineer at The Foundry before spending 9 years as a post-doctoral researcher and assistant professor at the Department of Electrical and Electronic Engineering at Trinity College Dublin where his research focused on applied image/video processing covering domains such as digital cinema post-production, electron microscopy, education and marine biology. Since 2017 he has been at the Huawei IRC working in the area of video AI and computer graphics. His current interest in the Game Rendering Lab is the application of neural networks to optimise the performance of real-time rendering on mobile devices.

11. Relevant Courses (EG and SIGGRAPH)

Order-Independent Transparency is a key technique in computer graphics, as it is often highlighted in industry talks, such as Game Developers Conference (GDC). However, there is a notable lack of dedicated tutorials on OIT in academic settings, such as Eurographics. While SIGGRAPH and Eurographics offer tutorials that either partly cover OIT or aspects of it (sorting, rasterization), there is a clear gap when it comes to comprehensive, focused resources specifically addressing transparency rendering. Furthermore, deep learning has gained increasing attention in recent years and is now making its mark in real-time rendering and OIT, with several mentions in recent tutorials. This section outlines courses related to our proposed tutorial.

- Practical Machine Learning for Rendering: From Research to Deployment - Eurographics 2022

This course addresses the challenges of deploying neural networks in rendering, focusing on data acquisition, development, training, and deployment. It provides insights into practical use cases and neural models for integrating neural rendering into production workflows, such as super resolution, frame extrapolation, and denoising.

- Open Problems in Real-Time Rendering - SIGGRAPH 2017

"Deep Learning: The Future of Real-Time Rendering?" examines the possibility of deep learning integrating with real-time rendering, discussing several future applications such as anti-aliasing, and shading.

- Open Problems in Real-Time Rendering - SIGGRAPH 2016

Features a section "Peering Through a Glass, Darkly at the Future of Real-Time Transparency" on transparency and order-independent transparency addressing the challenges of transparency, state of the art and practical solutions.

- Open Problems in Real-Time Rendering - SIGGRAPH 2015

"Anti-Aliasing: Are We There Yet?" addresses the issues of anti-aliasing, offering OIT as a possible solution to aliasing. "The Rendering Pipeline - Challenges & Next Steps" addresses several problems in real-time rendering pipelines in modern game engines. An extensive focus is given on the problem of transparency, discussing OIT (sorting fragments), scalability, and using OIT for hair and foliage rendering.

- Efficient Sorting and Searching in Rendering Algorithms - Eurographics 2014

This tutorial addresses the problem of sorting and search in rendering algorithms. It provides theoretical and experimental evidence that sorting performance is crucial for many rendering techniques, such as visibility determination (transparency).

- Advances in Real-Time Rendering in Games - SIGGRAPH 2013

"Solving Old Graphics Problems with New Data Structures" addresses the problem of fragment shader synchronization, which is a crucial issue to OIT. It features several applications, including a very extensive discussion on the implementation of OIT methods such as programmable blending, and k-Buffers. "Destiny: From Mythic Science Fiction to Rendering in Real-Time" presents the key design concepts and features of the video game destiny from a rendering point of view. Extensive mentions of transparency, and particle rendering using OIT.

- Advances in Real-Time Rendering in Games - SIGGRAPH 2010

Section "Real-Time Order Independent Transparency and Indirect Illumination using Direct3D 11" describes a method of constructing linked lists on the GPU taking advantage of new features in Direct3D 11 for OIT, discussing the integration in the graphics pipeline taking advantage of standard modern hardware features.

References

[ANAOÅF20] ANDERSSON, PONTUS, NILSSON, JIM, AKENINE-MÖLLER, TOMAS, OSKARSSON, MAGNUS, ÅSTRÖM, KALLE, and FAIRCHILD, MARK D. "FLIP: A Difference Evaluator for Alternating Images". *Proc. ACM Comput. Graph. Interact. Tech.* 3.2 (Aug. 2020). DOI: 10.1145/3406183. URL: https://doi.org/10.1145/3406183 3.

[ASM21] AIZENSHTEIN, MAKSIM, SMAL, NIKLAS, and MCGUIRE, MORGAN. "Wavelet Transparency". (Dec. 2021). DOI: 10.48550/arXiv.2201.00094 5.

[BCLCS07] BAVOIL, LOUIS, CALLAHAN, STEVEN P., LEFOHN, AARON, COMBA, JOÃO L. D., and SILVA, CLÁUDIO T. "Multi-Fragment Effects on the GPU Using the k-Buffer". *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D '07. Seattle, Washington: ACM, 2007, 97–104. DOI: 10.1145/1230100.1230117 1, 4.

[BM08] BAVOIL, LOUIS and MYERS, KEVIN. "Order Independent Transparency with Dual Depth Peeling". 2008 1, 2, 4, 5.

[Car84] CARPENTER, LOREN. "The A -Buffer, an Antialiased Hidden Surface Method". *SIGGRAPH Comput. Graph.* 18.3 (Jan. 1984), 103–108. DOI: 10.1145/964965.808585 1, 3, 8.

[DNSD22] DATTA, SAYANTAN, NOWROUZEZAHRAI, DEREK, SCHIED, CHRISTOPH, and DONG, ZHAO. "Neural Shadow Mapping". *ACM SIGGRAPH 2022 Conference Proceedings*. Vancouver, BC, Canada: ACM, 2022. DOI: 10.1145/3528233.3530700 2, 7, 8.

[Eve01] EVERITT, CASS. *Interactive Order-Independent Transparency*. Tech. rep. Nvidia Corporation, 2001 1, 4.

[HSK16] HOLDEN, DANIEL, SAITO, JUN, and KOMURA, TAKU. "Neural network ambient occlusion". *SIGGRAPH ASIA 2016 Technical Briefs* (2016). URL: https://api.semanticscholar.org/CorpusID:14173036 7, 8.

[JC99] JOUPPI, NORMAN P. and CHANG, CHUN-FA. "Z3: an economical hardware technique for high-quality antialiasing and transparency". *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. HWWS '99. Los Angeles, California, USA: Association for Computing Machinery, 1999, 85–93. ISBN: 1581131704. DOI: 10.1145/311534.311582. URL: https://doi.org/10.1145/311534.311582 3.

[MADTF18] MOUTAFIDOU, ANASTASIA, ADAMOPOULOS, GEORGIOS, DROSOU, ANASTASIOS, TZOVARAS, DIMITRIOS, and FUDOS, IOANNIS. "Multiple Material Layer Visualization for Cultural Heritage Artifacts". *Eurographics Workshop on Graphics and Cultural Heritage*. Ed. by SABLATNIG, ROBERT and WIMMER, MICHAEL. The Eurographics Association, 2018. ISBN: 978-3-03868-057-4. DOI: 10.2312/gch.20181353 8.

[MB13] MCGUIRE, MORGAN and BAVOIL, LOUIS. "Weighted Blended Order-Independent Transparency". *Journal of Computer Graphics Techniques (JCGT)* 2.2 (Dec. 2013), 122–141. URL: http://jcgt.org/published/0002/02/09/ 2, 5.

[MCTB11] MAULE, MARILENA, COMBA, JOÃO L.D., TORCHELSEN, RAFAEL P., and BASTOS, RUI. "A Survey of Raster-based Transparency Techniques". *Computers & Graphics* 35.6 (2011), 1023–1034. DOI: 10.1016/j.cag.2011.07.006 2.

[MCTB13] MAULE, MARILENA, COMBA, JOÃO, TORCHELSEN, RAFAEL, and BASTOS, RUI. "Hybrid Transparency". *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. Orlando, Florida: ACM, 2013, 103–118. DOI: 10.1145/2448196.2448212 2, 6, 7.

[Mes07] MESHKIN, HOUMAN. "Sort-independent alpha blending". GDC Session. 2007. URL: https://gdcvault.com/play/535/Sort-Independent-Alpha 5.

[MKKP18] MÜNSTERMANN, CEDRICK, KRUMPEN, STEFAN, KLEIN, REINHARD, and PETERS, CHRISTOPH. "Moment-Based Order-Independent Transparency". *Proc. ACM Comput. Graph. Interact. Tech.* 1.1 (July 2018). DOI: 10.1145/3203206 2, 5, 7.

[MMNL16] MARA, MICHAEL, MCGUIRE, MORGAN, NOWROUZEZAHRAI, DEREK, and LUEBKE, DAVID P. "Deep g-buffers for stable global illumination approximation". *High Performance Graphics*. 2016. URL: https://api.semanticscholar.org/CorpusID:11390429 8.

[MP01] MARK, WILLIAM R. and PROUDFOOT, KEKOA. "The F-buffer: a rasterization-order FIFO buffer for multi-pass rendering". *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. HWWS '01. Los Angeles, California, USA: Association for Computing Machinery, 2001, 57–64. ISBN: 158113407X. DOI: 10.1145/383507.383527. URL: https://doi.org/10.1145/383507.383527 3.

[OPvdWC23] OSMAN, BESM, PEREIRA, MESTIEZ, van de WETERING, HUUB, and CHAMBERLAND, MAXIME. "Voxlines: Streamline Transparency Through Voxelization and View-Dependent Line Orders". *Computational Diffusion MRI*. Ed. by KARAMAN, MUGE, MITO, REMIKA, POWELL, ELIZABETH, RHEAULT, FRANCOIS, and WINZECK, STEFAN. Cham: Springer Nature Switzerland, 2023, 92–103. ISBN: 978-3-031-47292-3 8.

[Ora] ORAMAVR. *Accelerating the world's transition to medical XR training.* https://oramavr.com/ 8.

[PD84] PORTER, THOMAS and DUFF, TOM. "Compositing digital images". 18.3 (Jan. 1984), 253–259. ISSN: 0097-8930. DOI: 10.1145/964965.808606. URL: https://doi.org/10.1145/964965.808606 2.

[Sha18] SHARPE, BRIAN. "Moment transparency". *Proceedings of the Conference on High-Performance Graphics, HPG 2018, Vancouver, Canada, August 10-12, 2018*. Ed. by MOLNAR, STEVEN. ACM, 2018, 8:1–8:4. DOI: 10.1145/3231578.3231585 5.

[SML11] SALVI, MARCO, MONTGOMERY, JEFFERSON, and LEFOHN, AARON E. "Adaptive Transparency". *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics 2011, Vancouver, Canada, August 5-7, 2011*. Ed. by DACHSBACHER, CARSTEN, MARK, WILLIAM, and PANTALEONI, JACOPO. ACM, 2011, 119–126. DOI: 10.1145/2018323.2018342 5.

[SV14] SALVI, MARCO and VAIDYANATHAN, KARTHIK. "Multi-Layer Alpha Blending". *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '14. San Francisco, California: ACM, 2014, 151–158. DOI: 10.1145/2556700.2556705 2, 6.

[TFV22] TSOPOURIDIS, GRIGORIS, FUDOS, IOANNIS, and VASILAKIS, ANDREAS-ALEXANDROS. "Deep Hybrid Order-Independent Transparency". *The Visual Computer* 38.9 (2022), 3289–3300. DOI: 10.1007/s00371-022-02562-7 2, 7.

[TVF24a] TSOPOURIDIS, GRIGORIS, VASILAKIS, ANDREAS, and FUDOS, IOANNIS. "Neural Moment Transparency". *Eurographics*. 2024. URL: https://api.semanticscholar.org/CorpusID:269292577 2, 7.

[TVF24b] TSOPOURIDIS, GRIGORIS, VASILAKIS, ANDREAS A., and FUDOS, IOANNIS. "Deep and Fast Approximate Order Independent Transparency". *Computer Graphics Forum* 43.6 (2024), e15071. DOI: https://doi.org/10.1111/cgf.15071. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.15071. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.15071 2, 7, 8.

[VF12] VASILAKIS, ANDREAS and FUDOS, IOANNIS. "S-buffer: Sparsity-aware Multi-fragment Rendering". *Eurographics (Short Papers)*. Cagliari, Sardinia: The Eurographics Association, 2012, 101–104. DOI: 10.2312/conf/EG2012/short/101-104 1, 3.

[VF14] VASILAKIS, ANDREAS A. and FUDOS, IOANNIS. "k^+-buffer: Fragment Synchronized k-buffer". *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '14. San Francisco, California: ACM, 2014, 143–150. DOI: 10.1145/2556700.2556702 1, 4.

[VVPM17] VASILAKIS, ANDREAS-ALEXANDROS, VARDIS, KONSTANTINOS, PAPAIOANNOU, GEORGIOS, and MOUSTAKAS, KONSTANTINOS. "Variable k-buffer using Importance Maps". *38th Annual Conference of the European Association for Computer Graphics, Eurographics 2017 - Short Papers, Lyon, France, April 24-28, 2017*. Ed. by PEYTAVIE, ADRIEN and BOSCH, CARLES. Eurographics Association, 2017, 21–24. DOI: 10.2312/EGSH.20171005 4.

[WBSS04] WANG, ZHOU, BOVIK, A.C., SHEIKH, H.R., and SIMONCELLI, E.P. "Image quality assessment: from error visibility to structural similarity". *IEEE Transactions on Image Processing* 13.4 (2004), 600–612. DOI: 10.1109/TIP.2003.819861 3.

[Wit01] WITTENBRINK, CRAIG. "R-buffer: A pointerless A-buffer hardware architecture". *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (Aug. 2001), 73–80. DOI: 10.1145/383507.383529 3.

[YHGT10] YANG, JASON C., HENSLEY, JUSTIN, GRÜN, HOLGER, and THIBIEROZ, NICOLAS. "Real-Time Concurrent Linked List Construction on the GPU". *Computer Graphics Forum* 29.4 (2010), 1297–1304. DOI: 10.1111/j.1467-8659.2010.01725.x 1.

[ZRWCNBEDKL23] ZELTNER, TIZIAN, ROUSSELLE, FABRICE, WEIDLICH, ANDREA, CLARBERG, PETRIK, NOV'AK, JAN, BITTERLI, BENEDIKT, EVANS, ALEX, DAVIDOVIC, TOMÁS, KALLWEIT, SIMON, and LEFOHN, AARON E. "Real-time Neural Appearance Models". *ACM Transactions on Graphics* 43 (2023), 1–17. URL: https://api.semanticscholar.org/CorpusID:258479840 7.