



---

## Bridging neural rendering and classical computer graphics

---

**HELD Jan**

PhD thesis

|                         |                                          |                      |
|-------------------------|------------------------------------------|----------------------|
| VAN DROOGENBROECK Marc  | University of Liège (Belgium)            | <i>Supervisor</i>    |
| CIOPPA Anthony          | University of Liège (Belgium)            | <i>Co-supervisor</i> |
| LOUPPE Gilles           | University of Liège (Belgium)            | <i>President</i>     |
| TAGLIASACCHI Andrea     | Simon Fraser University (Canada)         | <i>Jury</i>          |
| NIESSNER Matthias       | Technical University of Munich (Germany) | <i>Jury</i>          |
| DEGBOMONT Jean-François | EVS (Belgium)                            | <i>Jury</i>          |

March 2026

*Three splats for fields where radiance flows,  
In convex forms their light bestows,  
Smooth shapes that bend the hidden rays,  
And render truth in subtle ways.*

*Three meshes wrought with solid might,  
Opaque and firm in learned sight,  
Differentiable, sharp and true,  
They shape the world in shaded hue.*

*Three triangles swift, in real-time cast,  
Through scenes that flicker, dense and vast,  
They trace the light with speed and skill,  
And bend perception to their will.*

*But one to bind them, edge and plane,  
**One form to rise, the rest to tame—**  
**One triangle to rule them all,**  
**One triangle to splat them all,**  
**One triangle to bind them all,**  
**And in the renderer enthrall.<sup>1</sup>***

---

<sup>1</sup>This text was partially generated with ChatGPT.



# Abstract

Recent advances in neural rendering have improved both the quality and efficiency of novel view synthesis from multi-view images. In particular, 3D Gaussian Splatting stands out by representing scenes with volumetric primitives and yielding high visual fidelity in real time. Despite these successes, point-based primitives struggle with sharp geometry, irregular surfaces, and integration with classical graphics pipelines.

This thesis addresses these limitations by investigating radiance field representations based on alternative geometric primitives. The selection of a particular primitive significantly affects reconstruction accuracy, rendering performance, and compatibility with standard graphics pipelines. We introduce three complementary approaches that progressively enhance visual quality and integration with graphics pipelines: *smooth convexes*, *triangles*, and, finally, culminate in *meshes*.

First, we introduce 3D Convex Splatting, the first work to move beyond Gaussian primitives by introducing smooth convex shapes. Compared to Gaussians, convex shapes offer greater flexibility and more accurately represent flat surfaces and sharp edges with fewer primitives. Coupled with an efficient CUDA rasterizer, it improves fidelity while reducing memory usage.

Second, we revisit one of the most fundamental primitives in computer graphics: the triangle. The main advantage of triangles is that they are fully compatible with current rendering pipelines. Triangle Splatting develops a differentiable rendering pipeline that enables direct optimization of triangles through end-to-end gradients. It represents a promising first step toward mesh-aware neural rendering, bridging decades of GPU-accelerated graphics with modern differentiable frameworks.

Finally, MeshSplatting extends this line of work by optimizing fully opaque, connected triangles, jointly refining geometry and appearance through differentiable rendering. This produces coherent surfaces that integrate seamlessly with real-time rendering engines, enabling applications such as augmented and virtual reality without any postprocessing.

Together, these contributions show that alternative primitives provide a powerful complement to Gaussian-based radiance fields. By progressing from convex shapes to triangles and finally to meshes, this thesis bridges neural rendering and classical computer graphics, enabling high-quality reconstruction, geometric consistency, efficient rendering, and full compatibility with real-time rendering engines.

# Acknowledgment

I would like to take this opportunity to express my sincere gratitude to my supervisor, Marc Van Droogenbroeck and Anthony Cioppa, for their support and guidance throughout my PhD. I am especially grateful for the freedom they granted me to pursue my own research directions, including multiple research stays, while always encouraging my ambitions and goals. I deeply appreciate the trust, independence, and academic liberty they offered me during this journey. Thank you!

I would like to extend my appreciation to my co-workers, Silvio Giancola, and Renaud Vandeghen. It was always a great pleasure working with you, and even more so spending time together outside of university, enjoying drinks at conferences or simply sharing moments of friendship. Thank you, Silvio, for hosting me at KAUST and making my six months in Saudi Arabia unforgettable. I am definitely coming back one day to get a smoothie and shish kebab in Thuwal.

I would like to thank Bernard Ghanem for giving me the opportunity to join KAUST as a research intern. I drew great inspiration from your vision and ambitions.

I would also like to sincerely thank Andrea Tagliasacchi for hosting me in his lab at Simon Fraser University. I am grateful for your guidance, motivation, and constant support. I learned a lot from you, and I am already looking forward to sharing another German coffee with you. It was a pleasure working with you, and I look forward to doing so again in the future! Thank you for everything!

I would also like to thank all the people with whom I collaborated and everyone I met during this journey, whether in labs, conferences, or research visits!

For my parents: Schließlich möchte ich meinen Eltern, Michel Held und Monika Dries, meinen herzlichen Dank für ihre ständige Unterstützung aussprechen, dafür, dass sie stets Interesse an meiner Arbeit gezeigt haben, dass sie für mich um die ganze Welt gereist sind, um mich zu besuchen, und vor allem dafür, dass sie mir jeden Tag vorgelebt haben, dass Einsatz, Ausdauer und Leidenschaft der Schlüssel sind, um erfolgreich zu sein. Ohne euch wäre das alles nicht möglich gewesen. Danke!

Thank you to my brother, Tun Held, my friend Bernard Schmidt, and all my other friends. Even though I often talked to them about work, AI, and conferences, they always remained interested and supportive of what I was doing. Most importantly, they helped me maintain the right balance between hard work and enjoying life.

Thank you!

# Table of Contents

|                                                                   |           |
|-------------------------------------------------------------------|-----------|
| <b>1. Research overview</b>                                       | <b>2</b>  |
| <b>2. Introduction</b>                                            | <b>3</b>  |
| 2.1. Introduction to novel view synthesis . . . . .               | 3         |
| 2.2. Research objective . . . . .                                 | 7         |
| 2.3. Contributions . . . . .                                      | 8         |
| 2.3.1. List of publications . . . . .                             | 8         |
| 2.3.2. Software contributions and community recognition . . . . . | 10        |
| <b>3. 3D Convex Splatting</b>                                     | <b>12</b> |
| 3.1. Context . . . . .                                            | 12        |
| 3.2. Scientific contributions . . . . .                           | 13        |
| 3.3. Publication and recognition . . . . .                        | 14        |
| 3.4. Abstract . . . . .                                           | 15        |
| 3.5. Introduction . . . . .                                       | 15        |
| 3.6. Related Work . . . . .                                       | 16        |
| 3.7. Methodology . . . . .                                        | 17        |
| 3.5.1. Preliminaries on 3D Smooth Convexes . . . . .              | 17        |
| 3.5.2. 3DCS: Splatting 3D Smooth Convexes . . . . .               | 18        |
| 3.5.3. Optimization . . . . .                                     | 18        |
| 3.8. Experiments . . . . .                                        | 19        |
| 3.6.1. Experiments on Synthetic Data . . . . .                    | 19        |
| 3.6.2. Experimental Setup . . . . .                               | 19        |
| 3.6.3. Real-world Novel View Synthesis . . . . .                  | 20        |
| 3.6.4. Ablation Study and Discussion . . . . .                    | 21        |
| 3.9. Conclusion . . . . .                                         | 22        |
| 3.10. References . . . . .                                        | 23        |
| 3.11. Supplementary Material . . . . .                            | 25        |
| <b>4. Triangle Splatting</b>                                      | <b>31</b> |
| 4.1. Context . . . . .                                            | 31        |
| 4.2. Scientific contributions . . . . .                           | 32        |
| 4.3. Publication and recognition . . . . .                        | 33        |
| 4.4. Abstract . . . . .                                           | 34        |
| 4.5. Introduction . . . . .                                       | 35        |
| 4.6. Related Work . . . . .                                       | 35        |

|                                                              |           |
|--------------------------------------------------------------|-----------|
| 4.7. Method . . . . .                                        | 36        |
| 4.5.1. Differentiable Rasterization . . . . .                | 36        |
| 4.5.2. Adaptive Pruning and Splitting . . . . .              | 37        |
| 4.5.3. Optimization . . . . .                                | 38        |
| 4.8. Experiments . . . . .                                   | 38        |
| 4.9. Ablations . . . . .                                     | 39        |
| 4.10. Triangle Splatting in a Traditional Renderer . . . . . | 40        |
| 4.11. Conclusions . . . . .                                  | 41        |
| 4.12. References . . . . .                                   | 42        |
| 4.13. Supplementary Material . . . . .                       | 44        |
| <b>5. MeshSplatting</b>                                      | <b>49</b> |
| 5.1. Context . . . . .                                       | 49        |
| 5.2. Contributions . . . . .                                 | 51        |
| 5.3. Abstract . . . . .                                      | 52        |
| 5.4. Introduction . . . . .                                  | 52        |
| 5.5. Related Work . . . . .                                  | 53        |
| 5.6. Methodology . . . . .                                   | 54        |
| 5.5.1. Background . . . . .                                  | 54        |
| 5.5.2. Vertex-sharing Triangle Representation . . . . .      | 54        |
| 5.5.3. From Soups to Meshes . . . . .                        | 54        |
| 5.5.4. Optimizing Meshes with Opaque Triangles . . . . .     | 55        |
| 5.5.5. Optimization Details . . . . .                        | 56        |
| 5.7. Experiments . . . . .                                   | 56        |
| 5.6.1. Mesh-based NVS . . . . .                              | 56        |
| 5.6.2. Training Speed & Memory . . . . .                     | 57        |
| 5.6.3. Surface Reconstruction . . . . .                      | 58        |
| 5.6.4. Applications . . . . .                                | 58        |
| 5.6.5. Analysis . . . . .                                    | 59        |
| 5.8. Conclusions . . . . .                                   | 59        |
| 5.9. References . . . . .                                    | 60        |
| 5.10. Supplementary Material . . . . .                       | 63        |
| <b>6. Conclusions and perspectives</b>                       | <b>69</b> |
| 6.1. This thesis . . . . .                                   | 69        |
| 6.2. So, what's next? . . . . .                              | 70        |
| 6.2.1. The future of splatting . . . . .                     | 70        |
| 6.2.2. Novel view synthesis . . . . .                        | 71        |
| 6.2.3. Adoption by the industry . . . . .                    | 71        |
| <b>A. Appendix</b>                                           | <b>75</b> |
| A.1. General context . . . . .                               | 75        |
| A.2. Conclusion and perspectives . . . . .                   | 77        |

|        |                                                                                                            |     |
|--------|------------------------------------------------------------------------------------------------------------|-----|
| A.3.   | Contributions of this appendix . . . . .                                                                   | 77  |
| A.3.1. | List of publications included in this appendix . . . . .                                                   | 77  |
| A.3.2. | List of software related to this appendix . . . . .                                                        | 78  |
| A.3.3. | Participation in the organization of SoccerNet challenges . . . . .                                        | 78  |
| A.4.   | VARs: Video Assistant Referee System for Automated Soccer Decision Making from Multiple Views . . . . .    | 79  |
| A.4.1. | Abstract . . . . .                                                                                         | 79  |
| A.4.2. | Introduction . . . . .                                                                                     | 79  |
| A.4.3. | Related Work . . . . .                                                                                     | 80  |
| A.4.4. | SoccerNet-MVFouls Dataset . . . . .                                                                        | 81  |
| A.4.5. | Methodology . . . . .                                                                                      | 83  |
| A.4.6. | Experiments . . . . .                                                                                      | 84  |
| A.4.7. | Conclusion . . . . .                                                                                       | 86  |
| A.4.8. | References . . . . .                                                                                       | 87  |
| A.4.9. | Supplementary Material: VARs . . . . .                                                                     | 91  |
| A.5.   | Towards AI-Powered Video Assistant Referee System (VARs) for Association Football . . . . .                | 96  |
| A.5.1. | Abstract . . . . .                                                                                         | 96  |
| A.5.2. | 1 Introduction . . . . .                                                                                   | 96  |
| A.5.3. | 2 Methodology . . . . .                                                                                    | 98  |
| A.5.4. | 3 Experiments . . . . .                                                                                    | 99  |
| A.5.5. | 4 Human study . . . . .                                                                                    | 102 |
| A.5.6. | 5 Conclusion . . . . .                                                                                     | 104 |
| A.5.7. | References . . . . .                                                                                       | 104 |
| A.6.   | X-VARs: Introducing Explainability in Football Refereeing with Multi-Modal Large Language Models . . . . . | 110 |
| A.6.1. | Abstract . . . . .                                                                                         | 110 |
| A.6.2. | Introduction . . . . .                                                                                     | 110 |
| A.6.3. | Related Work . . . . .                                                                                     | 111 |
| A.6.4. | SoccerNet-XFoul Dataset . . . . .                                                                          | 112 |
| A.6.5. | Methodology . . . . .                                                                                      | 113 |
| A.6.6. | Experiments . . . . .                                                                                      | 115 |
| A.6.7. | Conclusion . . . . .                                                                                       | 117 |
| A.6.8. | References . . . . .                                                                                       | 118 |

*The triangle is dead,  
long live the triangle!*

# 1. Research overview

This thesis summarizes the research conducted during my doctoral studies in computer vision. Over the course of my PhD, my research evolved across two main directions.

*The first part* of my research focused on the application of artificial intelligence to sports analytics, with a particular emphasis on automated assistance for football refereeing. As a football referee myself, I was particularly interested in how computer vision methods could support referees in analyzing complex game situations. In this context, the research explored how deep learning models can automatically detect fouls, classify incidents, and provide decision support based on match footage.

In addition to recognizing incidents, the work also investigated how modern AI systems can generate explanations for their predictions to improve transparency and trust in automated decision-making. This line of research resulted in the development of the VARS and X-VARS frameworks, which combine video understanding with explainable AI techniques to assist refereeing decisions. This work led to three research papers presented in the appendix of this thesis.

*The second part* of my doctoral studies shifted toward a different challenge in computer vision, initially also motivated by its application in football: reconstructing and rendering 3D scenes from images. This problem is central to many applications, including virtual and augmented reality, robotics, and digital content creation.

The main focus of this thesis is the second research direction, which investigates 3D scene reconstruction through radiance field representations based on explicit primitives optimized via differentiable rendering. In particular, the thesis explores splatting-based representations that progressively increase the geometric structure of the underlying primitives, including smooth convexes, triangles, and meshes. By introducing more structured primitives, these approaches aim to improve reconstruction quality, efficiency, and compatibility with traditional graphics pipelines.

Finally, while the approaches developed in this thesis are generic and applicable to a wide range of 3D scenes, I have also explored their application to sports scenarios. This year, I co-organized the SoccerNet Novel View Synthesis Challenge, which investigates the combination of radiance field rendering with football broadcast footage. In collaboration with *EVS Broadcast*, we constructed a dataset to allow participants to generate novel viewpoints and evaluate their methods in realistic sports settings.

## 2. Introduction

### 2.1. Introduction to novel view synthesis

Humans have an extraordinary ability to imagine the world from viewpoints they have never directly observed. When looking at an image, we can often infer what lies just outside the frame or how the scene would appear from a slightly different angle. This intuitive understanding of 3D structure from limited observations is fundamental to how we perceive and interact with our environment.

Reproducing this capability in machines has been a long-standing goal in computer vision and computer graphics. The task, commonly referred to as *novel view synthesis*, aims to generate images of a scene from previously unseen viewpoints, given only a set of input observations. Figure 2.1 illustrates a typical setup in which a scene is captured from multiple camera poses with the goal of rendering it from previously unseen perspectives.



Figure 2.1.: **Overview of novel view synthesis.** The goal is to render the scene from novel camera poses that were not observed during training, while preserving the appearance and structure consistent with the input views. The image is taken from [10].



Achieving the long-standing goal of accurate novel view synthesis from a sparse set of observations would fundamentally change how we interact with visual data. Rather than being limited to static images or pre-recorded videos, scenes could be represented as fully navigable environments in which viewpoints are no longer constrained at capture time but generated on demand. This would enable a shift from passive viewing to active exploration, allowing users to move freely within a scene and observe it from arbitrary perspectives.

Such a capability has a wide range of application domains. In virtual and augmented reality, it enables immersive experiences to be created from sparse image collections, reducing the need for expensive capture setups. In robotics and autonomous systems, the ability to infer unseen viewpoints can improve scene understanding and decision-making in partially observed environments. In movies and series, it enables viewers to become more immersed in the narrative by exploring scenes from different perspectives. In sports, it enables free-viewpoint replays, allowing key moments to be visualized from any angle, thereby enhancing analysis, broadcasting, and viewer experience.

The first major breakthrough in achieving high-quality novel view synthesis was Neural Radiance Fields (NeRFs) [6]. In NeRFs, scenes are represented as continuous volumetric functions learned directly from images. To render an image from a given viewpoint, 5D coordinates are sampled along camera rays. These 5D coordinates consist of the 3D spatial position  $\mathbf{x} \in \mathbb{R}^3$  and the 2D viewing direction  $(\theta, \phi)$ .

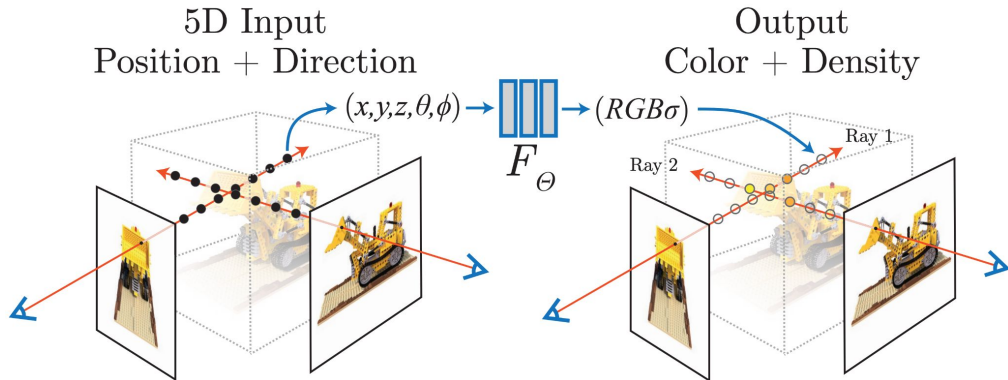


Figure 2.2.: **Overview of Neural Radiance Fields (NeRF).** (Left) Images are rendered by sampling 5D coordinates along camera rays. (Right) 5D coordinates are fed into an MLP to produce a color and volume density. Modified image taken from [6].

Those 5D coordinates are then fed to an MLP, which predicts the RGB color and the volume density  $\sigma$ . The volume density can be seen as how much a point contributes to blocking or emitting light along a ray. Finally, the color of any ray passing through the scene is rendered using principles from classical volume rendering [4].

The rendered color  $C(r)$  of a camera ray  $r(t) = \mathbf{o} + t\mathbf{d}$  is given by:

$$C(r) = \int_{t_n}^{t_f} T(t) \sigma(r(t)) c(r(t), \mathbf{d}) dt, \quad (2.1)$$

where

$$T(t) = \exp \left( - \int_{t_n}^t \sigma(r(s)) ds \right), \quad (2.2)$$

with  $t_n$  and  $t_f$  being the near and far bounds, and  $T(t)$  the accumulated transmittance along the ray from  $t_n$  to  $t$ . Figure 2.2 gives an overview of the method.

This approach produces extremely high-quality novel view synthesis results, but early neural radiance field models rely on neural networks evaluated at a high number of samples along each ray. This leads to slow rendering speeds and limits their applicability in interactive settings, as performance remains far from real-time requirements.

To address this limitation, another representation has emerged that represents scenes using collections of explicit primitives optimized through differentiable rendering. 3D Gaussian Splatting (3DGS) [5] demonstrated that scenes can be efficiently represented by thousands or millions of anisotropic Gaussian primitives.

More specifically, each Gaussian is defined by its 3D position  $\mathbf{x} \in \mathbb{R}^3$ , covariance, opacity, and color. For a given viewpoint, the 3D Gaussians are efficiently projected into 2D image space by applying the corresponding camera transformation followed by a differentiable projection of the 3D covariance into a 2D elliptical Gaussian in screen space. Then, each of the  $N$  image space Gaussians is sorted according to its depth (from front to back). The color of a pixel  $\mathbf{p}$  is then obtained via alpha compositing:

$$C(\mathbf{p}) = \sum_{n=1}^N \mathbf{c}_n \alpha_n \prod_{j=1}^{n-1} (1 - \alpha_j), \quad (2.3)$$

where  $\mathbf{c}_n$  denotes the view-dependent color of the  $n$ -th Gaussian (typically modeled using spherical harmonics and evaluated given the viewing direction), and  $\alpha_n$  is the influence of the  $n$ -th Gaussian at pixel  $\mathbf{p}$  in screen space. It is computed by evaluating the projected 2D Gaussian at that pixel, multiplied by the learned opacity parameter.

The entire rendering pipeline is fully differentiable, allowing image-space losses to be computed directly against the training views. This enables gradients to propagate back to the individual Gaussian parameters, which can then be optimized to accurately reconstruct the scene from multiple viewpoints. Figure 2.3 illustrates the individual 3D Gaussians in the scene representation, as well as the resulting final RGB rendering.

A main advantage of 3D Gaussian Splatting is its highly efficient rendering. This leads to significantly faster training and inference speeds, reaching several orders



Figure 2.3.: **3D Gaussian Splatting.** (Left) Visualization of individual Gaussians in the scene representation. (Right) Corresponding rendered RGB image obtained via differentiable splatting. Image taken and modified from [9].

of magnitude improvement over neural radiance field-based approaches while maintaining comparable reconstruction quality.

Despite their success, Gaussian-based splatting representations also present several limitations. Because Gaussian primitives represent volumetric blobs in space, accurately modeling sharp geometric structures such as edges and thin surfaces often requires a large number of primitives. Moreover, these primitives tend to disperse around surfaces rather than forming coherent geometric structures. With millions of Gaussians representing a single scene, tasks such as mesh extraction, scene editing, or integration into standard graphics pipelines become challenging. This is primarily because the representation remains inherently unstructured, with Gaussians distributed in space rather than forming coherent surface geometry, making it difficult to recover explicit meshes without postprocessing steps.

Moreover, this unstructured, volumetric formulation is fundamentally misaligned with classical computer graphics pipelines, which are designed around explicit surface primitives, most notably triangle or quad meshes. Rasterization pipelines assume well-defined surfaces with clear visibility ordering, enabling efficient techniques such as depth buffering, early-z rejection, and occlusion culling. In contrast, Gaussian splatting represents pixel color through accumulated density and alpha blending, requiring primitives to be sorted and composited at render time. This not only introduces additional computational overhead but also prevents the use of key hardware-accelerated features of modern GPUs.

Consequently, Gaussian-based methods rely on custom rendering pipelines that bypass the fixed-function rasterization hardware, reducing efficiency and scalability compared to mesh-based approaches. Their lack of explicit geometry limits compatibility with existing graphics ecosystems, including real-time rendering engines, mesh editing tools, and simulation frameworks, all of which are built to operate on structured surface representations.

## 2.2. Research objective

This thesis investigates the role of geometric primitives in radiance field representations and explores how novel primitives can improve reconstruction quality, efficiency, and compatibility with traditional graphics pipelines. The central hypothesis is that the choice of primitive strongly influences the properties of the resulting representation. By progressively increasing the geometric structure of primitives, it is possible to obtain representations that retain the advantages of splatting methods while improving geometric consistency and practical usability.

To explore this idea, this thesis introduces three complementary approaches based on increasingly structured primitives: smooth convex [3], triangles [2], and meshes [1].

The first contribution, presented in *3D Convex Splatting* (3DCS), replaces Gaussian primitives with smooth convex shapes for radiance field rendering. Convex primitives provide greater geometric expressiveness and can represent planar surfaces and sharp edges more effectively than Gaussian blobs. By combining these primitives with an efficient CUDA-based rasterization pipeline, the method achieves improved reconstruction quality while requiring fewer primitives and maintaining real-time rendering performance.

The second contribution revisits one of the most fundamental primitives in computer graphics: triangles. The work *Triangle Splatting* introduces a differentiable rendering framework that enables direct optimization of triangle primitives through end-to-end gradients. Instead of using triangles only for final surface extraction, triangles themselves become the representation used during optimization. By rendering triangles as differentiable splats, the method combines the efficiency and hardware compatibility of triangle rendering with the adaptive density of primitive-based radiance fields. This approach achieves high visual fidelity, fast convergence, and high rendering throughput.

Finally, the thesis extends this progression toward connected geometry with Mesh-Splatting, a mesh-based reconstruction framework that jointly optimizes geometry and appearance through differentiable rendering. Starting from an unstructured set of triangles, the method enforces mesh connectivity in a second stage using restricted Delaunay triangulation, producing coherent surfaces.

As the final optimized mesh consists of fully opaque and discrete triangles, it is directly compatible with real-time rendering engines, allowing efficient rasterization via modern GPU pipelines, including hardware-accelerated depth testing, occlusion culling, and shading. Moreover, the resulting representation integrates naturally with existing graphics workflows, supporting downstream applications such as editing, simulation, and asset export without requiring additional conversion steps.

This representation bridges neural rendering techniques with traditional mesh-based graphics pipelines used in applications such as augmented or virtual reality, and

interactive 3D environments.

Taken together, these contributions form a unified exploration of structured primitive representations for neural scene rendering. The thesis demonstrates that moving beyond Gaussian-based primitives toward more structured geometric representations can significantly improve scene modeling, rendering efficiency, and compatibility with classical computer graphics. By integrating differentiable rendering with structured geometric primitives, it opens the door to reconstruction methods that combine the flexibility of neural representations with the practicality and efficiency of mesh-based graphics pipelines.

## 2.3. Contributions

### 2.3.1. List of publications

*Publications included in this thesis:*

- Chapter 3 **Jan Held**, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. MeshSplatting: Differentiable rendering with opaque meshes. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Denver, CO, USA, Jun. 2026
- Chapter 4 **Jan Held\***, Renaud Vandeghen\*, Adrien Delière, Daniel Hamdi, Abdullah Rebain, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. In Int. Conf. 3D Vis. (3DV), pages 1–10, Vancouver, Can., Mar. 2026.
- Chapter 5 **Jan Held\***, Renaud Vandeghen\*, Abdullah Hamdi\*, Adrien Delière, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pages 21360–21369, Nashville, TN, USA, Jun. 2025

*Publications included in Appendix A:*

- Section A.4 **Jan Held\***, Anthony Cioppa\*, Silvio Giancola\*, Abdullah Hamdi, Bernard Ghanem, and Marc Van Droogenbroeck. VARS: Video assistant referee system for automated soccer decision making from multiple views. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pages 5086–5097, Vancouver, Can., Jun. 2023.
- Section A.5 **Jan Held\***, Hani Itani\*, Anthony Cioppa, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. X-VARS: Introducing explainability in foot-

ball refereeing with multi-modal large language models. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pages 3267–3279, Seattle, WA, USA, Jun. 2024.

Section A.6 **Jan Held**, Anthony Cioppa, Silvio Giancola, Abdullah Hamdi, Christel Devue, Bernard Ghanem, and Marc Van Droogenbroeck. Towards an AI-powered video assistant referee system (VARS) for association football. In Scand. Conf. Image Anal. (SCIA), volume 15725 of Lect. Notes Comput. Sci., pages 295–309, 2025.

*Publications not included in this thesis:*

- Victor Rong\*, **Jan Held**\*, Victor Chu, Daniel Rebain, Marc Van Droogenbroeck, Kiriakos N. Kutulakos, Andrea Tagliasacchi, and David B. Lindell. Nexels: Neurally-textured surfels for real-time novel view synthesis with sparse geometries. arXiv, abs/2512.13796, 2025.
- Victor Gaspar, Anthony Cioppa, **Jan Held**, Silvio Giancola, Marc Braham, Adrien Delière, Bernard Ghanem, and Marc Van Droogenbroeck. An end-to-end pipeline for virtual banner replacement in football broadcasts. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pages 5872–5883, Nashville, TN, USA, Jun. 2025.
- Anais Halin, Sebastien Pierard, Renaud Vandeghen, Benoit Gerin, Maxime Zanella, Martin Colot, **Jan Held**, Anthony Cioppa, Emmanuel Jean, Gianluca Bontempi, Said Mahmoudi, Benoit Macq, and Marc Van Droogenbroeck. Physically interpretable probabilistic domain characterization. In Asian Conf. Comput. Vis. Work. (ACCV Work.), volume 15482 of Lect. Notes Comput. Sci., pages 17–35, Hanoi, Vietnam., 2025.
- **Jan Held**, Anthony Cioppa, Silvio Giancola, Elaf Almahmoud, Katherine M. Collins, Umang Bhatt, Bernard Ghanem, and Marc Van Droogenbroeck. Enhancing football refereeing with AI: VARS and X-VARS for assisted decision-making. In MathSport Conference, Luxembourg, Jun. 2025.
- Silvio Giancola, Anthony Cioppa, Marc Gutierrez-Perez, **Jan Held**, Carlos Hinojosa, Victor Joos, Arnaud Leduc, Bernard Ghanem, Marc Van Droogenbroeck, and et al.. SoccerNet 2025 challenges results. arXiv, abs/2508.19182, 2025.
- Anthony Cioppa, Silvio Giancola, Vladimir Somers, Victor Joos, Floriane Magera, **Jan Held**, Bernard Ghanem, Marc Van Droogenbroeck, and et al. SoccerNet 2024 challenges results. arXiv, abs/2409.10587, 2024.
- Sushant Gautam, Mehdi Houshmand Sarkhoosh, **Jan Held**, Cise Midoglu, Anthony Cioppa, Silvio Giancola, Vajira Thambawita, Michael A. Riegler, Pal Halvorsen, and Mubarak Shah. SoccerNet-Echoes: A soccer game audio

---

\* means equal contribution

commentary dataset. Int. Symp. Multimedia (ISM), pages 71–78, Dec. 2024.

- Anthony Cioppa, Silvio Giancola, Vladimir Somers, Floriane Magera, Xin Zhou, Hassan Mkhallati, Adrien Deliege, **Jan Held**, Carlos Hinojosa, Amir M. Mansourian, Pierre Miralles, Olivier Barnich, Christophe De Vleeschouwer, Alexandre Alahi, Bernard Ghanem, Marc Van Droogenbroeck, and et al. SoccerNet 2023 challenges results. Sports Eng., 27(2):1–18, Jul. 2024.

### 2.3.2. Software contributions and community recognition

A major contribution of this thesis was the open access release of all code, enabling researchers to build on top of our work. On GitHub, 3D Convex Splatting, Triangle Splatting, and MeshSplatting have achieved a combined total of 2,000 stars, with Triangle Splatting alone reaching 1,100.

In addition to the GitHub recognition, the work has also attracted attention from the broader AI community. Several independent videos have been widely viewed. Most notably, a video by an AI content creator, *The Future of 3D Is... Triangles?!*, explains Triangle Splatting and its improvements over Gaussian Splatting and has accumulated over 180,000 views: <https://www.youtube.com/watch?v=F0H3NAHP9r0>.

Finally, the work presented in this thesis has also received recognition at major conferences. 3D Convex Splatting was selected as a *Highlight* paper at CVPR 2025.

# Part 1

---

*The First Novel Primitive*



## 3. 3D Convex Splatting

### 3.1. Context

3D Gaussian Splatting has recently transformed the field of real-time novel view synthesis by enabling high-quality scene reconstruction using a large set of explicit primitives. In 3DGS, a scene is represented with millions of 3D Gaussians, which can be efficiently optimized in less than an hour and rendered in real time while maintaining high visual quality.

Despite these advantages, 3D Gaussian Splatting exhibits several inherent limitations. Due to their smooth, rounded nature, Gaussians lack well-defined physical boundaries, making them poorly suited for representing sharp edges and surfaces.

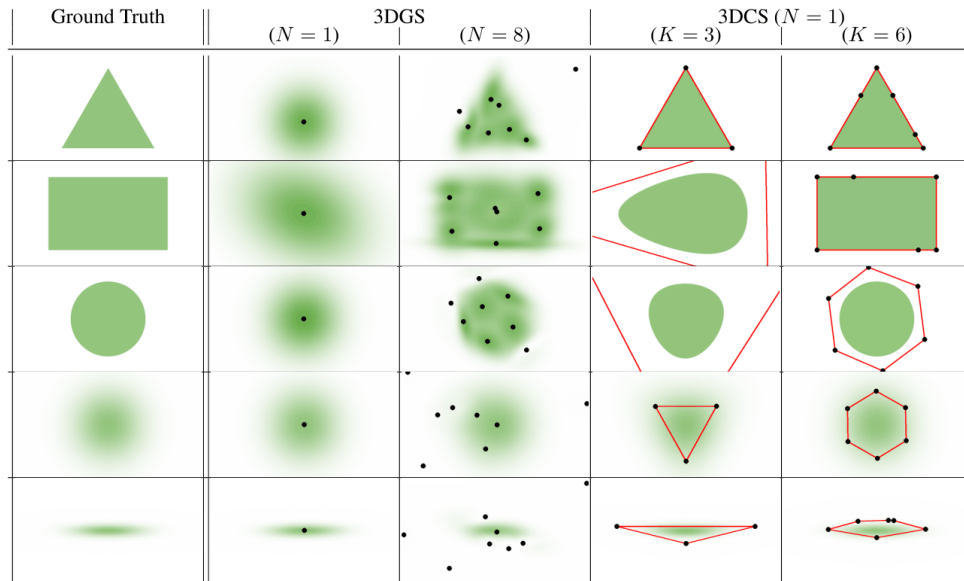


Figure 3.1.: **Reconstruction of Simple Shapes: Comparison of 3D Gaussians vs. 3D Convexes.** Smooth convex primitives are more expressive and can reconstruct simple shapes more accurately than Gaussians. Smooth convexes can form sharper geometric boundaries and are even capable of approximating Gaussian primitives. For 3D Convex Splatting (3DCS), the red lines indicate the convex hull, while the black dots represent the underlying point set. For 3DGS, the black dots denote the Gaussian centers.

As a result, accurately reconstructing high-frequency details or planar surfaces often requires a large number of overlapping Gaussians and higher memory consumption.

This raises the question: *are 3D Gaussians the best primitive for novel view synthesis?*

In this chapter, we address this question by introducing *3D smooth convexes* as a novel geometric primitive. In contrast to Gaussians, smooth convex primitives provide greater expressiveness and structural control. They are capable of forming compact, dense volumes that more accurately capture sharp features and detailed surfaces using fewer primitives.

A key aspect of our formulation is the introduction of explicit *smoothness* and *sharpness* parameters, which provide independent control over curvature and boundary transitions. This allows the representation to continuously interpolate between diffuse and sharply defined shapes, significantly expanding the range of geometries that can be modeled efficiently. As a result, complex structures that would otherwise require a large number of Gaussians can be captured using a smaller set of smooth convex primitives. Figure 3.1 illustrates the reconstruction capabilities on a simple 2D toy example. It shows that Gaussians struggle to accurately represent sharp geometry and require a large number of primitives to approximate sharp edges accurately. In contrast, smooth convex representations are expressive enough to reconstruct sharp boundaries while also capturing smooth transitions, and can even approximate Gaussian-like structures.

Beyond visual quality and memory efficiency, computational performance is a main factor. To this end, we adapt the Gaussian GPU-based rendering pipeline to support efficient splatting of smooth convexes. This implementation preserves real-time rendering capabilities while maintaining full differentiability, enabling efficient gradient-based optimization of smooth convexes.

Finally, and most importantly, introducing 3D smooth convexes as a new primitive demonstrates that the design space of splatting-based representations extends well beyond Gaussians. By moving away from the constraints inherent to Gaussian formulations, this work shows that alternative primitives can exhibit fundamentally different geometric and rendering properties. 3D Convex Splatting establishes the technical foundation for the subsequent developments in this thesis, including *Triangle Splatting* and *MeshSplatting*.

## 3.2. Scientific contributions

We summarize our main contributions as follows:

- (i) We introduce 3D Convex Splatting, a novel radiance field representation based on 3D smooth convex primitives. To the best of our knowledge, it is the first

real-time rendering approach built on non-Gaussian primitives.

- (ii) We develop a differentiable, GPU-based rendering framework for smooth convexes, enabling high-quality 3D scene reconstruction from multi-view images while maintaining real-time rendering performance.
- (iii) We establish a foundation for subsequent research, with 3D Convex Splatting serving as the basis for follow-up work by other researchers, as well as for the methods presented in Chapters 4 and 5.

### 3.3. Publication and recognition

The work included in this chapter was published as:

**Jan Held\***, Renaud Vandeghen\*, Abdullah Hamdi\*, Adrien Delière, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, USA, 2025.

The paper was accepted as a **Highlight paper** at CVPR 2025.

*Author contribution.* As a lead author, I co-designed the method, implemented major components of the training pipeline, and conducted experiments and analysis. Renaud Vandeghen supported the work through numerous discussions, assisted with running and validating experiments, and benchmarking concurrent work. Abdullah Hamdi contributed with his expertise by providing valuable guidance and scientific feedback throughout the development of the project.

# 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes

Jan Held\*<sup>1,2</sup> Renaud Vandeghen\*<sup>1</sup> Abdullah Hamdi\*<sup>3</sup> Adrien Deliege<sup>1</sup> Anthony Cioppa<sup>1</sup>  
Silvio Giancola<sup>2</sup> Andrea Vedaldi<sup>3</sup> Bernard Ghanem<sup>2</sup> Marc Van Droogenbroeck<sup>1</sup>

<sup>1</sup> University of Liège <sup>2</sup> KAUST <sup>3</sup> University of Oxford

\* Equal contribution

## Abstract

Recent advances in radiance field reconstruction, such as 3D Gaussian Splatting (3DGS), have achieved high-quality novel view synthesis and fast rendering by representing scenes with compositions of Gaussian primitives. However, 3D Gaussians present several limitations for scene reconstruction. Accurately capturing hard edges is challenging without significantly increasing the number of Gaussians, creating a large memory footprint. Moreover, they struggle to represent flat surfaces, as they are diffused in space. Without hand-crafted regularizers, they tend to disperse irregularly around the actual surface. To circumvent these issues, we introduce a novel method, named 3D Convex Splatting (3DCS), which leverages 3D smooth convexes as primitives for modeling geometrically-meaningful radiance fields from multi-view images. Smooth convex shapes offer greater flexibility than Gaussians, allowing for a better representation of 3D scenes with hard edges and dense volumes using fewer primitives. Powered by our efficient CUDA-based rasterizer, 3DCS achieves superior performance over 3DGS on benchmarks such as Mip-NeRF360, Tanks and Temples, and Deep Blending. Specifically, our method attains an improvement of up to 0.81 in PSNR and 0.026 in LPIPS compared to 3DGS while maintaining high rendering speeds and reducing the number of required primitives. Our results highlight the potential of 3D Convex Splatting to become the new standard for high-quality scene reconstruction and novel view synthesis. The project page is <https://convexsplatting.github.io>

## 1. Introduction

Reconstructing complex scenes and synthesizing novel views have been fundamental challenges in computer vision and graphics for decades [1, 12], with applications ranging from virtual reality to autonomous navigation [17, 34, 38]. Neural Radiance Fields (NeRF) [36] revolutionized this area by representing scenes as continuous volumetric radi-

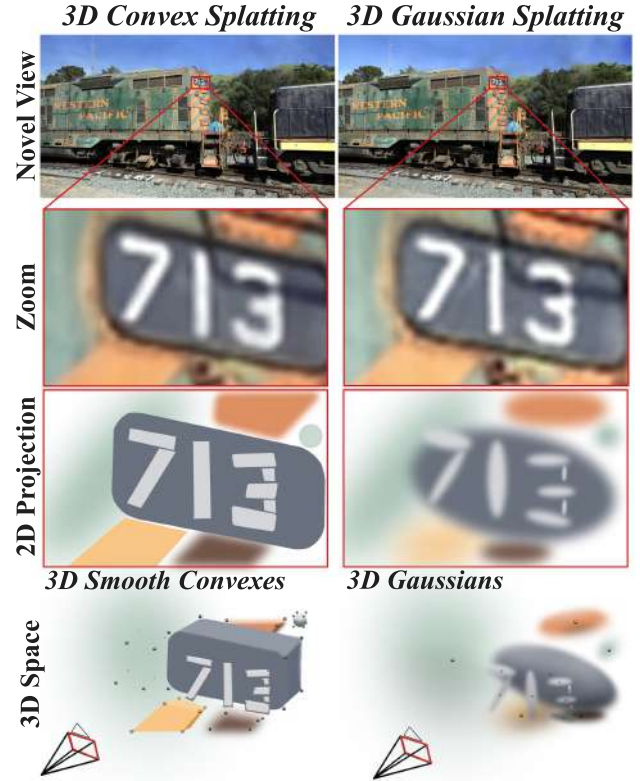


Figure 1. **3D Convex Splatting for Novel View Synthesis.** We introduce a novel primitive-based pipeline for novel view synthesis with 3D smooth convexes. Our 3D smooth convexes share the rendering speed of 3D Gaussians [25] and the flexible representation of smooth convexes [9]. As a result, 3D Convex Splatting better reconstructs scenes with fewer primitives.

ance fields, which are optimized to render novel views at high-quality. However, NeRF suffered from slow training and rendering times, limiting its practicality. To address these issues, 3D Gaussian Splatting (3DGS) [25] emerged as an efficient alternative, by representing scenes with millions of 3D Gaussian primitives. 3DGS significantly accelerated training and enabled real-time rendering while main-

taining high-quality outputs.

Despite this progress, Gaussian primitives have two inherent limitations. (1) Gaussians lack defined physical boundaries, making them unsuitable for accurately representing flat surfaces or enabling physically meaningful decompositions of scenes. (2) In addition to their specific smoothness and rounded nature, Gaussians are inadequate for capturing hard edges and geometric structures. Each Gaussian behaves similarly to an ellipsoid, with a symmetrical distribution, struggling to conform to angular boundaries or flat surfaces. This inherent limitation is highlighted by the sphere packing problem in computer graphics [8, 16], where densely packed spherical or ellipsoidal shapes leave gaps and result in inefficient coverage, especially along flat or sharp corners. As with spheres or ellipsoids, an impractically large number of Gaussian primitives would be needed to fill space without gaps, leading to increased memory consumption and computational overhead.

To overcome these limitations, we propose a novel method called *3D Convex Splatting (3DCS)*, which leverages *3D smooth convexes* as primitives for modeling and reconstructing geometrically accurate radiance fields from multi-view images. 3D smooth convexes offer greater flexibility than Gaussians, as they can form dense volumes that accurately capture hard edges and detailed surfaces using fewer primitives. Figure 1 illustrates this point, showing that 3DCS enables the rendering of 3D smooth convexes to generate high-quality novel views of complex scenes. Moreover, by incorporating *smoothness* and *sharpness* parameters, we can control the curvature and the diffusion of the smooth convexes, respectively. This enables the creation of shapes that are hard or soft, dense or diffuse. Figure 2 shows a toy example of how smooth convexes can represent a chair with hard edges with far fewer elements than Gaussians, while utilizing the same optimization. For novel view synthesis, we merge the benefits of the fast rendering process from Gaussians [25], with the flexibility of 3D smooth convexes [9]. We achieve this by rendering 3D smooth convexes using our efficient CUDA-based rasterizer, which enables real-time rendering and accelerates the optimization process. To the best of our knowledge, 3D Convex Splatting is the first method to leverage differentiable smooth convex shapes for novel view synthesis on realistic scenes, outperforming previous methods that use other primitives.

**Contributions.** We summarize our contributions as follows: (i) We introduce 3D Convex Splatting (3DCS), utilizing 3D smooth convexes as novel primitives for radiance field representation, addressing the limitations of Gaussian primitives in capturing dense volumetric features. (ii) We develop an optimization framework and a fast, differentiable GPU-based rendering pipeline for our 3D smooth convexes, enabling high-quality 3D scene representations from multi-view images and high rendering speeds.

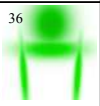
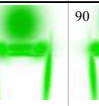
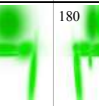
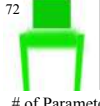
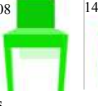
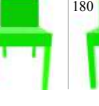
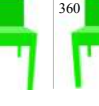
| # of Primitives | 4                                                                                  | 6                                                                                   | 8                                                                                   | 10                                                                                  | 20                                                                                  |
|-----------------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Input Image     |  |  |  |  |  |
|                 | 36                                                                                 | 54                                                                                  | 72                                                                                  | 90                                                                                  | 180                                                                                 |
| Input Image     |  |  |  |  |  |
|                 | 72                                                                                 | 108                                                                                 | 144                                                                                 | 180                                                                                 | 360                                                                                 |
| # of Parameters |                                                                                    |                                                                                     |                                                                                     |                                                                                     |                                                                                     |

Figure 2. **Toy Experiment of Modeling a Chair.** For the chair input image, we use Gaussians and smooth 6-point convexes to fit the chair with an increasing number of primitives. Note how the convexes efficiently represent the chair with fewer parameters.

(iii) 3DCS surpasses existing rendering primitives on MipNeRF360, Tanks and Temples, and Deep Blending datasets, achieving better performance than 3D Gaussian Splatting while using a reduced number of primitives per scene.

## 2. Related Work

**Neural radiance fields (NeRF).** Recovering the 3D structure of a scene from images captured from multiple viewpoints is a fundamental problem in computer vision [1, 12]. The introduction of Neural Radiance Fields (NeRF) [36] revolutionized this field by representing scenes as volumetric radiance fields, enabling high-quality novel view synthesis [2, 3, 46]. NeRF employs multi-layer perceptrons to encode scene geometry and view-dependent appearance, optimized via photometric loss through volume rendering [10, 23, 29, 35]. Enhancements to NeRF include grid-based representations for faster training [6, 13, 37, 45], baking techniques for accelerated rendering [20, 42, 43, 47], as well as addressing challenges such as antialiasing [3, 4], modeling unbounded scenes, and adapting to few-shot [11, 22, 26] and one-shot settings [5, 48]. In this work, we do not rely on neural networks to model radiance fields like other NeRFs, but instead optimize 3D smooth convexes to fit 3D scenes efficiently. Yet, 3D Convex Splatting allows for strong modeling capacity that rivals MipNeRF-360 [3] in visual fidelity but with real-time rendering speed.

**Primitive-based differentiable rendering.** Differentiable rendering techniques enable gradient computation through the rendering pipeline, facilitating the optimization of scene parameters from image observations [15, 24, 30–32]. Neural point-based rendering [24] represents scenes with points that store learned features for geometry and texture. 3D Gaussian Splatting (3DGS) [25] introduces Gaussian primitives parameterized by positions, covariances, and appearance attributes. By optimizing millions of Gaussians, 3DGS achieves high-quality rendering with significantly faster training times and real-time rendering capa-

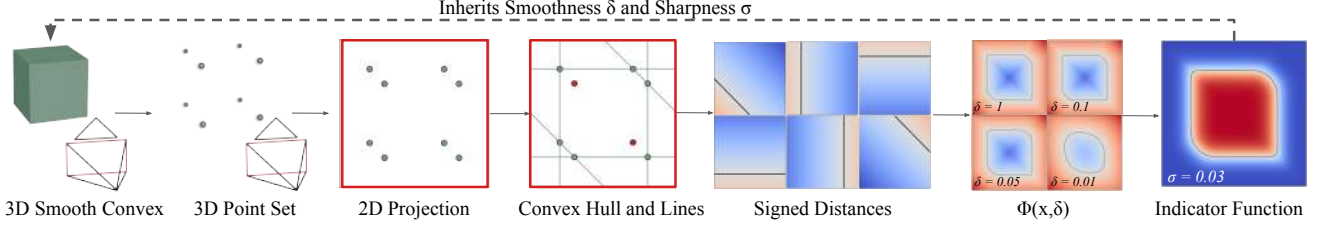


Figure 3. **Convex Splatting Pipeline.** The 3D smooth convex is represented with a point set that is projected in the 2D camera plane. We extract the line-delimited convex hull of the projected points and define the signed distance function for each line. The lines are combined to define an indicator function for each pixel based on smoothness  $\delta$  and sharpness  $\sigma$  of the 3D convex. The pipeline is differentiable end-to-end, which allows the parameters of the smooth convex primitives to be optimized based on the rendered images.

bilities. Enhancements to this approach include antialiasing techniques [49], and extensions to dynamic scene modeling [33]. However, Gaussian primitives have inherent limitations due to their very specific smoothness, making it challenging to capture hard edges and dense volumetric structures without significantly increasing the number of primitives. This leads to increased memory consumption and computational overhead, hindering scalability and efficiency. Alternative primitives have been explored to improve geometric representation. GES [18] utilizes generalized exponential functions to better capture signals with harder edges. 2D Gaussian splatting [21] collapses the 3D Gaussians into oriented planar Gaussians to better represent surfaces. In our work, we introduce 3D smooth convex shapes as a novel primitive for real-time rendering of novel views. These shapes address the limitations of the Gaussian primitives by efficiently capturing dense volumetric shapes.

**Convex shapes.** Convex shapes have been extensively studied in computer graphics and computer vision due to their geometric simplicity and flexibility in representing complex objects [14, 41]. In 3D reconstruction, convex shape representations have proven highly effective for decomposing complex structures into simpler components like planes, spheres, cubes, cylinders, and superquadrics [40]. CvxNet [9] and BSP-Net [7] introduce neural networks that learn hyperplanes to construct flexible convex shapes, enabling more accurate differentiable modeling of geometries with primitives. A concurrent work utilizes rigid convex polyhedra and differentiable mesh rendering to fit simple 3D shapes with few primitives using multi-view supervision [44]. However, these methods are limited to simple shapes with few optimized primitives and do not scale to large scenes or allow for accurate novel view synthesis.

In our work, we introduce splatting-based rasterization with an array of smooth, high-capacity primitives. This allows us to achieve rendering fidelity levels comparable to volume rendering techniques in modeling complex scenes.

### 3. Methodology

3D Convex Splatting (3DCS) combines smooth convexes (Sec. 3.1) from CvxNet [9] with the primitive-based representation from 3DGS [25] for efficient, real-time novel view synthesis. 3DCS uses a point-based convex shape representation and convex hull operations to enable straightforward differentiable projection of 3D convex shapes onto the 2D image plane (Sec. 3.2). Further operations for smoothing, splatting, and adaptive densification are used to optimize the representation from posed images (Sec. 3.3). Figure 3 shows an overview of our convex splatting pipeline.

#### 3.1. Preliminaries on 3D Smooth Convexes

Following CvxNet [9], we define a convex polyhedron with  $J$  planes ( $J > 3$ ). We define the signed distance  $L_j(\mathbf{p})$  between a point  $\mathbf{p} \in \mathbb{R}^3$  and a plane  $\mathcal{H}_j$  as follows:

$$L_j(\mathbf{p}) = \mathbf{n}_j \cdot \mathbf{p} + d_j \quad (1)$$

with  $\mathbf{n}_j$  being the plane normal pointing towards the outside of the shape and  $d_j$  its offset. The signed distance from  $\mathbf{p}$  to the convex shape is calculated as the maximum of all the  $J$  distances defined by  $\tilde{\phi}(\mathbf{p}) = \max_{j=1, \dots, J} L_j(\mathbf{p})$ . However, to create a smooth representation of the convex shape, we use a smooth approximate signed distance function  $\phi(\mathbf{p}) \approx \tilde{\phi}(\mathbf{p})$  using the LogSumExp function from CvxNet [9]:

$$\phi(\mathbf{p}) = \log \left( \sum_{j=1}^J \exp(\delta L_j(\mathbf{p})) \right), \quad (2)$$

where the *smoothness* parameter  $\delta > 0$  controls the curvature of the convex approximation. Larger values of  $\delta$  approximate  $\phi(\mathbf{p})$  to  $\tilde{\phi}(\mathbf{p})$  more closely, resulting in harder edges, while smaller values soften the vertices.

The indicator function  $I(\mathbf{p})$  of the smooth convex is then defined by applying a sigmoid function to the approximate signed distance function [9]:

$$I(\mathbf{p}) = \text{Sigmoid}(-\sigma \phi(\mathbf{p})), \quad (3)$$



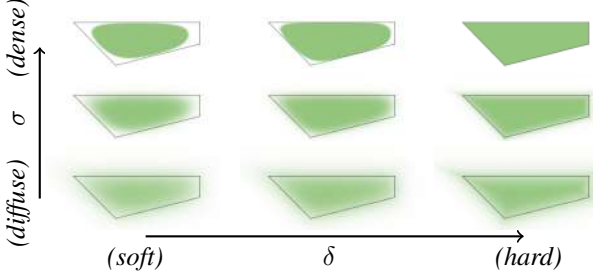


Figure 4. **Effects of  $\delta$  and  $\sigma$  on Splatting.** The smoothness  $\delta$  characterizes vertices and edges, from soft to hard, while the sharpness  $\sigma$  characterizes radiance field transitions, from diffuse to dense.

where the *sharpness* parameter  $\sigma > 0$  controls how rapidly the indicator function transitions at the boundary of the underlying convex shape. Higher values of  $\sigma$  result in a steeper transition, making the shape’s boundary more defined, whereas lower values result in a more diffuse shape. At the boundary of the convex shape, where  $\phi(\mathbf{p}) = 0$ , the indicator function of the smooth convex satisfies  $I(\mathbf{p}) = 0.5$ . More details about smooth convexes can be found in [9]. Fig. 4 illustrates the effect of sharpness  $\sigma$  and smoothness  $\delta$  on the indicator function  $I(\cdot)$ .

### 3.2. 3DCS: Splatting 3D Smooth Convexes

**Point-based 3D convex shape representation.** Plane-based representations of 3D convexes are impractical for camera plane projections. Unlike CvxNet [9], we define a 3D convex as the convex hull of a 3D point set  $\mathbb{S} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_K\}$ . During optimization, the 3D points can move freely, allowing for flexible positioning and morphing of the convex shape. Note that this set of  $K$  points does not necessarily correspond to the *explicit* vertices of a convex polyhedron, but rather the hull of the 3D convex shape.

**Differentiable projection onto the 2D image plane.** To be efficient, we do not explicitly build the 3D convex hull and project it into 2D, but instead, we project the 3D points into 2D and then construct its 2D convex hull. Specifically, we project each 3D point  $\mathbf{p}_k \in \mathbb{S}$  onto the 2D image plane using the pinhole perspective camera projection model. The projection involves the intrinsic camera matrix  $\mathbf{K}$  and extrinsic parameters (rotation  $\mathbf{R}$  and translation  $\mathbf{t}$ ):

$$\mathbf{q}_k = \mathbf{K}(\mathbf{R}\mathbf{p}_k + \mathbf{t}), \quad \forall k = 1, 2, \dots, K. \quad (4)$$

This projection is differentiable, allowing gradients to flow back to the 3D points during optimization.

**2D convex hull computation.** To construct the convex shape in 2D, we apply the Graham Scan algorithm [14], which efficiently computes the convex hull by retaining only the points that define the outer boundary of the projected shape. This approach ensures that the 2D projection

accurately represents the convex outline needed for rendering. The Graham Scan starts by sorting the points based on their polar angle relative to a reference point. After sorting, the Graham Scan algorithm is applied to construct the convex hull by iteratively adding points to the hull while maintaining convexity. The convexity is ensured by checking the cross product of the last two points  $\mathbf{q}_i$  and  $\mathbf{q}_j$  on the convex hull and the current point  $\mathbf{q}_k$ , removing points that form a right turn (negative cross product).

**Differentiable 2D convex indicator function.** We define the 2D convex indicator function of our convex hull by extending the smooth convex representation from 3D to 2D, reusing the equations introduced in Sec. 3.1. We define  $\phi(\mathbf{q})$  and  $I(\mathbf{q})$  as in Eqs. (2) and (3), but replace the 3D point  $\mathbf{p}$  with the 2D point  $\mathbf{q}$  and the planes delimiting the 3D convex hull by the lines delimiting the resulting 2D convex hull. The parameters  $\sigma$  and  $\delta$  are inherited from the 3D smooth convex (from Eqs. (2) and (3)) and still control the sharpness and smoothness of the projected 2D shape boundaries. To account for perspective effects in the 2D projection, we scale  $\delta$  and  $\sigma$  by the distance  $d$ , ensuring that the appearance of the convex shape remains consistent with respect to its distance to the camera.

**Efficient differentiable rasterizer.** To enable real-time rendering, we build our rasterizer following the 3DGS tile-based rasterizer [25], which allows for efficient backpropagation across an arbitrary number of primitives. All computations, including 3D-to-2D point projection, convex hull calculation, line segment definition, and indicator function implementation—are fully differentiable and executed within our custom CUDA kernels to maximize efficiency and rendering speed. During rendering, we rasterize the 3D shape into the target view using  $\alpha$ -blending. For a given camera pose  $\theta$  and  $N$  smooth convexes to render each pixel  $\mathbf{q}$ , we order the  $N$  convexes by sorting them according to increasing distance defined from the camera to their centers, and compute the color value for each pixel  $\mathbf{q}$ :

$$C(\mathbf{q}) = \sum_{n=1}^N \mathbf{c}_n o_n I(\mathbf{q}) \left( \prod_{i=1}^{n-1} (1 - o_i I(\mathbf{q})) \right), \quad (5)$$

where  $\mathbf{c}_n$  is the color of the  $n$ -th smooth convex, stored as spherical harmonics and converted to color based on the pose  $\theta$ ,  $o_n$  the opacity of the  $n$ -th shape, and  $I(\mathbf{q})$  is the indicator function adapted to our case from Eq. (3).

### 3.3. Optimization

**Initialization and losses.** We optimize the position of each point set in 3D,  $\delta$  and  $\sigma$  parameters, the opacity  $o$ , and the spherical harmonic color coefficients  $\mathbf{c}$ . To constrain opacity within the range  $[0, 1]$ , we apply a sigmoid activation function. For  $\delta$  and  $\sigma$ , we use an exponential activation to

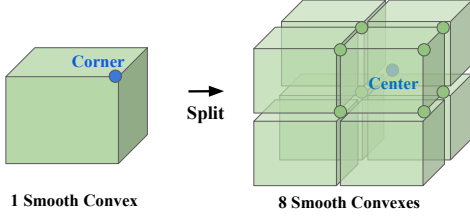


Figure 5. **Adaptive Convex Densification Scheme.** We divide each convex, here exemplified with  $K = 8$  points, into as many scaled-down occurrences of the convex, centered at the initial points, each with reduced opacity.

ensure their values remain positive. We initialize each convex shape with a set of points uniformly distributed around a sphere centered in the points of the point cloud, using the Fibonacci sphere algorithm. We define the size of each convex shape based on its average distance to the three nearest smooth convexes. This results in smaller smooth convexes in densely populated regions and larger shapes in sparser areas, allowing for adaptive scaling based on local geometry. Following the approach used in 3DGS, we apply a standard exponential decay scheduling technique for the learning rate similar to Plenoxels [13], but only for optimizing the position of the 3D points. We experimented with applying this technique to  $\delta$  and  $\sigma$  as well, but we did not observe any performance improvements. During training, we use the same regularization loss  $\mathcal{L}_m$  as in [28] to reduce the number of convexes. Our final loss function combines  $\mathcal{L}_1$  with a D-SSIM term and the loss for the mask, following [28]:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} + \beta\mathcal{L}_m, \quad (6)$$

where  $\lambda$  controls the balance between  $\mathcal{L}_1$  and  $\mathcal{L}_{\text{D-SSIM}}$ . For all tests, we use  $\lambda = 0.2$  as in 3DGS [25] and  $\beta = 0.0005$ .

**Adaptive convex shape refinement.** The initial set of smooth convexes is generated from a sparse point set obtained through Structure-from-Motion. Since this initial number of smooth convexes is insufficient to accurately represent complex scenes, we employ an adaptive control mechanism to add smooth convexes dynamically. In 3DGS, additional Gaussians are introduced by splitting or cloning those with large view-space positional gradients. However, in 3DCS, positional gradients do not consistently correspond to regions with missing geometric features (“under-reconstruction”) or areas where convexes over-represent large portions of the scene (“over-reconstruction”). Instead, we observe that 3DCS exhibits a large sharpness  $\sigma$  loss in both under-reconstructed and over-reconstructed regions. Rather than cloning and differentiating between small and large shapes, we consistently split our smooth convexes. Instead of splitting a smooth convex into just two new convexes, we split it directly into  $K$  new convexes. Each new convex shape is scaled down, and the centers of these new

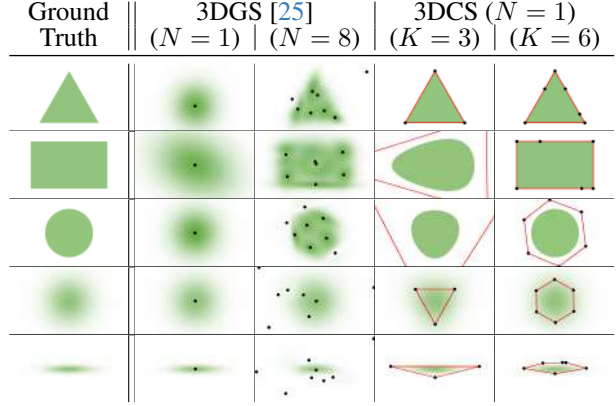


Figure 6. **Reconstruction of Simple Shapes with Primitives.** Smooth convex primitives reconstruct simple shapes better than Gaussians, as they can create sharper geometric boundaries. For 3DCS, the red lines describe the convex hull, whereas the black dots represent the point set. For 3DGS, the black dots represent the Gaussian centers.

convexes correspond to the  $K$  points defining the initial convex shape. By placing the centers of the new convexes at the 3D points of the initial convex shape, we ensure that the new shapes collectively cover the volume of the original convex to maintain the overall completeness of the 3D representation (see Fig. 5 for illustration). To encourage the formation of denser volumetric shapes during optimization, we increase the sharpness  $\sigma$  throughout splitting, while keeping the smoothness  $\delta$  the same. We prune convexes that have an opacity lower than a predefined threshold and convexes that are too large. More details are provided in the experimental setup Sec. 4.2.

## 4. Experiments

We first evaluate 3D Convex Splatting (3DCS) with synthetic experiments to showcase its superior shape representation over Gaussian primitives. Then, we describe the real-world setup, including tasks, datasets, baselines, metrics, and implementation, followed by results on novel view synthesis and an ablation study.

### 4.1. Experiments on Synthetic Data

Figure 6 compares the representation capabilities of using 1 or 8 Gaussian primitives against using a single convex shape defined by 3 or 6 points. The results demonstrate that smooth convexes effectively approximate a wide range of shapes, including both polyhedra and Gaussians, while requiring fewer primitives for accurate representation.

### 4.2. Experimental Setup

**Datasets.** To evaluate 3DCS on real-world novel view synthesis, we use the same datasets as 3DGS [25]. This



| Dataset<br>Method—Metric | Tanks&Temples      |                   |                   |                    |                  |                  | Deep Blending      |                   |                   |                    |                  |                  | Mip-NeRF360 Dataset |                   |                   |                    |                  |                  |
|--------------------------|--------------------|-------------------|-------------------|--------------------|------------------|------------------|--------------------|-------------------|-------------------|--------------------|------------------|------------------|---------------------|-------------------|-------------------|--------------------|------------------|------------------|
|                          | LPIPS <sup>↓</sup> | PSNR <sup>↑</sup> | SSIM <sup>↑</sup> | Train <sup>↓</sup> | FPS <sup>↑</sup> | Mem <sup>↓</sup> | LPIPS <sup>↓</sup> | PSNR <sup>↑</sup> | SSIM <sup>↑</sup> | Train <sup>↓</sup> | FPS <sup>↑</sup> | Mem <sup>↓</sup> | LPIPS <sup>↓</sup>  | PSNR <sup>↑</sup> | SSIM <sup>↑</sup> | Train <sup>↓</sup> | FPS <sup>↑</sup> | Mem <sup>↓</sup> |
| Instant-NGP[37]          | 0.305              | 21.92             | 0.745             | 7m                 | 14.4             | 48MB             | 0.390              | 24.96             | 0.817             | 8m                 | 2.79             | 48MB             | 0.331               | 25.59             | 0.699             | 7.5m               | 9.43             | 48MB             |
| ZipNeRF[4]               | -                  | -                 | -                 | -                  | -                | -                | -                  | -                 | -                 | -                  | -                | -                | 0.189               | 28.54             | 0.828             | 5h                 | 0.184            | 569MB            |
| Mip-NeRF360[3]           | 0.257              | 22.22             | 0.759             | 48h                | 0.14             | 8.6MB            | 0.245              | 29.40             | 0.901             | 48h                | 0.09             | 8.6MB            | 0.237               | 27.69             | 0.792             | 48h                | 0.06             | 8.6MB            |
| 3DGS[25]                 | 0.183              | 23.14             | 0.841             | 26m                | 154              | 411MB            | 0.243              | 29.41             | 0.903             | 36m                | 137              | 676MB            | 0.214               | 27.21             | 0.815             | 42m                | 134              | 734MB            |
| GES [18]                 | 0.198              | 23.35             | 0.836             | 21m                | 210              | 222MB            | 0.252              | 29.68             | 0.901             | 30m                | 160              | 399MB            | 0.250               | 26.91             | 0.794             | 32m                | 186              | 377MB            |
| 2DGS[21]†                | 0.212              | 23.13             | 0.831             | 14m                | 122              | 200MB            | 0.257              | 29.50             | 0.902             | 28m                | 76               | 353MB            | 0.252               | 27.18             | 0.808             | 29m                | 64               | 484MB            |
| <b>3DCS (light)</b>      | 0.170              | 23.71             | 0.842             | 46m                | 40               | 83 MB            | 0.245              | 29.61             | 0.901             | 84m                | 30               | 110 MB           | 0.266               | 26.66             | 0.769             | 53m                | 47               | 77MB             |
| <b>3DCS</b>              | 0.157              | 23.95             | 0.851             | 60m                | 33               | 282MB            | 0.237              | 29.81             | 0.902             | 71m                | 30               | 332 MB           | 0.207               | 27.29             | 0.802             | 87m                | 25               | 666MB            |

Table 1. **Comparative Analysis of Novel View Synthesis Methods.** We conduct a quantitative comparison of our 3DCS method across three datasets. 3DCS achieves higher-quality results in novel view synthesis with reduced memory consumption, all while achieving fast rendering performance. No codebooks or post-processing compression [28, 39] are used to reduce the size of any of the methods. The best performances are shown in red and the second best in orange. † indicates reproduced results.

includes two scenes from Deep Blending (DB) [19], two scenes from Tanks and Temples (T&T) [27], and all scenes from the Mip-NeRF360 dataset [3].

**Baselines.** We compare our 3D Convex Splatting method against methods that use three other basic primitives for novel-view synthesis: 3D Gaussians [25], Generalized Exponential Functions (GES) [18], and 2D Gaussians [21]. While many follow-up studies have built upon 3DGS and introduced various enhancements, we focus on the basic primitives for comparison. This choice ensures that the evaluation is based on the core principles of each approach. We compare our method to ZipNeRF [4] and Mip-NeRF360[3] for quality and Instant-NGP [37] for speed.

**Metrics.** We assess visual quality using SSIM, PSNR, and LPIPS and report training time, rendering speed, and memory usage for a detailed comparison with other methods.

**Implementation details of 3DCS.** For each experiment, we initialize the number of points per convex shape to  $K = 6$  and a spherical harmonic degree of 3, resulting in a total of 69 parameters per convex shape. For each 3D Gaussian, 59 parameters are needed. In the following sections, we compare two variants of 3DCS: a best-performing model and a lightweight variant. Our best model employs different hyperparameters for indoor and outdoor scenes, whereas the lightweight model uses a unified set of parameters. For our light version, we increase the threshold criterion for densifying convexes, effectively reducing the number of shapes. Furthermore, we store the 3D convex parameters with 32-bit precision for the high-quality model and 16-bit precision for the lightweight version. A list of hyperparameters can be found in the *Supplementary Material*. Notably, no compression methods are applied to reduce memory usage.

### 4.3. Real-world Novel View Synthesis

**Main results.** Table 1 presents the quantitative results. As can be seen, our 3DCS method consistently matches or surpasses the rendering quality of existing methods across all evaluated datasets. Specifically, 3DCS outperforms 3DGS, GES and 2DGS in most metrics on the T&T and DB

datasets, while also achieving the second highest PSNR and lowest LPIPS on the Mip-NeRF360 dataset. 3DCS effectively balances memory usage and training time, sitting in between the ones of Mip-NeRF360 and 3DGS. Particularly, while it consumes more memory than Mip-NeRF360, it significantly reduces training time, requiring only 63 minutes compared to the 48 hours of Mip-NeRF360. Moreover, it delivers better visual quality, especially on the T&T dataset, where 3DCS demonstrates a notable performance advantage of over 1.73 PSNR compared to Mip-NeRF360. In comparison with 3DGS, 3DCS exhibits a slightly longer training time and lower rendering speed. Yet, 3DCS still operates within real-time rendering capabilities. Thanks to its greater adaptability, 3DCS efficiently utilizes only 70% of the memory needed by 3DGS, while achieving higher visual quality. Figure 7 strikes a qualitative comparison between 3DCS, 3DGS and 2DGS. Notably, our method achieves sharp and detailed rendering even in challenging regions, *e.g.* the background in the *Train* scene. In contrast, Gaussian primitives tend to oversmooth areas, resulting in images with pronounced artifacts, as observed in the *Flower*, *Bicycle*, and *Truck* scenes. The convex-based approach, however, produces results that closely align with the ground truth, showcasing higher fidelity and a superior ability to realistically represent 3D environments.

3DCS light outperforms 3DGS and GES on the T&T and DP dataset, while using less memory. Figure 8 contains a visual comparison between 3DGS and light 3DCS.

**Indoor versus outdoor scenes.** Table 2 presents a comparative analysis of indoor versus outdoor scenes from the Mip-NeRF360 dataset. Indoor scenes consist of structured, flat surfaces with hard edges, while outdoor scenes generally have more unstructured surfaces. This structural difference advantages convex shapes, which are better suited for capturing the geometric characteristics of indoor environments. In fact, for indoor scenes, it can be seen that 3DCS significantly outperforms 3DGS with an improvement of 0.9 PSNR, 0.007 SSIM, and 0.023 LPIPS, surpassing all other Gaussian-based methods. Moreover, 3DCS achieves

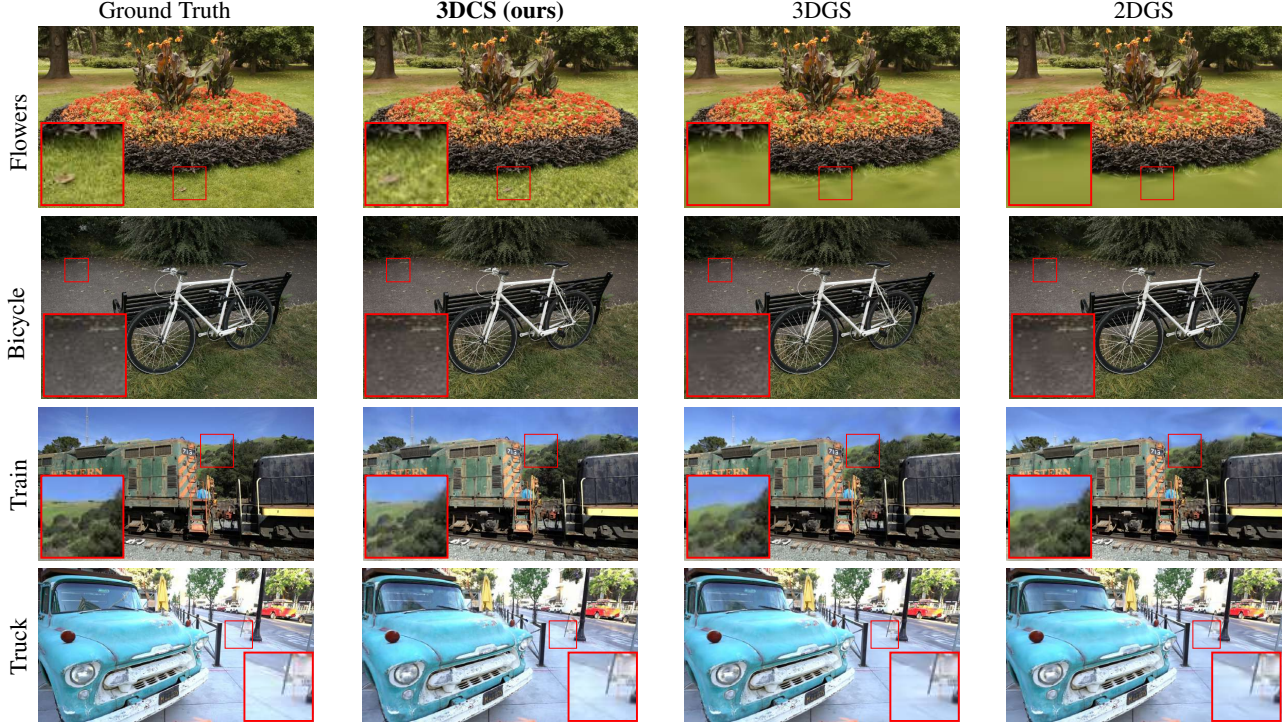


Figure 7. **Qualitative Comparison between 3DCS, 3DGS and 2DGS.** Our 3DCS captures finer details and provides a more accurate approximation of real-world scenes compared to Gaussian splatting methods, which often produce blurrier results.

|             | Outdoor Scene |        |        | Indoor scene |        |        |
|-------------|---------------|--------|--------|--------------|--------|--------|
|             | LPIPS ↓       | PSNR ↑ | SSIM ↑ | LPIPS ↓      | PSNR ↑ | SSIM ↑ |
| MipNeRF360  | 0.283         | 24.47  | 0.691  | 0.180        | 31.72  | 0.917  |
| 3DGS        | 0.234         | 24.64  | 0.731  | 0.189        | 30.41  | 0.920  |
| GES         | 0.243         | 24.46  | 0.724  | 0.189        | 30.85  | 0.922  |
| 2DGS        | 0.246         | 24.34  | 0.717  | 0.195        | 30.40  | 0.916  |
| 3DCS (ours) | 0.238         | 24.07  | 0.700  | 0.166        | 31.33  | 0.927  |

Table 2. **Quantitative Results on Mip-NeRF 360 [3] Dataset.** We evaluate our method on both indoor and outdoor scenes, demonstrating substantial performance improvements over all 3DGS-based methods in indoor scenes and surpassing MipNeRF360 in SSIM and LPIPS metrics.



Figure 8. **Visual Comparison Between our Light Model and 3DGS.** The light model (right) shows high visual quality compared to 3DGS (left), using less than 15% of the memory.

superior results in terms of SSIM and LPIPS metrics compared to Mip-NeRF360. Even in outdoor scenes containing a lot of human-made structures—such as the *Truck* and

*Train* scenes from T&T, 3DCS substantially outperforms 3DGS, demonstrating its ability to effectively handle structured geometries. However, in outdoor scenes dominated by nature and unstructured elements like trees and vegetation, the strengths of 3DCS become less pronounced. While 3DGS and 3DCS achieve comparable LPIPS results, 3DGS achieves better PSNR and SSIM. Yet, qualitatively, we can see in Fig. 7 that 3DCS appears significantly closer to the ground truth in terms of visual quality. Specifically, in the highlighted region of the *Flower* scene, our reconstruction better represents the real grass even though the PSNR of this area is 20.17 for 3DCS and 21.65 for 3DGS. This showcases the popularly observed mismatch between PSNR and perceived visual quality. This is mainly due to the fact that PSNR is highly sensitive to pixel-level differences and, therefore, tends to favor blurrier images.

#### 4.4. Ablation Study and Discussion

We analyze the impact of densification strategies, the number of points per convex shape, and the influence of reducing the number of shapes on rendering quality.

**Densification strategy.** We evaluate the effectiveness of splitting each convex shape into new convex shapes, as described in Sec. 3.3. Specifically, we analyze the impact of dividing a convex shape defined initially by  $K = 6$  points into 2, 3, or 6 (default value) new convex shapes. For split-





Figure 9. **Ablation of densification strategy.** From left to right, we split each convex into 2, 3, or 6 new convexes.

|            | $LPIPS\downarrow$ | $PSNR\uparrow$ | $SSIM\uparrow$ | Train $\downarrow$ |
|------------|-------------------|----------------|----------------|--------------------|
| 3DCS (K=3) | 0.241             | 22.40          | 0.794          | 44m                |
| 3DCS (K=4) | 0.159             | 23.73          | 0.848          | 52m                |
| 3DCS (K=5) | 0.160             | 23.70          | 0.848          | 60m                |
| 3DCS (K=6) | <b>0.157</b>      | <b>23.90</b>   | 0.850          | 71m                |
| 3DCS (K=7) | <b>0.157</b>      | <b>23.90</b>   | <b>0.851</b>   | 73m                |

Table 3. **Ablation Study of the Number of Points per Convex.** We study the impact of the number of points per convex on reconstruction quality and training time on the T&T dataset. With only 4 points, our 3DCS performs better than 3DGS.

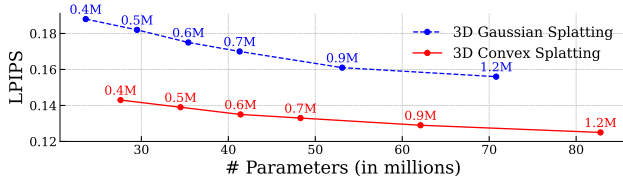


Figure 10. **# Parameters vs.  $LPIPS\downarrow$  (Truck scene).** The number of primitives is indicated for each point. 3DCS achieves a better regime than 3DGS for a comparable number of parameters.

ting into 2 or 3 shapes, the new convex shapes are centered on 2 or 3 randomly selected points from the original convex shape. As can be seen in Fig. 9, splitting a convex shape into more shapes results in higher visual quality, particularly in capturing finer details in the background.

**Number of points per shape.** Increasing  $K$  provides greater flexibility in representing convex shapes but comes at the cost of longer training times. Notably, the case of 3 points represents a special configuration, resulting in a non-volumetric triangle in 3D space, analogous to the 2D Gaussian Splatting approach [21] in terms of its dimensionality constraints. Table 3 shows that using  $K \geq 4$  points per convex consistently outperforms 3DGS. However, increasing beyond 6 points has no significant performance gain.

**Influence of less primitives on rendering quality.** Figure 10 shows how LPIPS on the T&T dataset changes with the number of primitives and parameters. Notably, our



Figure 11. **3DCS vs. 3DGS with fewer shapes.** Convex splatting (left) can decompose objects into meaningful convex shapes, enabling a realistic and compact 3D representation of the world.

method 3DCS consistently outperforms 3DGS.

**Physically meaningful 3D representations.** Figure 11 visually compares the performance of 3DGS and 3DCS when the number of primitives  $N$  is reduced. With fewer shapes, 3DGS produces blurry images due to the limited flexibility of 3D Gaussians, which struggle to form visually meaningful representations of real objects. In contrast, 3DCS preserves image clarity by effectively decomposing objects into convex shapes. 3DCS represents the leaves on the stump as either a single convex shape or a collection of convex shapes, with each shape capturing a physically meaningful part of the real-world object. Ultimately, 3DCS offers a significant advantage by delivering more physically meaningful 3D representations. By leveraging the adaptability of convex shapes, we bridge the gap between visual accuracy and interpretability, enabling high-quality, geometrically meaningful 3D modeling.

## 5. Conclusion

We introduce 3D Convex Splatting (3DCS), a novel method for radiance field rendering that leverages 3D smooth convexes to achieve high-quality novel view synthesis. Particularly, our method overcomes the limitations of 3D Gaussian Splatting, delivering denser representations with fewer primitives and parameters. Furthermore, 3DCS demonstrates substantial improvements on the novel view synthesis task. By combining the adaptability of convex shapes with the efficiency of primitive-based radiance field rendering, 3DCS achieves high-quality, real-time, and flexible radiance field reconstruction. We envision this new primitive to set the ground for further research in the field.

**Acknowledgments.** J. Held, A. Deliege and A. Cioppa are funded by the F.R.S.-FNRS. The research reported in this publication was supported by funding from KAUST Center of Excellence on GenAI, under award number 5940. This work was also supported by KAUST Ibn Rushd Postdoc Fellowship program. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.

## References

- [1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. *Commun. ACM*, 54(10):105–112, 2011. [1](#), [2](#)
- [2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5835–5844, Montréal, Can., 2021. [2](#)
- [3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5460–5469, New Orleans, LA, USA, 2022. [2](#), [6](#), [7](#)
- [4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 19640–19648, Paris, Fr., 2023. [2](#), [6](#)
- [5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini de Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16102–16112, New Orleans, LA, USA, 2022. [2](#)
- [6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensorRF: Tensorial radiance fields. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 333–350. Springer Nat. Switz., 2022. [2](#)
- [7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-net: Generating compact meshes via binary space partitioning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 42–51, Seattle, WA, USA, 2020. [3](#)
- [8] John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer New York, 1999. [2](#)
- [9] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable convex decomposition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 31–41, Seattle, WA, USA, 2020. [1](#), [2](#), [3](#), [4](#)
- [10] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *ACM Int. Conf. Comput. Graph. Interact. Tech. (SIGGRAPH)*, pages 65–74, Atlanta, Georgia, USA, 1988. [2](#)
- [11] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4970–4980, Vancouver, Can., 2023. [2](#)
- [12] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Eur. Conf. Comput. Vis. (ECCV)*, pages 563–578. Springer Berl. Heidelb., 1992. [1](#), [2](#)
- [13] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5491–5500, New Orleans, LA, USA, 2022. [2](#), [5](#)
- [14] Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4): 132–133, 1972. [3](#), [4](#)
- [15] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kauffmann Publ. Inc., 2007. [2](#)
- [16] Thomas C. Hales. The sphere packing problem. *J. Comput. Appl. Math.*, 44(1):41–76, 1992. [2](#)
- [17] Abdullah Hamdi, Bernard Ghanem, and Matthias Nießner. SPARF: Large-scale learning of 3D sparse radiance fields from few input images. In *IEEE/CVF Int. Conf. Comput. Vis. Work. (ICCV Work.)*, pages 2922–2932, Paris, Fr., 2023. [1](#)
- [18] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. GES: Generalized exponential splatting for efficient radiance field rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 19812–19822, Seattle, WA, USA, 2024. [3](#), [6](#)
- [19] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6):1–15, 2018. [6](#)
- [20] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5855–5864, Montréal, Can., 2021. [2](#)
- [21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conf. Pap.*, pages 1–11, Denver, CO, USA, 2024. [3](#), [6](#), [8](#)
- [22] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5865–5874, Montréal, Can., 2021. [2](#)
- [23] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH Comput. Graph.*, 18(3): 165–174, 1984. [2](#)
- [24] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3907–3916, Salt Lake City, UT, USA, 2018. [2](#)
- [25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):1–14, 2023. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#)
- [26] Mijeong Kim, Seonguk Seo, and Bohyung Han. InfoNeRF: Ray entropy minimization for few-shot neural volume rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 12902–12911, New Orleans, LA, USA, 2022. [2](#)
- [27] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):1–13, 2017. [6](#)

- [28] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21719–21728, Seattle, WA, USA, 2024. 5, 6
- [29] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990. 2
- [30] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.*, 37(6):1–11, 2018. 2
- [31] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 7707–7716, Seoul, South Korea, 2019.
- [32] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 154–169. Springer Int. Publ., 2014. 2
- [33] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *Int. Conf. 3D Vis. (3DV)*, pages 800–809, Davos, Switzerland, 2024. 3
- [34] Jinjie Mai, Wenxuan Zhu, Sara Rojas, Jesus Zarzar, Abdullah Hamdi, Guocheng Qian, Bing Li, Silvio Giancola, and Bernard Ghanem. TrackNeRF: Bundle adjusting NeRF from sparse and noisy views via feature tracks. In *Eur. Conf. Comput. Vis. (ECCV)*, Milan, Italy, 2024. 1
- [35] Nelson Max. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 1(2):99–108, 1995. 2
- [36] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 405–421. Springer Int. Publ., 2020. 1, 2
- [37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):1–15, 2022. 2, 6
- [38] Richard A. Newcombe, Andrew Fitzgibbon, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, and Steve Hodges. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. Mix. Augment. Real.*, pages 127–136, Basel, Switzerland, 2011. 1
- [39] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian splatting for accelerated novel view synthesis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10349–10358, Seattle, WA, USA, 2024. 6
- [40] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10336–10345, Long Beach, CA, USA, 2019. 3
- [41] Franco P. Preparata and Sie June Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2):87–93, 1977. 3
- [42] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 14315–14325, Montréal, Can., 2021. 2
- [43] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph.*, 42(4):1–12, 2023. 2
- [44] Daxuan Ren, Haiyi Mei, Hezi Shi, Jianmin Zheng, Jianfei Cai, and Lei Yang. Differentiable convex polyhedra optimization from multi-view images. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 251–269. Springer Nat. Switz., 2024. 3
- [45] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5449–5459, New Orleans, LA, USA, 2022. 2
- [46] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5481–5490, New Orleans, LA, USA, 2022. 2
- [47] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. BakedSDF: Meshing neural SDFs for real-time view synthesis. In *ACM SIGGRAPH Conf. Proc.*, pages 1–9, Los Angeles, CA, USA, 2023. 2
- [48] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4576–4585, Nashville, TN, USA, 2021. 2
- [49] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3D Gaussian splatting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 19447–19456, Seattle, WA, USA, 2024. 3

# 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes

## Supplementary Material

### 6. Hyperparameters & Initialization

We initialize each convex shape using the COLMAP points as centers and uniformly distribute *new* points around them with the Fibonacci sphere algorithm to ensure an even distribution. The initial sphere radius is set to 1.2 times the mean distance to the three nearest neighbors in the point cloud. This adaptive initialization ensures that dense 3D regions contain many small convex shapes, while sparser regions are represented by larger convexes. The initial values for the smoothness parameter  $\delta$  and sharpness parameter  $\sigma$  are set to 0.1 and  $9.5e^{-4}$ , respectively. These values are chosen to initially produce more diffuse shapes, as this configuration was empirically found to result in better performance during optimization. The initial opacity is set to 0.1. A detailed list of the hyperparameters can be found in Tab. 4.

### 7. Methodology Details

**2D equations.** We define the 2D convex indicator function for our convex hull by adapting the smooth convex representation from 3D to 2D, utilizing the equations introduced in Sec. 3.1. Specifically, we define  $\phi(\mathbf{q})$  and  $I(\mathbf{q})$  as in Eqs. (2) and (3), but substitute the 3D point  $\mathbf{p}$  with the 2D point  $\mathbf{q}$  and replace the planes delimiting the 3D convex hull with the lines that delimit the resulting 2D convex hull.

$$\phi(\mathbf{q}) = \log \left( \sum_{t=1}^T \exp(d \delta L_j(\mathbf{q})) \right), \quad (7)$$

where  $T$  is the total number of lines delimiting the 2D convex shape.

The indicator function  $I(\mathbf{q})$  of the smooth convex is then defined by:

$$I(\mathbf{q}) = \text{Sigmoid}(-d \sigma \phi(\mathbf{q})). \quad (8)$$

### 8. Ablation Study

**Perspective-Aware Scaling in 2D Projection.** To incorporate perspective effects in the 2D projection, we scale  $\delta$  and  $\sigma$  by the distance  $d$ , ensuring that the appearance of the convex shape remains consistent regardless of its distance from the camera. Table 5 provides an ablation study demonstrating the necessity of scaling  $\delta$  and  $\sigma$  as well as analyzing the impact of the scaling magnitude.

### 9. More Results

**Experiments on Synthetic Data.** To highlight the effectiveness of our smooth convexes for rendering different shapes, we designed four 2D synthetic examples, where the optimization is performed in 2D, with a single image as ground truth. Figure 12 illustrates the optimization process of our smooth convexes on those four distinct shapes during training. Our convex shapes are highly versatile and capable of approximating a wide range of different shapes.

**Real-world Novel View Synthesis.** Figure 13 shows additional qualitative results, highlighting the capabilities of 3D Convex Splatting compared to 3D Gaussians and 2D Gaussians. The inherent softness of Gaussian primitives often leads to blurrier images and noticeable artifact-prone regions. While PSNR favors such blurrier images due to imprecise image alignment, they lack high-quality detail. Compared to Gaussian, 3D Convex Splatting does not produce any blurry areas and often results in a rendering much closer to the ground truth. For instance, in the *Bicycle* scene, Gaussian methods produce blurry artifacts on the street and in the grass, whereas 3D Convex Splatting achieves a result that closely matches the ground truth. Tables 6 to 11 show the complete quantitative results for each scene. 3D Convex Splatting outperforms 3DGS, 2DGS, and GES across all metrics on indoor scenes, the Deep Blending dataset, and the Tanks & Temples dataset.

**Additional Visualizations.** Figure 14 shows the depth maps obtained using 3D convexes vs. Gaussians, illustrating our better geometric quality, producing fewer artifacts and smoother depths. Furthermore, Fig. 15 shows the representation of a wall. It confirms that convexes generate denser and more physically accurate wall representations, whereas Gaussians produce more diffused shapes.



Table 4. Hyperparameters for Convex Splatting

| Parameter                     | Indoor           |                  | Outdoor          |                  |
|-------------------------------|------------------|------------------|------------------|------------------|
|                               | Normal           | Light            | Normal           | Light            |
| Number of training iterations |                  | 30,000           |                  |                  |
| position_lr_delay_mult        |                  | 0.01             |                  |                  |
| position_lr_max_steps         |                  | 30,000           |                  |                  |
| feature_lr                    |                  | 0.0025           |                  |                  |
| opacity_lr                    |                  | 0.01             |                  |                  |
| lambda_dssim                  |                  | 0.2              |                  |                  |
| densification_interval        |                  | 200              |                  |                  |
| densify_from_iter             |                  | 500              |                  |                  |
| opacity_reset                 |                  | 0.2              |                  |                  |
| opacity_reset_interval        |                  | 3000             |                  |                  |
| remove_size_threshold         |                  | 0.3              |                  |                  |
| min_opacity                   |                  | 0.03             |                  |                  |
| lr_mask                       |                  | 0.01             |                  |                  |
| lr_delta                      |                  | 0.005            |                  |                  |
| nb_points                     |                  | 6                |                  |                  |
| convex_size                   |                  | 1.2              |                  |                  |
| set_opacity                   |                  | 0.1              |                  |                  |
| set_delta                     |                  | 0.1              |                  |                  |
| opacity_cloning               |                  | 0.5              |                  |                  |
| delta_scaling_cloning         |                  | 1                |                  |                  |
| shifting_cloning              |                  | 1                |                  |                  |
| set_sigma                     | $9 e^{-4}$       | $9.5 e^{-4}$     | $1 e^{-3}$       | $9.5 e^{-4}$     |
| densify_grad_threshold        | $6 e^{-6}$       | $2.5 e^{-5}$     | $1 e^{-6}$       | $2.5 e^{-5}$     |
| lr_sigma                      | $4.5 e^{-3}$     | $4.5 e^{-3}$     | $4 e^{-3}$       | $4.5 e^{-3}$     |
| reset_opacity_until           | 5000             | 5000             | 18000            | 5000             |
| scaling_cloning               | 0.7              | 0.63             | 0.6              | 0.63             |
| sigma_scaling_cloning         | 0.85             | 0.88             | 0.88             | 0.88             |
| mask_threshold                | $2 e^{-2}$       | $1 e^{-2}$       | $1 e^{-2}$       | $1 e^{-2}$       |
| lr_convex_points_init         | $4 e^{-4}$       | $5 e^{-4}$       | $5 e^{-4}$       | $5 e^{-4}$       |
| lr_convex_points_final        | $4 e^{-6}$       | $5 e^{-6}$       | $5 e^{-6}$       | $5 e^{-6}$       |
| densify_until_iter            | 9500             | 9000             | 9000             | 9000             |
| storage                       | 32-bit precision | 16-bit precision | 32-bit precision | 16-bit precision |

| Magnitude  | Truck | Train | DrJohnson | Playroom |
|------------|-------|-------|-----------|----------|
| 1          | 19.47 | 19.14 | 29.17     | 28.82    |
| $\sqrt{d}$ | 25.49 | 21.41 | 29.49     | 29.98    |
| $d$        | 25.65 | 22.23 | 29.54     | 30.08    |
| $d^2$      | 7.08  | 8.91  | 8.42      | 8.99     |

Table 5. **Perspective-Aware Scaling in 2D Projection.** We evaluate the PSNR under varying scaling magnitudes.

|      | Truck | Train | DrJohnson | Playroom |
|------|-------|-------|-----------|----------|
| 3DGS | 0.148 | 0.218 | 0.244     | 0.241    |
| 2DGS | 0.173 | 0.251 | 0.257     | 0.257    |
| GES  | 0.162 | 0.232 | 0.249     | 0.252    |
| 3DCS | 0.125 | 0.187 | 0.238     | 0.237    |

Table 6. LPIPS score for T&amp;T and DB datasets.

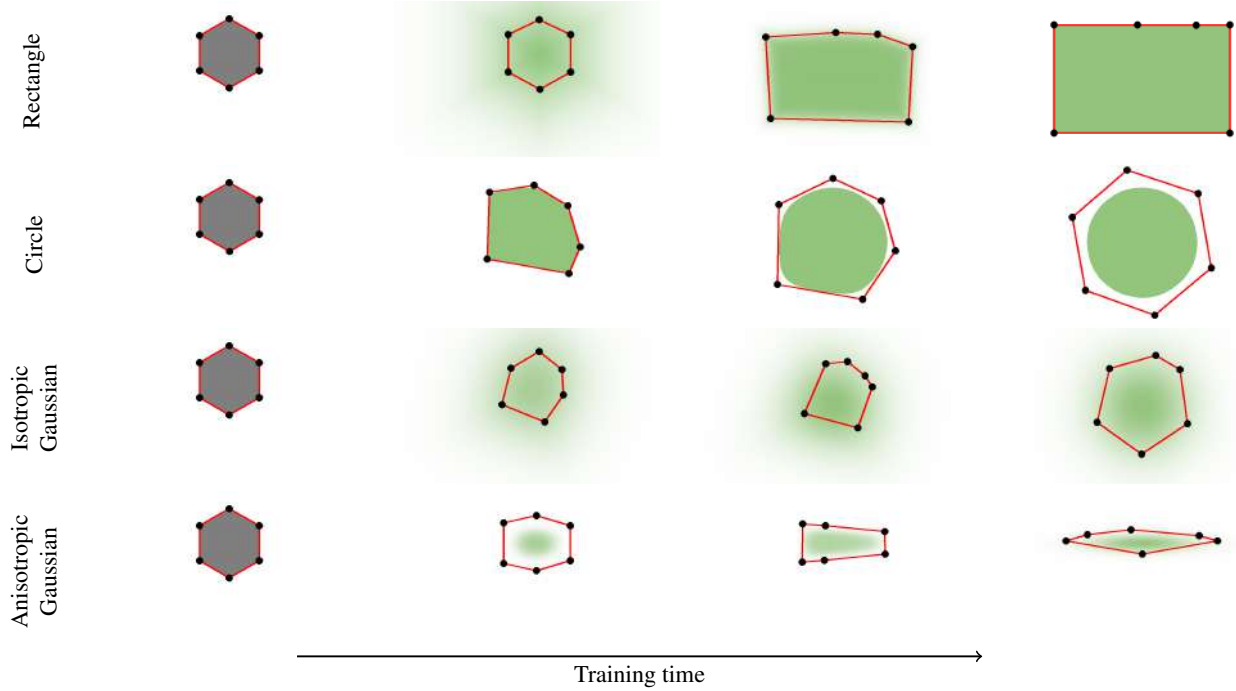


Figure 12. Smooth convexes can represent a wide variety of shapes, whether hard or soft, dense or diffuse. They effectively approximate diverse geometries, including both polyhedra and Gaussians, while requiring fewer primitives for accurate representation. The red lines describe the convex hull, whereas the black dots represent the point set.

|      | Truck | Train | DrJohnson | Playroom |
|------|-------|-------|-----------|----------|
| 3DGS | 25.18 | 21.09 | 28.76     | 30.04    |
| 2DGS | 25.12 | 21.14 | 28.95     | 30.05    |
| GES  | 25.07 | 21.75 | 29.24     | 30.06    |
| 3DCS | 25.65 | 22.23 | 29.54     | 30.08    |

Table 7. PSNR score for T&T and DB datasets.

|      | Bicycle | Flowers | Garden | Stump | Treehill | Room  | Counter | Kitchen | Bonsai |
|------|---------|---------|--------|-------|----------|-------|---------|---------|--------|
| 3DGS | 25.24   | 21.52   | 27.41  | 26.55 | 22.49    | 30.63 | 28.70   | 30.31   | 31.98  |
| 2DGS | 24.87   | 21.15   | 26.95  | 26.47 | 22.27    | 31.06 | 28.55   | 30.50   | 31.52  |
| GES  | 24.76   | 21.33   | 26.89  | 26.06 | 22.31    | 31.03 | 28.88   | 31.21   | 31.94  |
| 3DCS | 24.72   | 20.52   | 27.09  | 26.12 | 21.77    | 31.70 | 29.02   | 31.96   | 32.64  |

Table 10. PSNR score for the MipNerf360 dataset.

|      | Truck | Train | DrJohnson | Playroom |
|------|-------|-------|-----------|----------|
| 3DGS | 0.879 | 0.802 | 0.899     | 0.906    |
| 2DGS | 0.874 | 0.789 | 0.900     | 0.906    |
| GES  | 0.872 | 0.800 | 0.899     | 0.902    |
| 3DCS | 0.882 | 0.820 | 0.902     | 0.902    |

Table 8. SSIM score for T&T and DB datasets.

|      | Bicycle | Flowers | Garden | Stump | Treehill | Room  | Counter | Kitchen | Bonsai |
|------|---------|---------|--------|-------|----------|-------|---------|---------|--------|
| 3DGS | 0.771   | 0.605   | 0.868  | 0.775 | 0.638    | 0.914 | 0.905   | 0.922   | 0.938  |
| 2DGS | 0.752   | 0.588   | 0.852  | 0.765 | 0.627    | 0.912 | 0.900   | 0.919   | 0.933  |
| GES  | 0.727   | 0.600   | 0.846  | 0.768 | 0.631    | 0.910 | 0.899   | 0.920   | 0.939  |
| 3DCS | 0.737   | 0.575   | 0.850  | 0.746 | 0.595    | 0.925 | 0.909   | 0.930   | 0.945  |

Table 11. SSIM score for the MipNerf360 dataset.

|      | Bicycle | Flowers | Garden | Stump | Treehill | Room  | Counter | Kitchen | Bonsai |
|------|---------|---------|--------|-------|----------|-------|---------|---------|--------|
| 3DGS | 0.205   | 0.336   | 0.103  | 0.210 | 0.317    | 0.220 | 0.204   | 0.129   | 0.205  |
| 2DGS | 0.218   | 0.346   | 0.115  | 0.222 | 0.329    | 0.223 | 0.208   | 0.133   | 0.214  |
| GES  | 0.272   | 0.342   | 0.110  | 0.218 | 0.331    | 0.220 | 0.202   | 0.127   | 0.206  |
| 3DCS | 0.216   | 0.322   | 0.113  | 0.227 | 0.317    | 0.193 | 0.182   | 0.117   | 0.182  |

Table 9. LPIPS score for the MipNerf360 dataset.



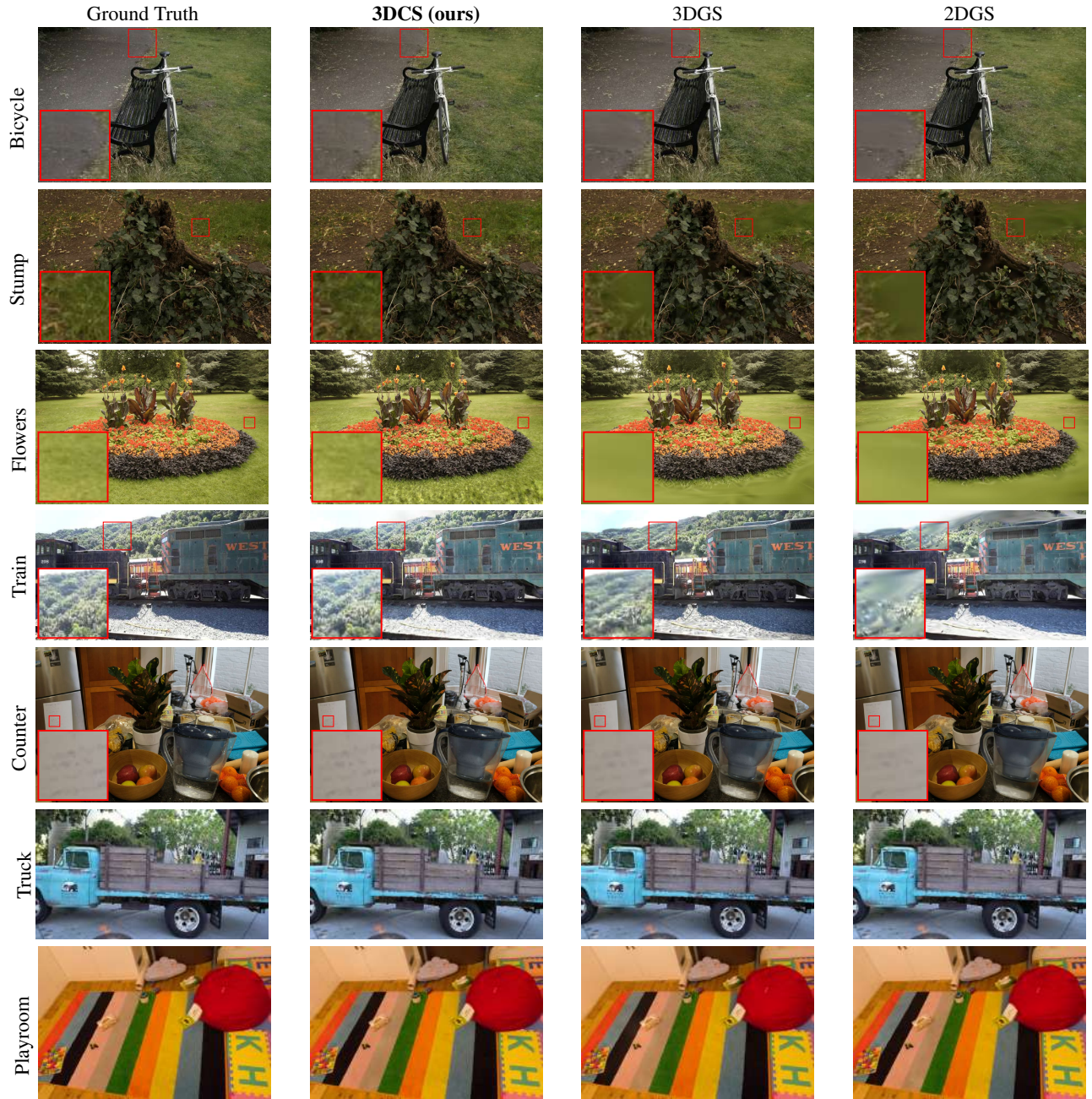


Figure 13. **Qualitative Comparison between 3DCS, 3DGS and 2DGS.** 3D Convex Splatting achieves high-quality novel view synthesis and fast rendering by representing scenes with 3D smooth convexes. In contrast, the softness of Gaussian primitives often results in blurring and loss of detail, while 3D Convex Splatting effectively captures sharp edges and fine details.

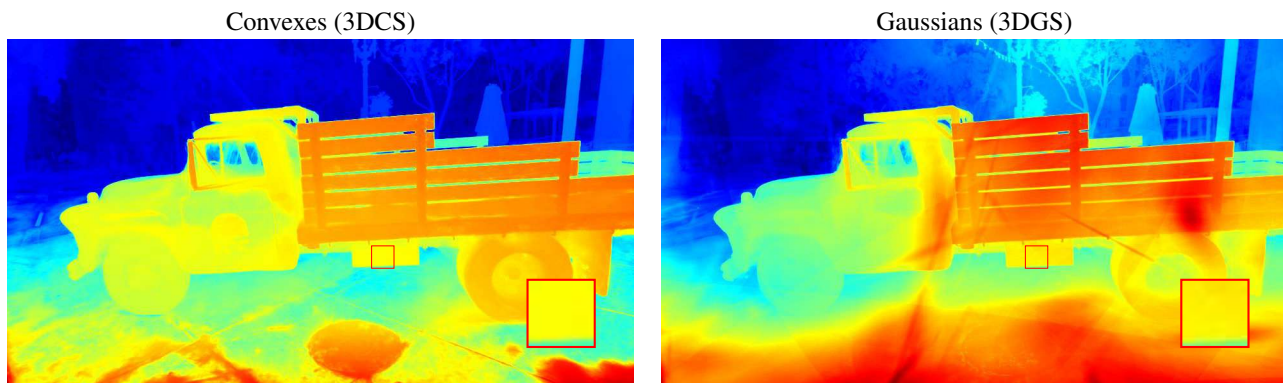


Figure 14. **Depthmap Comparison.** 3DCS generates fewer artifacts and a cleaner depth map than 3DGS.

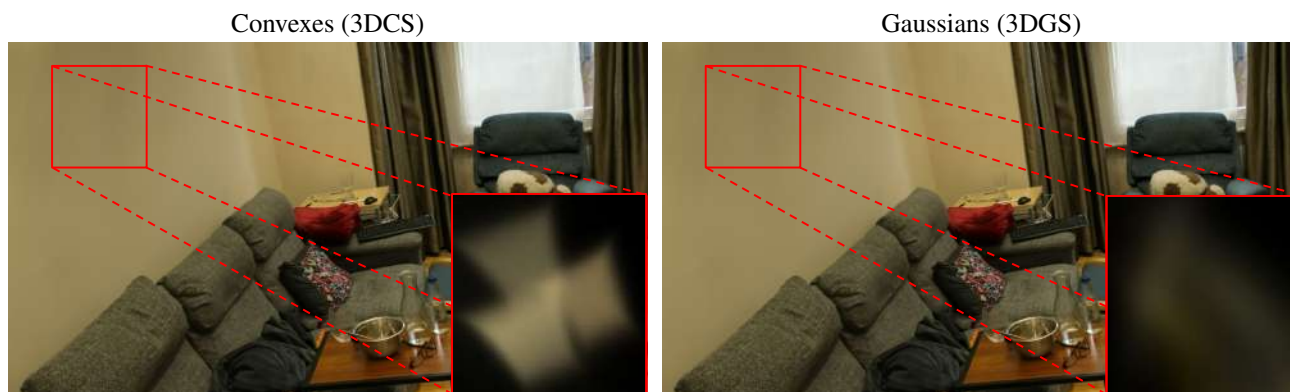


Figure 15. **Wall Approximation.** Example showing that 3DCS approximates a wall more uniformly and accurately than 3DGS.

## Part 2

---

*The Return of the Triangle*

## 4. Triangle Splatting

### 4.1. Context

3D Convex Splatting provides a foundation for exploring more expressive primitives beyond Gaussians. To design an improved primitive, it is essential to first analyze the convex formulation to understand where it can be improved.

First, 3D Convex Splatting remains a volumetric representation, which introduces inherent limitations compared to surface-based primitives. For each smooth convex, a 2D convex hull must be computed in image space to determine its projection. This additional geometric processing increases rendering complexity, resulting in higher computational cost during training and reduced frame rates at inference time.

Second, smooth convex primitives are not natively compatible with classical computer graphics pipelines, which are typically designed around explicit surface representations.

Finally, 3D Convex Splatting is not directly compatible with standard computer graphics pipelines, which are fundamentally built around rasterizing discrete surface primitives rather than volumetric shapes. This mismatch limits straightforward integration into existing rendering engines and requires additional adaptations.

These limitations motivate a return to the most fundamental primitive in computer graphics: the triangle. At first, this choice may appear straightforward, as triangles are widely used in rendering systems. However, their introduction in the context of splatting represents a conceptual shift, from volumetric representations to explicit surface modeling.

A triangle is the simplest geometric element that defines a surface in 3D space. With only three vertices, it uniquely determines a planar patch, making it the minimal building block for representing continuous surfaces. This simplicity is what enables the efficiency of modern graphics pipelines: triangles can be rasterized independently, their visibility can be resolved deterministically via depth buffering, and their regular structure allows parallelization on GPUs. In this sense, triangle-based representations are not only a practical choice, but the result of decades of co-evolution between geometry and hardware.

However, this transition also reveals deeper challenges. While the physical world is three-dimensional, graphics pipelines operate on 2D primitives projected onto

the image plane. Triangles therefore act as an interface between these domains, approximating continuous 3D geometry through a collection of discrete 2D surface elements. This dimensional reduction is non-trivial and introduces fundamental questions: how should surfaces be discretized, how should connectivity be defined, and how can complex shapes be faithfully approximated using only planar elements? This reflects a general geometric difficulty: moving between dimensions is inherently non-trivial, as structure can be lost or reinterpreted under projection.

Even simple geometric concepts illustrate these difficulties. A line, for instance, cannot be directly represented by triangles and must instead be approximated by thin surface elements. Similarly, a planar quadrilateral must be decomposed into two triangles, raising ambiguity in the choice of triangulation. These examples highlight that triangle-based representations are inherently tied to questions of topology and discretization, rather than being a neutral geometric choice.

Despite these challenges, triangles offer a decisive advantage over volumetric primitives such as Gaussians or convexes: they provide an explicit, structured surface representation that is directly compatible with rasterization pipelines. This compatibility enables the use of highly optimized GPU hardware, including depth buffering, early-z rejection, and occlusion culling, which are not accessible to volumetric splatting approaches relying on sorting and alpha blending.

However, triangles introduce a new challenge due to their inherently discrete and non-differentiable nature. Their sharp boundaries prevent direct gradient-based optimization. To address this limitation, we propose a novel differentiable window function that models soft triangle boundaries, enabling smooth transitions at edges and stable end-to-end gradient flow to triangle parameters during training.

Triangle Splatting therefore represents more than a simple change of primitive. It marks a step toward bridging continuous 3D scene representations with hardware-efficient graphics pipelines. While it operates on soft, unconnected triangles, it lays the foundation for template-free mesh optimization using gradient-based methods. The remaining challenge of producing a discrete, connected triangle mesh is addressed in the next chapter through MeshSplatting, where connectivity and topology are explicitly incorporated into the representation.

## 4.2. Scientific contributions

We summarize our main contributions as follows:

- (i) We propose Triangle Splatting, a novel differentiable rendering pipeline that directly optimizes triangles, bridging neural rendering and computer graphics. The resulting triangle representation can be seamlessly integrated into standard computer graphics pipelines without requiring any post-processing.

- (ii) We introduce a novel differentiable window function designed specifically for soft triangle boundaries, allowing end-to-end gradients to flow into the triangle parameters during training.
- (iii) By using triangles as the fundamental primitive, Triangle Splatting establishes a strong foundation for subsequent work toward seamless integration into traditional graphics pipelines and game engines. This also includes the follow-up method presented in Chapter 5, which directly builds upon and extends Triangle Splatting.

### 4.3. Publication and recognition

The work included in this chapter was published as:

**Jan Held\***, Renaud Vandeghen\*, Adrien Deliege, Daniel Hamdi, Abdullah Rebaï, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle Splatting for Real-Time Radiance Field Rendering. In International Conference on 3D Vision (3DV), Vancouver, Canada, 2026.

The Triangle Splatting project received significant recognition from the research community. The GitHub repository has accumulated over 1,100 stars: <https://github.com/trianglesplatting/triangle-splatting>, and several independent videos have been viewed widely, including one with over 180,000 views: <https://www.youtube.com/watch?v=FOH3NAHP9r0>.

*Author contribution.* As a lead author, I co-designed the method, implemented major components of the training pipeline, and conducted experiments and analysis. Renaud Vandeghen supported the work through numerous discussions, assisted with running and validating experiments, benchmarking concurrent work, and was responsible for all experiments in *Unity* and other mesh-splatting renderers.



# Triangle Splatting for Real-Time Radiance Field Rendering

Jan Held\*<sup>1,2</sup> Renaud Vandeghen\*<sup>1</sup> Adrien Deliege<sup>1</sup> Abdullah Hamdi<sup>4</sup>  
Silvio Giancola<sup>2</sup> Daniel Rebain<sup>3</sup> Anthony Cioppa<sup>1</sup> Bernard Ghanem<sup>2</sup>  
Andrea Vedaldi<sup>4</sup> Andrea Tagliasacchi<sup>5,6,7</sup> Marc Van Droogenbroeck<sup>1</sup>

<sup>1</sup> University of Liège <sup>2</sup> KAUST <sup>3</sup> University of British Columbia <sup>4</sup> University of Oxford  
<sup>5</sup> Simon Fraser University <sup>6</sup> University of Toronto <sup>7</sup> Google DeepMind

\* Equal contribution



Figure 1. We propose a new representation for differentiable rendering based on the most classical of 3D primitives, the *triangle*. We show how a triangle soup (*i.e.* unstructured, disconnected triangles) can be optimized effectively, generating state-of-the-art novel view synthesis images while being immediately compatible with classical rendering pipelines. The figure shows the final rendered output (left), a visualization of soft blending (middle), and the rendering of a random subset of triangles to highlight their structure (right).

## Abstract

The field of computer graphics was revolutionized by models such as NeRF and 3D Gaussian Splatting, displacing triangles as the dominant representation for photogrammetry. In this paper, we argue for a triangle comeback. We develop a differentiable renderer that directly optimizes triangles via end-to-end gradients. We achieve this by rendering each triangle as differentiable splats, combining the efficiency of triangles with the adaptive density of representations based on independent primitives. Compared to popular 2D and 3D Gaussian Splatting methods, our approach achieves competitive rendering and convergence speed, and demonstrates high visual quality. On the Mip-

NeRF360 dataset, our method outperforms concurrent non-volumetric primitives in visual fidelity and achieves higher perceptual quality than the state-of-the-art Zip-NeRF on indoor scenes. Triangles are simple, compatible with standard graphics stacks and GPU hardware, and highly efficient. Our results highlight the efficiency and effectiveness of triangle-based representations for high-quality novel view synthesis. Triangles bring us closer to mesh-based optimization by combining classical computer graphics with modern differentiable rendering frameworks. The project page is <https://trianglesplatting.github.io/>



## 1. Introduction

One of the enduring challenges in 3D vision and graphics is identifying a truly *universal* primitive for representing 3D content in a differentiable form, enabling gradient-based optimization of geometry and appearance. Despite extensive research, no single data structure has emerged as a silver bullet. Instead, researchers have explored a variety of approaches, including neural fields [28], hash tables [29], convex primitives [7, 14], and Gaussians [20], among others. Conversely, in conventional graphics pipelines, the triangle remains the undisputed workhorse. Game engines and other real-time systems primarily rely on triangles, as GPUs feature dedicated hardware pipelines for ultra-efficient triangle processing and rendering. Although other primitives exist (*e.g.*, quads in 2D or tetrahedra in 3D), they can always be subdivided into triangles. Moreover, surface reconstruction in 3D vision and graphics predominantly relies on triangle meshes to represent continuous, watertight geometry in an efficient, renderable form [19].

Despite their ubiquity, triangles are difficult to optimize in differentiable frameworks due to their discrete nature. Early attempts at differentiable optimization softened the non-differentiable occlusion at polygon edges, enabling gradients from image loss to flow into geometry and appearance parameters [18, 25]. However, these methods require a predefined mesh template, making them unsuitable when the scene’s topology is unknown a priori. As a result, they struggle to capture fine geometric details and adapt to novel structures. To address these challenges, researchers adopted volumetric primitives, such as anisotropic 3D Gaussians in 3DGS [20], which can be optimized for high-quality novel view synthesis. However, the unbounded support of Gaussians makes it difficult to define the “surface” of the representation, and their inherent smoothness hinders accurate modeling of sharp details. Surface structures can be partially restored using 2D Gaussian Splatting [15] or 3D convex polytopes [14]. Yet, a pivotal question remains:

*Can triangles themselves be optimized directly?*

Learning to optimize a “triangle soup” (*i.e.* unstructured, disconnected triangles), as shown in the right side of Figure 1, via gradient-based methods could represent a major step towards the goal of template-free mesh optimization. Such an approach leverages decades of GPU-accelerated triangle processing and the mature mesh processing literature, making it easier to integrate these techniques within differentiable rendering pipelines.

In this work, we introduce **Triangle Splatting**, a real-time differentiable renderer that splats a soup of triangles into screen space while enabling end-to-end gradient-based optimization. Triangle Splatting merges the adaptability of Gaussians with the efficiency of triangle primitives, surpassing 3D Gaussian Splatting (3DGS), 2D Gaussian Splatting (2DGS), and 3D Convex Splatting (3DCS) in visual

fidelity, training speed, and rendering throughput. The optimized triangle soup is directly compatible with any standard mesh-based renderer. To our knowledge, Triangle Splatting is the first splatting-based approach to directly optimize triangles for novel-view synthesis and 3D reconstruction, delivering state-of-the-art results while bridging classical rendering pipelines with modern differentiable frameworks.

**Contributions.** (i) We propose Triangle Splatting, a novel approach that directly optimizes triangles, bridging computer graphics and radiance fields. (ii) We introduce a differentiable window function for soft triangle boundaries, enabling effective gradient flow. (iii) Triangle Splatting delivers high visual quality and rendering speed, surpassing Zip-NeRF on indoor scenes in perceptual quality (LPIPS) and structural similarity (SSIM). (iv) The optimized triangles are compatible with standard mesh-based renderers, enabling seamless integration into traditional graphics pipelines.

## 2. Related work

Neural radiance fields have become the de-facto standard for image-based 3D reconstruction [27]. A large body of work has since addressed NeRF’s slow training and rendering by introducing multi-resolution grids or hybrid representations [5, 9, 23, 29], or baking procedures for real-time playback [6, 13, 31]. Improvements in robustness include anti-aliasing [1–3], handling unbounded scenes [2], and few-shot generalization [4, 8, 16]. Despite their success, implicit fields still require costly volume integration at render time. Our Triangle Splatting sidesteps this by optimizing *explicit* triangles that are traced once per pixel, leading to comparable fidelity but orders-of-magnitude faster rendering. For example, our triangles render ten times faster than Instant-NGP [29], while matching its optimization speed and achieving significantly higher visual fidelity.

### Primitive-based differentiable rendering.

Differentiable renderers back-propagate image loss to scene parameters, enabling end-to-end optimization of explicit primitives such as points [10, 18], voxels [9], meshes [18, 25, 26], and Gaussians [20]. 3D Gaussian Splatting [20] demonstrated that millions of Gaussians can be fitted in minutes and rendered in real time. Because Gaussians have infinite support and inherently smooth fall-off, they struggle with sharp creases and watertight surfaces. Convex Splatting [14] replaced Gaussians with smooth convexes, capturing hard edges more faithfully, but at the cost of slow optimization time and larger memory footprints. Compared with Gaussian [15, 20] or Convex Splatting [14], which explored either *volumetric* (*e.g.* Gaussian, voxel) or *solid* (*e.g.* convex, tetrahedral) primitives, Triangle Splatting proposes *surface* primitives, aligning with the surface

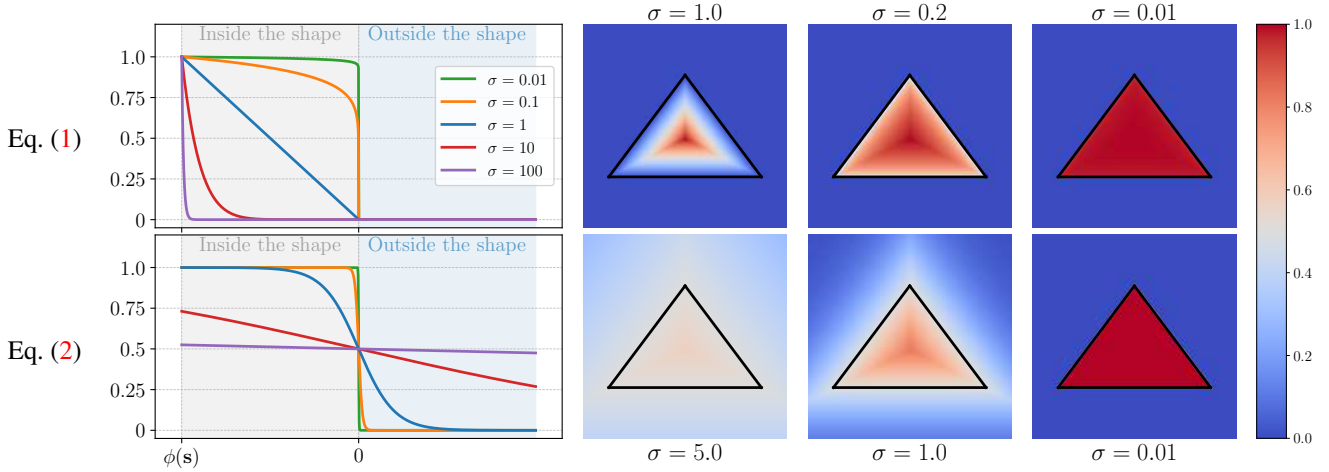


Figure 2. **Triangle window function (1D and 2D).** We visualize the window functions of prior works [7, 14] (bottom) vs. the one introduced in our paper (top) in both 1D (left) and 2D (right). We show how the window function changes as we vary the smoothness control parameter  $\sigma$ . As  $\sigma$  decreases, note that both can approximate the window function of a triangle. However, as  $\sigma$  increases, the support of Eq. (2) exceeds the footprint of the triangle, making it unsuitable for *rasterization* workloads. In the limit, Eq. (2) becomes globally supported, with a window value of 0.5 everywhere, causing every triangle to contribute to the color of every pixel in the image.

of solid objects most typically found in real-world scenes. In extensive experiments, we show that TriangleSplatting surpasses 3DGS [20], 2DGS [15], and 3DCS [14] in visual quality and speed of rendering and optimization.

### 3. Method

We address the problem of reconstructing a photorealistic 3D scene from multiple images. We propose a scene representation that enables efficient, differentiable rendering and can be directly optimized by minimizing a rendering loss. Similar to prior work, the scene is represented by a large collection of simple geometric primitives. Where 3DGS used 3D Gaussians [20], 3DCS used 3D convex hulls [14], and 2DGS used 2D Gaussians [15], we propose the simplest and most efficient primitive, *triangles*. First, Sec. 3.1 explains how these triangles are rendered on an image. Then, Sec. 3.2 describes how we adaptively prune and densify the triangle representation. Finally, Sec. 3.3 describes how to optimize the triangles’ parameters to fit the input images.

#### 3.1. Differentiable rasterization

Our primitives are 3D triangles  $T_{3D}$ , each defined by three vertices  $\mathbf{v}_i \in \mathbb{R}^3$ , a color  $\mathbf{c}$ , a smoothness parameter  $\sigma$  and an opacity  $o$ . The vertices can move freely during optimization. To render a triangle, we first project each vertex  $\mathbf{v}_i$  to the image plane using a standard pinhole camera model. The projection involves the intrinsic camera matrix  $\mathbf{K}$  and the camera pose (parameterized by rotation  $\mathbf{R}$  and translation  $\mathbf{t}$ ):  $\mathbf{q}_i = \mathbf{K}(\mathbf{R}\mathbf{v}_i + \mathbf{t})$ , with  $\mathbf{q}_i \in \mathbb{R}^2$  forming the projected triangle  $T_{2D}$  in the 2D image space. Instead of rendering the triangle as fully opaque, we weigh its influence smoothly using a window function  $I$  mapping pixels  $\mathbf{p}$  to

values in the  $[0, 1]$  range. As we discuss below, the choice of this function is of critical importance. Once the triangles are projected, the color of each image pixel  $\mathbf{p}$  is computed by accumulating contributions from all overlapping triangles, in depth order, treating the value  $I(\mathbf{p})$  as opacity. The rendering equation is the same as the one used in prior works [14, 20].

**A new window function.** We first describe how the window function  $I$  is defined, which is one of our core contributions. We start by defining the *signed distance field* (SDF)  $\phi$  of the 2D triangle in image space. This is given by:

$$\phi(\mathbf{p}) = \max_{i \in \{1, 2, 3\}} L_i(\mathbf{p}), \quad L_i(\mathbf{p}) = \mathbf{n}_i \cdot \mathbf{p} + d_i,$$

where  $\mathbf{n}_i$  are the unit normals of the triangle edges pointing outside the triangle, and  $d_i$  are offsets such that the triangle is given by the zero-level set of the function  $\phi$ . The signed distance field  $\phi$  thus takes positive values outside the triangle, negative values inside, and equals zero on its boundary. Let  $\mathbf{s} \in \mathbb{R}^2$  be the *incenter* of the projected triangle  $T_{2D}$  (i.e., the point inside the triangle with minimum signed distance). With this, we define our new window function  $I$  as:

$$I(\mathbf{p}) = \text{ReLU} \left( \frac{\phi(\mathbf{p})}{\phi(\mathbf{s})} \right)^\sigma$$

(1)

such that  $I(\mathbf{p}) \begin{cases} = 1 & \text{at the triangle incenter,} \\ = 0 & \text{at the triangle boundary,} \\ = 0 & \text{outside the triangle.} \end{cases}$

Here, the parameter  $\sigma > 0$  controls the *smoothness* of the window function.  $\phi(\mathbf{p})$  is negative inside the triangle, and  $\phi(\mathbf{s})$  is its smallest (most negative) value, so the ratio  $\phi(\mathbf{p})/\phi(\mathbf{s})$  is positive inside the triangle, equal to one at the

incenter, and equal to zero at the boundary. This formulation has three important properties: (i) there is a point (the incenter) inside the triangle where the window function obtains the maximum value of one; (ii) the window function is zero at the boundary and outside the triangle so that its support tightly fits the triangle; and (iii) a single parameter can easily control the smoothness of the window function.

Figure 2 shows that all three properties are satisfied for different values of  $\sigma$ . For  $\sigma \rightarrow 0$  our window function converges to the window function of the triangle. For larger values, the window function transitions smoothly from zero at the boundary to one in the middle, and for  $\sigma \rightarrow \infty$  the window function becomes a delta function at the incenter.

**Window function alternatives.** Related works [7, 14] use the LogSumExp function to approximate max in the signed distance field:  $\phi(\mathbf{p}) = \log \sum_{i=1}^3 \exp L_i(\mathbf{p})$ . However, we observed that, for small triangles, this max approximation is poor, to the point that *only one* of the three vertices has any influence on the final shape. We thus opted to use the actual max function which, while not smooth everywhere, accurately defines the signed distance field. Further, related work [7, 14, 25] also use a different definition for the window function  $I$  based on sigmoid:

$$I(\mathbf{p}) = \text{sigmoid}(-\sigma^{-1} \phi(\mathbf{p}))$$

$$\text{such that } I(\mathbf{p}) \begin{cases} > \frac{1}{2} & \text{inside the triangle,} \\ = \frac{1}{2} & \text{at the triangle boundary,} \\ < \frac{1}{2} & \text{outside the triangle.} \end{cases} \quad (2)$$

This definition fails to meet properties (i) and (ii) above, as the maximum can be less than 1, and the support of the window function can be significantly larger than the triangle. This is illustrated in Figure 2, where  $\sigma \rightarrow \infty$  results in a constant value everywhere in  $\mathbb{R}^2$ .

**Simpler depth-dependent scaling.** In 3D Gaussian Splatting, each 3D Gaussian is defined in world space by a full covariance matrix, which is mapped to image space by accounting for the projective transformation, resulting in a 2D covariance matrix inversely proportional to depth. The effect is a 2D Gaussian whose size scales with depth. In Convex Splatting [14], the 2D convex hull scales automatically with depth, but not the window function smoothness parameter  $\sigma$ . Because the latter is defined in pixel units, it must be scaled “manually” to achieve a depth-consistent effect. In our case, this is unnecessary because of the normalization in Eq. (1):  $\sigma$  results in consistently scaled 2D window functions for all depth values.

**A simpler and more efficient primitive.** Triangle Splatting reduces complexity compared to concurrent approaches. Unlike 3DGS, Triangle Splatting avoids the need for EWA, a core component of 3DGS that is complex to implement and whose quality degrades when moving away from the optical center. Also, compared to 3D Convex



Figure 3. **Triangle pruning.** To reduce floaters, we prune triangles seen in fewer than two views with over one pixel of coverage, removing those overfitted to a single training view.

Splatting, Triangle Splatting avoids the complex and costly kernel required to compute active edges and the need to calculate the convex hull per primitive, simplifying implementation and improving speed.

### 3.2. Adaptive pruning and splitting

Triangles have a compact spatial domain (and, therefore, a compact gradient); hence, we need a mechanism to control coverage of the spatial domain by the triangles, modulating their density and thus representation power at different locations. This is achieved by pruning and densification routines (respectively decreasing and increasing the representation power), analogously to 3DGS [20].

**Pruning.** During rasterization, we calculate the maximum volume rendering blending weight  $T \cdot o$  (where  $T$  is transmittance, and  $o$  is opacity) for each triangle, and prune all triangles whose maximum weight is less than a defined threshold  $\tau_{\text{prune}}$  across all training views. Additionally, we prune all triangles that are not rendered at least *twice* with more than one pixel. In other words, we remove triangles that explain small amounts of data within a single view and are therefore likely to have overfitted to the training data. Figure 3 illustrates the impact of this pruning strategy.

**Densification.** Instead of relying on manually tuned heuristics for adding shapes, we adopt the probabilistic framework based on MCMC introduced by Kheradmand et al. [21]. At each densification step, we sample from a probability distribution to guide where new shapes should be added. Kheradmand et al. [21] stochastically allocates new Gaussians proportionally to the *opacity*, and we extend this idea to our representation by incorporating the sharpness parameter  $\sigma$ . Since both opacity and  $\sigma$  are learned during training, we build the probability distribution directly from these parameters by alternating between using the inverse of  $\sigma$  and the opacity for Bernoulli sampling. In particular, we preferentially sample triangles with low  $\sigma$  values, *i.e.* *solid* triangles. Because of our window function, the triangle’s influence is bounded by its projected geometry, and the diffusion remains confined within the triangle itself. In high-density regions, many triangles overlap at each pixel, allowing each shape to adopt a higher  $\sigma$  and thus a softer contribution. In contrast, in low-density regions, where fewer triangles in-

fluence a pixel, each triangle must contribute more to the reconstruction. As a result, it adopts a lower  $\sigma$  to increase the contribution across its interior, ensuring maximal coverage within the geometric bounds and producing a more solid appearance.

Further, taking inspiration from Kheradmand et al. [21], we design updates to avoid disrupting the sampling process. In particular, we require that the probability of the state (*i.e.* the current set of parameters of all triangles) remains *unchanged* before and after the transition, allowing it to be interpreted as a move between equally probable samples, preserving the integrity of the Markov chain. To preserve a consistent representation across sampling steps, we apply *midpoint subdivision* to the selected triangles. Each triangle is split into four smaller ones by connecting the midpoints of its edges, ensuring that the combined area and spatial region of the new triangles match that of the original. As in our parametrization, a triangle is defined by 3D vertices, making this operation straightforward to perform. Finally, if a triangle is smaller than the threshold  $\tau_{\text{small}}$ , we do not split it. Instead, we clone it and add random noise along the triangle’s plane orientation.

### 3.3. Optimization

Our method starts from a set of images and their corresponding camera parameters, calibrated via SfM [32], which also produces a sparse point cloud. We create a 3D triangle for each 3D point in the sparse point cloud. We optimize the 3D vertex positions  $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ , sharpness  $\sigma$ , opacity  $o$ , and spherical harmonic color coefficients  $\mathbf{c}$  of all such 3D triangles by minimizing the rendering error from the given posed views. The initialization is done as follows. Let  $\mathbf{q} \in \mathbb{R}^3$  be a SfM 3D point and let  $d$  be the average Euclidean distance to its three nearest neighbors. We initialize the corresponding 3D triangle to be approximately equilateral, randomly oriented, and with a size proportional to  $d$ . To do this, we sample uniformly at random three vertices  $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$  from the unit sphere, we scale them by  $d$ , and we add  $\mathbf{q}$  to center them at the point  $\mathbf{q}$ :  $\mathbf{v}_i = \mathbf{q} + k \cdot d \cdot \mathbf{u}_i$ , where  $k \in \mathbb{R}$  is a scaling constant. Our training loss combines the photometric  $\mathcal{L}_1$  and  $\mathcal{L}_{\text{D-SSIM}}$  terms [20], the opacity loss  $\mathcal{L}_o$  [21], and the distortion  $\mathcal{L}_d$  and normal  $\mathcal{L}_n$  losses [15]. Finally, we add a size regularization term  $\mathcal{L}_s = 2 \|(\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)\|_2^{-1}$ , to encourage larger triangles. The final loss  $\mathcal{L}$  is given by:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} + \beta_1\mathcal{L}_o + \beta_2\mathcal{L}_d + \beta_3\mathcal{L}_n + \beta_4\mathcal{L}_s. \quad (3)$$

The full list of thresholds and hyperparameters is detailed in the *Supplementary Material*.

## 4. Experiments

We compare our method to concurrent approaches on the standard benchmarks Mip-NeRF360 [2] and Tanks and Temples (T&T) [22]. We consider 3DCS [14], which is the most closely related method, as well as to non-volumetric primitives such as BBSplat [33] and 2DGS [15]. We also consider primitive-based volumetric methods, including 3DGS [20], 3DGS-MCMC [21], and DBS [24]. Additionally, we evaluate against implicit neural rendering methods such as Instant-NGP [29], Mip-NeRF360 [2], and the state-of-the-art in novel view synthesis Zip-NeRF [3]. We evaluate the visual quality using standard metrics: SSIM, PSNR, and LPIPS. We report training time, rendering speed, and memory usage, measured on an NVIDIA A100.

**Implementation details.** We use spherical harmonics of degree 3, resulting in 59 parameters per triangle, matching the number of parameters for a single 3D Gaussian [20].

### 4.1. Novel-view synthesis

Table 1 presents the quantitative results on the T&T dataset, as well as on the indoor and outdoor scenes from the Mip-NeRF360 dataset. In comparison with planar primitive methods, Triangle Splatting achieves a higher visual quality, with a significant improvement in LPIPS (the metric that best correlates with human visual perception). Specifically, Triangle Splatting improves over 2DGS and BBSplat by 25% and 19% on Mip-NeRF360, respectively. Triangle Splatting achieves consistently better LPIPS scores in outdoor scenes, surpassing both 2DGS and BBSplat. Similarly, on the T&T dataset, Triangle Splatting yields a substantial improvement over 2DGS and BBSplat.

While our method yields slightly lower PSNR values, this metric does not fully capture visual quality due to its inherent limitations (PSNR generally rewards overly smooth reconstructions that regress to the mean). As a result, smooth representations, such as Gaussian-based primitives, tend to perform better under PSNR, whereas sharper transitions from solid shapes may be penalized. Figure 4 illustrates this limitation of PSNR: although our reconstruction looks visually superior to that of 2DGS, the PSNR in the *highlighted region* is 3 PSNR higher for 2DGS.

Against volumetric primitive methods, Triangle Splatting achieves high visual quality, with a significant improvement in LPIPS. Specifically, Triangle Splatting improves over 3DGS and 3DCS by 10% and 7% respectively on Mip-NeRF 360. Compared to implicit methods, Triangle Splatting matches the visual quality of the state-of-the-art Zip-NeRF, with only a 0.002 difference in LPIPS, while delivering over  $500\times$  faster rendering performance.

Triangles work particularly well in indoor or structured outdoor scenes with walls, cars, and other well-defined surfaces, where they can closely approximate geometry.



|                           | Outdoor Mip-NeRF 360 |              |              | Indoor Mip-NeRF 360 |              |              | Aver. Mip-NeRF 360 |              |              | Tanks & Temples |              |              |            |
|---------------------------|----------------------|--------------|--------------|---------------------|--------------|--------------|--------------------|--------------|--------------|-----------------|--------------|--------------|------------|
|                           | LPIPS ↓              | PSNR ↑       | SSIM ↑       | LPIPS ↓             | PSNR ↑       | SSIM ↑       | LPIPS ↓            | PSNR ↑       | SSIM ↑       | LPIPS ↓         | PSNR ↑       | SSIM ↑       | FPS ↑      |
| Implicit Methods          |                      |              |              |                     |              |              |                    |              |              |                 |              |              |            |
| Instant-NGP [29]          | -                    | -            | -            | -                   | -            | -            | 0.331              | 25.59        | 0.699        | 0.305           | 21.92        | 0.745        | 14.4       |
| Mip-NeRF360 [2]           | 0.283                | 24.47        | 0.691        | 0.179               | 31.72        | 0.917        | 0.237              | 27.69        | 0.792        | 0.257           | 22.22        | 0.759        | 0.14       |
| Zip-NeRF [3]              | 0.207                | 25.58        | 0.750        | 0.167               | 32.25        | 0.926        | 0.189              | 28.54        | 0.828        | -               | -            | -            | -          |
| Volumetric Primitives     |                      |              |              |                     |              |              |                    |              |              |                 |              |              |            |
| 3DGS [20]                 | 0.234                | 24.64        | 0.731        | 0.189               | 30.41        | 0.920        | 0.214              | 27.21        | 0.815        | 0.183           | 23.14        | 0.841        | 154        |
| 3DGS-MCMC [21] ‡          | 0.210                | 25.51        | 0.76         | 0.208               | 31.08        | 0.917        | 0.210              | 27.98        | 0.829        | 0.19            | 24.29        | 0.86         | 129        |
| DBS [24] †                | 0.246                | 25.10        | 0.751        | 0.22                | 32.29        | 0.936        | 0.234              | 28.29        | 0.833        | 0.140           | 24.85        | 0.870        | 150        |
| 3DCS [14]                 | 0.238                | 24.07        | 0.700        | 0.166               | 31.33        | 0.927        | 0.207              | 27.29        | 0.802        | 0.156           | 23.94        | 0.851        | 33         |
| Non-Volumetric Primitives |                      |              |              |                     |              |              |                    |              |              |                 |              |              |            |
| BBSplat [33] †            | 0.281                | 23.55        | 0.669        | 0.178               | 30.62        | 0.921        | 0.236              | 26.69        | 0.781        | 0.172           | <b>25.12</b> | <b>0.868</b> | 66         |
| 2DGS [15]                 | 0.246                | <b>24.34</b> | 0.717        | 0.195               | 30.40        | 0.916        | 0.252              | 27.04        | 0.805        | 0.212           | 23.13        | 0.831        | 122        |
| <b>Triangle Splatting</b> | <b>0.217</b>         | 24.27        | <b>0.722</b> | <b>0.160</b>        | <b>30.80</b> | <b>0.928</b> | <b>0.191</b>       | <b>27.16</b> | <b>0.814</b> | <b>0.143</b>    | 23.14        | 0.857        | <b>165</b> |

Table 1. **Quantitative results.** We evaluate our method on both indoor and outdoor scenes, achieving state-of-the-art performance on the *indoor* benchmark. Across all datasets, our approach consistently outperforms other non-volumetric primitives. Bold scores indicate the best results among *non-volumetric* methods. † denotes reproduced results, while ‡ marks results reported in [24].

this makes Triangle Splatting especially effective indoors, achieving state-of-the-art performance and outperforming 3DCS and Zip-NeRF. Unstructured outdoor scenes are more challenging for planar primitives due to sparse or ambiguous geometry, making it harder to maintain visual consistency across views. Even so, Triangle Splatting narrows the performance gap and surpasses 3DGS and 3DCS on the T&T dataset, achieving a lower LPIPS.

Figure 5 presents a qualitative comparison between Triangle Splatting, 3DCS, and 2DGS. We consistently produce sharper reconstructions, particularly in high-frequency regions. For instance, in the *Bicycle* scene, it more accurately captures fine details, as highlighted.

**Speed & Memory.** Table 2 compares the memory consumption and rendering speed of concurrent methods. Although BBSplat uses fewer primitives, it suffers from considerably slower training and inference. Triangle Splatting shows strong scalability, despite using more primitives, it renders  $4\times$  faster than BBSplat and achieves a 40% speedup over 2DGS. Triangle Splatting significantly outperforms 3DCS, achieving  $2\times$  faster training and  $4\times$  faster inference.

|         | Train ↓    | FPS ↑     | Memory ↓     |
|---------|------------|-----------|--------------|
| ZipNerf | 5h         | 0.18      | 569MB        |
| 3DGS    | 42m        | 134       | 734MB        |
| 3DCS    | 87m        | 25        | 666MB        |
| BBSplat | 96m        | 25        | <b>175MB</b> |
| 2DGS    | <b>29m</b> | 64        | 484MB        |
| Ours    | 39m        | <b>97</b> | 795MB        |

Table 2. **Speed & Memory.** Triangle Splatting scales efficiently, achieving faster training and rendering even with more primitives.

Unlike 3DCS, Triangle Splatting does not require computing a 2D convex hull and rendering is more efficient. Triangle Splatting computes the signed distance for only three lines per pixel, whereas 3DCS requires calculations for six lines, effectively doubling the per-pixel computational cost.

## 5. Ablations

**Loss terms.** Table 3 shows the impact on performance when removing different components of our pipeline on Mip-NeRF360. The opacity regularization  $\mathcal{L}_o$  is the most



Figure 4. **Limitations of PSNR.** Due to its smoothness, Gaussians tends to perform better on the PSNR metric, which evaluates pixel-wise differences, despite being blurrier. In the highlighted areas, our method achieves a PSNR of 18.41, compared to 21.27 for 2DGS.

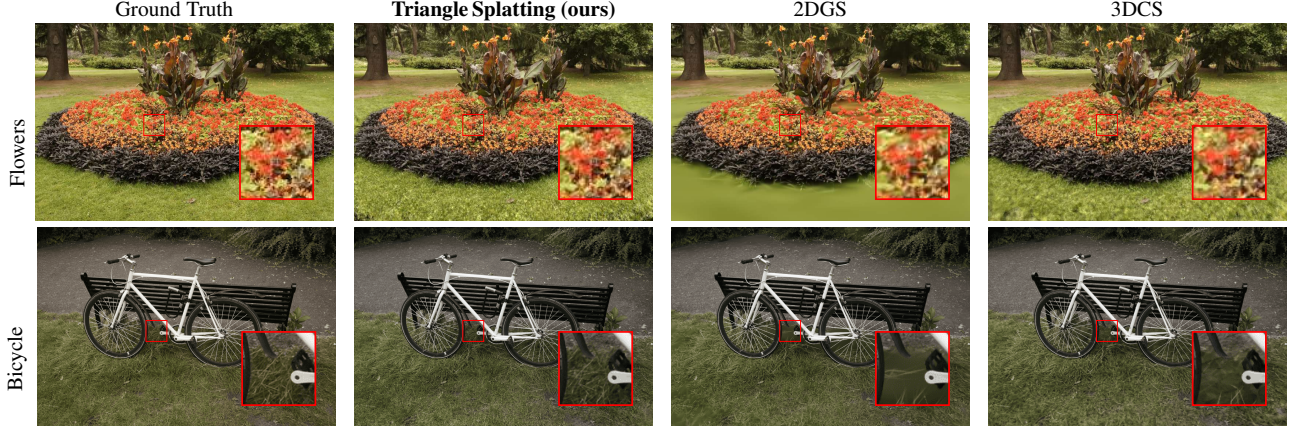


Figure 5. **Qualitative results.** We visually compare our method to 3DCS [14] and 2DGS [15]. Triangle Splatting captures finer details and produces more accurate renderings of real-world scenes, with less blurry results than 2DGS, and a higher visual quality than 3DCS [14].

|                                       | LPIPS ↓      | PSNR ↑       | SSIM ↑       |
|---------------------------------------|--------------|--------------|--------------|
| Triangle Splatting                    | <b>0.191</b> | <b>27.14</b> | <b>0.814</b> |
| w/o $\mathcal{L}_s$                   | <b>0.191</b> | 26.97        | 0.812        |
| w/o $\sigma^{-1}$ sampling            | 0.193        | 27.03        | 0.811        |
| w/o $o$ sampling                      | 0.193        | 27.02        | 0.811        |
| w/o $\mathcal{L}_d$ & $\mathcal{L}_n$ | 0.194        | 27.11        | 0.811        |
| w/o $\mathcal{L}_o$                   | 0.207        | 26.38        | 0.794        |

Table 3. **Ablation study.** We isolate the impact of each component by removing them individually.

impactful, encouraging lower opacity values so that unnecessary triangles become transparent and are reallocated. The regularization term  $\mathcal{L}_s$  encourages larger triangles, increasing PSNR, particularly in indoor scenes. The initial point cloud is often extremely sparse along walls, frequently with few or no initial triangles. Without  $\mathcal{L}_s$ , triangles expand too slowly to reach scene boundaries. By promoting larger shapes, this regularization enables faster growth, allowing triangles to extend into underrepresented regions. Sampling based on  $\sigma^{-1}$  or opacity alone yields similar performance, while combining both leads to improved results.

**Window functions.** Figure 6 compares the sigmoid-based window function with our proposed version. In regions with sparse initial point cloud, particularly in the background, the sigmoid function fails to recover the scene structure. Since the sigmoid is not bounded by its geometry’s vertices and can grow arbitrarily large, the optimizer tends to increase the sigma values instead of moving vertices to cover empty areas. This results in small yet very smooth shapes, making them difficult to optimize. In contrast, our normalized window function enforces spatial bounds, which encourages vertices to move and fill underrepresented regions. As the size of each shape is explicitly defined, the optimization process becomes more stable and effective.

**Triangle vs. convex splatting.** In 3DCS, each convex is defined by six vertices. When the number is reduced to



Figure 6. **Ablation study (window function).** We compare against the Sigmoid function (left) which fails to recover background regions accurately, while ours doesn’t (right).



Figure 7. **Ablation study (triangles as convexes).** We compare our method (left) with 3DCS using three-vertex convexes (right), which results in degenerate geometry, as seen in the zoom-ins.

three, the shape degenerates into a triangle. In Figure 7, we present a visual comparison between Triangle Splatting and 3DCS using triangles. Triangle Splatting does not produce line artifacts, which often appear in 3DCS when handling degenerate triangles. Furthermore, Triangle Splatting obtains a higher visual quality with an improvement of 0.05 LPIPS, 0.61 PSNR and 0.045 SSIM on Mip-NeRF360.

## 6. Triangle splatting in a traditional renderer

While the main goal of this work is to introduce triangles as a novel primitive for high-quality novel view synthesis, their benefits extend beyond this task. To obtain a model with mesh-like topology, we set opacity to 1 and  $\sigma$  to 0, yielding opaque, hard triangles that require only vertex positions and per-triangle colors to be stored in a standard mesh format (e.g., .off), with no post-processing. This marks a sig-



nificant improvement over 2DGS. The final triangle soup can be integrated into *any* mesh renderer. Alternatively, the full triangle splatting model can be incorporated into traditional rendering pipelines using order-independent transparency to correctly handle semi-transparent triangles. To demonstrate practicality, we evaluate triangle splatting in a traditional renderer across both use cases.

**Stochastic Triangle Splatting renderer.** Alpha blending-based models like 3DGS can be implemented with z-buffer rasterization by employing stochastic transparency. This approach is particularly well suited to Triangle Splatting, as it relies on invoking the fragment shader many times for each final pixel to reduce noise. Unlike Gaussian-based models, which must evaluate the Gaussian PDF in the fragment shader, Triangle Splatting can perform almost all computations in the vertex shader, lowering the cost of additional samples. We include a link in the *Supplementary Material* to a browser-based demo of this method, which includes a naive implementation of temporal filtering for noise reduction. It should be noted that more sophisticated temporal anti-aliasing pipelines are a standard feature in modern game engines. Table 4 shows that even a consumer laptop with an Iris Xe renders 4.3M triangles at high speed, while high-end GPUs like the RTX 4090 exceed 1000 FPS. As shown in Figure 8, Stochastic Triangle Splatting also preserves high visual quality when integrated into a traditional renderer.

**Mesh-based novel-view synthesis.** Table 5 compares mesh-based novel view synthesis between Triangle Splatting and 2DGS. For 2DGS, the mesh was extracted via TSDF fusion and textured with a neural color field trained for 5k iterations following 30k iterations of main training (see MILo [12] for more details). In contrast, Triangle Splatting requires no additional processing, only the opacity is set to 1 and  $\sigma$  to 0. Compared to Triangle Splatting, 2DGS relies on post-processing, lacks direct game engine compatibility, and achieves lower visual quality. For Triangle Splatting we simply set the opacity to 1 and  $\sigma$  to 0, without any additional processing. 2DGS requires post-processing, lacks direct game engine compatibility, and delivers lower visual quality. Triangle Splatting improves PSNR by 37%, SSIM by 12%, and reduces LPIPS by 2.7%, yielding better visuals without post-processing.

Triangle Splatting uses far fewer triangles ( $\approx 1\text{M}$  vs.  $\approx 7\text{M}$ ). It reconstructs background regions more accurately, while 2DGS tends to focus on the object center and often leaves backgrounds sparse or missing, reducing visual quality. Triangle Splatting opens the door to future work, such as developing training strategies specifically for high visual quality in game engines (for example, gradually pushing opacity and  $\sigma$  toward 1 and 0, respectively), or connecting triangles during training to obtain a watertight mesh.

| Hardware   | OS      | TFLOPS | HD   | Full HD | 4k  |
|------------|---------|--------|------|---------|-----|
| Iris Xe    | Windows | 2      | 33   | 23      | 15  |
| MacBook M4 | MacOS   | 8      | 120  | 90      | 40  |
| RTX4090    | Linux   | 48     | 1058 | 759     | 327 |

Table 4. **Stochastic renderer: FPS across different hardware and resolutions.** Evaluated on *Garden* ( $\approx 4.3\text{M}$  triangles) with 1 sample per pixel. OS stands for operating system.



Figure 8. **Qualitative rendering results with Stochastic Triangle Splatting.** Most computations run in the vertex shader, enabling real-time rendering in a game engine without quality loss.

|                | Post-processing | PSNR $\uparrow$ | LPIPS $\downarrow$ | SSIM $\uparrow$ | #T                  |
|----------------|-----------------|-----------------|--------------------|-----------------|---------------------|
| 2DGS $\dagger$ | Yes             | 15.36           | 0.474              | 0.498           | $\approx 7\text{M}$ |
| <b>Ours</b>    | No              | <b>21.05</b>    | <b>0.462</b>       | <b>0.558</b>    | $\approx 1\text{M}$ |

Table 5. **Mesh-based novel view synthesis.** Comparison between Triangle Splatting and 2DGS on MipNeRF 360. We achieve higher image quality (higher PSNR/SSIM, lower LPIPS), and require far fewer triangles (#T).  $\dagger$  results obtained from [12].

## 7. Conclusions

We have introduced **Triangle Splatting**, a novel differentiable rendering technique that directly optimizes triangle primitives for novel-view synthesis. Leveraging the same primitive used in classical mesh representations, our method bridges the gap between neural rendering and traditional graphics pipelines. We have shown that Triangle Splatting outperforms concurrent methods in visual quality while retaining the efficiency, and compatibility of the triangle. Moreover, it opens several directions for future research, including connecting triangles during training to obtain watertight meshes and optimizing  $\sigma$  and opacity for high visual quality in game engines. These results establish Triangle Splatting as a promising step toward mesh-aware neural rendering, unifying decades of GPU-accelerated graphics with modern differentiable frameworks.

**Acknowledgments.** J. Held, A. Deliege and A. Cioppa are funded by the F.R.S.-FNRS. The research reported in this publication was supported by funding from KAUST Center of Excellence on GenAI, under award number 5940. This work was also supported by KAUST Ibn Rushd Post-doc Fellowship program. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.

## References

- [1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5835–5844, Montréal, Can., 2021. Inst. Electr. Electron. Eng. (IEEE). 2
- [2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5460–5469, New Orleans, LA, USA, 2022. Inst. Electr. Electron. Eng. (IEEE). 2, 5, 6
- [3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 19640–19648, Paris, Fr., 2023. Inst. Electr. Electron. Eng. (IEEE). 2, 5, 6
- [4] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini de Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16102–16112, New Orleans, LA, USA, 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensorRF: Tensorial radiance fields. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 333–350, Tel Aviv, Israël, 2022. Springer Nat. Switz. 2
- [6] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16569–16578, Vancouver, Can., 2023. IEEE. 2
- [7] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable convex decomposition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 31–41, Seattle, WA, USA, 2020. Inst. Electr. Electron. Eng. (IEEE). 2, 3, 4
- [8] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4970–4980, Vancouver, Can., 2023. Inst. Electr. Electron. Eng. (IEEE). 2
- [9] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5491–5500, New Orleans, LA, USA, 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [10] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kauffmann Publ. Inc., San Francisco, CA, USA, 2007. 2
- [11] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5354–5363, Seattle, WA, USA, 2024. Inst. Electr. Electron. Eng. (IEEE). 12
- [12] Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. Milo: Mesh-in-the-loop gaussian splatting for detailed and efficient surface reconstruction. *ACM Transactions on Graphics*, 44(6): 1–15, 2025. 8
- [13] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5855–5864, Montréal, Can., 2021. Inst. Electr. Electron. Eng. (IEEE). 2
- [14] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Delière, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21360–21369, Nashville, TN, USA, 2025. Inst. Electr. Electron. Eng. (IEEE). 2, 3, 4, 5, 6, 7, 12, 13
- [15] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conf. Pap.*, pages 1–11, Denver, CO, USA, 2024. ACM. 2, 3, 5, 6, 7, 12, 13
- [16] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5865–5874, Montréal, Can., 2021. Inst. Electr. Electron. Eng. (IEEE). 2
- [17] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 406–413, Columbus, OH, USA, 2014. Inst. Electr. Electron. Eng. (IEEE). 12
- [18] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3907–3916, Salt Lake City, UT, USA, 2018. Inst. Electr. Electron. Eng. (IEEE). 2
- [19] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Symp. Geom. Process.*, pages 61–70, Cagliari, Italy, 2006. Eurographics Assoc. 2
- [20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):1–14, 2023. 2, 3, 4, 5, 6, 12, 13
- [21] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as Markov chain Monte Carlo. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 80965–80986, Vancouver, Can., 2024. Curran Assoc. Inc. 4, 5, 6
- [22] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):1–13, 2017. 5

- [23] Jonas Kulhanek and Torsten Sattler. Tetra-NeRF: Representing neural radiance fields using tetrahedra. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 18412–18423, Paris, Fr., 2023. Inst. Electr. Electron. Eng. (IEEE). 2
- [24] Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable beta splatting. In *Proc. Spec. Interes. Group Comput. Graph. Interact. Tech. Conf. Conf. Pap.*, pages 1–11, Vancouver, Can., 2025. ACM. 5, 6
- [25] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 7707–7716, Seoul, South Korea, 2019. Inst. Electr. Electron. Eng. (IEEE). 2, 4
- [26] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 154–169, Zürich, Switzerland, 2014. Springer Int. Publ. 2
- [27] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 405–421, Virtual conference, 2020. Springer Int. Publ. 2
- [28] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF. *Commun. ACM*, 65(1):99–106, 2021. 2
- [29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):1–15, 2022. 2, 5, 6
- [30] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. *ACM Trans. Graph.*, 43(4):1–17, 2024. 11
- [31] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph.*, 42(4):1–12, 2023. 2
- [32] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4104–4113, Las Vegas, NV, USA, 2016. Inst. Electr. Electron. Eng. (IEEE). 5
- [33] David Svitov, Pietro Morerio, Lourdes Agapito, and Alessio Del Bue. BillBoard splatting (BBSplat): Learnable textured primitives for novel view synthesis. *arXiv*, abs/2411.08508, 2024. 5, 6

## 8. Supplementary Material

### 8.1. Methodology

**Depth-dependant scaling.** In Triangle Splatting, the influence  $I(p)$  depends only on the normalized ratio  $\phi / \min \phi$ . Since a projection is a uniform in-plane scaling, a single exponent  $\sigma$  suffices for all depths. Indeed, if we scale a triangle by  $a > 0$ , then each projected vertex  $v_i$  and pixel  $p$  transforms as  $v_i \mapsto a v_i$  and  $p \mapsto a p$ , so the signed-distance  $d_i(p)$  to each edge satisfies  $d'_i(p') = a d_i(p)$ , implying  $\phi'(p') = \max(d'_i(p')) = a \phi(p)$  and  $\min \phi' = a \min \phi$ . Hence

$$\frac{\phi'(p')}{\min \phi'} = \frac{a \phi(p)}{a \min \phi} = \frac{\phi(p)}{\min \phi}, \quad (4)$$

and so  $I'(p') = (\phi'(p') / \min \phi')^\sigma = (\phi(p) / \min \phi)^\sigma = I(p)$ .

**Tile assignment in Triangle Splatting** Triangle Splatting is a tile-based renderer that assigns triangles to pixel tiles by computing their screen-space intersections. Unlike 3D Gaussian Splatting, where the shape has a soft spatial extent, Triangle Splatting is precisely bounded by the projection of its three vertices. A simple and efficient initial guess for determining tile coverage is to compute the minimum and maximum x and y coordinates of the projected vertices. While this method is computationally simple, it is conservative, resulting in unnecessary computations for pixels that are not influenced. As  $\sigma$  increases or opacity  $o$  decreases, the influence region may not reach the triangle’s vertices, causing the rasterizer to process pixels that have no contribution. To avoid this, we compute a more precise bounding box by determining the maximum distance  $d$  a pixel can be from the triangle edge while still contributing at least  $\tau_{\text{cutoff}}$  influence. We define the influence function as:

$$\tau_{\text{cutoff}} = \left( \frac{d}{L(s)} \right)^\sigma \cdot o. \quad (5)$$

where  $L(s)$  is the signed distance from the triangle’s edges to the incenter. Rearranging gives:

$$d = L(s) \cdot \left( \frac{\tau_{\text{cutoff}}}{o} \right)^{\frac{1}{\sigma}}. \quad (6)$$

We then subtract the distance  $d$  from the offset in each edge equation, effectively tightening the triangle’s boundary. The minimum and maximum of the updated edge intersections define a more accurate bounding box. As  $\sigma$  increases or  $o$  decreases, this bounding box shrinks accordingly, significantly reducing unnecessary rasterization.

**Depth sorting.** During rasterization, the triangles are currently sorted based on their center, which can lead to popping and blending artifacts during view rotation. Instead,

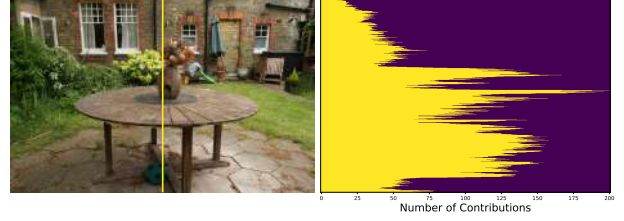


Figure 9. **Number of contributions per pixel.** In background regions where the initial point cloud is sparse, triangles reduce their  $\sigma$  to increase coverage across their interior. This leads to more solid shapes and, consequently fewer contributions per pixel.

future work can implement per-pixel sorting of the triangles, as proposed in [30], where Radl *et al.* introduce a hierarchical tile-based rasterization approach that performs local per-pixel sorting to ensure consistent visibility and eliminate popping artifacts.

| Method                | Outdoor | Indoor  |
|-----------------------|---------|---------|
| feature_lr            | 0.0025  | 0.0025  |
| opacity_lr            | 0.014   | 0.014   |
| lr_convex_points_init | 0.0018  | 0.0015  |
| lr_sigma              | 0.0008  | 0.0008  |
| lambda_normals        | 0.0001  | 0.00004 |
| lambda_opacity        | 0.0055  | 0.0055  |
| lambda_size           | 1e-8    | 5e-8    |
| max_noise_factor      | 1.5     | 1.5     |
| opacity_dead          | 0.014   | 0.014   |
| split_size            | 24.0    | 24.0    |
| importance_threshold  | 0.022   | 0.0256  |

Table 6. **Hyperparameters**

**Densification.** We prioritize sampling triangles with low  $\sigma$  values, corresponding to more *solid* triangles. Our window function ensures that each triangle’s influence is strictly limited to its projected area, preventing any influence beyond its geometric bounds. In regions with high triangle density, multiple shapes contribute to each pixel, allowing individual triangles to adopt larger  $\sigma$  values and produce softer, more diffuse effects. Conversely, in sparse regions where fewer triangles are present, each triangle must account for more of the pixel-wise reconstruction, leading it to adopt a smaller  $\sigma$  and thus contribute more across its surface. Figure 9 visualizes the number of contributions per pixel. In sparse background regions, triangles adopt lower  $\sigma$  values, resulting in solid shapes and fewer overlapping contributions per pixel. In contrast, densely sampled regions exhibit higher per-pixel contributions due to many overlapping triangles. As we prioritize adding new triangles to low-



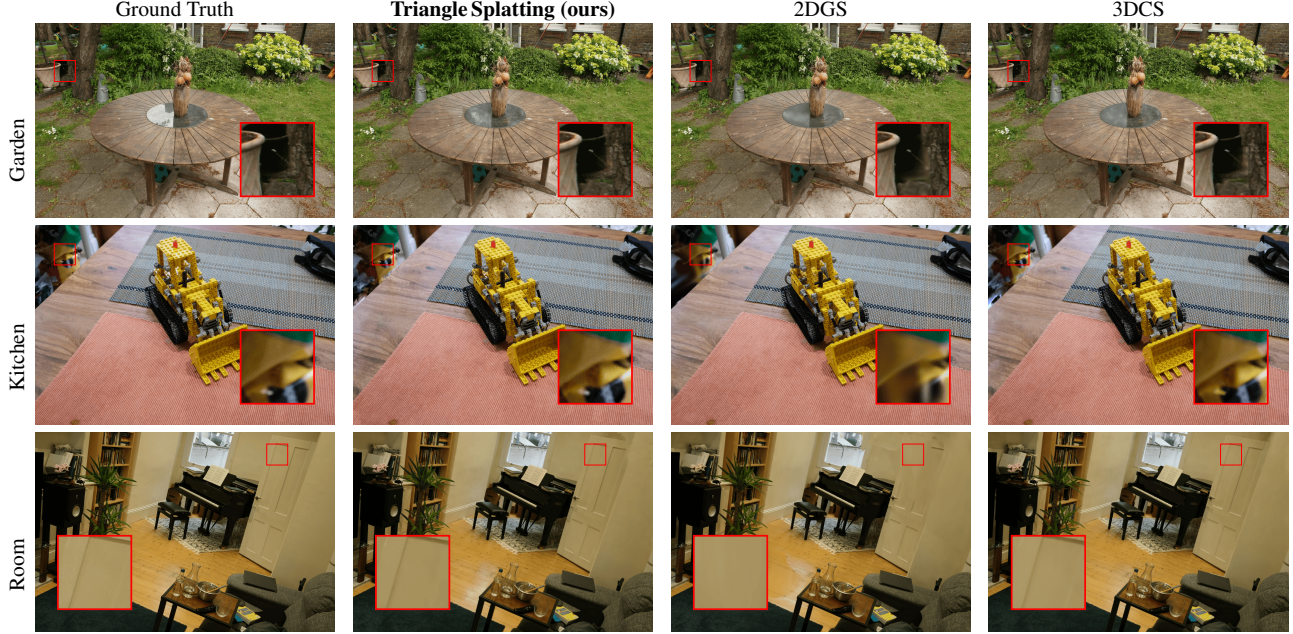


Figure 10. **Qualitative results.** We visually compare our method to 2DGS [15] and 3DCS [14]. Triangle Splatting captures finer details and produces more accurate renderings of real-world scenes, with less blurry results than 2DGS, and a higher visual quality than 3DCS [14].

density regions, we sample from a probability distribution defined over the inverse of  $\sigma$ .

## 8.2. Initialization & Hyperparameters.

**Initialization.** At initialization, each triangle is assigned a fixed opacity of 0.28, and a sigma value of 1.16. The scaling constant  $k$  of the convex hull is set to 2.2, which defines the spatial extent of the triangle in the 3D scene. These parameter values were determined empirically to ensure stable initialization and consistent rendering behavior across scenes.

**Densification.** We perform densification every 500 iterations, starting from iteration 500 until iteration 25,000. At each densification step, we increase the number of shapes by 30%.

All other hyperparameters are defined in Table 6.

## 8.3. More novel-view synthesis results

Tables 8 to 11 present a detailed analysis of the results on the Mip-NeRF360 and Tanks and Temples datasets, while Figure 10 presents additional qualitative results.

## 8.4. Geometry analysis of Triangle Splatting

Our triangle soup representation is already compatible with standard mesh-based renderers and can be rendered directly without modification. However, constructing a connected mesh from this representation still requires post-processing. A promising direction for future research is to design meshing strategies that take full advantage of the explicit triangle structure inherent to our approach. For completeness, we applied the meshing method used in 2D Gaussian Splatting and present the quantitative results on the DTU dataset [17] in Table 7 and some quantitative results in Figure 11. While this highlights compatibility with existing techniques, we advocate for future work to explore more principled and direct meshing methods tailored specifically to triangle-based representations. Figure 12 shows the normal map produced

| Method             | 24   | 37   | 40   | 55   | 63   | 65   | 69   | 83   | 97   | 105  | 106  | 110  | 114  | 118  | 122  | Mean |
|--------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 3DGS [20]          | 2.14 | 1.53 | 2.08 | 1.68 | 3.49 | 2.21 | 1.43 | 2.07 | 2.22 | 1.75 | 1.79 | 2.55 | 1.53 | 1.52 | 1.50 | 1.96 |
| SuGaR [11]         | 1.47 | 1.33 | 1.13 | 0.61 | 2.25 | 1.71 | 1.15 | 1.63 | 1.62 | 1.07 | 0.79 | 2.45 | 0.98 | 0.88 | 0.79 | 1.33 |
| 2DGS [15]          | 0.48 | 0.91 | 0.39 | 0.39 | 1.01 | 0.83 | 0.81 | 1.36 | 1.27 | 0.76 | 0.70 | 1.40 | 0.40 | 0.76 | 0.52 | 0.80 |
| Triangle Splatting | 0.98 | 1.07 | 1.07 | 0.51 | 1.67 | 1.44 | 1.17 | 1.32 | 1.75 | 0.98 | 0.96 | 1.11 | 0.56 | 0.93 | 0.72 | 1.06 |

Table 7. **Chamfer distance on the DTU dataset**[17]. We report the Chamfer distance on 15 scenes from DTU dataset.



Figure 11. **Mesh extraction from depth maps.** We extract meshes by applying TSDF fusion to the predicted depth maps, as followed in 2DGS [15].

by Triangle Splatting. The triangles are well aligned with the underlying geometry. For example, on the *Garden* table, all triangles share a consistent orientation and lie flat on the surface.

| Method             | LPIPS ↓      |              | PSNR ↑       |              | SSIM ↑       |              |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|
|                    | Truck        | Train        | Truck        | Train        | Truck        | Train        |
| 3DGS [20]          | 0.148        | 0.218        | 25.18        | 21.09        | 0.879        | 0.802        |
| 2DGS [15]          | 0.173        | 0.251        | 25.12        | 21.14        | 0.874        | 0.789        |
| 3DCS [14]          | 0.125        | 0.187        | <b>25.65</b> | <b>22.23</b> | 0.882        | 0.820        |
| Triangle Splatting | <b>0.108</b> | <b>0.179</b> | 24.94        | 21.33        | <b>0.889</b> | <b>0.823</b> |

Table 8. LPIPS, PSNR, and SSIM scores for the Truck and Train scenes of the T&T dataset.

|                    | Bicycle      | Flowers      | Garden       | Stump        | Treehill     | Room         | Counter      | Kitchen      | Bonsai       |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 3DGS               | 0.205        | 0.336        | <b>0.103</b> | <b>0.210</b> | 0.317        | 0.220        | 0.204        | 0.129        | 0.205        |
| 2DGS               | 0.218        | 0.346        | 0.115        | 0.222        | 0.329        | 0.223        | 0.208        | 0.133        | 0.214        |
| 3DCS               | 0.216        | 0.322        | 0.113        | 0.227        | 0.317        | 0.193        | 0.182        | 0.117        | 0.182        |
| Triangle Splatting | <b>0.190</b> | <b>0.284</b> | 0.106        | 0.214        | <b>0.289</b> | <b>0.186</b> | <b>0.171</b> | <b>0.115</b> | <b>0.169</b> |

Table 9. LPIPS score for the MipNerf360 dataset.

|                    | Bicycle      | Flowers      | Garden       | Stump        | Treehill     | Room         | Counter      | Kitchen      | Bonsai       |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 3DGS               | <b>25.24</b> | <b>21.52</b> | <b>27.41</b> | <b>26.55</b> | <b>22.49</b> | 30.63        | 28.70        | 30.31        | 31.98        |
| 2DGS               | 24.87        | 21.15        | 26.95        | 26.47        | 22.27        | 31.06        | 28.55        | 30.50        | 31.52        |
| 3DCS               | 24.72        | 20.52        | 27.09        | 26.12        | 21.77        | <b>31.70</b> | <b>29.02</b> | <b>31.96</b> | <b>32.64</b> |
| Triangle Splatting | 24.90        | 20.85        | 27.20        | 26.29        | 21.94        | 31.05        | 28.90        | 31.32        | 31.95        |

Table 10. PSNR score for the MipNerf360 dataset.

|                    | Bicycle      | Flowers      | Garden       | Stump        | Treehill     | Room         | Counter      | Kitchen      | Bonsai       |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 3DGS               | <b>0.771</b> | 0.605        | <b>0.868</b> | <b>0.775</b> | <b>0.638</b> | 0.914        | 0.905        | 0.922        | 0.938        |
| 2DGS               | 0.752        | 0.588        | 0.852        | 0.765        | 0.627        | 0.912        | 0.900        | 0.919        | 0.933        |
| 3DCS               | 0.737        | 0.575        | 0.850        | 0.746        | 0.595        | 0.925        | 0.909        | <b>0.930</b> | 0.945        |
| Triangle Splatting | 0.765        | <b>0.614</b> | 0.863        | 0.759        | 0.611        | <b>0.926</b> | <b>0.911</b> | 0.929        | <b>0.947</b> |

Table 11. SSIM score for the MipNerf360 dataset.

## 8.5. Triangle splatting in a traditional renderer

**Stochastic Triangle Splatting renderer.** With the stochastic Triangle Splatting renderer, the optimized triangle soup (including opacity and  $\sigma$  values) can be loaded on a wide range of machines and hardware, running in real time even



Figure 12. **Normal map and rendered image.** The normal map reveals a smooth surface, with the triangle orientations consistently aligned to follow the local geometry.

| Hardware   | OS      | TFLOPS | HD    | Full HD | 4k   |
|------------|---------|--------|-------|---------|------|
| MacBook M4 | MacOS   | 8      | 500   | 370     | 160  |
| RTX5000    | Windows | 11     | 570   | 380     | 290  |
| RTX4090    | Linux   | 48     | 2,400 | 1900    | 1050 |

Table 12. **FPS for different hardware and resolutions.** Evaluated on *Garden* ( $\approx 2M$  triangles). OS stands for operating system.



Figure 13. **Byproduct of the triangle-based representation.** In a game engine, we render at 3,000 FPS at 1280×720 resolution on a RTX4090.

at high resolutions. We implemented a browser-based demo that runs on consumer laptops while still producing high visual quality. This prototype includes a simple temporal filtering scheme to reduce noise, **with the best quality obtained when the camera is static and a high number of samples per pixel are accumulated.** In practice, modern game engines already include advanced temporal anti-aliasing pipelines, which would further suppress noise and yield visual quality comparable to that achieved at the end of training.

The browser-based demo can be found under [the following link](#).

The code of the browser-based demo can be found under [the following link](#).

**Mesh-based novel-view synthesis.** To obtain higher visual quality, we naively prune low-opacity triangles. Specifically, during the final 5,000 training iterations, we remove all triangles with opacity below a threshold, retaining only



solid ones. Additionally, we introduce a loss term to encourage higher opacity and lower  $\sigma$ , ensuring that the final triangles are mostly solid and opaque. The conversion from an optimized triangle representation to a solid triangle representation is straightforward. The opacity and  $\sigma$  variables are set to 1 and 0, respectively, to obtain opaque and hard triangles. Finally, the vertex positions and the color of each triangle are stored in a format compatible with standard game engines (such as .off). No post-processing is required. Figure 13 presents qualitative results, while Table 12 reports rendering speeds across hardware and resolutions. Our method reaches 500 FPS at HD resolution on a consumer laptop and 2,400 FPS on an RTX 4090 within a game engine, highlighting both efficiency and practical usability. Compared to *stochastic rendering*, speed is higher since no sorting or additional computations are required, as all triangles are opaque and solid. Future work could explore training strategies tailored for game engine deployment, as our current setup targets novel-view synthesis rather than in-engine visual quality. This opens the door to integrating radiance fields directly into AR/VR and gaming pipelines.

## Part 3

---

*Rise of the Mesh*

## 5. MeshSplatting

### 5.1. Context

Triangle Splatting significantly bridges neural rendering and classical computer graphics by replacing Gaussians with triangles as the fundamental rendering primitive. However, it still exhibits two key limitations that prevent seamless integration into existing graphics engines. First, it produces an unstructured *triangle soup*, where triangles are optimized independently without shared vertices and therefore lack the topological connectivity required for downstream applications such as physics-based simulation or consistent surface editing. Second, smoothness and opacity are treated as independent free parameters during optimization, which leads to semi-transparent triangles. In contrast, standard game engines assume fully opaque triangles to efficiently leverage depth buffers and occlusion culling.

To this end, we introduce *MeshSplatting*, an end-to-end optimization pipeline for learning structured mesh-based scene representations. Traditional rendering engines are highly optimized for polygonal meshes and rely on well-established mechanisms such as depth buffering, back-face culling, and occlusion handling. In contrast, splatting-based approaches typically depend on expensive sorting and alpha blending, which complicates direct integration and often requires custom rendering pipelines or non-differentiable post-processing steps to extract usable geometry.

MeshSplatting directly addresses these challenges by learning a globally consistent, connected, and fully opaque triangle mesh representation. Unlike prior splatting-based methods, it produces a final representation that can be directly imported into standard real-time 3D engines without any intermediate mesh extraction or conversion. This makes the representation inherently compatible with hardware-accelerated graphics pipelines and allows it to fully benefit from established rendering techniques such as depth testing and occlusion culling.

MeshSplatting achieves this by fundamentally changing the underlying parameterization of Triangle Splatting. Instead of treating each triangle as an isolated primitive with its own dedicated vertices and attributes, which results in a disconnected “triangle soup,” MeshSplatting defines triangles through a shared pool of vertices. In this updated structure, each vertex individually stores its own spatial position, color, and opacity. A single triangle is then formed simply by referencing three specific indices from this shared vertex set. This ensures that during the optimization pro-

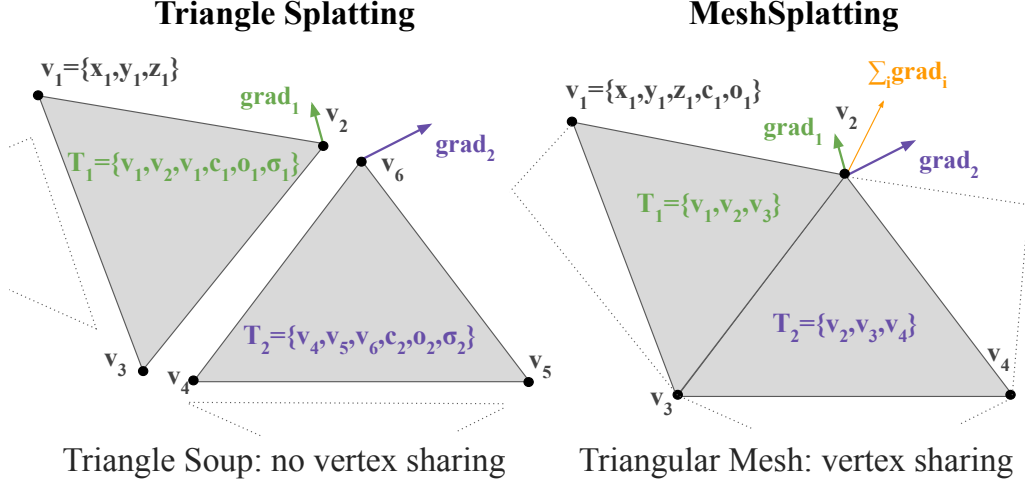


Figure 5.1.: **Mesh parametrization.** (left) In Triangle Splatting, each triangle  $\mathbf{T}_m$  is defined independently by three vertices  $\mathbf{v}_i$ ,  $\mathbf{v}_j$ ,  $\mathbf{v}_k$ , a color  $\mathbf{c}_m$ , a smoothness parameter  $\sigma_m$ , and an opacity  $o_m$ , without sharing vertices with neighboring triangles. This results in a triangle soup with no connectivity between triangles. (right) MeshSplatting parameterizes a triangle  $\mathbf{T}_m$  through a shared vertex set, where each vertex  $\mathbf{v}_i$  stores  $x_i, y_i, z_i, c_i$ , and  $o_i$ . Each triangle is defined by the three indices in the vertex set that compose it. During the backward pass, gradients from all adjacent triangles are accumulated at shared vertices. The smoothness parameter  $\sigma$  is shared across all triangles. This parametrization enforces connectivity between triangles, allowing them to form a coherent mesh topology.

cess, gradients from all adjacent triangles naturally accumulate at their common vertices. Figure 5.1 illustrates the difference in parametrization between Triangle Splatting and MeshSplatting. We further enforce structural consistency by refining vertex positions initialized from a restricted Delaunay triangulation. This induces a coherent global mesh topology where triangles are no longer independent primitives but are connected through shared vertices, transforming an unstructured triangle set into a fully connected surface representation.

Furthermore, MeshSplatting introduces an opacity-aware optimization strategy in which the training process is explicitly designed to produce fully opaque triangles.

MeshSplatting represents the final step toward unifying neural rendering and classical computer graphics. To the best of our knowledge, it is the first end-to-end framework capable of reconstructing large-scale real-world scenes as connected, fully opaque, colored triangle meshes without relying on post-hoc mesh extraction or conversion. This direct output enables immediate deployment in existing graphics ecosystems and unlocks a wide range of downstream applications.

## 5.2. Contributions

We summarize our main contributions as follows:

- (i) We propose MeshSplatting, an end-to-end framework for reconstructing large-scale real-world scenes as connected, fully opaque, and colored triangle meshes, eliminating the need for post-processing.
- (ii) We extend the Triangle Splatting parameterization to allow triangles to share vertices, enabling a structured and consistent surface representation.
- (iii) We obtain a globally connected mesh by optimizing vertex positions within a restricted Delaunay triangulation.
- (iv) The training process progressively enforces full opacity, resulting in triangles that are directly compatible with standard game engines.

## Publication and recognition

The work included in this chapter was published as:

**Jan Held**, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. *MeshSplatting: Differentiable Rendering with Opaque Meshes*. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Denver, USA, 2026.

The MeshSplatting project received significant recognition from the research community. The GitHub repository has already accumulated over 500 stars while being very recent: <https://github.com/meshsplatting/mesh-splatting>.



# MeshSplatting: Differentiable Rendering with Opaque Meshes

Jan Held<sup>1,2</sup> Sanghyun Son<sup>3</sup> Renaud Vandeghen<sup>1</sup>  
Daniel Rebain<sup>4</sup> Matheus Gadelha<sup>6</sup> Yi Zhou<sup>6</sup> Anthony Cioppa<sup>1</sup>  
Ming C. Lin<sup>3</sup> Marc Van Droogenbroeck<sup>1</sup> Andrea Tagliasacchi<sup>2,5</sup>

<sup>1</sup>University of Liège <sup>2</sup>Simon Fraser University <sup>3</sup>University of Maryland

<sup>4</sup>University of British Columbia <sup>5</sup>University of Toronto <sup>6</sup>Adobe Research

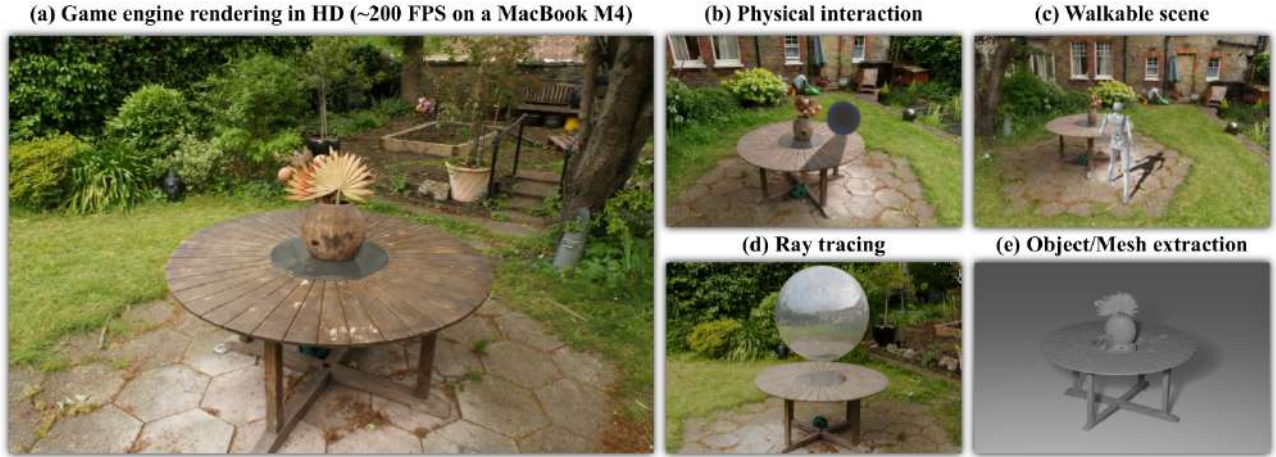


Figure 1. **MeshSplatting** produces a *connected mesh* composed only of *opaque triangles*, achieving high-quality novel view synthesis through end-to-end optimization, with a  $2\times$  training speed-up and  $2\times$  lower memory usage over current state-of-the-art methods. (a) Our representation is compatible with standard game engines, requiring no a-posteriori conversion and/or custom rendering routines for transparency, and natively supports (b) physical interactions, (c) interactive walkthroughs, and (d) ray tracing. (e) MeshSplatting enables straightforward object extraction, allowing scene elements to be directly exported and imported into game engines.

## Abstract

*Primitive-based splatting methods like 3D Gaussian Splatting (3DGS) have revolutionized novel view synthesis with real-time rendering. However, their point-based representations remain incompatible with mesh-based pipelines that power AR/VR and game engines. We present MeshSplatting, a mesh-based reconstruction approach that jointly optimizes geometry and appearance through differentiable rendering. By enforcing connectivity via restricted Delaunay triangulation and refining surface consistency, MeshSplatting creates end-to-end smooth, high-fidelity meshes that render efficiently in real-time engines. On Mip-NeRF360 and Tanks&Temples, it boosts PSNR by +0.69 dB, while training  $2\times$  faster and using  $2\times$  less memory, bridging neural rendering and interactive 3D graphics for seamless real-time scene interaction. The project page is available at <https://meshsplatting.github.io/>.*

## 1. Introduction

Recent advances in novel view synthesis, like 3D Gaussian Splatting [26], have enabled photo-realistic reconstruction of extremely complex scenes. 3D Gaussian Splatting encodes scenes with millions of 3D Gaussian primitives, achieving real-time rendering while also training efficiently. However, while 3DGS renders with high visual fidelity, Gaussian primitives are not immediately compatible with classical graphics pipelines used in simulators, games, and AR/VR applications, as these are typically based on *polygonal meshes*.

Therefore, integrating 3DGS in classical pipelines either requires *engineering* rendering engines [30, 38, 49] and simulators [36] to support them. However, this is non-trivial since 3DGS relies on *sorting* and *alpha blending*, preventing the use of standard techniques like depth buffers and occlusion culling [1, 22]. Another line of work is *converting*

Gaussian radiance fields into meshes [14, 20, 54]. Although somewhat effective, all of these conversion approaches rely on complex post-processing and typically lead to loss of visual quality, as the conversion step is non-differentiable. Rather than relying on conversion a-posteriori, we consider optimizing a mesh directly by making the rasterization process differentiable. Early approaches, such as Kato et al. [25] and Liu et al. [32], enabled differentiable optimization of solid polygonal surfaces, but they required very careful initialization for optimization to converge effectively and are mainly limited to object-level setups rather than large realistic scenes.

Instead, Held et al. [17] recently proposed to optimize (potentially transparent) triangles via volumetric rendering, therefore effectively replacing Gaussians with *triangles*. However, when their triangles are rendered in a game engine, a noticeable drop in visual quality occurs, as game engines assume triangles to be opaque. Moreover, Held et al. [17] outputs a *soup of triangles*, rather than a connected polygonal mesh, often needed for physics-based simulation.

**Key Contributions.** We introduce *MeshSplatting* to address all the aforementioned limitations: (i) an end-to-end optimization of mesh-based scene representations that retains visual quality while training  $2\times$  faster than current state-of-the-art methods; (ii) rather than a polygon soup, we generate a connected mesh by refining the vertex locations of a restricted Delaunay triangulation; (iii) triangles are naturally connected to each other, and quantities stored within vertices are smoothly interpolated across each triangle; (iv) the optimization is aware that the triangles should be opaque, and therefore allowing direct high-quality rendering in standard game engines (see Fig. 1), opening the door for classical techniques like the use of depth buffers and occlusion culling [1, 22].

MeshSplatting achieves higher visual fidelity and captures finer geometric detail compared to modern *mesh-based* novel-view synthesis approaches [14, 20, 53, 54]. It is the first method to reconstruct large-scale real-world meshes end-to-end, directly producing connected, opaque, and colored triangle meshes *without* post-hoc extraction. Our representation can be *directly* imported into standard game engines, enabling a wide range of downstream applications including physics-based simulation, interactive walkthroughs, ray tracing, and scene editing, some of which are illustrated in Figure 1.

## 2. Related work

Differentiable rendering enables end-to-end optimization by propagating image-based losses back to scene parameters, allowing for the learning of explicit representations such as point clouds [12, 25], voxel grids [10], polygonal meshes [25, 32, 33], and more recently, Gaussian prim-

itives [26]. The advent of 3D Gaussian Splatting [26] showed that it is possible to fit millions of anisotropic Gaussians in minutes, enabling real-time rendering with high fidelity. Since then, various directions have been explored to improve the Gaussian primitive, including the use of 2D Gaussians [20], generalized Gaussians [15, 45], alternative kernels [21], and learnable basis functions [6]. Other works moved beyond Gaussians entirely, investigating different primitives such as smooth 3D convexes [16], linear primitives [47], sparse voxel fields [44], or radiance foams [11]. More recently, Held et al. [17] advocated for the comeback of triangles, the most classical primitive in computer graphics. Several researchers have explored this direction, proposing triangle-based representations for efficient scene modeling [4, 24]. However, existing triangle-based methods usually result in an unstructured *triangle soup*, with no connectivity between adjacent triangles, and they fail to produce opaque triangles at the end of training, limiting usability in downstream applications. We propose a method that enforces *connectivity*, resulting in a mesh of opaque triangles directly compatible with game engines.

**Mesh reconstruction from images.** Implicit and explicit methods have made significant progress in reconstructing 3D scenes, but they remain largely incompatible with traditional game engines that primarily rely on mesh-based rasterization with depth buffers. Some methods propose strategies to convert implicit radiance fields into meshes. BakedSDF [51] learns a neural signed distance field and appearance and then bakes them into a textured triangle mesh. Binary Opacity Fields [40] drives densities toward near binary opacities so surfaces can be extracted as a mesh, and MobileNeRF [7] distills a NeRF into a compact set of textured polygons. However, these methods introduce overhead and increase the overall training time.

Several methods have built upon 3DGS and proposed ways to extract a mesh from an optimized Gaussian scene. 2DGS [20] and RaDe-GS [54] rely on Truncated Signed Distance Fields for mesh extraction. Other approaches extract meshes by sampling a surface-aligned Gaussian level set followed by Poisson reconstruction [13], or by defining a Gaussian opacity level set and applying Marching Tetrahedra on Gaussian-induced tetrahedral grids [53]. All of these, however, treat mesh extraction as a *separate* post-processing step, decoupled from the optimization process.

More recently, MiLo [14] integrates surface mesh extraction directly into the optimization, jointly refining both the mesh and the Gaussian representation. However, while MiLo optimizes the mesh geometry during training, color still has to be learned separately. Another line of work employs differentiable meshes for 3D reconstruction [42, 43], but these methods primarily target synthetic objects and do not generalize to real-world scenes. In contrast, MeshSplatting directly optimizes opaque triangles together with their

vertex colors, making the result immediately compatible with any game engine without additional post-processing steps.

### 3. Methodology

We now overview the key components of our method. Section 3.1 reviews Triangle Splatting [17], which is used as volume-renderable primitives for differentiable rendering in this work. We give an overview of our MeshSplatting representation in Section 3.2, and describe the optimization stages to convert the triangle soup into a connected mesh in Section 3.3. For the representation to be *natively* compatible with game engines, triangles need to be *opaque*, and the process to achieve this objective is detailed in Section 3.4. We conclude by detailing other optimization details in Section 3.5, such as densification/pruning, and training losses.

#### 3.1. Background

In Triangle Splatting [17], each triangle  $\mathbf{T}_m$  is defined by three vertices  $\mathbf{v}_i \in \mathbb{R}^3$ , a color  $\mathbf{c}_m$ , a smoothness parameter  $\sigma_m$  and an opacity  $o_m$ . The rasterization process begins by projecting each 3D vertex  $\mathbf{v}_i$  of a triangle  $\mathbf{T}_m$  onto the image plane using a standard pinhole camera model<sup>1</sup>. To determine the influence of a triangle on a pixel  $\mathbf{p}$  and make the splatting process differentiable, the *signed distance field*  $\phi$  of the 2D triangle in image space is defined as:

$$\phi(\mathbf{p}) = \max_{i \in \{1,2,3\}} L_i(\mathbf{p}), \quad L_i(\mathbf{p}) = \mathbf{n}_i \cdot \mathbf{p} + d_i, \quad (1)$$

where  $\mathbf{n}_i$  are the unit normals of the triangle edges pointing outside the triangle, and  $d_i$  are offsets such that the triangle is given by the zero-level set of the function  $\phi$ . The signed distance field  $\phi$  thus takes positive values outside the triangle, negative values inside, and equals zero on its boundary. The window function  $I$  is then defined as:

$$I(\mathbf{p}) = \left( \text{ReLU} \left( \frac{\phi(\mathbf{p})}{\phi(\mathbf{s})} \right) \right)^\sigma \quad (2)$$

with  $\mathbf{s} \in \mathbb{R}^2$  be the *incenter* of the projected triangle (*i.e.*, the point inside the triangle with minimum signed distance). Note how the indicator evaluates to 1 at the triangle incenter, 0 at the boundary and 0 outside the triangle.  $\sigma$  is a smoothness parameter that controls the transition between the incenter and boundary of the triangle. More specifically, as  $\sigma \rightarrow 0$ , the representation converges to a solid triangle, while larger values of  $\sigma$  yield a smooth window function that gradually increases from zero at the boundary to one at the center. This triangle parameterization has two main limitations: triangles remain isolated without vertex sharing, and treating  $\sigma$  and  $o$  as independent free parameters prevents them from becoming fully opaque after training.

<sup>1</sup>Unlike 3DGS, linearization of Gaussian projections is required [55], since perspective projection preserves linearity, 3D triangles remain triangles in 2D screen space.

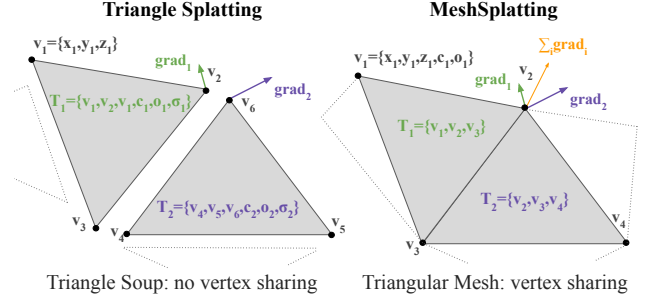


Figure 2. **Mesh parametrization.** (left) In a triangle soup, each triangle  $\mathbf{T}_m$  is defined independently by three vertices  $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ , a color  $\mathbf{c}_m$ , a smoothness parameter  $\sigma_m$ , and an opacity  $o_m$ , without sharing vertices with neighboring triangles. (right) MeshSplatting parameterize a triangle  $\mathbf{T}_m$  through a shared vertex set, where each vertex  $\mathbf{v}_i$  stores  $x_i, y_i, z_i, c_i$ , and  $o_i$ . Each triangle is defined by the three indices in the vertex set that compose it. During the backward pass, gradients from all adjacent triangles are accumulated at shared vertices. The smoothness parameter  $\sigma$  is shared across all triangles.

#### 3.2. Vertex-sharing triangle representation

In MeshSplatting, we define our mesh vertices as

$$\mathcal{V} = \{\mathbf{v}_i \in \mathbb{R}^3 \mid i = 1, \dots, N\}, \quad (3)$$

with  $N$  denoting their cardinality. Similarly to Son et al. [43] each vertex is parameterized as  $\mathbf{v}_i = (x_i, y_i, z_i, c_i, o_i)$ , where  $(x_i, y_i, z_i) \in \mathbb{R}^3$  denotes its 3D position,  $\mathbf{c}_i \in \mathbb{R}^3$  the vertex color and  $o_i \in [0, 1]$  the vertex opacity.<sup>2</sup> A triangle is defined by a triplet of vertices  $\mathbf{T}_m = \{v_i, v_j, v_k\}$ , its opacity is set to  $o_{\mathbf{T}_m} = \min(o_i, o_j, o_k)$ , and its color at a point inside the triangle is obtained by interpolating vertex colors with *barycentric coordinates*. During differentiation, vertex positions, colors, and opacities therefore receive the accumulated gradients from all triangles connected to it, as shown in Figure 2 (right).

#### 3.3. From soups to meshes

Our optimization executes in two stages that *gradually* convert an unstructured into a structured representation, as illustrated in Figure 3. This design exploits the fact that, early in training, unstructured representations are *easier to optimize*, as they impose less constraints on the representation.

**Stage 1. Triangle soup optimization .** We start by taking as input a set of posed images and corresponding camera parameters obtained from structure-from-motion (SfM) [41], which also provides a sparse point cloud. For each 3D point, we initialize an equilateral triangle centered at that point, with its size proportional to the average distance to its three nearest neighbors, and random orientation. All triangles are initially defined as semi-transparent (we selected  $o_i=0.28$  via hyper-parameter tuning).

<sup>2</sup>After training, the opacity parameter is *discarded*, and all triangles are treated as fully opaque for compatibility with standard game engines.



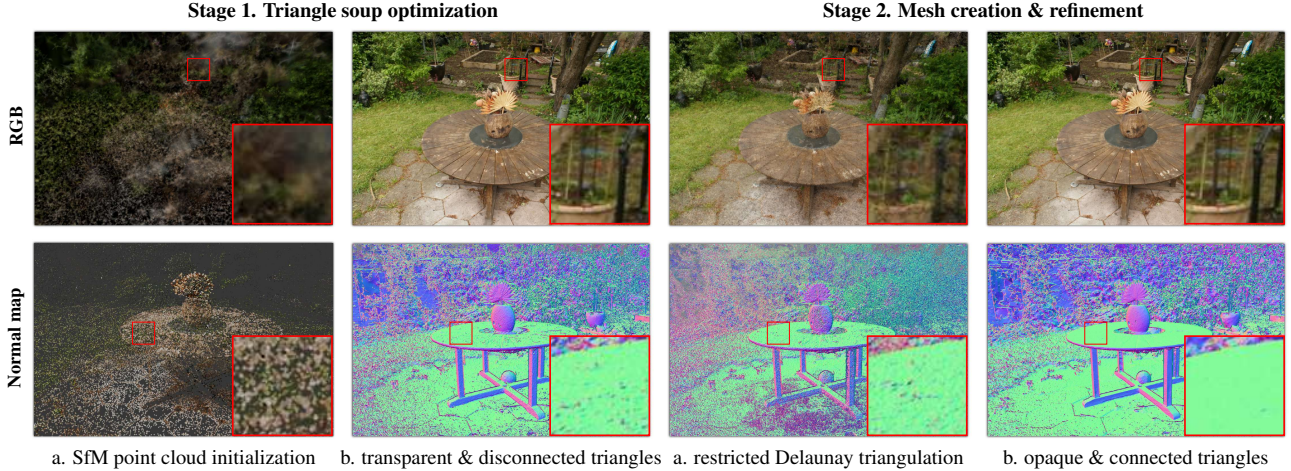


Figure 3. **From triangle soups to meshes.** (1a) We initialize semi-transparent triangles and scale them based on local density. (1b) We optimize a semi-transparent triangle soup without shared vertices, leading to disconnected triangles. (2a) Applying restricted Delaunay triangulation restores global connectivity but introduces geometric artifacts and a loss of visual quality, as vertex colors no longer accurately align with the underlying geometry. (2b) The final fine-tuning stage refines the connected mesh, producing smooth surfaces, accurate geometry, and restoring the visual fidelity lost during triangulation. Using only opaque triangles, our method achieves high visual quality compared to the semi-transparent and isolated triangle soup.

We begin the optimization with this *unstructured* triangle soup, i.e. without any connectivity or manifold constraints between triangles. Each triangle is optimized independently and can move freely under the influence of image-space gradients. This unconstrained formulation behaves similarly to point-based splatting, enabling rapid coverage of the visible scene and fast adaptation to local geometry and appearance. Note that, while similar, this is not the same as executing Held et al. [17], as we optimize *interpolated* vertex quantities within a triangle, rather than keeping them uniform.

**Stage 2. Mesh creation & refinement.** To transform our triangle soup into a connected mesh, we execute a restricted Delaunay triangulation on the triangle soup [8]. This operation first compute a standard Delaunay tetrahedralization, and then identifies tetrahedral faces whose dual Voronoi edges intersect the surface of the input triangle soup. The restricted Delaunay triangulation generates a mesh that *approximates the surface*, while at the same time, *maintaining Delaunay properties* (high quality meshing) locally restricted to it. Note that, in contrast to other reconstruction [28, 34, 35], restricted Delaunay triangulation does not introduce new vertices or modify their position. Instead, it *reuses* the optimized vertices directly, preserving both spatial accuracy and learned appearance.

Given the mesh connectivity from our restricted Delaunay triangulation, we then continue optimization so to fine-tune both vertex locations, as well as their appearance. As vertices are shared among adjacent triangles, gradients from neighboring faces are accumulated at shared vertices, ensuring that each vertex is updated consistently according to all incident triangles; see Figure 2. We do not need to introduce additional facets or vertices, as the unstructured trian-

gle soup optimization has already produced a sufficiently dense set of triangles to capture all spatial regions accurately. After this fine-tuning stage, we obtain a fully opaque *mesh* capable to render the scene at *high photometric quality* in conventional rendering engines. In the final iterations of training, we enable supersampling, allowing even small triangles to receive gradients and be properly optimized.

### 3.4. Optimizing meshes with opaque triangles

Optimizing meshes composed only of *opaque* triangles introduces new challenges compared to classical NeRF/3DGS optimization. To enable gradient propagation through occlusions and allow the representation to be optimized effectively, the representation must remain semi-transparent in the early phases of training. There are two degrees of freedom that we can control in this regard: the per-vertex opacity parameter  $o$ , and the smoothness parameter  $\sigma$ .

**Opacity parameter scheduling.** We optimize opacity freely during the initial  $5k$  iterations. Subsequently, we reparameterize opacity so that the optimizer is encouraged to make triangles more opaque. We achieve this by reparameterizing opacity values as

$$o'(o) = O_t + (1 - O_t) \cdot \text{sigm}(o), \quad (4)$$

where  $\text{sigm}(\cdot)$  is the sigmoid function, and scheduling  $O_t$  over time. Note that if  $O_t = 0$  the sigmoid parameterizes opacities smoothly between 0 and 1. Conversely, if  $O_t = 1$  all the opacities are mapped to a value of 1 (fully opaque triangles). By linearly increasing  $O_t$  from 0 to 1 during optimization, we can control this behavior smoothly.

**Window parameter scheduling.** We initialize the window parameter  $\sigma=1.0$ , corresponding to a linear transition

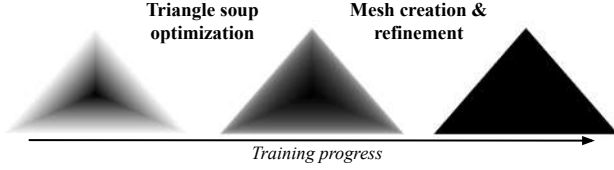


Figure 4. **Window parameter scheduling.** To ensure stable gradient flow during training, we begin with smooth triangles ( $\sigma=1.0$ , left) and linearly decrease  $\sigma$  throughout training, resulting in sharper triangles by the end. We visualize  $\sigma$  for a prototypical triangle at the *beginning* and *end* of each optimization stage.

from the incenter to the triangle boundary, and treat it as a *single parameter*, shared across all triangles. Throughout the training stages detailed in Section 3.3,  $\sigma$  is linearly annealed from  $1.0 \rightarrow 0.0001$ , ensuring strong gradient flow in the early stages while converging to opaque triangles at the end of the optimization process; see Figure 4. Note that this is in direct contrast to Triangle Splatting, as Held et al. [17] lets every triangle optimize its own window parameter.

### 3.5. Optimization details

We now detail our densification/pruning strategies, training losses, and observations about our rendering process.

**Densification.** As the initial triangle soup may not be sufficiently dense, we adapt the ideas from 3DGS-MCMC [29] to spawn additional triangles. At each densification step, candidate triangles are selected by sampling from a probability distribution constructed directly from their opacity  $o$  using Bernoulli sampling. Following Held et al. [17], new triangles are generated through *midpoint subdivision*: the midpoints of the three edges of a selected triangle are connected, splitting it into four smaller triangles. The new midpoints are added to the vertex set  $\mathcal{V}$  and assigned the average color and opacity of their two adjacent vertices. We already exploit the connectivity in the early stages, as it drastically reduces the number of newly created vertices, resulting in only 6 new vertices after a split when connected, compared to 12 in a triangle-soup setting.

**Pruning.** At iteration  $5k$  (Stage 1, just before opacity scheduling starts), we prune all triangles with opacity  $o < 0.2$ , eliminating roughly 70% of primitives. During the rest of Stage 1, we monitor the *maximum* volume rendering blending weight  $w = T \cdot o$  across views, and prune whenever  $w < O_t$ , hence eliminating *occluded* triangles as the representation becomes more opaque. While pruning is disabled during Stage 2, we perform a final pruning pass over all training views at the end of training to remove triangles that were never rendered.

**Training losses.** We optimize the 3D vertex positions  $\mathbf{v}_i$ , opacity  $o_i$ , and spherical harmonic color coefficients  $\mathbf{c}_i$  of all vertices. Our training loss combines the photometric  $\mathcal{L}_1$  and  $\mathcal{L}_{D-SSIM}$  terms from 3DGS [26], the opacity loss  $\mathcal{L}_o$

from Kheradmand et al. [29], the *depth alignment loss*  $\mathcal{L}_z$  (detailed below), the normal loss  $\mathcal{L}_n$  from [20], and the depth loss  $\mathcal{L}_d$  from Kerbl et al. [27]:

$$\mathcal{L} = \mathcal{L}_{3DGS} + \beta_o \mathcal{L}_o + \beta_z \mathcal{L}_z + \beta_n \mathcal{L}_n + \beta_d \mathcal{L}_d. \quad (5)$$

We follow Kerbl et al. [27], and employ Depth Anything v2 [50] to align the predicted depths using their scale-and-shift procedure. The normal loss  $\mathcal{L}_n$  can be supervised either by an external normal estimation network [18], or the self-supervised normal regularization from 2DGS [20], or both. We employ both supervision sources in all experiments, except for the mesh quality evaluation (Section 4.3), where we demonstrate that MeshSplatting also performs effectively even in a *self-supervised* setting.

**Depth alignment loss.** To promote the creation of manifolds, we align triangles to the observed depth map using a vertex-to-surface depth loss. To achieve this, for each rendered vertex  $v_i$  with predicted depth  $z_i$  and screen coordinates  $(x_i, y_i)$ , we sample the predicted depth  $z_i^*$  from the rendered depth map, and (robustly via L1 losses) penalize the depth difference:  $\mathcal{L}_z = \frac{1}{N} \sum_{i=1}^N |z_i - z_i^*|$ . Unlike losses like normal consistency [3, 9] or Laplacian [3, 9, 48] that rely on local mesh connectivity, this formulation acts on each vertex independently.

**Rendering equation.** The final color of each image pixel  $\mathbf{p}$  is computed by accumulating contributions from all overlapping triangles in depth order:  $C(\mathbf{p}) = \sum_{n=1}^N \mathbf{c}_{T_n} o_{T_n} I(\mathbf{p}) \left( \prod_{i=1}^{n-1} (1 - o_{T_i} I(\mathbf{p})) \right)$ . Analogously to Chen et al. [7], at the end of training, this simplifies to  $C(\mathbf{p}) = \mathbf{c}_{T_n} I(\mathbf{p})$ , so that only a *single* evaluation per pixel is required, significantly accelerating the rendering process (i.e. *over-drawing is zero*).

## 4. Experiments

We compare our method to concurrent approaches on Mip-NeRF360 [2] and Tanks and Temples (T&T) [31]. We evaluate the visual quality using standard metrics: SSIM, PSNR, and LPIPS. Our the total number of used vertices and training times are reported on an NVIDIA A100 (40GB). Following Guédon et al. [14], we focus our evaluation on the task of *Mesh-Based Novel View Synthesis*, whose evaluation protocol attempts to measure how well reconstructed meshes reproduce complete scenes. Existing datasets such as DTU, MipNeRF360, and Tanks&Temples are limited: MipNeRF360 lacks ground-truth geometry, DTU only contains simple objects, and T&T provides sparse annotations with only foreground regions. As a solution, we measure novel view synthesis quality through the visual consistency between mesh renderings and reference images, capturing (i) geometric alignment and surface artifacts, (ii) mesh completeness, and (iii) background reconstruction, even when ground-truth geometry is unavail-

|                           | Characteristics |       |         |       | Mip-NeRF360 dataset |                    |                   |                 | Tanks & Temples   |                    |                   |                 |
|---------------------------|-----------------|-------|---------|-------|---------------------|--------------------|-------------------|-----------------|-------------------|--------------------|-------------------|-----------------|
|                           | Mesh            | Color | Connect | Ready | PSNR <sup>↑</sup>   | LPIPS <sup>↓</sup> | SSIM <sup>↑</sup> | V  <sup>↓</sup> | PSNR <sup>↑</sup> | LPIPS <sup>↓</sup> | SSIM <sup>↑</sup> | V  <sup>↓</sup> |
| 3DGS[26]                  | –               | –     | –       | ✗     | 27.21               | 0.214              | 0.815             | –               | 23.14             | 0.183              | 0.841             | –               |
| Triangle Splatting [17]   | –               | –     | –       | ✗     | 27.16               | 0.191              | 0.814             | –               | 23.14             | 0.143              | 0.857             | –               |
| 2DGS [20]                 | ✗               | ✗     | ✓       | ✓     | 15.36               | 0.474              | 0.498             | 2M              | 14.23             | 0.485              | 0.569             | 16M             |
| GOF [53]                  | ✗               | ✗     | ✓       | ✓     | 20.78               | 0.465              | 0.573             | 33M             | 21.69             | 0.326              | 0.690             | 12M             |
| RaDe-GS [54]              | ✗               | ✗     | ✓       | ✓     | 23.56               | 0.361              | 0.668             | 31M             | 20.51             | 0.344              | 0.659             | 10M             |
| MiLo [14]                 | ✓               | ✗     | ✓       | ✓     | 24.09               | 0.323              | 0.688             | 7M              | 21.46             | 0.348              | 0.706             | 4M              |
| Triangle Splatting [17] † | ✓               | ✓     | ✗       | ✓     | 21.05               | 0.462              | 0.558             | 3M              | 17.27             | 0.402              | 0.600             | 6M              |
| <b>MeshSplatting</b>      | ✓               | ✓     | ✓       | ✓     | 24.78               | 0.310              | 0.728             | 3M              | 20.52             | 0.287              | 0.745             | 2M              |

Table 1. **Mesh-based novel view synthesis on the Mip-NeRF360 dataset.** MeshSplatting significantly outperforms all concurrent methods both in visual quality and in compactness, requiring far fewer vertices to achieve superior results. *Mesh* indicates whether a method directly produces a mesh (vs. requiring post-processing). *Color* denotes whether the mesh is already colored or requires some form of post-processing (e.g., coloring by fine-tuning). *Connect* specifies whether the final mesh consists of a connected component. *Ready* means the output is directly usable in standard game engines without custom rendering shaders. † with only opaque triangles.

able. Finally, to quantitatively assess the mesh quality of our method, we compute the Chamfer Distance on DTU.

**Baselines.** We compare our method against Triangle Splatting [17], and meshes derived from MiLo [14], 2DGS [20], Gaussian Opacity Fields (GOF) [53], and RaDe-GS [54]. For MiLo, surface mesh extraction is integrated into the optimization itself, so no additional post-processing is required. In contrast, 2DGS, GOF, and RaDe-GS rely on mesh extraction as a post-processing operation. Furthermore, *all* these methods require an additional post-processing stage to color the mesh, achieved by training a neural color field for  $5k$  iterations; see MiLo for details [14]. For Triangle Splatting, we use the *opaque*<sup>†</sup> triangles version, as it produces *game engine* outputs without additional post-processing needed. For reference, we also compare against 3DGS [26] to highlight the contrast in rendering quality between volumetric and mesh-based novel view synthesis. Qualitatively, we compare against current state-of-the-art MiLo [14] and Triangle Splatting [17], which represents the closest line of work to ours.

**Implementation details.** We set the spherical harmonics to degree 3, which yields 51 parameters per vertex (48 from the SH coefficients and 3 from the vertex position) and 3 parameters per triangle (the vertex indices). In comparison, a single 3D Gaussian requires 59 parameters.

#### 4.1. Mesh-based NVS – Table 1 and Figure 5

Table 1 reports quantitative results on the Mip-NeRF360 and Tanks & Temples datasets for mesh-based novel view synthesis. Compared to 2DGS and Triangle Splatting, our method uses a similar number of vertices, yet it achieves a 4–10 dB higher PSNR and a significantly lower LPIPS. Compared to GOF, RaDe-GS and MiLo, MeshSplatting uses 2 to 10 times fewer vertices while obtaining significantly higher SSIM and lower LPIPS. In terms of LPIPS (the metric that best correlates with human visual perception), MeshSplatting significantly outperforms all concurrent methods.

| Method               | Train ↓ | FPS ↑ (HD) | FPS ↑ (Full HD) | Memory ↓ |
|----------------------|---------|------------|-----------------|----------|
| GOF                  | 74m     | OOM        | OOM             | 1.5GB    |
| RaDe-GS              | 84m     | OOM        | OOM             | 1.1GB    |
| MiLo                 | 106m    | 170        | 160             | 253MB    |
| <b>MeshSplatting</b> | 48m     | 220        | 190             | 100MB    |

Table 2. **Speed & memory on MipNeRF-360.** MeshSplatting achieves *faster training and lower memory usage* than concurrent methods. FPS were measured on a costumer MacBook M4.

On the Mip-NeRF360 dataset, our method achieves substantially higher PSNR than GOF, RaDe-GS and MiLo. On T&T, GOF and MiLo attain a higher PSNR but exhibit noticeably lower SSIM and higher LPIPS scores. This suggests that although GOF and MiLo produce highly detailed meshes, their renderings contain more artifacts, which degrade perceptual quality and lead to worse LPIPS and SSIM scores. We illustrate this qualitatively in Figure 5 and provide more examples in the *supplementary material*. MeshSplatting reconstructs fine structures and details more accurately and produces less noisy renderings.

Additionally, note that 2DGS, GOF, and RaDe-GS require two additional post-processing steps after training: first *extracting* a mesh, and then *coloring* it. MiLo directly outputs a mesh, but still relies on a post-processing stage to *texture* the mesh. These extra steps limit the practicality of such methods, and increase their overall complexity. In contrast, we *directly produce colored opaque meshes that are immediately compatible* with any game engine, *without* requiring additional steps.

#### 4.2. Training speed & memory – Table 2

We report the average training time and final mesh size on the Mip-NeRF360 dataset. MeshSplatting trains in only 48 minutes on average, achieving a 35–55% speedup over concurrent mesh-yielding methods. Our optimized restricted Delaunay triangulation runs in under two minutes, contributing negligibly to the total training time. MiLo performs Delaunay triangulation at *every iteration*, whereas we run it only once. This results in a training time of



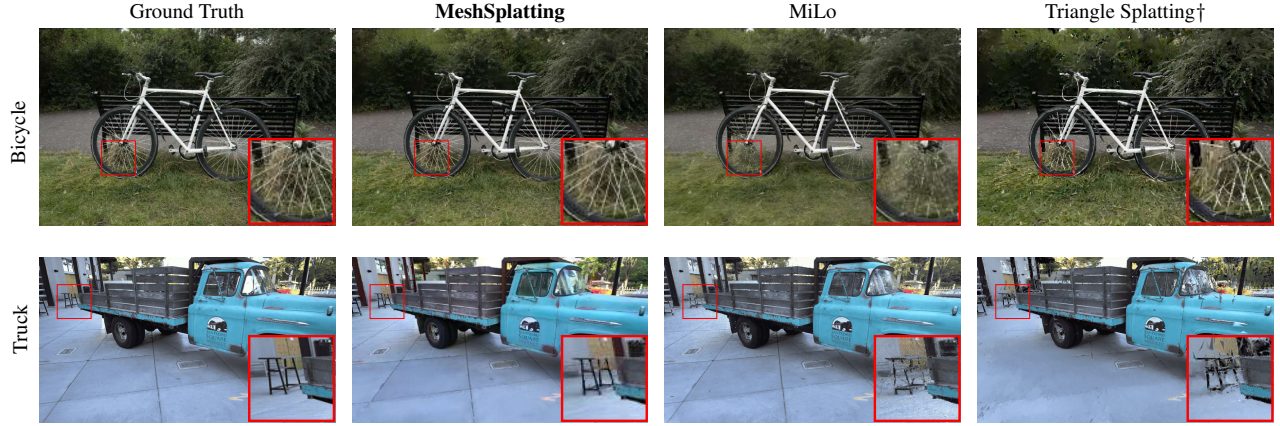


Figure 5. **Qualitative results.** Comparison of our method with ground truth, the current state-of-the-art MiLo [14] and opaque<sup>†</sup> Triangle Splatting [17]. Our approach produces renderings that are *closer to the ground truth, with sharper details and finer structures* (see the *Bicycle* spokes), and with fewer artifacts (see the table in the *Truck* scene). More visualizations are available in the *supplementary material*.

106 minutes for MiLo, compared to just 48 minutes for MeshSplatting. The final mesh representation produced by MeshSplatting is substantially more compact, with only 100 MB, which corresponds to a  $2.5\text{--}15\times$  reduction compared to concurrent methods. GOD and RaDe-GS require 1.5 GB and 1.1 GB of memory, respectively, making them impractical for real-world applications. This compactness enables MeshSplatting to run efficiently even on consumer hardware, significantly opening its applicability to lightweight rendering and real-time use-cases. MeshSplatting renders  $\approx 25\%$  faster on a consumer M4 MacBook compared to existing approaches. GOF and RaDe-GS exceed the device’s memory capacity, resulting in an out-of-memory error.

### 4.3. Surface reconstruction

For evaluation on the DTU dataset, we use only self-supervised regularization ( $\beta_d$  to zero) to ensure a fair comparison with other methods and to show that MeshSplatting also performs well in a self-supervised setting. Although we designed MeshSplatting for mesh-based novel-view synthesis in large and complex real scenes, rather than surface reconstruction, it achieves mesh quality comparable to concurrent methods. Across the 15 scenes, MeshSplatting attains the lowest Chamfer distance in 5 of them. A detailed comparison table is provided in the *supplementary material*.

### 4.4. Applications

We now illustrate a few application demos that are quite difficult to implement with semi-transparent representations, yet almost trivial once the scene is represented with opaque meshes like in MeshSplatting.

**Physical simulation.** As our representation contains no semi-transparent primitives, the triangles may directly be interpreted as hard surfaces for the purpose of physical simulation. We demonstrate this by using an off-the-shelf non-convex mesh collider: the one provided in the *Unity game*

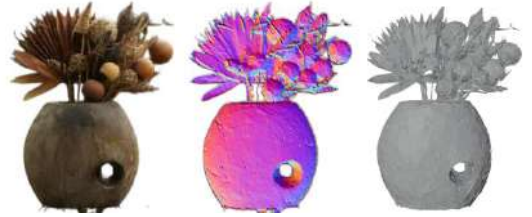


Figure 6. **Object segmentation.** With MeshSplatting, objects can be easily extracted or removed from a reconstructed scene, in this case the flower pot from the *Garden* scene. From left to right: generated RGB image of the object, estimated surface normals, and resulting mesh representation.

*engine*. With no post-processing, our mesh can be loaded into Unity, and used for physics interaction with dynamic objects and characters. Figure 1 showcases a selection of downstream applications, with additional visualizations in the *supplementary material*.

**Object segmentation.** Current 3D Gaussian Splatting methods for object extraction or removal [5, 52] face a fundamental challenge: a single pixel is rendered by accumulating the contributions of many primitives. This makes it non-trivial to decide whether a primitive belongs to a given object. To address this, prior work learns object associations during optimization [5, 52]. In contrast, MeshSplatting requires no additional tweak. Since each pixel is covered by exactly one triangle, mapping objects from image space to 3D space becomes straightforward: given a 2D mask of an object, all triangles contributing to pixels within that mask are directly identified as part of the object. By iterating over all training views, we recover the complete set of triangles belonging to the object. The object masks are generated using SAMv2 [39], which enables the selection of single or multiple objects that can then be removed or extracted from a scene with minimal complexity. Figure 6 shows a qualitative example of the extracted *flower pot* from the *Garden* scene of the Mip-NeRF360 dataset. Beyond object re-

|                     | 0  | 1   | 2   | 3   | 4+  | Mean |
|---------------------|----|-----|-----|-----|-----|------|
| Restricted Delaunay | 1% | 2%  | 5%  | 11% | 81% | 6.1  |
| With pruning        | 9% | 22% | 25% | 19% | 25% | 2.5  |
| Final mesh          | 2% | 9%  | 16% | 20% | 53% | 3.7  |

Table 3. **Connectivity.** Distribution of triangle connectivity on the *Garden* scene. The final mesh mostly consists of triangles connected to three or more neighboring triangles, indicating a well-connected mesh.

|                     | PSNR $\uparrow$ | LPIPS $\downarrow$ | SSIM $\uparrow$ |
|---------------------|-----------------|--------------------|-----------------|
| <b>Baseline</b>     | <b>24.78</b>    | <b>0.31</b>        | <b>0.728</b>    |
| w/o SH              | -2.07           | +0.06              | -0.069          |
| w/o $\mathcal{L}_d$ | +0.05           | -0.04              | +0.006          |
| w/o $\mathcal{L}_z$ | +0.02           | -0.01              | +0.002          |
| w/o $\mathcal{L}_n$ | +0.10           | -0.02              | +0.004          |

Table 4. **Ablations (Mip-NeRF360).** We assess the impact of each design choice by removing each one of them individually.

moval and extraction, this capability allows the scene to be segmented into distinct sub-meshes by disconnecting triangles belonging to different objects, enabling structured and object-aware 3D scene editing.

#### 4.5. Analysis

**Mesh connectivity – Table 3.** We report the distribution of triangles according to their connectivity with neighboring triangles. After the restricted Delaunay triangulation, most triangles are already well connected, with about 92% having at least three neighbors. If pruning is applied immediately after this step, a large number of triangles are removed because the Delaunay mesh contains many very small triangles that do not contribute to rendering, significantly reducing the overall connectivity. Instead, pruning is performed only after training. We make a final pass over all training views and remove all triangles that did not contribute to any rendered image. On average, each triangle is connected to approximately 3.7 triangles, and fewer than 2% of triangles remain isolated. Overall, most triangles are connected to exactly three neighboring triangles.

**Ablations – Table 4 and Figure 7.** While the regularization terms  $\mathcal{L}_d$ ,  $\mathcal{L}_z$ , and  $\mathcal{L}_n$  slightly reduce visual quality, they significantly improve geometric accuracy and yield smoother surfaces. The decrease in visual fidelity arises because the spherical harmonics representation cannot fully compensate for the reduced geometric flexibility, nor can it “cheat” by positioning triangles in non-physical configurations to better reproduce local colors or textures. This is further confirmed by replacing spherical harmonics with simple RGB colors, which results in an average drop of about 2 PSNR. This highlights that expressive color representations are essential for maintaining high visual fidelity with fully opaque triangles. Future work could decouple geometry and appearance. By employing a more expressive appearance model, such as neural textures, the representation

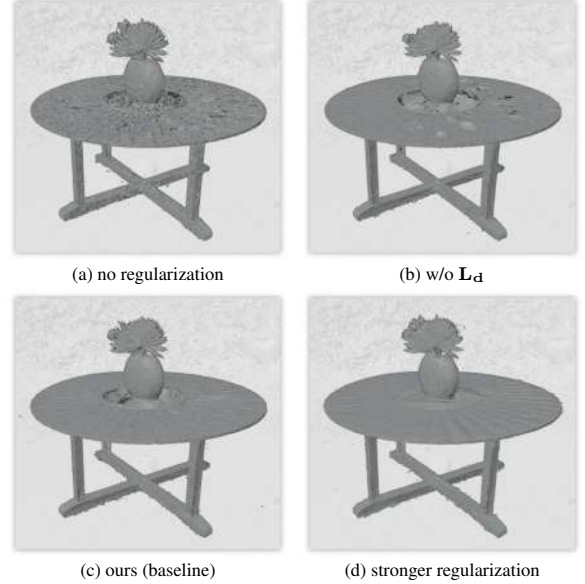


Figure 7. **Regularization vs. mesh quality.** (a) Without any regularization, the rendered views have high visual quality, but the underlying geometry is inaccurate. (b) The normal loss  $\mathcal{L}_n$  encourages smoother surfaces, yet without the depth regularization  $\mathcal{L}_d$ , a few local regions show minor geometric inaccuracies. (c) Our baseline model achieves both smooth and geometrically consistent surfaces. (d) Increasing the regularization strength yields even smoother geometry, but the visual fidelity decreases as spherical harmonics fail to capture fine appearance details.

could achieve smoother and more accurate geometry without losing appearance information, thereby preserving high visual fidelity even under strong smoothness constraints. More ablations are provided in the *supplementary material*.

## 5. Conclusions

We introduce a differentiable framework for end-to-end optimization of mesh-based scene representations. By reformulating the triangle parameterization to enable vertex sharing, our method produces connected meshes while maintaining high visual fidelity. A redefined training strategy moves the optimization toward opaque triangles and connectivity, resulting in a unified representation that combines high quality appearance and accurate geometry within a compact, real-time-renderable mesh. MeshSplatting bridges radiance field optimization with traditional graphics pipelines, paving the way for the practical integration of neural scene representations into interactive VR applications, game engines, and simulation environments.

**Acknowledgments.** We thank Bernhard Kerbl and George Kopanas for their helpful feedback and for proofreading the paper. J. Held is funded by the F.R.S.-FNRS. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.

## References

- [1] Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering*. AK Peters/CRC Press, Boca Raton, Florida, USA, 4 edition, 2018. 1, 2
- [2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5460–5469, New Orleans, LA, USA, 2022. 5
- [3] Mark Boss, Zixuan Huang, Aaryaman Vasishta, and Varun Jampani. SF3D: Stable fast 3D mesh reconstruction with UV-unwrapping and illumination disentanglement. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16240–16250, Nashville, TN, USA, 2025. 5
- [4] Nathaniel Burgdorfer and Philippos Mordohai. Radiant triangle soup with soft connectivity forces for 3D reconstruction and novel view synthesis. *arXiv*, abs/2505.23642, 2025. 2
- [5] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3D Gaussians. In *AAAI Conf. Artif. Intell.*, pages 1971–1979, Philadelphia, PA, USA, 2025. 7
- [6] Haodong Chen, Runnan Chen, Qiang Qu, Zhaoqing Wang, Tongliang Liu, Xiaoming Chen, and Yuk Ying Chung. Beyond Gaussians: Fast and high-fidelity 3D splatting with linear kernels. *arXiv*, abs/2411.12440:1–14, 2024. 2
- [7] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16569–16578, Vancouver, Can., 2023. 2, 5, 3
- [8] Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Shewchuk. *Delaunay Mesh Generation*. Chapman and Hall/CRC, 2013. 4
- [9] Jaehoon Choi, Rajvi Shah, Qinbo Li, Yipeng Wang, Ayush Saraf, Changil Kim, Jia-Bin Huang, Dinesh Manocha, Suhil Alsian, and Johannes Kopf. LTM: Lightweight textured mesh extraction and refinement of large unbounded scenes for efficient storage and real-time rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5053–5063, Seattle, WA, USA, 2024. 5
- [10] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinlong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5491–5500, New Orleans, LA, USA, 2022. 2
- [11] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. *arXiv*, abs/2502.01157, 2025. 2
- [12] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kauffmann Publ. Inc., San Francisco, CA, USA, 2007. 2
- [13] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5354–5363, Seattle, WA, USA, 2024. 2
- [14] Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. MLo: Mesh-in-the-loop Gaussian splatting for detailed and efficient surface reconstruction. *ACM Trans. Graph.*, 44(6):1–15, 2025. 2, 5, 6, 7, 3
- [15] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. GES: Generalized exponential splatting for efficient radiance field rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 19812–19822, Seattle, WA, USA, 2024. 2
- [16] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Delière, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21360–21369, Nashville, TN, USA, 2025. 2
- [17] Jan Held, Renaud Vandeghen, Adrien Delière, Daniel Hamdi, Abdullah Rebain, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. In *Int. Conf. 3D Vis. (3DV)*, pages 1–10, Vancouver, Can., 2026. 2, 3, 4, 5, 6, 7
- [18] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):10579–10596, 2024. 5
- [19] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi. *ACM Trans. Graph.*, 38(6):1–16, 2019. 1
- [20] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conf. Pap.*, pages 1–11, Denver, CO, USA, 2024. 2, 5, 6, 3
- [21] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable radial kernel splatting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21513–21523, Nashville, TN, USA, 2025. 2
- [22] John Hughes, Andries van Dam, Morgan McGuire, David Sklar, James D. Foley, Steven Feiner, and Kurt Akeley. *Computer Graphics: Principles and Practice*. Addison-Wesley, 3 edition, 2014. 1, 2
- [23] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 406–413, Columbus, OH, USA, 2014. 3
- [24] Changjian Jiang, Kerui Ren, Linning Xu, Jiong Chen, Jiangmiao Pang, Yu Zhang, Bo Dai, and Mulin Yu. HaloGS: Loose coupling of compact geometry and Gaussian splats for 3D scenes. *arXiv*, abs/2505.20267, 2025. 2
- [25] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *IEEE/CVF Conf. Comput. Vis. Pat-*



- tern Recognit. (CVPR)*, pages 3907–3916, Salt Lake City, UT, USA, 2018. 2
- [26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):1–14, 2023. 1, 2, 5, 6, 3
- [27] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3D Gaussian representation for real-time rendering of very large datasets. *ACM Trans. Graph.*, 43(4):1–15, 2024. 5
- [28] Muhammad Umar Karim Khan and Chong-Min Kyung. Poisson mixture model for high speed and low-power background subtraction. *Smart Sensors and Systems*, pages 1–23, 2020. 4
- [29] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as Markov chain Monte Carlo. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 80965–80986, Vancouver, Can., 2024. 5
- [30] KIRI Innovations. 3DGS Render: 3D Gaussian splatting renderer for Blender. <https://github.com/Kiri-Innovation/3dgs-render-blender-addon>, 2025. 1
- [31] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):1–13, 2017. 5
- [32] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 7707–7716, Seoul, South Korea, 2019. 2
- [33] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 154–169, Zürich, Switzerland, 2014. 2
- [34] Nissim Maruani, Roman Klovov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. VoroMesh: Learning watertight surface meshes with Voronoi diagrams. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 14519–14528, Paris, Fr., 2023. 4
- [35] Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. PoNQ: A neural QEM-based mesh representation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3647–3657, Seattle, WA, USA, 2024. 4
- [36] Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David I. W. Levin. Simplicitics: Mesh-free, geometry-agnostic elastic simulation. *ACM Trans. Graph.*, 43(4):1–11, 2024. 1
- [37] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. *ACM SIGGRAPH 2005 Courses*, page 7, 2005. 1
- [38] Aras Pranckevičius. UnityGaussianSplatting: 3D Gaussian splatting in Unity. <https://github.com/aras-p/UnityGaussianSplatting>, 2023. 1
- [39] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Juntao Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *Int. Conf. Learn. Represent. (ICLR)*, pages 1–44, Singapore, 2025. 7
- [40] Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *ACM Trans. Graph.*, 43(4):1–14, 2024. 2
- [41] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4104–4113, Las Vegas, NV, USA, 2016. 3
- [42] Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C. Lin, and Yi Zhou. DMesh: A differentiable mesh representation. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 12035–12077, Vancouver, Can., 2024. 2
- [43] Sanghyun Son, Matheus Gadelha, Yang Zhou, Matthew Fisher, Zexiang Xu, Yi-Ling Qiao, Ming C. Lin, and Yi Zhou. DMesh++: An efficient differentiable mesh for complex shapes. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 26590–26599, Honolulu, HI, USA, 2025. 2, 3
- [44] Cheng Sun, Jaesung Choe, Charles Loop, Wei-Chiu Ma, and Yu-Chiang Frank Wang. Sparse voxels rasterization: Real-time high-fidelity radiance field rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16187–16196, Nashville, TN, USA, 2025. 2
- [45] Maria Taktasheva, Lily Goli, Alessandro Fiorini, Zhen Li, Daniel Rebain, and Andrea Tagliasacchi. 3D Gaussian flats: Hybrid 2D/3D photometric scene reconstruction. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 1–26, San Diego, CA, USA, 2025. 2
- [46] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, Ilhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon,

- Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-Baeza. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods*, 17(3):261–272, 2020. 1
- [47] Nicolas von Lütow and Matthias Nießner. LinPrim: Linear primitives for differentiable volumetric rendering. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 1–27, San Diego, CA, USA, 2025. 2
- [48] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. *Computer Vision - ECCV 2018*, pages 55–71, 2018. 5
- [49] XVERSE. XScene-UEPlugin: 3D Gaussian splatting plugin for Unreal Engine 5. <https://github.com/xverse-engine/XScene-UEPlugin>, 2024. 1
- [50] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiao-gang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 1–37, Vancouver, Can., 2024. 5
- [51] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. BakedSDF: Meshing neural SDFs for real-time view synthesis. In *ACM SIGGRAPH Conf. Proc.*, pages 1–9, Los Angeles, CA, USA, 2023. 2
- [52] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3D scenes. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 162–179, 2024. 7
- [53] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Trans. Graph.*, 43(6):1–13, 2024. 2, 6, 3
- [54] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiao-Xiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *ACM Transactions on Graphics*, 45(2): 1–14, 2026. 2, 6, 3
- [55] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA splatting. *IEEE Trans. Vis. Comput. Graph.*, 8(3):223–238, 2002. 3



# MeshSplatting: Differentiable Rendering with Opaque Meshes

## Supplementary Material

### 5.1. Methodology.

**Mesh creation & refinement - Figure 8.** We provide an additional visualization that illustrates both the visual and geometric improvements achieved across the different stages. Figure 8 shows how the geometry evolves by rendering the mesh without color in *Blender*. Fine-tuning the output of the restricted Delaunay triangulation not only enhances visual fidelity but also leads to more accurate geometry.

**Restricted Delaunay triangulation.** As mentioned earlier, to construct the restricted Delaunay triangulation we first compute the standard Delaunay tetrahedralization of the input vertices, using the implementation in SciPy [46]. For each triangular face  $F$  in this tetrahedralization, we identify the two tetrahedra  $T_1^F$  and  $T_2^F$  that are adjacent to  $F$ ; faces on the boundary of the tetrahedralization that have only a single incident tetrahedron are discarded. We then compute the circumcenters of  $T_1^F$  and  $T_2^F$ , which are vertices of the dual Voronoi diagram of the Delaunay tetrahedralization. The dual edge associated with  $F$  is obtained by connecting the circumcenters of  $T_1^F$  and  $T_2^F$ .

After computing the dual edges  $E$  associated with the Delaunay faces, we determine which of these edges intersect the triangles in our optimized triangle soup. To perform intersection tests efficiently, we build a bounding volume hierarchy over these triangles. During traversal, when testing an edge in  $E$  against a BVH node, we first check for overlap between their axis-aligned bounding boxes. If the node is internal, we continue the traversal to its children; if it is a leaf node, we apply a precise ray-triangle intersection test based on the Möller-Trumbore algorithm [37]. Whenever an intersection is detected, we mark the corresponding Delaunay face as part of the output mesh. We implement this detection pipeline in Taichi [19], whose automatic CPU/GPU parallelization yields a substantial speedup.

**Hyperparameters - Table 5.** The code will be released together with all hyperparameters used, ensuring that all results are fully reproducible. For completeness, we also list the most important hyperparameters here.

### 5.2. Addition experimental results

**Detailed results - Table 6 & Table 7 & Table 8 & Figure 13 & Figure 14.** We provide the per-scene results of MeshSplatting to facilitate detailed comparison and reproducibility. In addition, we provide further qualitative results demonstrating that MeshSplatting produces renderings with fewer artifacts and less noise compared to the current state of the art, MiLo. While MiLo generates high-quality meshes and therefore achieves a strong PSNR on the T&T

|                           | Outdoor | Indoor  |
|---------------------------|---------|---------|
| feature learning rate     | 0.0016  | 0.004   |
| opacity learning rate     | 0.03    | 0.05    |
| vertex pos. learning rate | 0.0015  | 0.0015  |
| densification start       | 500     | 500     |
| densification end         | 10000   | 10000   |
| densification interval    | 500     | 500     |
| start pruning             | 4000    | 3000    |
| pruning threshold         | 0.235   | 0.235   |
| mesh creation             | 11000   | 11000   |
| $\beta_z$                 | 0.00025 | 0.00025 |
| $\beta_n$                 | 0.0001  | 0.0001  |
| $\beta_d$                 | 0.01    | 0.0     |
| $\beta_o$                 | 2e-06   | 0.0     |

Table 5. **Hyperparameters** List of the most important hyperparameters.

|                    | Truck | Train |
|--------------------|-------|-------|
| PSNR $\uparrow$    | 22.32 | 18.72 |
| LPIPS $\downarrow$ | 0.237 | 0.337 |
| SSIM $\uparrow$    | 0.799 | 0.693 |

Table 6. Per scene LPIPS, PSNR, and SSIM scores for the Truck and Train scenes of the T&T dataset.

|                    | Bicycle | Flowers | Garden | Stump | Treehill |
|--------------------|---------|---------|--------|-------|----------|
| PSNR $\uparrow$    | 23.04   | 19.34   | 24.70  | 24.78 | 20.53    |
| LPIPS $\downarrow$ | 0.348   | 0.417   | 0.217  | 0.316 | 0.428    |
| SSIM $\uparrow$    | 0.641   | 0.480   | 0.762  | 0.678 | 0.540    |

Table 7. Per-scene PSNR, LPIPS, and SSIM scores for outdoor MipNeRF360 scenes.

|                    | Room  | Counter | Kitchen | Bonsai |
|--------------------|-------|---------|---------|--------|
| PSNR $\uparrow$    | 28.52 | 26.51   | 27.42   | 28.19  |
| LPIPS $\downarrow$ | 0.271 | 0.279   | 0.227   | 0.294  |
| SSIM $\uparrow$    | 0.873 | 0.846   | 0.858   | 0.876  |

Table 8. Per-scene PSNR, LPIPS, and SSIM scores for indoor MipNeRF360 scenes.

dataset, its outputs exhibit more noise, leading to a higher LPIPS and a lower SSIM compared to MeshSplatting.

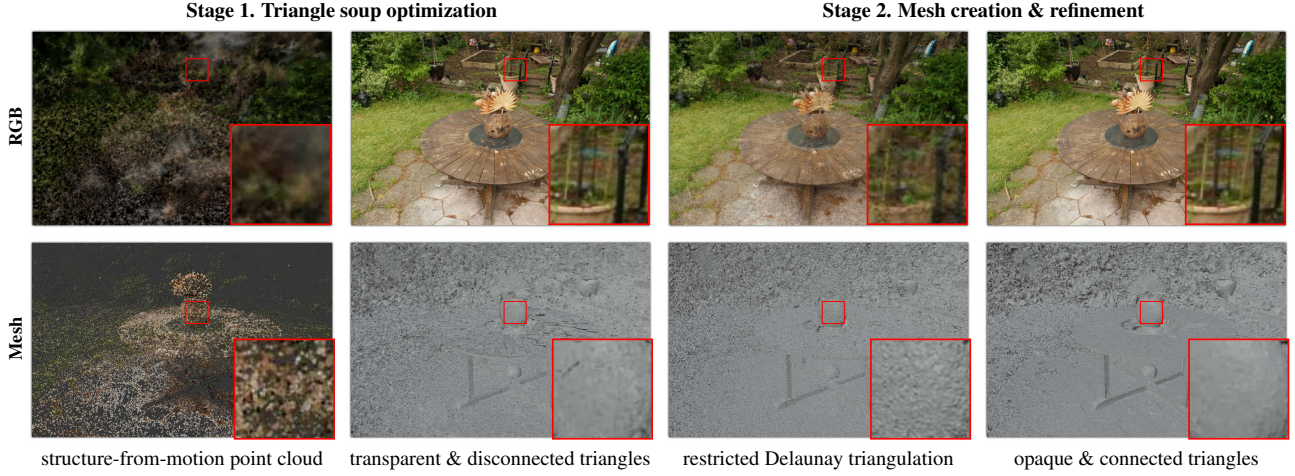


Figure 8. **From soups to triangle meshes.** The geometry improves significantly after refining the mesh.

**Surface reconstruction - Table 10.** MeshSplatting attains the best Chamfer distance on 5 of the 15 scenes. It achieves a significantly better mean score than Triangle Splatting and 3DGS. Compared to methods specifically tailored for mesh extraction, our results remain competitive and are on par with 2DGS. Even though our focus is mesh-based novel view synthesis, these results show that MeshSplatting also produces accurate surface meshes.

Fig. 9 shows the visual comparison of the reconstructed meshes with the current SOTA Milo. We acknowledge that our surface reconstruction metrics are slightly weaker than some surface reconstruction-focused methods, as our optimization is driven by rendering losses rather than explicit surface reconstruction objectives. Nevertheless, our geometric accuracy remains competitive with prior splatting-based approaches such as 2DGS, while significantly improving efficiency and memory. We agree that reducing surface noise and further improving geometric fidelity are important directions for future work, for example, by incorporating stronger geometric regularizers or topology-aware smoothing. Most downstream applications, such as physical interaction and ray tracing, remain fully compatible, since they rely on local surface geometry rather than strict manifoldness. As needed, standard mesh cleanup steps can be applied as lightweight post-processing.

|                    | 2M    | 3M    | 4M    | 5M    |
|--------------------|-------|-------|-------|-------|
| PSNR $\uparrow$    | 22.14 | +0.15 | +0.36 | +0.46 |
| LPIPS $\downarrow$ | 0.39  | -0.02 | -0.05 | -0.06 |

Table 9. **Number of vertices vs visual quality.** MeshSplatting scales effectively with the number of vertices, showing consistent improvements in visual quality as the vertex count increases. All improvements are shown relative to 2M.

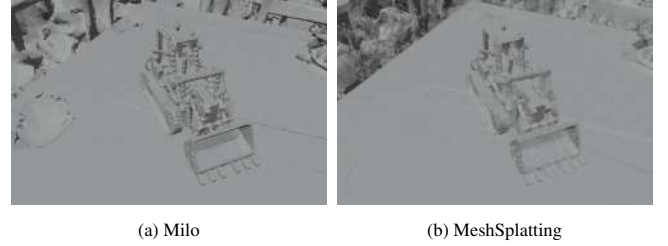


Figure 9. **Surface quality.** While  $2\times$  faster and using **50%** memory, MeshSplatting achieves competitive surface reconstruction. Please zoom in to see details.

**Impact of triangle count on visual quality - Table 9.** We evaluate the visual quality across different triangle counts on the outdoor scenes of MipNeRF360, as these scenes offer more room for improvement compared to indoor ones. Increasing the number of triangles consistently enhances visual quality, with clear gains in both PSNR and especially in LPIPS. The results show that MeshSplatting scales well with triangle count, with visual quality improving consistently as more triangles are used.

**Object extraction - Figure 10.** Given a 2D mask from SAMv2, we identify all visible triangles contributing to the masked pixels and aggregate them across all training views, yielding the complete set of triangles belonging to the selected object. The entire process takes up to two minutes, depending on the scene and the number of input views. Concretely, we first store the object masks for each training image, then iterate over all views to mark triangles as part of the object whenever they render at least one pixel within the mask. The extracted triangles can then be removed or exported as a standalone submesh without retraining.

### 5.3. Additional ablations - Table 11

**Hard pruning step.** An effective pruning strategy is crucial for achieving high visual quality with opaque primi-

| Method                  | 24   | 37   | 40   | 55   | 63   | 65   | 69   | 83   | 97   | 105  | 106  | 110  | 114  | 118  | 122  | Mean |
|-------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 3DGS [26]               | 2.14 | 1.53 | 2.08 | 1.68 | 3.49 | 2.21 | 1.43 | 2.07 | 2.22 | 1.75 | 1.79 | 2.55 | 1.53 | 1.52 | 1.50 | 1.96 |
| 2DGS [20]               | 0.48 | 0.91 | 0.39 | 0.39 | 1.01 | 0.83 | 0.81 | 1.36 | 1.27 | 0.76 | 0.70 | 1.40 | 0.40 | 0.76 | 0.52 | 0.80 |
| GOF [53]                | 0.50 | 0.82 | 0.37 | 0.37 | 1.12 | 0.74 | 0.73 | 1.18 | 1.29 | 0.68 | 0.77 | 0.90 | 0.42 | 0.66 | 0.49 | 0.74 |
| RaDe-GS [54]            | 0.46 | 0.73 | 0.33 | 0.38 | 0.79 | 0.75 | 0.76 | 1.19 | 1.22 | 0.62 | 0.70 | 0.78 | 0.36 | 0.68 | 0.47 | 0.68 |
| MiLo [14]               | 0.43 | 0.74 | 0.34 | 0.37 | 0.80 | 0.74 | 0.70 | 1.21 | 1.22 | 0.66 | 0.62 | 0.80 | 0.37 | 0.76 | 0.48 | 0.68 |
| Triangle Splatting [17] | 0.98 | 1.07 | 1.07 | 0.51 | 1.67 | 1.44 | 1.17 | 1.32 | 1.75 | 0.98 | 0.96 | 1.11 | 0.56 | 0.93 | 0.72 | 1.06 |
| <b>MeshSplatting</b>    | 0.77 | 0.72 | 0.74 | 0.60 | 0.89 | 1.00 | 0.81 | 1.09 | 1.19 | 0.58 | 0.68 | 0.93 | 0.63 | 0.66 | 0.59 | 0.79 |

Table 10. **Chamfer distance on the DTU dataset [23]**. MeshSplatting achieves performance comparable to concurrent methods.



Figure 10. **Object extraction.** Additional visual examples demonstrating the object extraction capabilities of MeshSplatting. From left to right: generated RGB image of the object, estimated surface normals, and resulting mesh representation.

|                          | PSNR $\uparrow$ | LPIPS $\downarrow$ | SSIM $\uparrow$ |
|--------------------------|-----------------|--------------------|-----------------|
| Baseline                 | 24.78           | 0.31               | 0.728           |
| w/o hard pruning         | -0.67           | +0.02              | -0.021          |
| w/o stage 2              | -8.56           | +0.25              | -0.260          |
| w/o supersampling        | -0.80           | +0.04              | -0.040          |
| w/o $w$ pruning          | -0.62           | +0.05              | 0.045           |
| w/o SH                   | -2.07           | +0.06              | -0.07           |
| prune w/o conn.          | -0.19           | +0.01              | -0.01           |
| w/o $\mathcal{L}_d$      | +0.05           | -0.04              | +0.006          |
| w/o $\mathcal{L}_z$      | +0.02           | -0.01              | +0.002          |
| w/o $\mathcal{L}_n$      | +0.10           | -0.02              | +0.004          |
| w/o sigma decay          | -7.96           | +0.27              | -0.329          |
| cosine opacity schedule  | -0.20           | +0.01              | -0.01           |
| cosine $\sigma$ schedule | -0.76           | +0.03              | -0.028          |

Table 11. **Detailed ablations (Mip-NeRF360).** We isolate the impact of each design choice by removing them individually.

tives. Without the hard pruning step at iteration  $5k$ , visual quality degrades noticeably. Unnecessary triangles remain in the scene and can no longer be removed.

**W/o stage 2.** Running restricted Delaunay triangulation directly on the SfM point cloud produces a connected mesh that is not yet geometrically consistent, leaving many regions, particularly in the background, uncovered. Training with only connected triangles restricts the degrees of freedom of both vertices and triangles, making optimization

considerably harder and leading to a drop in visual quality.

**W/o supersampling.** To reduce aliasing from opaque triangles, similar to Chen et al. [7], we render at  $s \times$  the target resolution, and then downsample to the final resolution using area interpolation, which averages over input pixel regions to implement a box anti-aliasing filter. By disabling supersampling during the final training iterations and testing, visual quality decreases. This occurs because the representation uses fully opaque triangles, and metrics such as PSNR penalize small pixel-level shifts. When supersampling followed by average downsampling is applied, transitions between triangles become smoother, leading to more continuous transitions and higher visual quality. We applied anti-aliased rendering for both training and testing.

**W/o  $w$  pruning.** When pruning is based only on *opacity* rather than the maximum blending weight  $w$ , many unnecessary triangles remain in the scene. As these triangles become opaque, they introduce artifacts and reduce visual quality.

**W/o SH.** When training with only RGB colors, the visual quality drops significantly. The opaque and connected triangles struggle to accurately reproduce fine texture details. Real-world scenes exhibit complex spatial variations in color and illumination, which would require an impractically large number of triangles to model using RGB alone. Incorporating spherical harmonics enables each triangle to capture part of this variation, resulting in a noticeably improved appearance. This observation suggests that a more expressive appearance model could further enhance visual fidelity. Future work could explore learning per-triangle textures either jointly during training or as a post-processing refinement, as done in recent mesh-based methods. Such an approach could further narrow the visual quality gap between 3D Gaussian-based representations and fully opaque triangle-based ones.

**Pruning w/o connectivity.** During the *first stage*, we exploit triangle connectivity by applying midpoint subdivision while ensuring that all four newly created triangles remain connected. This approach reduces the number of introduced vertices by half: subdividing each triangle would yield 12 vertices, whereas maintaining shared connectivity requires only 6. By connecting triangles, we can allocate a larger



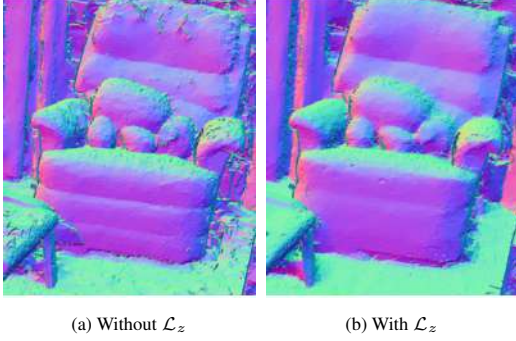


Figure 11. **Impact of the  $\mathcal{L}_z$  regularization term.** With the regularization we obtain smoother surfaces.

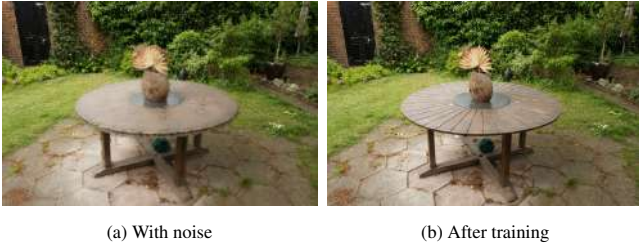


Figure 12. **Noisy vertices.** Even with geometric noises on the vertices, MeshSplatting remains robust and delivers high visual quality.

portion of the memory budget to rendering more vertices and triangles, which leads to improved visual quality.

**W/o  $\mathcal{L}_z$  - Figure 11.** Even with normal regularization, some triangles remain misoriented. By pushing the vertices closer to the surface, we improve both mesh smoothness and normal consistency. The vertex regularization term  $\mathcal{L}_{\text{vertex}}$  reduces artifacts and promotes smoother surfaces, particularly in background regions with weaker supervision.

**Cosine scheduler.** Compared to a linear schedule for increasing *opacity* and decreasing  $\sigma$ , we also experimented with a cosine scheduler. However, performance, particularly for  $\sigma$ , drops noticeably. The main reason is that the linear schedule maintains higher  $\sigma$  values for longer, allowing gradients to remain stable during the final iterations, whereas the cosine scheduler reaches low  $\sigma$  values earlier, causing gradients to vanish prematurely. Finally, we also analyze the impact of initialization. Instead of starting from soft triangles (i.e.,  $\sigma = 1.0$ ) that gradually converge toward solid ones, we evaluate the case where training begins directly with fully solid triangles. In this case, gradients can only flow through the opacity term, causing optimization to stagnate. As a result, both performance and visual quality degrade noticeably due to severe vanishing gradients.

**Noisy input to restricted Delaunay triangulation.** We analyze the impact of noisy vertex positions as input to the restricted Delaunay triangulation. We inject scale-normalized Gaussian noise with a standard deviation equal

to 1% of the scene extent into triangle-soup vertices prior to triangulation. Fig. 12 shows that even under strong geometric noise, the method exhibits only a minor performance degradation, with only +0.22 PSNR and -0.02 in LPIPS. In practice, before triangulation we already remove low-opacity triangles, and enforce depth and normal regularization’s that removes floating artifacts and large geometric outliers during the triangle-soup stage.

#### 5.4. Mesh connectivity.

The resulting mesh exhibits a *vertex-face ratio* of 0.48, which is close to the theoretical 0.5 expected for a closed manifold surface, indicating that the global mesh topology is already compact and near-manifold despite local non-manifold regions remaining. For reference, a completely isolated mesh with no vertices-sharing, would have a ratio of 1.5. We further analyze vertex connectivity by measuring the number of incident triangles (vertex valence) across the reconstructed mesh. The mean valence is 6.2 with a median of 5, which aligns with the expected value of 6 for regular triangular tessellations but with a broader distribution. Roughly 35% of vertices exhibit low valence ( $\leq 4$ ), indicating remaining boundary regions and small disconnected fragments, while about 30% fall within the regular interior range (5–8). A small fraction ( $< 10\%$ ) shows high valence ( $> 12$ ), corresponding to locally dense or merged zones. This distribution shows that while the mesh is largely connected and compact, it remains not fully watertight.

#### 5.5. Limitations

**Limitations.** MeshSplatting achieves high visual quality and accurate reconstruction in regions where the initial point cloud is dense. In contrast, background areas with sparse coverage still exhibit incomplete geometry and reduced fidelity. Moreover, when moving outside the orbit of training views, the visual quality degrades. While the softness and opacity of Gaussian-based approaches may still provide slightly plausible results in such cases, our use of opaque triangles makes the artifacts more pronounced. Furthermore, transparent objects such as glasses or bottles remain difficult to represent using only opaque triangles. Finally, MeshSplatting does not explicitly enforce watertightness; however, the resulting meshes can be directly used in many downstream applications, offering an effective trade-off between simplicity, usability, and high visual quality. Future work could incorporate additional regularization terms to constrain vertex motion and prevent self-intersections or overlaps.



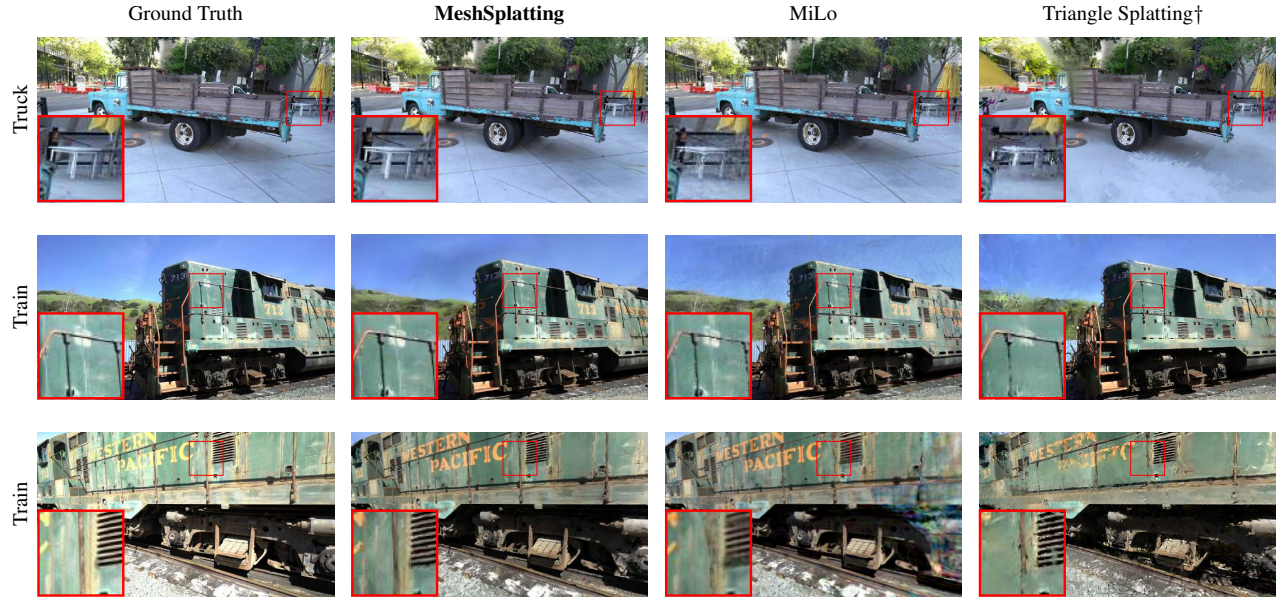


Figure 13. **More qualitative results on the *Tanks and Temples* dataset.** MeshSplatting reconstructs more details and finer structures compared to concurrent works. While MiLo achieves a higher PSNR, MeshSplatting produces fewer artifacts and less noisy renderings, which results in a significantly lower LPIPS and a higher SSIM.



Figure 14. **More qualitative results on the *MipNeRf-360*.** MeshSplatting is able to reconstruct fine details with only *opaque triangles*.





## 6. Conclusions and perspectives

### 6.1. This thesis

This thesis introduces a series of improved splatting representations for radiance field rendering, progressively moving from flexible volumetric primitives to explicit surface-based geometry. The central objective of this work was to bridge the gap between modern neural rendering approaches and classical computer graphics while maintaining the efficiency and differentiability of splatting-based rendering.

First, we introduced *3D Convex Splatting*, a radiance field representation based on smooth convex primitives. By replacing Gaussian primitives with adaptive convex shapes, this approach enables denser scene representations using fewer primitives and parameters. The resulting representation improves reconstruction quality, particularly on challenging scenes such as Tanks&Temples and indoor environments from the Mip-NeRF360 dataset. By combining the flexibility of convex geometry with efficient primitive-based rendering, 3D Convex Splatting demonstrates that richer geometric primitives can improve reconstruction accuracy.

Building on this idea, we continued with *Triangle Splatting*, a differentiable rendering technique that directly optimizes triangle primitives for novel view synthesis. By leveraging the fundamental primitive used in traditional mesh representations, this approach connects neural rendering with decades of GPU-accelerated graphics research. Triangle Splatting achieves high visual fidelity while maintaining fast rendering speeds, demonstrating that explicit surface primitives can serve as an effective alternative to volumetric or implicit representations. However, while the resulting triangle soups can already be rendered using standard graphics pipelines, generating a connected mesh still requires additional processing. Furthermore, large-scale outdoor scenes can occasionally produce floaters, as surface primitives receive less training supervision compared to volumetric representations that are visible from more viewpoints.

Finally, we introduced *MeshSplatting*, a differentiable framework for end-to-end optimization of connected mesh representations, representing the final step toward unifying neural rendering and classical computer graphics. By reformulating the triangle parameterization to enable vertex sharing, our method produces coherent meshes while preserving the high visual fidelity achieved by splatting-based rendering. A modified optimization strategy encourages opaque surfaces and mesh connectivity,

resulting in a unified representation that combines accurate geometry, compact storage, and real-time rendering performance. This representation further strengthens the connection between neural radiance field optimization and traditional graphics pipelines. To the best of our knowledge, it is the first end-to-end framework capable of reconstructing large-scale real-world scenes as connected, fully opaque, colored triangle meshes without relying on any postprocessing methods. This direct output enables immediate deployment in existing graphics ecosystems and unlocks a wide range of downstream applications.

Taken together, these contributions demonstrate a progression toward increasingly structured scene representations. Starting from convex volumetric primitives, moving to unstructured triangle surfaces, and ultimately arriving at connected mesh representations, this work explores how geometric structure can be integrated into splatting-based rendering frameworks. The results show that introducing explicit geometry can improve interpretability, efficiency, and compatibility with existing graphics tools while maintaining the differentiability required for modern optimization-based reconstruction methods.

Overall, this thesis highlights the potential of structured splatting representations as a bridge between neural rendering and classical computer graphics. By combining efficient differentiable rendering with explicit geometric primitives, these methods move toward practical scene representations that are both high quality and directly usable in real-time graphics pipelines.

## **6.2. So, what’s next?**

### **6.2.1. The future of splatting**

MeshSplatting opens the door to a range of promising research directions. First, the current formulation does not explicitly enforce watertightness. Future work could incorporate additional regularization terms to better constrain vertex motion and prevent self-intersections or overlaps, ultimately leading to cleaner and more physically consistent meshes.

Another interesting direction is to move towards a more natural and elegant approach to learning the connectivity of triangles during optimization. The model should be able to decide which triangles should be connected and which should remain separate during optimization. Recent concurrent work suggests that such strategies are do-able, at least at object-level reconstruction [8, 7].

Finally, improving the representation of appearance remains an important direction for future research. MeshSplatting currently relies on spherical harmonics, which are not ideal for modeling complex view-dependent effects and are not natively compatible with standard game engines. Learning explicit texture maps during

optimization could both improve visual fidelity and enable more seamless integration into existing graphics pipelines.

### 6.2.2. Novel view synthesis

A key challenge for achieving high-quality novel view synthesis has traditionally been the availability of dense training views. In particular, having a large number of images with accurate camera poses directly leads to significantly improved reconstruction quality. While this dependence on input coverage remains mostly unchanged, recent advances in image and video generation models have opened the door to a fundamentally different paradigm.

Instead of relying on densely captured scenes, it is now becoming possible to generate immersive environments from sparse inputs, such as a single image or even a textual description. This shift moves novel view synthesis beyond reconstruction toward generative scene creation, where entire worlds can be synthesized and explored without extensive data capture.

### 6.2.3. Adoption by the industry

A final question remains: can Triangle Splatting and MeshSplatting overcome the current dominance of Gaussian-based representations and become the preferred primitives in industry?

Despite the advantages of Triangle Splatting and MeshSplatting in terms of structure, efficiency, and compatibility with established graphics pipelines, 3D Gaussian Splatting currently benefits from two decisive advantages: widespread adoption and a large, active community. Since its introduction, Gaussian-based representations have attracted significant attention in both academia and industry, resulting in a rapidly growing ecosystem of optimized implementations, open-source libraries, and practical tools. This strong momentum lowers the barrier to entry and makes Gaussians an appealing choice for rapid deployment and reliable baseline performance.

Nevertheless, Triangle Splatting and MeshSplatting offer multiple advantages that outperform Gaussians even with a smaller research community.

First, both representations are aligned with modern graphics hardware. Triangles remain the native rendering primitive of GPUs, allowing Triangle Splatting and MeshSplatting to directly benefit from decades of hardware acceleration and highly optimized rendering pipelines. This leads to efficient rasterization, and straightforward integration into existing real-time engines, without the need for specialized CUDA kernels or custom rendering backends. This advantage becomes particularly important in scenarios, where large-scale scenes must be rendered under strict latency and power constraints on limited hardware such as in augmented and virtual

reality. In this context, Triangle Splatting and MeshSplatting enable the efficient rendering of a large number of primitives while maintaining real-time performance, making them especially well suited for VR headsets and other resource-constrained devices.

Additionally, compatibility with existing production pipelines is a key advantage. Most industrial workflows in gaming, visual effects, and simulation are built around mesh-based representations. Triangle Splatting and MeshSplatting can be integrated into these established pipelines with minimal adaptation, enabling a seamless transition from learned representations to directly deployable assets. In contrast, Gaussian-based methods typically require additional conversion steps or specialized rendering solutions, which introduce overhead and increase complexity.

Taken together, these properties suggest that Triangle Splatting and MeshSplatting are not only competitive with Gaussian representations in terms of quality, but also more suitable for long-term industrial deployment, where efficiency and straightforward integration into existing pipelines are critical requirements.



# Bibliography

- [1] Jan Held, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. MeshSplatting: Differentiable rendering with opaque meshes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Denver, CO, USA, Jun. 2026.
- [2] Jan Held, Renaud Vandeghen, Adrien Delière, Daniel Hamdi, Abdullah Rebain, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. In *Int. Conf. 3D Vis. (3DV)*, pages 1–10, Vancouver, Can., Mar. 2026.
- [3] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Delière, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21360–21369, Nashville, TN, USA, Jun. 2025.
- [4] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH Comput. Graph.*, 18(3):165–174, Jan. 1984.
- [5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):1–14, Jul. 2023.
- [6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 12346 of *Lect. Notes Comput. Sci.*, pages 405–421, Virtual conference, 2020.
- [7] Sanghyun Son, Matheus Gadelha, Yang Zhou, Matthew Fisher, Zexiang Xu, Yi-Ling Qiao, Ming C. Lin, and Yi Zhou. DMesh++: An efficient differentiable mesh for complex shapes. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 26590–26599, Honolulu, HI, USA, Oct. 2025.
- [8] Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C. Lin, and Yi Zhou. DMesh: A differentiable mesh representation. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 37, pages 12035–12077, Vancouver, Can., Dec. 2024. Curran Assoc. Inc.
- [9] Ruiqi Wu, Linlin Jiao, Gang Liu, Li Zhu, Xuan Fei, Yashuang Mu, and Chao Fan. LFGS: A lightweight framework for efficient 3D Gaussian splatting with minimal memory footprint. *Comput. & Graph.*, 131, Oct. 2025.

- [10] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 12–22, Paris, Fr., Oct. 2023.

# A. Appendix

## A.1. General context

*Association football*, also named *football* or *soccer*, is one of the most widely followed sports in the world, where referees are responsible for enforcing the rules of the game and ensuring fair competition. However, refereeing decisions are often difficult to make in real time due to the speed and complexity of the game. Incidents such as fouls, tackles, and physical duels between players can be ambiguous and require careful interpretation of the Laws of the Game. Incorrect decisions may significantly influence the outcome of matches and can have substantial sporting and financial consequences for teams and organizations.

Over the past decades, technological assistance for referees has progressively evolved. Initially, referees relied solely on their perception and communication with assistant referees on the field. Later, wireless communication systems allowed referees to coordinate more effectively during matches. Further technological advances led to the introduction of goal-line technology, which automatically determines whether the ball has fully crossed the goal line. More recently, the Video Assistant Referee (VAR) system was introduced to allow referees to review potentially game-changing incidents using video replays. These developments illustrate the increasing role of technology in improving decision-making in professional football.

Despite its benefits, the VAR system still has several limitations. First, it requires significant infrastructure, including multiple calibrated cameras, specialized equipment, and additional referees operating in a video control room. Consequently, such systems are typically only available in top-tier professional leagues and major international competitions. Lower-tier leagues and amateur competitions often lack the financial resources and personnel necessary to deploy this technology. In addition, referee shortages worldwide further complicate the deployment of systems that require additional officials.

Another challenge concerns the consistency and transparency of refereeing decisions. Even when video assistance is available, different referees may interpret similar situations differently, leading to perceived inconsistencies between matches or leagues. Furthermore, the reasoning behind decisions is often difficult for players, coaches, and spectators to understand, which can reduce trust in the decision-making process. These issues highlight the need for automated systems capable not only of

assisting referees but also of providing consistent and transparent decision support.

Recent advances in computer vision and deep learning have opened new possibilities for analyzing sports videos automatically. In particular, modern video understanding methods can recognize complex actions, track players, and interpret interactions between athletes in dynamic environments. These techniques have already shown promising results in several sports analytics tasks, including player detection, action recognition, and tactical analysis. However, the automatic interpretation of refereeing decisions remains a challenging problem due to the fine-grained and often subjective nature of football fouls.

To address these challenges, we first introduced the Video Assistant Referee System (VARs), a computer vision framework designed to analyze football incidents automatically from multiple camera views and assist referees in making decisions. VARs leverages multi-view video analysis to classify whether an action constitutes a foul, determine the type of foul, and estimate the appropriate disciplinary sanction. The system is supported by a dedicated dataset of football fouls captured from multiple broadcast viewpoints and annotated by professional referees, enabling the development and evaluation of automated refereeing models. The full architecture and dataset description are provided in section A.4.

In our second publication, we introduce an attention mechanism that enables the model to identify which viewpoints are most informative for its prediction by assigning them higher weights than others. This attention is learned during training and can later be interpreted as an indicator of which views the model relies on most for its decision. This information can be leveraged for camera selection in human refereeing or for broadcasting applications to highlight the most relevant viewpoint for viewers. Further details are provided in section A.5.

While automated decision systems can improve consistency and scalability, they also raise important questions regarding transparency and explainability. Modern deep learning models often operate as black boxes, making it difficult to understand the reasoning behind their predictions. In domains where decisions have significant consequences, such as healthcare, autonomous driving, or sports officiating, the ability to explain model predictions is essential to build trust and facilitate adoption.

To address this limitation, we extended the VARs framework by introducing X-VARS, an explainable video assistant referee system based on multi-modal large language models. X-VARS combines visual information extracted from video clips with language modeling capabilities to generate explanations for refereeing decisions. In addition to predicting whether a foul occurred and determining the appropriate sanction, the system can describe the observed action, answer refereeing questions, and justify its decisions using natural language.

The development of X-VARS was supported by the creation of a new dataset, SoccerNet-XFoul, which contains thousands of video clips paired with refereeing

questions and detailed answers provided by experienced referees. This dataset enables the training and evaluation of models capable of both understanding football incidents and explaining their reasoning in accordance with the Laws of the Game. Further details are provided in section A.6.

## A.2. Conclusion and perspectives

Together, VARS and X-VARS represent important steps toward automated and explainable referee assistance systems. By combining advances in multi-view video understanding with recent progress in multi-modal language models, these approaches aim to improve the accuracy, accessibility, and transparency of decision support tools for football refereeing. Ultimately, such systems could support referees across all levels of the sport, from elite competitions to amateur leagues. In professional settings, VAR referees could benefit from faster decision-making through a VARS-based assistant that immediately highlights the most informative viewpoints. In amateur leagues, where video assistant refereeing is often infeasible due to financial and human resource constraints, matches are typically officiated by only a few referees, and sometimes even a single referee. In this context, a lightweight VARS-like system could help reduce critical errors and improve decision quality, contributing to fairer play across all levels of competition.

Integrating novel view synthesis into X-VARS can further enhance its capabilities. By generating alternative viewpoints, referees would be able to inspect fouls from the most informative angles, improving their ability to assess fouls and make accurate decisions.

A natural next step is the development of an AI system that automatically identifies the optimal viewpoint for each situation, selecting the view that maximizes relevant visual information for human decision-making. Such a system would not only improve the quality of refereeing decisions, but also increase their speed, as the most informative camera angle could be provided instantly and adaptively without manual selection.

## A.3. Contributions of this appendix

### A.3.1. List of publications included in this appendix

Section A.4 **Jan Held\***, Anthony Cioppa\*, Silvio Giancola\*, Abdullah Hamdi, Bernard Ghanem, and Marc Van Droogenbroeck. VARS: Video assistant referee system for automated soccer decision making from multiple views. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pages 5086–5097,



Vancouver, Can., Jun. 2023.

*Author contribution.* As lead author, I co-designed the method, implemented the training pipeline, conducted the experiments, and constructed the dataset. Hani Itani contributed through extensive discussions and assisted with parts of the training pipeline implementation.

Section A.5 **Jan Held**, Anthony Cioppa, Silvio Giancola, Abdullah Hamdi, Christel Devue, Bernard Ghanem, and Marc Van Droogenbroeck. Towards an AI-powered video assistant referee system (VARS) for association football. In Scand. Conf. Image Anal. (SCIA), volume 15725 of Lect. Notes Comput. Sci., pages 295–309, 2025.

Section A.6 **Jan Held\***, Hani Itani\*, Anthony Cioppa, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. X-VARS: Introducing explainability in football refereeing with multi-modal large language models. In IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pages 3267–3279, Seattle, WA, USA, Jun. 2024.

*Author contribution.* As lead author, I co-designed the method, implemented the training pipeline, conducted the experiments, and constructed the dataset. Anthony Cioppa and Silvio Giancola provided valuable guidance and expertise, contributed to the method design, and supported the writing and refinement of the manuscript.

### A.3.2. List of software related to this appendix

A major contribution of this thesis is the open-source release of all code, enabling other researchers to build upon our work. VARS and X-VARS have also gained close to 100 GitHub stars, reflecting strong community interest and adoption.

### A.3.3. Participation in the organization of SoccerNet challenges

I was part of the organizing team of the SoccerNet challenges 2023, 2024, 2025, and 2026, and served as lead organizer for the Multi-View Foul Recognition Challenge in 2024 and 2025, as well as the Novel View Synthesis Competition in 2026 (currently ongoing). The Multi-View Foul Recognition Challenge attracted more than 40 participants over two editions. As lead organizer, I was also featured in the official YouTube presentations of the challenges: <https://www.youtube.com/watch?v=UDeSd0R9Ing> and <https://www.youtube.com/watch?v=J3c4YTME5ZQ>.

# VARs: Video Assistant Referee System for Automated Soccer Decision Making from Multiple Views

Jan Held<sup>\*,1</sup> Anthony Cioppa<sup>\*,1,2</sup> Silvio Giancola<sup>\*,2</sup>  
Abdullah Hamdi<sup>2</sup> Bernard Ghanem<sup>2</sup> Marc Van Droogenbroeck<sup>1</sup>  
<sup>1</sup> University of Liège <sup>2</sup> KAUST

## Abstract

The Video Assistant Referee (VAR) has revolutionized association football, enabling referees to review incidents on the pitch, make informed decisions, and ensure fairness. However, due to the lack of referees in many countries and the high cost of the VAR infrastructure, only professional leagues can benefit from it. In this paper, we propose a Video Assistant Referee System (VARs) that can automate soccer decision-making. VARs leverages the latest findings in multi-view video analysis, to provide real-time feedback to the referee, and help them make informed decisions that can impact the outcome of a game. To validate VARs, we introduce SoccerNet-MVFoul, a novel video dataset of soccer fouls from multiple camera views, annotated with extensive foul descriptions by a professional soccer referee, and we benchmark our VARs to automatically recognize the characteristics of these fouls. We believe that VARs has the potential to revolutionize soccer refereeing and take the game to new heights of fairness and accuracy across all levels of professional and amateur federations.

## 1. Introduction

Over the past decades, the technology used by referees in soccer has undergone a drastic evolution. Before the beginning of this century, referees and their assistants only relied on their own judgment, and the communication between them was based on eye contact and body language. The French and Scottish refereeing trios were the first ones to be linked through wireless mini-earphones during league matches, facilitating communication among them [56]. Nowadays, wireless headsets are essential pieces of equipment for referees worldwide, made mandatory for high-level competitions. Another important breakthrough in professional soccer was the introduction of goal-line

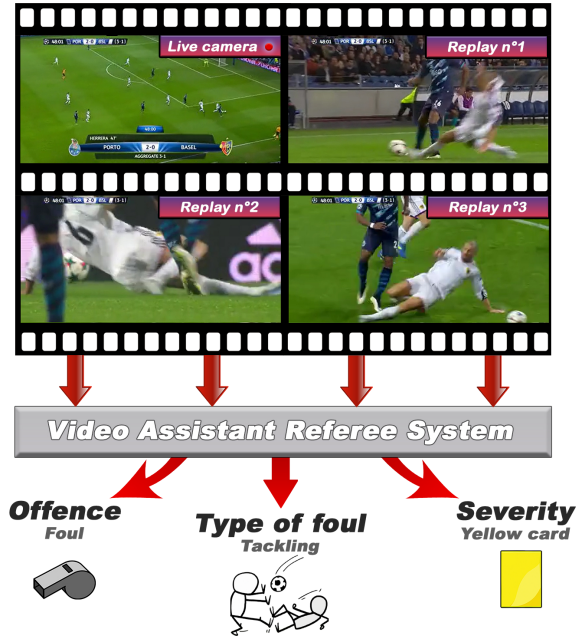


Figure 1. **Video Assistant Referee System (VARs)**. We propose an automated VARs for automatically classifying whether an action is a foul, determining the type of foul (e.g., ‘Tackling’, ‘Pushing’, etc.), and the appropriate punishment the player should receive for the foul (i.e., ‘No card’, ‘Yellow card’, or ‘Red card’) from a multi-view camera setup.

technology, which uses a combination of cameras and sensors to determine whether the entire ball has crossed the goal line or not. This technology aims to prevent controversial goals such as the famous “ghost goal” scored by Geoff Hursts in the 1966 World Cup final against Germany, where the ball may not have fully crossed the line and led to England receiving the world champion title [17]. Successively, the International Football Association Board (IFAB) approved the introduction of extra referees, namely the video assistant referees (VAR), to prevent game-changing errors. More recently, artificial intelligence systems appeared for the first time during the World Cup 2022 in Qatar. Semi-

(\*) Denotes equal contributions

Contacts: jan.held@student.uliege.be, silvio.giancola@kaust.edu.sa, anthony.cioppa@uliege.be. Data/code available at [www.soccer-net.org](http://www.soccer-net.org).

automated offside technology now supports the VAR to help referees make faster, more accurate, and more reproducible offside decisions [18]. This new system relies on 12 well-calibrated cameras to track the ball and the player’s body pose, sending an automated offside alert to the video assistant referee inside the video operation room. This shows that soccer is moving towards more assistance or even automated systems to help referees make better decisions.

However, despite its intention to improve the accuracy of referee decisions, VAR has become a source of frustration and anger for many football fans around the world. Since we have a different video assistant referee for each game, we do not always have consistent decisions. Sometimes the VAR predicts different outcomes for similar situations in different games and leagues. Moreover, the implementation of the VAR technology and infrastructure requires a substantial financial investment, limiting its accessibility to only the top-tier leagues and clubs. As a result, semi-professional or amateur leagues are unable to benefit from the VAR due to financial constraints. Additionally, the shortage of referees worldwide makes it impossible in staffing additional referees as Video Assistant Referees, except for the professional leagues.

In this work, we propose a first step towards a fully automated “Video Assistant Referee System” (VARS) which could support or replace the current VAR. We attempt to automatically predict all fouls and suggest appropriate sanctions to the players. In case the on-field referee makes a significant mistake, our VARS could intervene to suggest a revision. It is intended that, just like regular VAR, our VARS serves as a support system for the referee, but the final decision remains in the hands of the on-field referee. To achieve this objective, we rely on multi-view uncalibrated camera video streams, which are already leveraged to edit broadcast games. Specifically, we release a new dataset comprising 3,901 actions with multi-view clips of 5 seconds around the action, annotated by a professional referee. We focus our analysis on the classification of foul types and evaluate their severity to identify the sanction for the player. Practically, our VARS analyses the different streams and combines the information from the multiple cameras. We show that using a multi-view system largely improves the performance compared to a single view and that we reach good performance on our video recognition tasks.

**Contributions.** We summarize our contribution as follows: (i) We publicly release *SoccerNet-MVFouls*, a new multi-view video dataset containing video clips captured by multiple cameras, annotated with 10 properties. (ii) We propose VARS, a new multi-camera video recognition system for classifying fouls and their severity. (iii) We propose a thorough study of using multiple views and how different types of camera views can influence the performance of VARS on two new video recognition tasks.

## 2. Related work

**Sports understanding.** As a research topic, sports video understanding has increased in popularity thanks to its challenging and fine-grained nature [41, 54]. Nowadays, most state-of-the-art automatic methods are based on deep learning and have shown impressive performance on tasks such as player detection and tracking [7, 39, 58], tactics analysis [53], pass feasibility [2] and prediction in soccer [25], talent scouting [11], or player re-identification in occluded scenarios [50]. Video classification started as a key area of research in this field [65], with approaches proposed to recognize specific actions [32, 45] or distinguish between different game phases [8]. With the growing interest in temporal activity localization [3], the task of action spotting [4, 6, 10, 26, 48, 49, 67] has gained interest as it provides precise localization of specific actions within a soccer game.

The progress in those tasks was made possible thanks to the availability of large-scale datasets [28, 43, 46, 57, 66]. Giancola *et al.* [20] introduced the SoccerNet dataset, which has grown to be the most extensive collection of data and annotations for video understanding in soccer, including benchmarks for 10 different tasks, ranging from broadcast understanding [12], field understanding [5] and player understanding [9]. The SoccerNet team also organizes yearly competitions on these different tasks to foster research in the field [21]. The dataset presented in this paper extends SoccerNet by proposing a novel multi-view video collection including foul annotations for video recognition tasks.

**Video understanding.** For a long time, video understanding lagged behind image understanding due to the lack of large-scale video datasets such as ImageNet or CIFAR-100 [13, 34] in the video domain. However, the release of large video understanding datasets such as UCF101 [51], ActivityNet [3], YouTube-8M [1], and Kinetics [31] has led to a surge in popularity and interest in the field. Video understanding tasks include video classification [16, 30, 42], action recognition [47, 61], video captioning [19, 33, 63], and video generation [36].

The interest in developing video classification models that capture spatio-temporal information has significantly grown. Temporal Segment Network (TSN) [62] aggregates features across multiple temporal video segments to improve recognition performance. Tran *et al.* [55] proposed a new spatio-temporal convolutional block R(2+1)D and analyze its effect on action recognition models. Recently, the Multiscale Vision Transformer (MViT) [15, 37] came as a way to combine the strengths of both convolutional neural networks (CNNs) and transformers for video classification, capturing both spatial and temporal attentions. In this work, we train different video representations to learn per-clip features that we aggregate from multiple views to identify the different properties of the fouls.

| Dataset                          | Context        | Task              | Videos/Images | View        | Type   |
|----------------------------------|----------------|-------------------|---------------|-------------|--------|
| Kinetics 400 [31]                | Human actions  | Classification    | 650,000       | Single-view | Videos |
| NTU RGB+D 120 [38]               | Human actions  | Classification    | 114,480       | Multi-view  | Videos |
| Northwestern-UCLA Multiview [60] | Human actions  | Classification    | 1,493         | Multi-view  | Videos |
| UWA3D Multiview II [44]          | Human actions  | Classification    | 900           | Multi-view  | Videos |
| SoccerNet-v2 (Actions) [21]      | Soccer         | Classification    | 110,458       | Single-View | Videos |
| SoccerNet-v3 (Re-id.) [5]        | Soccer         | Re-identification | 33,986        | Multi-View  | Images |
| <b>SoccerNet-MVFouls (Ours)</b>  | Soccer (Fouls) | Classification    | 8,923         | Multi-view  | Videos |

Table 1. **Video action understanding datasets.** Comparative overview of relevant datasets for multi/single-view action recognition in videos/images. Our dataset is the only one providing multi-view videos for classification in sports, with 10 annotated properties per action.

**Multi-view understanding.** Su *et al.* [52] introduces the idea of training image encoders to recognize 3D objects from multiple views, benefiting from the mature 2D computer vision. Most effort focused on informative aggregation between views, introducing cross-view confidence [29], group convolutions to learn rotation-equivariant representations [14], graph convolutions to learn view aggregation [64]. Alternatively, MVTN [23] predicted the viewpoints from a differentiable 3D renderer. In the video domain, synthetic views (*e.g.* 3D motion or optical flow) are created for single-stream videos as a way to obtain better representation learned in a self-supervised fashion [35, 59]. In this work, we leverage a simple multi-view pipeline for video understanding, trained in a fully supervised fashion, that incorporates multiple replay streams from soccer broadcast videos.

### 3. SoccerNet-MVFouls dataset

In this section, we introduce our novel multi-view foul classification soccer dataset, called *SoccerNet-MVFouls*.

Table 1 presents an overview of our dataset and compares it with other datasets that propose action recognition using either single or multiple views. Our dataset is the only one for multi-view video action recognition in sports, and the first dataset to focus specifically on referee’s decisions.

*SoccerNet-MVFouls* gathers 3,901 actions extracted from 500 soccer games from six main European leagues, covering three seasons from 2014 to 2017, extracted from the SoccerNet dataset [12, 20]. Each action is composed of at least two videos depicting the live action and at least one replay. The actions are annotated with 10 different properties describing the characteristics of the foul from a referee’s perspective (*e.g.* the severity of the foul, the type of foul, *etc.*). To ensure high-quality annotations, all these properties were manually annotated by a professional soccer referee with 6 years of experience and more than 300 official games. The referee watched the videos from all available views at any speed to accurately characterize the foul.

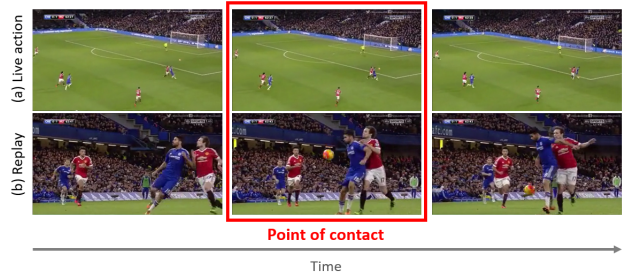


Figure 2. **Example of a multi-view sequence from our dataset.** Each foul has at least (a) one live-action clip (usually taken from the main camera) and (b) one synchronized replay clip (usually a close-up view). We annotate the exact frame where the point of contact happens (red box). The ground-truth properties for this example are: “Offence”, “Challenge”, “No card”, “With contact”, “Upper body”, “Use of shoulder”, “Ball is not played”, “Tried to play the ball”, “No handball”, and “No handball offence”.

#### 3.1. Dataset collection

The dataset was collected in three steps: (i) we extracted the relevant action clips from soccer broadcast videos, (ii) we temporally aligned the clips related to the same action, and (iii) we annotated several foul properties.

**Clip extraction.** As a starting point, we used the SoccerNet-v2 dataset [12], which contains timestamp annotations of fouls for 500 full broadcast games. Furthermore, the SoccerNet dataset also provides annotations of the replays of some of the fouls, allowing us to retrieve, for the same action, different viewpoints. Since our goal is to design a multi-view video assistant referee system, we only keep actions for which we have access to at least two different points of view. In most cases, the extracted clips should cover sufficient information to determine all the foul properties. Also, to prevent bias towards the on-field referee’s decision, the 5 second clips should not contain the decision of the referee (*e.g.* if the player is given a yellow card). Therefore, we extracted 5 second clips per action, starting 3 seconds before and ending 2 seconds after the timestamp



annotation. In the following, “live action clip” will refer to the clips taken from the main camera, while “replay clips” will denote all the replay clips typically taken from closer shots. Figure 2 shows an example of such extracted clip.

**Clips alignment.** We build a Multi-View Foul Annotator tool with a similar interface to a VAR room to ensure the quality of the annotations performed by our referee. At first, the referee is presented with all available clips of an action simultaneously on a grid layout. Our annotator tool enables users to modify the annotated point of contact (see Figure 2) for each clip individually and adjust the speed and offset of the clips to align them temporally, taking into account the fact that replays are frequently broadcasted at a slower speed. The referee may browse simultaneously the synchronized videos either at regular speed or frame by frame to accurately understand and describe the properties of the action. More information and an example of annotation using our annotator may be found in the supplementary material.

**Property annotations.** The SoccerNet-v2 dataset provides annotations for fouls and yellow/red cards given by the actual game referee. However, the on-field referee has only his own point of view to characterize the foul. Judging foul play incidents from the referee’s position at playing time leads to an average error rate of 14% [40]. Our referee annotator has no time pressure and access to multiple perspectives, which results in more accurate decisions compared to the on-field referee who has to take a quick decision and only has a single view. To ensure a high-quality dataset and avoid any bias, our professional soccer referee manually annotated all properties without seeing the on-field referee’s decision.

We defined several properties for each action that are necessary for the referee to take the final decision. These properties include (i) if the clip contains a foul (*i.e.* an action which breaks/violates the *Laws of the Game* [27]), (ii) the class of the foul, (iii) the severity of the foul, (iv) if the player plays the ball, (v) if the player tries to play the ball, (vi) if any player touches the ball with his hand or arm, deliberately or not, (vii) and whether it is an offence according to the *Laws of the Game* [27]), (viii) if there is contact between two players, (ix) the action foul relates to the upper or underbody, and, finally, (x) we further discriminate for the upper body between arms and shoulders. We have special labels corresponding to grey areas for the property (i), we use the label “Between” when both “Foul” and “No foul” decisions are equally valid and there is no obvious decision. For property (iii), we use the labels “Borderline No card/Yellow card” and “Borderline Yellow card/Red card” to indicate a grey area when either “No card” or “Yellow card”, (resp. “Yellow card” or “Red card”), would be the correct decision.

| Fouls        |       | Offence    |       | Offence Severity |       |
|--------------|-------|------------|-------|------------------|-------|
| Class        | Prob. | Class      | Prob. | Class            | Prob. |
| St. tackling | 43.6  | Offence    | 85.8  | No card          | 55.3  |
| Tackling     | 15.6  | No offence | 10.7  | Yellow card      | 26.6  |
| Challenge    | 13.0  | Between    | 3.4   | NC/YC            | 15.2  |
| Holding      | 12.5  |            |       | YC/RD            | 1.7   |
| Elbowing     | 5.9   |            |       | Red card         | 1.1   |
| High leg     | 3.5   |            |       |                  |       |
| Pushing      | 2.9   |            |       |                  |       |
| Dive         | 0.9   |            |       |                  |       |

Table 2. **Distribution of classes in our SoccerNet-MVFouls dataset.** The distribution of the classes for the “Offence”, “Severity” and “Type of foul” properties is highly imbalanced. The distribution for the other properties is shown in supplementary. “St. tackling” stands for standing tackling.

| Foul Class        | Succ. Rate  | Severity Distribution |             |             |  |
|-------------------|-------------|-----------------------|-------------|-------------|--|
|                   |             | No Card               | Yellow Card | Red Card    |  |
| Standing Tackling | <b>0.94</b> | 0.79                  | 0.18        | 0.02        |  |
| Tackling          | 0.87        | 0.37                  | <b>0.58</b> | <b>0.04</b> |  |
| High Leg          | 0.87        | 0.31                  | <b>0.63</b> | <b>0.06</b> |  |
| Holding           | 0.90        | 0.60                  | 0.40        | 0.00        |  |
| Pushing           | 0.84        | <b>0.99</b>           | 0.01        | 0.00        |  |
| Elbowing          | <b>0.93</b> | 0.43                  | <b>0.53</b> | 0.03        |  |
| Challenge         | 0.75        | <b>0.94</b>           | 0.05        | 0.01        |  |
| Dive              | /           | 0.00                  | 1.00        | 0.00        |  |

Table 3. **Referees success rate and severity per foul class.** Referees are successful in most classes but struggle with “Challenge”. Some classes are less likely to return a card, *e.g.* “Tackling” or “High leg”. The success rate for “Dive” is unknown, as we cannot know if a referee whistle for the foul or the dive.

### 3.2. Dataset statistics

**Number of views.** On average, we have 2.29 clips per foul action, around 75% of them have two viewpoints (live and replay), 20% have a second replay, and around 5% have a third replay video. No foul has more than four views.

**Properties distribution.** Table 2 shows the distribution of the properties “Offence”, “Severity” and “Type of foul”. We can see that in all three cases, the distribution is highly unbalanced towards “No card”, “Offence” and “Standing tackling”, respectively. This analysis follows our intuition of soccer, where yellow and red cards are usually rarer than simple free-kicks given after a foul.

**Success rate of the referees.** As we only have extracted fouls for which the on-field referee has given a foul in the game, we can analyze the success rate of the referees by analyzing the property “No offence”. From the 3901 fouls given by the referees in the games, our referee annotated 368 fouls as “No offence”, leading to an error rate of 10.7%. “Standing tackling” and “Elbowing” are the most well classified with 94% success rate, as shown in Table 3. For the



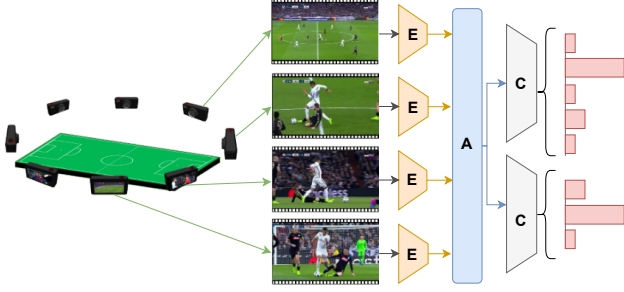


Figure 3. **VARS: Video Assistant Referee System.** From multi-view video clips input, our system **encodes per-view video features (E)**, **aggregates the view features (A)**, and classifies different properties of the foul action (**C**).

remaining action classes, the referees have a similar success rate of approximately 87%, except for the foul class “Challenge”, where the referees have an error rate of 25%. Our analysis is aligned with the finding of Mallo *et al.* [40].

**Severity for different foul classes.** The distribution of the severity among different foul classes can provide insight into how often certain types of fouls result in a card. The results are presented in Table 3. “Tackling”, “High leg”, and “Elbowing” are three types of fouls that very often result in a yellow card, as they represent fouls that are dangerous for opponents. Contrarily, some classes like “Pushing” or “Challenge” are very unlikely to get a yellow or red card.

## 4. Methodology

Our VAR system is a multi-view video architecture, that automatically identifies different properties for an action. We illustrate our proposed VARS in Figure 3.

### 4.1. Classification tasks

We formally define two tasks for our dataset.

**Task 1: Fine-grained foul classification.** Given multiple clips of the same foul instance, the objective is to classify the foul into one of 8 fine-grained foul classes: “Standing tackling”, “Tackling”, “High leg”, “Pushing”, “Holding”, “Elbowing”, “Challenge”, “Dive/Simulation”.

**Task 2: Offence severity classification.** Given multiple clips of the same foul instance, the objective is to classify whether the foul constitutes an offence, as well as the severity of the foul. We have defined four classes: “No offence”, “Offence + No card”, “Offence + Yellow card”, and “Offence + Red card”. We put aside clips labeled “Between” as well as the clips annotated as “Borderline”. Therefore, for this particular task, we use a subset of our SoccerNet-MVFoul dataset.

### 4.2. Video Assistant Referee System (VARS)

We propose a novel Video Assistant Referee System (VARS) for the task of video recognition from multiple camera views. The pipeline of the VARS is presented in Figure 3. Our VARS takes multiple video clips denoted by  $\mathbf{v} = \{v_i\}_1^n$  as input, showing the same action from  $n$  different views. A video  $v_i$  is fed into a video encoder  $\mathbf{E}$  with parameters  $\theta_E$  to extract a vector  $f_i$  containing the spatio-temporal features for that specific view:

$$f_i = \mathbf{E}_{\theta_E}(v_i) . \quad (1)$$

We aggregate the feature vectors through a function  $\mathbf{A}$  that outputs a single multi-view representation  $\mathbf{R}$  following:

$$\mathbf{R} = \mathbf{A}(\{f_i\}_{i=1}^n) , \quad (2)$$

with  $\mathbf{A}$  being a max or mean aggregation function. For the single-task classifier, we input the pooled features through a classification head  $\mathbf{C}$  with parameters  $\theta_C$ . VARS predicts the final class from the maximum probability score of the classification head, as given by:

$$\text{VARS} = \arg \max \mathbf{C}_{\theta_C}(\mathbf{R}) . \quad (3)$$

We train our model to minimize the following loss:

$$\mathcal{L} = \mathbf{L}(\mathbf{C}_{\theta_C}(\mathbf{A}(\{\mathbf{E}_{\theta_E}(v_i)\}_{i=1}^n)), y) , \quad (4)$$

with  $\mathbf{L}$  being the cross entropy loss function, and  $y$  the ground truth associated to  $\{v_i\}_{i=1}^n$ . For the offence severity classification, VARS has to understand the game of soccer in order to correctly classify fouls into “No card”, “Yellow card”, and “Red card”. In fact, bringing contextual information about the type of foul inside the network is essential to determine the severity of the offence. As the foul and offence classifiers share common features, we train a model to perform both tasks simultaneously. Our multi-task VARS learns to leverage these shared features to improve its predictions for both tasks. For the multi-task classifier, we define two heads,  $\mathbf{C}^{\text{foul}}$  and  $\mathbf{C}^{\text{off}}$ , respectively for the tasks of fine-grained foul classification and offence severity classification. From the probability vector of each task, the VARS will take the maximum as the final prediction:

$$\text{VARS}^t = \arg \max \mathbf{C}_{\theta_{\text{Ct}}}^t(\mathbf{R}) \quad \forall t \in \{\text{foul}, \text{off}\} . \quad (5)$$

We train our model by minimizing both tasks loss with:

$$\alpha_{\text{foul}} \mathcal{L}^{\text{foul}} + \alpha_{\text{off}} \mathcal{L}^{\text{off}} . \quad (6)$$

By choosing different values for  $\alpha$ , we can assign more or less importance to tasks. This scaling is necessary when the losses have significantly different magnitudes. In the case of our two tasks, the losses have a similar order of magnitude, so we typically select  $\alpha_{\text{foul}} = \alpha_{\text{off}} = 1$ .

**Video encoder E.** We considered different encoders to extract features from the video clips: (i) ResNet [24] may be used on videos by running the network on each frame independently and then using a max or mean pooling operation on the features across the frames to obtain a single feature vector that represents the entire video. While this approach works well for extracting spatial features, it does not capture temporal dynamics. (ii) R(2+1)D [55] extends the 2D CNN architecture with an additional temporal convolutional layer that operates on a sequence of frames to capture the temporal dynamics of the video. The advantage compared to ResNet is that it both captures spatial and temporal features directly. (iii) MViT [15, 37] integrates a multiscale feature representation with a transformer-based architecture to capture both spatial and temporal information from video clips. The feature encoders are typically pre-trained on ImageNet [13] (ResNet) or Kinetics [31] (R2+1D and MViT).

**Multi-view aggregator A.** To combine the extracted features from multiple views, we introduce two different pooling strategies [22], in particular: (i) Mean pooling takes the average value for each feature, and (ii) Max pooling which takes the maximum value per feature.

**Classification heads C.** Our classification heads consist of two dense layers with softmax activation. The output is a probability vector with dimensions that match the number of classes in the classification problems.

## 5. Experiments

### 5.1. Experimental setup

**Training details.** For both classification tasks, we leverage clips of 16 frames, spanning temporally for 1 second, with a spatial dimension of  $224 \times 398$  pixels. Specifically, the clips contain 8 frames before the foul and 8 frames after the foul. The encoders E are pre-trained as detailed in the methodology, and the classifier C is trained from scratch, while both are trained in an end-to-end fashion. We use a cross-entropy loss, optimized with Adam with an exponential decreasing learning rate starting at  $10^{-4}$  and a batch size of 8. The model starts overfitting after 10 epochs, and it takes around 9 hours to train on a single Nvidia V100 GPU.

**Evaluation metrics.** We report the classification accuracy, which is defined as the ratio of actions correctly classified with respect to the total number of actions. We also provide the top-2 accuracy (where a sample is considered well classified if the class appears in the top two highest confidence predictions) to get more insight into the model’s performance. As our dataset is unbalanced, we also provide the balanced accuracy, which is defined as follows:

$$\text{Balanced Accuracy (BA)} = \frac{1}{N} \sum_{i=1}^N \frac{TP_i}{P_i}, \quad (7)$$

| Feature Extractor | Pooling | Acc. @1     | Acc. @2     | BA          |
|-------------------|---------|-------------|-------------|-------------|
| ResNet [24]       | Mean    | 0.31        | 0.56        | 0.28        |
| ResNet [24]       | Max     | 0.32        | 0.60        | 0.28        |
| R(2+1)D [55]      | Mean    | 0.31        | 0.55        | 0.34        |
| R(2+1)D [55]      | Max     | 0.32        | 0.56        | 0.33        |
| MViT [15, 37]     | Mean    | 0.40        | 0.65        | <b>0.45</b> |
| MViT [15, 37]     | Max     | <b>0.47</b> | <b>0.69</b> | 0.43        |

Table 4. **Main results for the multi-view video foul classification.** We compare three feature encoders and two pooling methods. The best performance is obtained with MViT and a max pooling between the views. BA indicates the balanced accuracy after normalizing by the frequency of that class.

with  $N$  the number of classes,  $TP$  (True Positives) is the number of times where the model correctly predicted the class  $i$  and  $P_i$  (Positives) is the number of ground-truth samples for that class in the dataset.

### 5.2. Main Results

**Task 1: Fine-grained foul classification.** Our results may be found in Table 4. By extracting spatio-temporal features with MViT, we achieve significant improvements in performance compared to ResNet and R(2+1)D. This indicates that using a more advanced feature encoder can significantly enhance the model’s ability to identify and classify the type of foul. The influence of the pooling method on the performance is however not significant, although max pooling shows slightly better results. In general, max pooling might be better when not all views are equally informative. Taking the max values helps identify the most important features for the most informative views while ignoring less useful information. In contrast, mean-pooling takes into account the information from all views, including those with a poor perspective. Overall, the best performance is obtained by using MViT as video encoder and max pooling.

**Task 2: Offence severity classification.** For the offence severity classification, we study the same feature encoders and pooling techniques. The top part of Table 5 shows the results obtained by our single-task classifier. Regardless of the used feature extractor or pooling technique, the model has more difficulties in classifying the actions. These difficulties are mainly due to two factors. First, the dataset exclusively consists of actions that were awarded a free kick by the on-field referee. As a result, the “No offence” actions are visually similar to a foul, and not to clear “No offence” actions. The model often struggles to differentiate these actions from actual fouls, which can be further seen in the Supplementary Material. Secondly, the visual appearance of an offence with no card, yellow card, or red card can vary greatly. In Figures 4a and 4b, we compare two frames of two different foul classes that have a little visual similarity. However, in both cases, the defender acted with disregard

| Feature Extractor | Pooling | Task   | Acc.        | BA          |
|-------------------|---------|--------|-------------|-------------|
| ResNet [24]       | Mean    | Single | 0.25        | 0.26        |
| ResNet [24]       | Max     | Single | 0.22        | 0.25        |
| R(2+1)D [55]      | Mean    | Single | 0.28        | 0.30        |
| R(2+1)D [55]      | Max     | Single | 0.27        | 0.29        |
| MViT [15, 37]     | Mean    | Single | 0.32        | 0.23        |
| MViT [15, 37]     | Max     | Single | 0.29        | 0.27        |
| ResNet [24]       | Mean    | Multi  | 0.34        | 0.25        |
| ResNet [24]       | Mean    | Multi  | 0.32        | 0.24        |
| R(2+1)D [55]      | Mean    | Multi  | 0.34        | 0.30        |
| R(2+1)D [55]      | Max     | Multi  | 0.39        | 0.31        |
| MViT [15, 37]     | Mean    | Multi  | 0.38        | 0.31        |
| MViT [15, 37]     | Max     | Multi  | <b>0.43</b> | <b>0.34</b> |

Table 5. **Multi-view video offence and severity classification.** We evaluate our VARS with different feature encoders and pooling methods on a single and multi-task setup. BA stands for the balanced accuracy.



Figure 4. **Example of fouls.** (a) The defender uses his arm as a tool to gain an unfair advantage and ignores the potential danger for his opponent. (b) The defender makes a tackle while taking the risk of his opponent being injured. (c) The defender tries to play the ball in no dangerous way. (d) The defender has no intention to play the ball and only aims to harm his opponent.

for the safety of his opponent and therefore resulting in a yellow card. In contrast, Figures 4c and 4d depict fouls that are visually more similar than the previous two fouls, yet one resulted in “No card” while the other resulted in a “Red card”. Minor differences such as the point of contact, the speed of the foul, the distance to the ball, and the intention to play the ball or not, can lead to different classifications.

**Multi-task classifier.** Training a multi-task classifier on related tasks allows the model to utilize the learned information from one task to improve the performance on other tasks. In the bottom part of Table 5, we can see that the multi-task classifier outperforms the single-task classifier regardless of the feature encoder or pooling technique for offence severity classification. Using ResNet to extract spa-

| Performance  | Viewing Setup |      |             |             |             |
|--------------|---------------|------|-------------|-------------|-------------|
|              | L             | R1   | L+R1        | R1+R2       | L+R1+R2     |
| $Acc_{T1}$   | 0.31          | 0.47 | 0.50        | 0.56        | <b>0.57</b> |
| $Acc_{T1}@2$ | 0.54          | 0.68 | 0.70        | 0.69        | <b>0.72</b> |
| $BA_{T1}$    | 0.29          | 0.38 | 0.36        | <b>0.44</b> | 0.39        |
| $Acc_{T2}$   | 0.38          | 0.39 | <b>0.43</b> | 0.39        | 0.40        |
| $Acc_{T2}@2$ | 0.67          | 0.70 | 0.72        | 0.73        | <b>0.75</b> |
| $BA_{T2}$    | 0.38          | 0.27 | 0.34        | 0.27        | <b>0.39</b> |

Table 6. **Single vs. multi-view classification.** We compare the performance for single vs multi-views and the influence of the type of view (Live  $L$  and replay  $R$ ). We use MViT [15, 37] as feature extractor and max pooling. For both tasks, the best performance is mostly obtained with all three views. BA stands for balanced accuracy, T1 stands for task 1 (foul classification) and T2 stands for task 2 (offence severity classification).

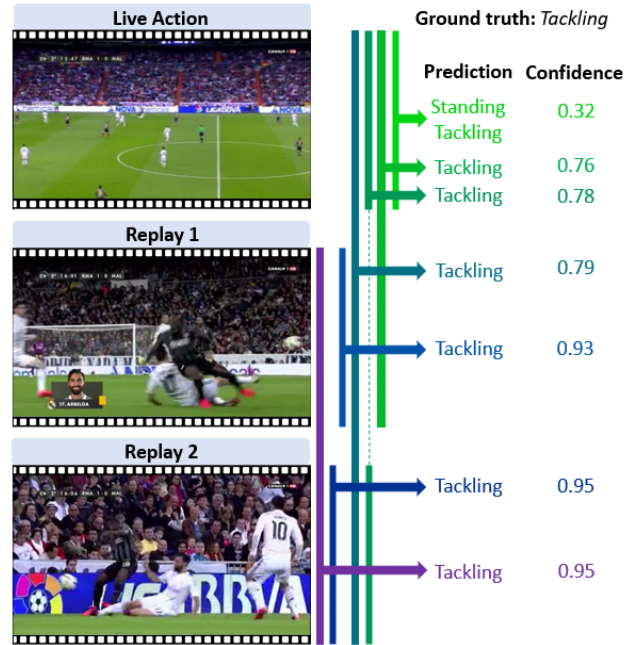


Figure 5. **Qualitative results.** VARS predictions for different combinations of views as input. The best performance is obtained with the two replay views.

tial features for the type of foul and the offence severity classification does not perform well for either task. The body movements over time and the speed of the players involved in an action are important factors that can greatly impact the outcome of the classification. The multi-task classifier combined with MViT as encoder and max pooling shows promising results in classifying actions into their corresponding offence severity class. Furthermore, the multi-task classifier shows similar results as obtained for the single-task type of foul classification.

### 5.3. Detailed analysis

**Single vs. multi-view analysis.** We now study the improvement of using multiple views over a single view. To do so, we first created a subset of the test set for which we have clips with two replays and one live action. As evidenced by the top part of Table 6, the type of view has a significant impact on the VARS’s ability to detect the correct type of foul. Although the live-action view alone provides worse performance than the replays, combining the live-action view with a replay improves the accuracy slightly compared to using only the replay view in both tasks. This implies that even a poor-quality view can slightly improve the performance. A highly informative view can boost the performance, as we can see by comparing the two replays with a single replay for the type of foul classification. For the offence and severity classification, the VARS seems to benefit more from live actions compared to replays for the offense and severity classification task. One possible explanation is that for the live actions, the VARS takes into account the position of the action on the field, allowing it to learn that the likelihood of a ‘No card’ or ‘Yellow card’ is higher in specific areas of the field. For both tasks, we achieved better results by using multiple views, and for most of the metrics, the best performance was achieved by using a live-action clip with two replays. This demonstrates the effectiveness of using multiple views to improve model performance in the type of foul and offence severity classification.

In Figure 5, we show the predictions of the foul classification models while changing the number and type of views. By only using the live action, the VARS is not able to detect the correct type of foul, as confirmed in Table 6. By adding 1 or 2 replays as input to the model, it is able to detect the foul class with a confidence score ranging from 76% to 95%. By analyzing the confidence scores, we can see that the view has a big impact of the prediction, which agrees with the results found in Table 6.

**Temporal analysis.** We investigated the temporal context needed to identify fouls and offence severity. In particular, we increased the video length, by reducing the frame rate, in order to maintain the same number of frames to process. Table 7, shows the results of the temporal analysis. We observed that as we increase the temporal context while decreasing the frame rates, the performance of our model decreases. This is likely because the most useful information for our classification tasks is concentrated within a narrow temporal window immediately preceding and following the foul. Adding more temporal context to the model results in the inclusion of frames that do not offer much additional information. By default, we used a frame rate of 16 frames per second, with a temporal context of 1 second, which seemed to strike the best balance between capturing sufficient temporal information and excluding unnecessary frames.

| Frame rate (FPS)            | 5    | 8    | 12   | 16   |
|-----------------------------|------|------|------|------|
| Temporal context            | 3.2s | 2.0s | 1.3s | 1.0s |
| Accuracy (Foul class.)      | 0.36 | 0.38 | 0.44 | 0.47 |
| Accuracy (Off. sev. class.) | 0.39 | 0.41 | 0.43 | 0.43 |

Table 7. **Temporal analysis.** We experiment with various temporal context while maintaining a fixed number of 16 frames. In all scenarios, we include 8 frames before and after the foul.

**Per class analysis.** We further analyze the performance per class. The confusion matrices for both tasks are in the supplementary material. We saw that performance varies considerably across classes. For the fine-grained foul classification, the VARS struggles to distinguish between illegal arm movements due to their shared characteristics. It performs well in detecting “Tackling”, but often confuses it with “Dive” due to the challenge of distinguishing genuine from deceptive actions in soccer games. The most difficult class for the VARS is “Challenge”, as it shares visual similarities with many other classes, making proper generalization during training difficult. Regarding offence classification, the VARS tends to make bad predictions in neighboring classes of the ground truth. For instance, it may classify a foul as “Offence + Yellow card” instead of “Offence + No card”. However, the model struggles with “Offence + Red card” due to the limited number of samples in the dataset.

## 6. Conclusion

In summary, our Video Assistant Referee System (VARS) has the potential to bring about a significant improvement in soccer refereeing by ensuring fairness and accuracy at all levels of professional and amateur play. VARS utilizes the latest advances in multi-view video analysis and provides referees with real-time feedback and assists them in making informed decisions that can impact the outcome of soccer games. To prove the effectiveness of VARS, we introduced a novel dataset, SoccerNet-MVFoul, that curates relevant fouls in soccer broadcasts from multiple views and includes foul properties. Our benchmarking results demonstrate that VARS can recognize foul characteristics based on multi-view video processing. By integrating the specific requirements of referees, VARS offers an unbiased and reliable decision-making process for soccer matches.

**Acknowledgement.** This work was partly supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding and the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI). A. Cioppa is funded by the F.R.S.-FNRS.



## References

- [1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016. [2](#)
- [2] Adrià Arbués Sangüesa, Adrià Martín, Javier Fernández, Coloma Ballester, and Gloria Haro. Using player’s body-orientation to model pass feasibility in soccer. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3875–3884, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [3] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 961–970, Boston, MA, USA, Jun. 2015. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [4] Alejandro Cartas, Coloma Ballester, and Gloria Haro. A graph-based method for soccer action spotting using unsupervised player classification. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 93–102, Lisbon, Port., Oct. 2022. ACM. [2](#)
- [5] Anthony Cioppa, Adrien Delière, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. Scaling up SoccerNet with multi-view spatial localization and re-identification. *Sci. Data*, 9(1):1–9, Jun. 2022. [2](#), [3](#)
- [6] Anthony Cioppa, Adrien Delière, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. A context-aware loss function for action spotting in soccer videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 13123–13133, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [7] Anthony Cioppa, Adrien Delière, Noor Ul Huda, Rikke Gade, Marc Van Droogenbroeck, and Thomas B. Moeslund. Multimodal and multiview distillation for real-time player detection on a football field. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 3846–3855, Seattle, WA, USA, Jun. 2020. [2](#)
- [8] Anthony Cioppa, Adrien Delière, and Marc Van Droogenbroeck. A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 1846–1855, Salt Lake City, UT, USA, Jun. 2018. [2](#)
- [9] Anthony Cioppa, Silvio Giancola, Adrien Deliege, Le Kang, Xin Zhou, Zhiyu Cheng, Bernard Ghanem, and Marc Van Droogenbroeck. SoccerNet-tracking: Multiple object tracking dataset and benchmark in soccer videos. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 3490–3501, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [10] Abdulrahman Darwish and Tallal El-Shabrawy. STE: Spatio-temporal encoder for action spotting in soccer videos. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 87–92, Lisbon, Port., Oct. 2022. ACM. [2](#)
- [11] Tom Decroos, Lotte Bransen, Jan Van Haaren, and Jesse Davis. Actions speak louder than goals. In *ACM SIGKDD Int. Conf. Knowl. Discov. & Data Min.*, page 1851–1861. ACM, Jul. 2019. [2](#)
- [12] Adrien Delière, Anthony Cioppa, Silvio Giancola, Meisam J. Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. SoccerNet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 4508–4519, Nashville, TN, USA, Jun. 2021. Best CVSports paper award. [2](#), [3](#)
- [13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 248–255, Miami, FL, USA, Jun. 2009. Inst. Electr. Electron. Eng. (IEEE). [2](#), [6](#)
- [14] Carlos Esteves, Yinshuang Xu, Christine Allec-Blanchette, and Kostas Daniilidis. Equivariant multi-view networks. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1568–1577, Seoul, South Korea, Oct. 2019. Inst. Electr. Electron. Eng. (IEEE). [3](#)
- [15] Haoqi Fan, Bo Xiong, Kartikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 6804–6815, Montreal, QC, Canada, Oct. 2021. Inst. Electr. Electron. Eng. (IEEE). [2](#), [6](#), [7](#)
- [16] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 6201–6210, Seoul, South Korea, Oct. 2019. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [17] FIFA. Goal-line technology. <https://www.fifa.com/technical/football-technology/football-technologies-and-innovations-at-the-fifa-world-cup-2022/goal-line-technology>, 2022. [1](#)
- [18] FIFA. Semi automated offside technology. <https://www.fifa.com/technical/football-technology/football-technologies-and-innovations-at-the-fifa-world-cup-2022/semi-automated-offside-technology>, 2023. [2](#)
- [19] Lianli Gao, Zhao Guo, Hanwang Zhang, Xing Xu, and Heng Tao Shen. Video captioning with attention-based LSTM and semantic consistency. *IEEE Trans. Multimedia*, 19(9):2045–2055, Sept. 2017. [2](#)
- [20] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. SoccerNet: A scalable dataset for action spotting in soccer videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 1792–179210, Salt Lake City, UT, USA, Jun. 2018. Inst. Electr. Electron. Eng. (IEEE). [2](#), [3](#)
- [21] Silvio Giancola, Anthony Cioppa, Adrien Delière, Floriane Magera, Vladimir Somers, Le Kang, Xin Zhou, Olivier Barnich, Christophe De Vleeschouwer, Alexandre Alahi, Bernard Ghanem, Marc Van Droogenbroeck, Abdulrahman



- Darwish, Adrien Maglo, Albert Clapés, Andreas Luyts, Andrei Boiarov, Artur Xarles, Astrid Orcesi, Avijit Shah, Baoyu Fan, Bharath Comandur, Chen Chen, Chen Zhang, Chen Zhao, Chengzhi Lin, Cheuk-Yiu Chan, Chun Chuen Hui, Dengjie Li, Fan Yang, Fan Liang, Fang Da, Feng Yan, Fufu Yu, Guanshuo Wang, H. Anthony Chan, He Zhu, Hongwei Kan, Jiaming Chu, Jianming Hu, Jianyang Gu, Jin Chen, João V. B. Soares, Jonas Theiner, Jorge De Corte, José Henrique Brito, Jun Zhang, Junjie Li, Junwei Liang, Leqi Shen, Lin Ma, Lingchi Chen, Miguel Santos Marques, Mike Azatov, Nikita Kasatkin, Ning Wang, Qiong Jia, Quoc Cuong Pham, Ralph Ewerth, Ran Song, Rengang Li, Rikke Gade, Ruben Debien, Runze Zhang, Sangrok Lee, Sergio Escalera, Shan Jiang, Shigeyuki Odashima, Shimin Chen, Shoichi Masui, Shouhong Ding, Sin-wai Chan, Siyu Chen, Tallal El-Shabrawy, Tao He, Thomas B. Moeslund, Wan-Chi Siu, Wei Zhang, Wei Li, Xiangwei Wang, Xiao Tan, Xiaochuan Li, Xiaolin Wei, Xiaoqing Ye, Xing Liu, Xinying Wang, Yandong Guo, Yaqian Zhao, Yi Yu, Yingying Li, Yue He, Yujie Zhong, Zhenhua Guo, and Zhiheng Li. SoccerNet 2022 challenges results. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 75–86, Lisbon, Port., Oct. 2022. ACM. [2](#), [3](#)
- [22] Abdullah Hamdi, Faisal AlZahrani, Silvio Giancola, and Bernard Ghanem. MVTN: Learning multi-view transformations for 3D understanding. *CoRR*, abs/2212.13462, 2022. [6](#)
- [23] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. MVTN: Multi-view transformation network for 3D shape recognition. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1–11, Montreal, QC, Canada, Oct. 2021. Inst. Electr. Electron. Eng. (IEEE). [3](#)
- [24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 770–778, Las Vegas, NV, USA, Jun. 2016. Inst. Electr. Electron. Eng. (IEEE). [6](#), [7](#)
- [25] Yutaro Honda, Rei Kawakami, Ryota Yoshihashi, Kenta Kato, and Takeshi Naemura. Pass receiver prediction in soccer using video and players’ trajectories. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3502–3511, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [26] James Hong, Haotian Zhang, Michaël Gharbi, Matthew Fisher, and Kayvon Fatahalian. Spotting temporally precise, fine-grained events in video. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 13695 of *Lect. Notes Comput. Sci.*, pages 33–51. Springer Nat. Switz., 2022. [2](#)
- [27] IFAB. Laws of the game. Technical report, The International Football Association Board, Zurich, Switzerland, 2022. [4](#), [13](#), [15](#)
- [28] Yudong Jiang, Kaixu Cui, Leilei Chen, Canjin Wang, and Changliang Xu. SoccerDB: A large-scale database for comprehensive video understanding. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, page 1–8. ACM, Oct. 2020. [2](#)
- [29] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5010–5019, Salt Lake City, UT, USA, Jun. 2018. Inst. Electr. Electron. Eng. (IEEE). [3](#)
- [30] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1725–1732, Columbus, OH, USA, Jun. 2014. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [31] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. [2](#), [3](#), [6](#)
- [32] Muhammad Zeeshan Khan, Summra Saleem, Muhammad A. Hassan, and Muhammad Usman Ghanni Khan. Learning deep C3D features for soccer video event detection. In *Int. Conf. Emerg. Technol. (ICET)*, pages 1–6, Islamabad, Pakistan, Nov. 2018. [2](#)
- [33] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 706–715, Venice, Italy, Oct. 2017. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [34] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009. Technical report, University of Toronto. [2](#)
- [35] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Unsupervised learning of view-invariant action representations. *Advances in neural information processing systems*, 31, 2018. [3](#)
- [36] Yitong Li, Martin Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. *AAAI Conf. Artif. Intell.*, 32(1), Apr. 2018. [2](#)
- [37] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022. [2](#), [6](#), [7](#)
- [38] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C. Kot. NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(10):2684–2701, Oct. 2020. [3](#)
- [39] Adrien Maglo, Astrid Orcesi, and Quoc-Cuong Pham. Efficient tracking of team sport players with few game-specific annotations. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3460–3470, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). [2](#)
- [40] Javier Mallo, Pablo Gonzalez Frutos, Daniel Ju’arez, and Enrique Navarro. Effect of positioning on the accuracy of decision making of association football top-class referees and assistant referees during competitive matches. *Journal of Sports Sciences*, 30(13):1437–1445, Sept. 2012. [4](#), [5](#)
- [41] Thomas B. Moeslund, Graham Thomas, and Adrian Hilton. *Computer vision in sports*. Springer, 2014. [2](#)

- [42] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4694–4702, Boston, MA, USA, Jun. 2015. Inst. Electr. Electron. Eng. (IEEE). 2
- [43] Luca Pappalardo, Paolo Cintia, Alessio Rossi, Emanuele Massucco, Paolo Ferragina, Dino Pedreschi, and Fosca Giannotti. A public data set of spatio-temporal match events in soccer competitions. *Sci. Data*, 6(1):1–15, Oct. 2019. 2
- [44] Hossein Rahmani, Arif Mahmood, Du Huynh, and Ajmal Mian. Histogram of oriented principal components for cross-view action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(12):2430–2443, Dec. 2016. 3
- [45] Himangi Saraogi, Rahul Anand Sharma, and Vijay Kumar. Event recognition in broadcast soccer videos. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 1–7, Dec. 2016. 2
- [46] Atom Scott, Ikuma Uchida, Masaki Onishi, Yoshinari Kameda, Kazuhiro Fukui, and Keisuke Fujii. SoccerTrack: A dataset and tracking algorithm for soccer with fish-eye and drone videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3568–3578, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [47] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 27, pages 568–576, Dec. 2014. 2
- [48] João V. B. Soares and Avijit Shah. Action spotting using dense detection anchors revisited: Submission to the SoccerNet challenge 2022. *CoRR*, abs/2206.07846, 2022. 2
- [49] Joao V. B. Soares, Avijit Shah, and Topojoy Biswas. Temporally precise action spotting in soccer videos using dense detection anchors. In *IEEE Int. Conf. Image Process. (ICIP)*, pages 2796–2800, Bordeaux, France, Oct. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [50] Vladimir Somers, Christophe De Vleeschouwer, and Alexandre Alahi. Body part-based representation learning for occluded person Re-Identification. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, pages 1613–1623, Waikoloa, HI, USA, Jan. 2023. Inst. Electr. Electron. Eng. (IEEE). 2
- [51] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 2
- [52] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 945–953, Santiago, Chile, Dec. 2015. Inst. Electr. Electron. Eng. (IEEE). 3
- [53] Genki Suzuki, Sho Takahashi, Takahiro Ogawa, and Miki Haseyama. Team tactics estimation in soccer videos based on a deep extreme learning machine and characteristics of the tactics. *IEEE Access*, 7:153238–153248, 2019. 2
- [54] Graham Thomas, Rikke Gade, Thomas B. Moeslund, Peter Carr, and Adrian Hilton. Computer vision for sports: current applications and research topics. *Comput. Vis. Image Underst.*, 159:3–18, Jun. 2017. 2
- [55] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 6450–6459, Salt Lake City, UT, USA, Jun. 2018. Inst. Electr. Electron. Eng. (IEEE). 2, 6, 7
- [56] UEFA. New age for referees. <https://www.uefa.com/insideuefa/news/0194-0f844fe11b14-503126864653-1000--new-age-for-referees/>, Aug. 2003. 1
- [57] Gabriel Van Zandycke, Vladimir Somers, Maxime Istasse, Carlo Del Don, and Davide Zambrano. DeepSportradar-v1: Computer vision dataset for sports understanding with high quality annotations. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 1–8, Lisbon, Port., Oct. 2022. ACM. 2
- [58] Renaud Vandeghen, Anthony Cioppa, and Marc Van Droogenbroeck. Semi-supervised training to improve player and ball detection in soccer. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, CVsports, pages 3480–3489, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [59] Shruti Vyas, Yogesh S. Rawat, and Mubarak Shah. Multi-view action recognition using cross-view video prediction. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 12372 of *Lect. Notes Comput. Sci.*, pages 427–444. Springer Int. Publ., 2020. 3
- [60] Jiang Wang, Xiaohan Nie, Yin Xia, Ying Wu, and Song-Chun Zhu. Cross-view action modeling, learning, and recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2649–2656, Columbus, OH, USA, Jun. 2014. Inst. Electr. Electron. Eng. (IEEE). 3
- [61] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 9912 of *Lect. Notes Comput. Sci.*, pages 20–36. Springer Int. Publ., 2016. 2
- [62] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(11):2740–2755, Nov. 2019. 2
- [63] Xin Wang, Wenhui Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. Video captioning via hierarchical reinforcement learning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4213–4222, Salt Lake City, UT, USA, Jun. 2018. Inst. Electr. Electron. Eng. (IEEE). 2
- [64] Xin Wei, Ruixuan Yu, and Jian Sun. View-GCN: View-based graph convolutional network for 3D shape analysis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1847–1856, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). 3
- [65] Fei Wu, Qingzhong Wang, Jian Bian, Haoyi Xiong, Ning Ding, Feixiang Lu, Jun Cheng, and Dejing Dou. A survey on video action recognition in sports: Datasets, methods and applications. *CoRR*, abs/2206.01038, 2022. 2
- [66] Junqing Yu, Aiping Lei, Zikai Song, Tingting Wang, Hengyou Cai, and Na Feng. Comprehensive dataset of

- broadcast soccer videos. In *IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, pages 418–423, Miami, FL, USA, Apr. 2018. Inst. Electr. Electron. Eng. (IEEE). 2
- [67] He Zhu, Junwei Liang, Chengzhi Lin, Jun Zhang, and Jianming Hu. A transformer-based system for action spotting in soccer videos. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 103–109, Lisbon, Port., Oct. 2022. ACM. 2

## 7. Supplementary

### 7.1. Video assistant referee system software

The design of both the VARS annotator and the VARS interface draws inspiration from the VAR room. To enhance the user experience, a grid layout is used to display all available perspectives synchronously. The objective is to have an easy to use interface to annotate and to predict different properties of an action.

**VARS annotator.** To increase the speed of the annotation task, we build a VARS annotator (Figure 6a), which shows all the available clips of an action simultaneously. The VARS annotator allows for individual adjustment of the annotated moment for each clip to achieve temporal alignment, speed adjustment for replays, and annotation of all the properties.

**VARS interface.** The VARS interface has the same interface as the annotator. It enables easy access to all available perspectives for a particular action. The multi-task VARS, which achieved the best results on the test set, is built directly into the interface, allowing for immediate analysis of selected videos. The VARS interface offers top two predictions for the type of foul classification, as well as the offense and severity classification for the selected videos. Furthermore, for each prediction, the VARS interface shows the confidence score of his prediction.

In Figure 6b, we can see an example of how the VARS interface looks. In this example, the VARS correctly predicts the type of foul, and tells that the action was indeed a foul, which leads in a penalty for the attacking team in this example. For the severity classification, the VARS was uncertain whether to assign no card or a yellow card.

### 7.2. Dataset

#### 7.2.1 Property explanations

This section explains in detail each property. Furthermore, we illustrate multiple examples of our dataset in Figure 7.

**Was it a foul?** The task of identifying fouls is a critical and challenging aspect of the role of a referee and VAR. They must determine whether an action is a foul or not, in accordance to the laws of the game [27]. For each action, we determined whether an action is an (i) offence (an action which breaks/violates the Laws of the Game [27]), (ii) no offence (did not break/violate the Laws of the Game), or (iii) between (if the action lays inside a grey area). In some cases, both decisions may be correct and the final decision depends on the interpretation of the rules of an individual.

It is worth mentioning that, for each clip of the same foul, we make the same annotation. During the annotation process, we looked at all the clips and took a global and final decision which is the same for each clip of the same foul.

**Was there any contact?** Another important property which we annotated was if there was any contact between two players during an action. We annotated for each foul (i) with contact (if there was contact between players), or (ii) without contact (if there was no contact). This property is important because a foul with contact such as a tackling, holding, elbowing result in a direct free kick, while a foul without contact such as a simulation or dangerous play will result in an indirect free kick.

**Did the player touch the ball with his hand/arm?** This property annotates if the player touches the ball, deliberately or not, with his hand or arm. We annotate (i) handball (if a players touches the ball with his hand/arm), or (ii) no handball (if the ball did not touch the hand/arm). This property only states whether the ball touched the hand or arm and does not indicate whether the handball is punishable or not. An important note to make is that the upper boundary of the arm is in line with the bottom of the armpit [27].

**Was the upper or under body used in the action?** This property annotates which part of the body was used during an action. We differentiated between (i) under body (which corresponds to the use of the foot or the leg), or (ii) upper body (which corresponds to the use of shoulder or the use of arms).

**With which part of the upper body was the action made?** In the case where we annotated the previous property with "upper body", we further split between (i) use of shoulders, or (ii) use of arms.

**Class of the action.** This property annotates the type of action. In total, we have 9 different classes:

1. **Tackling.** The sliding movement of a player towards an opponent who is in possession of the ball and legally or illegally using his foot or leg to try to take the ball away.
2. **Standing tackling.** The movement (not sliding) of a player towards an opponent who is in possession of the ball and legally or illegally using his foot or leg to try to take the ball away.
3. **Holding.** Occurs when a player's contact with an opponent's body or equipment impedes the opponent's movement [27].
4. **Pushing.** The action of using the upper body to push an opponent away.
5. **Challenge.** Physical challenge against an opponent, using the shoulder and/or the upper arm [27].
6. **Elbowing.** The use of arms (and frequently the elbows) as a tool or a weapon to gain an unfair advantage in aerial challenges, physical battles, to create space or to intimidate other players.





(a) **VARS annotator.** The annotator may browse simultaneously the synchronised videos either at regular speed or frame by frame. He can annotate all 10 properties and adjust the annotated point of contact for each clip separately and temporal align the different clips by modifying the speed and offset of the clips.



(b) **Video Assistant Referee System interface.** The interface of the VARS shows the ground truth of the action and his top 2 predictions for the foul classification task, and the offence and severity classification task. In this example, the VARS correctly predicts the foul resulting in a penalty.

Figure 6. **Views of the VARS interfaces.** (a) shows the interface for the annotation process. (b) shows the interface of the VARS for displaying its results.

7. **High leg.** A movement where a player swings his foot close to and above the waist of an opponent.

8. **Dive.** An action which creates a wrong/false impression that something has occurred when it has not, com-



mitted by a player to gain an unfair advantage. [27]

9. **Don't know.** Corresponds to anything which can not be classified in one of the classes above.

**How severe was the foul?** For each foul, we annotated the severity of the foul by a scale from 1 to 5:

- **1:** a careless foul which is when a player shows a lack of attention or consideration when making a challenge or acts without precaution. No disciplinary sanction is needed. (No card) [27]
- **2:** a borderline foul between careless and reckless. We are in a grey area where both “no card” or “yellow card” would be correct.
- **3:** a reckless foul which is when a player acts with disregard to the danger to, or consequences for, an opponent and must be cautioned. (Yellow card) [27]
- **4:** a borderline foul between reckless and violent. We are in a grey area where both “yellow card” or “red card” would be correct.
- **5:** a violent foul where a player exceeds the necessary use of force and/or endangers the safety of an opponent and must be sent off. (Red card) [27].

**Did the player try to play the ball?** When a player commits an offence against an opponent within their own penalty area which denies an opponent an obvious goal-scoring opportunity and the referee awards a penalty kick, the offender is cautioned if the offence was an attempt to play the ball; the offender is sent off if there was no possibility to play the ball [27]. We annotated, (i) “Yes”, if the player tried to play the ball, or “No”, if there was no possibility to play the ball.

**Did the player play the ball?** The final property annotates (i) “Yes” if the defender touches the ball, (ii) “No” when the defender did not play the ball, or (iii) “Maybe” in the case where the quality of the video or the viewpoint on the foul is not sufficient to determine if the player touched the ball or not.

## 7.2.2 Dataset distribution

The distribution of all classes is provided in Tables 8 and 9. Most of the properties in the dataset have a high degree of imbalance, particularly the “Handball” property that determines if a player has made contact with the ball using their arm or hand. Nearly 99% of the actions recorded in the dataset do not involve any handball. Similar imbalances are observed in the “Contact”, “Try to play the ball”, “Played the ball”, and “Offence” properties.

The “Bodypart” and “Upperbody part” properties are relatively less unbalanced, with a distribution of approximately 66% for the superior class and 34% for the inferior class.

## 8. Experiments

### 8.1. Per class analysis

#### Foul classification task

The performances for each class are summarized in the confusion matrix shown in Figure 8. Our analysis shows that the performance varies considerably across classes. The VARS often confuses all the illegal use of arm classes, like “Holding”, “Pushing”, and “Elbowing” as these fouls share some common characteristics and can involve similar physical movements. The model performs well in detecting “Tackling”, but confuses it often with “Dive” as it struggles to distinguish between genuine and deceptive actions, which can be challenging due to the complex and dynamic nature of soccer games. However, the most challenging class for the VARS is “Challenge”, which shares visual similarities with many other classes, making it difficult for the system to generalize properly during training.

#### Offence and severity classification task

Figure 9 displays the confusion matrix for the offence and severity classification, revealing that the model frequently confuses classes with their neighboring classes. For instance, when the ground truth is “Offence + No card”, the VARS often mistakes it for “No offence” or “Offence + Yellow card”.

Indeed, the visual similarity between all the classes, especially with the neighbor classes, is very high. Small details, which very often can only be seen in a couple of frames, can differ between the actual class. Factors such as the speed of the foul, the point of contact, or the intention of playing the ball are critical criteria for deciding which class an action corresponds to. However, these criteria can be challenging to spot for a model and to differentiate between the different classes. Furthermore, there is only a small number of instances of “Offence + Red card” in the dataset, making it more challenging for the model to generalize. Despite all these difficulties, the VARS is still able to achieve an accuracy of 0.43.



(a) “Offence”, “Tackling”, “Yellow card”, “With contact”, “Under body”, “f”, “Played the ball”, “Tried to play the ball”, “No handball”



(b) “Offence”, “High leg”, “Red card”, “With contact”, “Under body”, “f”, “Ball is not played”, “Tried to play the ball”, “No handball” and “No handball offence”



(c) “Offence”, “Challenge”, “No card”, “With contact”, “Upper body”, “Use of shoulder”, “Ball is not played”, “Tried to play the ball”, “No handball” and “No handball offence”



(d) “Offence”, “standing tackling”, “Yellow card”, “With contact”, “Under body”, “f”, “Did not play the ball”, “Tried to play the ball”, “No handball”

Figure 7. **Dataset overview and ground truth.** We annotated the exact frame where the point of contact happens (depicted by the back box).

| Fouls        |       | Severity    |       | Offence    |       | Handball |       | Handball offence |       |
|--------------|-------|-------------|-------|------------|-------|----------|-------|------------------|-------|
| Class        | Prob. | Class       | Prob. | Class      | Prob. | Class    | Prob. | Class            | Prob. |
| St. tackling | 0.43  | No card     | 0.55  | Offence    | 0.85  | Yes      | 0.99  | Yes              | 0.82  |
| Tackling     | 0.15  | Yellow card | 0.26  | No offence | 0.10  | No       | 0.01  | No               | 0.18  |
| Challenge    | 0.13  | NC/YC       | 0.15  | Between    | 0.03  |          |       |                  |       |
| Holding      | 0.12  | YC/RD       | 0.02  |            |       |          |       |                  |       |
| Elbowing     | 0.05  | Red card    | 0.01  |            |       |          |       |                  |       |
| High leg     | 0.03  |             |       |            |       |          |       |                  |       |
| Pushing      | 0.02  |             |       |            |       |          |       |                  |       |
| Dive         | 0.01  |             |       |            |       |          |       |                  |       |

Table 8. Distribution of classes in our SoccerNet-MVFouls dataset.

| Bodypart  |       | Upperbody part |       | Try to play the ball |       | Played the ball |       | Contact |       |
|-----------|-------|----------------|-------|----------------------|-------|-----------------|-------|---------|-------|
| Class     | Prob. | Class          | Prob. | Class                | Prob. | Class           | Prob. | Class   | Prob. |
| Upperbody | 0.36  | Arms           | 0.66  | Yes                  | 0.92  | Yes             | 0.10  | With    | 0.97  |
| Underbody | 0.64  | Shoulder       | 0.33  | No                   | 0.08  | No              | 0.87  | Withou  | 0.03  |
|           |       |                |       |                      |       | Maybe           | 0.02  |         |       |

Table 9. Distribution of classes in our SoccerNet-MVFouls dataset.

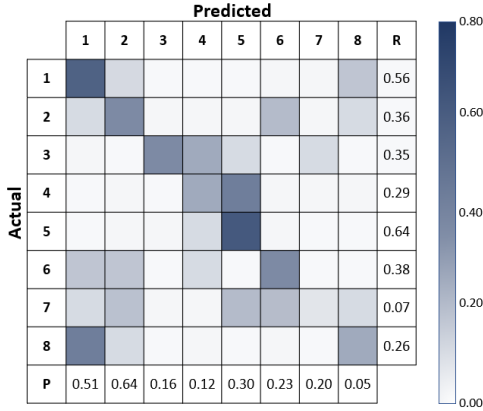


Figure 8. **Confusion matrix for the type of foul classification.** The VARS demonstrates good performance in classifying “Standing Tackling”, “Tackling”, and “Elbowing”. However, the model struggles with “Challenge” and frequently confuses it with other classes. 1: Standing Tackling, 2: Tackling, 3: High Leg, 4: Pushing, 5: Elbowing, 6: Holding, 7: Challenge, 8: Dive, R: Recall and P: Precision.

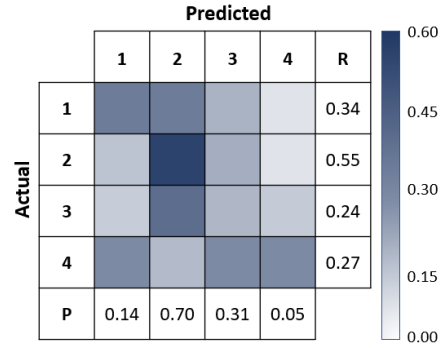


Figure 9. **Confusion matrix for the offence and severity classification.** The VARS shows good performance for the “Offence + No Card” class. The model confuses the classes “No Offence” and “Offence + Yellow Card” with “Offence + No Card”. For the class “Offence + Red Card” the model is not able to provide good results due to the low amount of samples in the dataset. 1: No offence, 2: Offence + No card, 3: Offence + Yellow card, 4: Offence + Red card, R: Recall and P: Precision.

# Towards AI-Powered Video Assistant Referee System (VARs) for Association Football

Jan Held<sup>1\*</sup>, Anthony Cioppa<sup>1,2\*</sup>, Silvio Giancola<sup>2\*</sup>, Abdullah Hamdi<sup>3</sup>, Christel Devue<sup>1</sup>, Bernard Ghanem<sup>2</sup> and Marc Van Droogenbroeck<sup>1</sup>

<sup>1</sup>University of Liege (ULiège), Belgium.

<sup>2</sup>King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

<sup>3</sup>University of Oxford, United Kingdom.

\*Corresponding author(s). E-mail(s): [jan.held@liege.be](mailto:jan.held@liege.be); [anthony.cioppa@uliege.be](mailto:anthony.cioppa@uliege.be); [silvio.giancola@kaust.edu.sa](mailto:silvio.giancola@kaust.edu.sa);

## Abstract

Over the past decade, the technology used by referees in football has improved substantially, enhancing the fairness and accuracy of decisions. This progress has culminated in the implementation of the Video Assistant Referee (VAR), an innovation that enables backstage referees to review incidents on the pitch from multiple points of view. However, the VAR is currently limited to professional leagues due to its expensive infrastructure and the lack of referees worldwide. In this paper, we present the Video Assistant Referee System (VARs) that leverages the latest findings in multi-view video analysis. Our VARs achieves a new state-of-the-art on the *SoccerNet-MVFoul* dataset by recognizing the type of foul in **50%** of instances and the appropriate sanction in **46%** of cases. Finally, we conducted a comparative study to investigate human performance in classifying fouls and their corresponding severity and compared these findings to our VARs. The results of our study highlight the potential of our VARs to reach human performance and support football refereeing across all levels of professional and amateur federations.

**Keywords:** Football, Soccer, Artificial Intelligence, Computer Vision, Video Recognition, Automated Decision, Video Assistant Referee, Referee Success Rate, Fouls evaluation

## 1 Introduction

In recent years, technology has played an increasing role in football, revolutionizing how the game is played, coached, and officiated. This transformation extends into the domain of sports video analysis, which encompasses a diverse range of challenging tasks, including player detection and tracking [1–4], spotting actions in untrimmed videos [5–12], pass feasibility and prediction [13, 14], summarizing [15–18], camera calibration[19],

player re-identification in occluded scenarios [20], or dense video captioning for football broadcasts commentaries [21, 22]. Solving these tasks has been taken to a higher level thanks to the emergence of deep learning techniques [23–25]. Similar to many other fields in which deep learning has been used, the advancements in sports video understanding heavily rely on the availability of large-scale datasets [26–30]. SoccerNet [31–39]



stands among the largest and most comprehensive sports dataset, with extensive annotations for video understanding in football.

In refereeing, the biggest revolution was introduced by the Video Assistant Referee (VAR) in 2016 [40]. The system involves a team of referees located in a video operation room outside the stadium. These referees have access to all available camera views and check all decisions taken by the on-field referee. If the VAR indicates a probable “clear and obvious error” (*E.g.* when the referee misses a penalty or a red card, gives a yellow card to the wrong player, etc.), it will be communicated to the on-field referee who can then review his decision in the referee review area before taking a final decision. The VAR helps to ensure greater fairness in the game by reducing the impact of incorrect decisions on the outcome of games. Notably, in 8% of the matches, the VAR has a decisive impact on the result of the game [41] and it slightly reduces the unconscious bias of referees towards home teams [42]. On average, away teams now score more goals and receive fewer yellow cards [43]. Controversial referee mistakes like the famous “hand of God” goal by Diego Maradona during the quarter-final match Argentina versus England of the 1986 FIFA World Cup, Josip Šimunić getting three yellow cards in a single game at the 2006 FIFA World Cup, or Thierry Henry’s handball preventing the Republic of Ireland from qualifying for the World Cup could have been avoided with the VAR and would have changed football history.

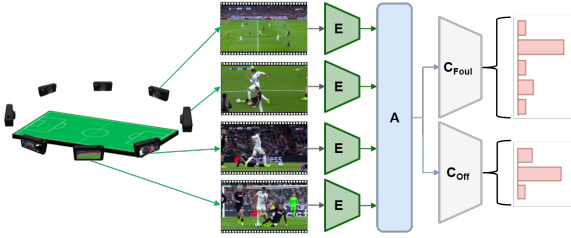
Despite its potential benefits, the use of the VAR technology remains limited to professional leagues. The infrastructure of the VAR is expensive, including multiple cameras to analyze the incident from different angles, video operation rooms in various locations, and VAR officials hired to analyze the footage. Leagues with financial limitations cannot afford the necessary infrastructure to operate the VAR. In addition to the upfront costs of the infrastructure, there is also an ongoing expense associated with using the VAR. The officials who serve as Video Assistant Referees require specialized training [44] and monetary compensation following each game. Given the implementation and operational costs of VAR, its use is currently restricted to professional leagues. A further obstacle is the shortage of referees world-wide. In Germany, there were only 50,241 active

referees during the 2020/2021 season, whereas the number of games played each weekend was around 90,000 [45, 46]. The introduction of the VAR requires an additional team of referees per game, which is not feasible for semi-professional or amateur leagues. Finally, each referee interprets the Laws of the Game [47] slightly differently, resulting in different decisions for similar actions. Given that the video assistant referee (VAR) changes from one game to another, inconsistencies may arise, with the VAR making different decisions for similar actions across different matches.

In this paper, we present the “Video Assistant Referee System” (VARS), which could support or extend the current VAR. Our VARS fulfills the same objectives and tasks as the VAR. By analyzing fouls from a single or a multi-camera video feed, it indicates a probable “clear and obvious error”, and can communicate this information to the referee, who will then decide whether to initiate a “review”. The proposed VARS automatically analyzes potential incidents that can then be shown to the referee in the referee review area. Just like the regular VAR, our VARS serves as a support system for the referee and only alerts him in the case of potential game-changing mistakes, but the final decision remains in the hands of the main referee. The main benefit of our VARS is that it no longer requires additional referees, making it the perfect tool for leagues that do not have enough financial or human resources.

**Contributions.** We summarize our contributions and novelties as follows: **(i)** We propose an upgraded version of the *VARS* presented by Held *et al.* [35]. We introduce an attention mechanism on the different views and calculate an importance score to allocate more attention to more informative views before aggregating the views. **(ii)** We present a thorough study on the influence of using multiple views and different types of camera views on the performance of our VARS. **(iii)** We present a comprehensive human study where we compare the performance of human referees, football players, and our VARS on the task of type of foul classification and offense severity classification. Our human study also illustrates the subjectivity of refereeing decisions by examining the inter-rater agreement among referees.





**Fig. 1: Architecture of our Video Assistant Referee System.** From multi-view video clips input, our system **encodes per-view video features (E)**, **aggregates the view features (A)**, and classifies different properties (**C<sub>Foul</sub>** and **C<sub>Off</sub>**). The figure is inspired by [35].

## 2 Methodology

We propose an upgraded version of the Video Assistant Referee System, which adds an advanced pooling technique to combine the information from multiple views, extracting the most relevant information based on our attention mechanisms.

The architecture is shown in Figure 1. Formally, the VARS takes multiple video clips  $\mathbf{v} = \{v_i\}_1^n$  as input. Each video clip shows the same action from  $n$  different perspectives. Each clip  $v_i$  is fed into a video encoder **E** to extract a spatio-temporal feature vector  $f_i$  of dimension  $d$  for each clip  $v_i$ . All feature vectors  $f_i$  are then stored in a matrix  $\mathbf{f}$  as follows:

$$\mathbf{f} = [f_1, f_2, \dots, f_n]^T. \quad (1)$$

An aggregation block **A** takes  $\mathbf{f}$  as input and outputs a single multi-view representation  $\mathbf{R}$ . A multi-head classifier, **C<sub>Foul</sub>** and **C<sub>Off</sub>**, simultaneously predicts the fine-grained type of foul class and the offense severity class. For each task, the VARS selects the value with the highest confidence from the respective confidence vector as the final prediction, following:

$$\mathbf{VARS}^t \leftarrow \operatorname{argmax} \mathbf{C}_{\theta_{C^t}}^t(\mathbf{R}), \forall t \in \{\text{foul}, \text{off}\}, \quad (2)$$

where  $\theta_{C^t}$  corresponds to the parameters of the classification head for task  $t \in \{\text{foul}, \text{off}\}$ . The model is trained by minimizing the unweighted summation of both task losses  $\mathcal{L}^{\text{foul}}$  and  $\mathcal{L}^{\text{off}}$ .

**Video Encoder E.** Based on the work presented in [35], the best performance is obtained with a video encoder that extracts spatial and temporal

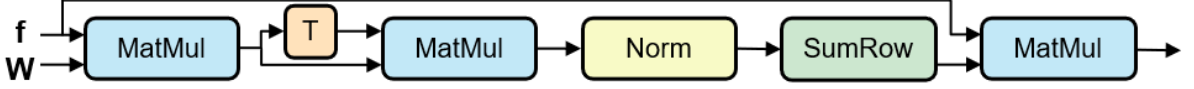
features. In the following, we use the state-of-the-art video encoder MViT [48, 49] pretrained on Kinetics [50], which incorporates a transformer-based architecture with a multiscale feature representation, allowing it to capture spatial and temporal information from video clips.

**Multi-view aggregation block A.** The original paper [35] used simple mean or max pooling operations to gather the multi-view information into a unique representation. A major drawback of these pooling approaches is that the combination of the feature vectors is fixed and ignores the relationship between the views. Instead, we propose a new aggregation technique based on an attention mechanism to model such relationships.

Our approach is inspired by the ‘‘Integrating Block’’ presented in [51], where each view is associated with an attention score. However, instead of aggregating multi-view images, we extend the operation to multi-view videos. Technically, we assign an attention score to each view and then calculate the final representation by a weighted sum of the feature vectors. There exist several strategies to assign an attention score to a view. A first naive approach consists of passing each feature vector individually into a learned function. However, this would neglect the relationships between the views and would not provide a relative attention score of the views. A better approach consists of determining the attention score of each view based on its relationships with the other views. To do so, we first take the dot product (denoted by  $\cdot$ ) of  $\mathbf{f}$  multiplied by a matrix  $\mathbf{W} \in \mathbb{R}^{d \times d}$  of trainable weights and its transpose:

$$\mathbf{S} = \mathbf{f}\mathbf{W} \cdot (\mathbf{f}\mathbf{W})^T. \quad (3)$$

By multiplying the matrix  $\mathbf{f}$  with its transpose, we compute the dot product between each pair of feature vectors, which measures the similarity between two vectors. The obtained symmetric similarity matrix  $\mathbf{S}$  is of dimension  $n \times n$ , where the value at row  $i$  and column  $j$  corresponds to the similarity score between view  $i$  and view  $j$ . A higher score indicates a higher similarity between the vectors, while a lower score suggests a lower similarity. Next, we normalize the similarity scores to get a probability-like distribution, by passing the matrix  $\mathbf{S}$  through a *ReLU* layer and divide it by the sum of the matrix  $\mathbf{S}$ , following:



**Fig. 2: Architecture of the attention block.** “MatMul” represents matrix multiplication, “T” denotes transpose, “Norm” signifies normalization, and “SumRow” indicates the process of summing each row.

$$\mathbf{N} = \frac{\text{ReLU}(\mathbf{S})}{\sum_{i=1}^n \sum_{j=1}^n \text{ReLU}(\mathbf{S}_{i,j})}. \quad (4)$$

To obtain the attention score for each view, we sum the values in each row of the normalized similarity matrix  $\mathbf{N}$ . The attention score for a view  $i$  represents the sum of its normalized similarity scores with all other views. By summing the values in each row of the normalized similarity matrix  $\mathbf{N}$ , we aggregate the normalized similarity scores for each view. This aggregation reflects how similar a particular view is to all other views collectively. Consequently, the resulting attention score captures a view’s overall relevance within the set of views. The reasoning behind this approach is that if a view is highly similar to many other views, it is considered important because it shares visual content with multiple views. On the other hand, if a view is dissimilar to other views, it might be considered less important since it does not contribute significantly to the collective visual information. Formally, we take the sum per row to obtain the attention score  $\mathbf{A}$  per view:

$$\mathbf{A} = \sum_{i=1}^n \mathbf{N}_{i,j}, \quad (5)$$

where  $\mathbf{A}$  is a vector of size  $n$ , where the value  $j$  corresponds to the attention score of the view  $j$  regarding all other views and itself. The final representation is given by the sum of the extracted feature vector weighted by their calculated attention score, following:

$$\mathbf{R}_i = \sum_{j=1}^n \mathbf{f}_{i,j} \times \mathbf{A}_j. \quad (6)$$

**Classification heads C.** A multi-task classification approach is used to classify simultaneously the type of foul, whether it is an offense or not, and its severity. As both tasks are related, learning them together can lead to improved generalization and a better understanding of each task. The model can leverage the relationships between the

two tasks to make better predictions. Each classification head consists of two dense layers and takes as input the aggregated representation. The output is a vector whose dimensions correspond to the number of classes in each of the classification problems.

## 3 Experiments

### 3.1 Experimental setup

**Tasks.** We test our VARS on the two classification tasks introduced by the SoccerNet-MVFouls dataset [35]: *Fine-grained foul classification*, which is the task of classifying a foul into one of 8 fine-grained foul classes (*i.e.*, “Standing tackling”, “Tackling”, “High leg”, “Pushing”, “Holding”, “Elbowing”, “Challenge”, and “Dive/Simulation”), and *Offence severity classification*, which is the task of classifying whether an action is an offence, as well as the severity of the foul, defined by four classes: “No offence”, “Offence + No card”, “Offence + Yellow card”, and “Offence + Red card”.

**Data.** The SoccerNet-MVFoul dataset contains 3,901 actions, composed of at least two videos, the live action and at least one replay, see Figure 1. The views were manually synchronized by a human and no pre-processing of the video clips is necessary. Our VARS is trained on clips of 16 frames, mostly 8 frames before the foul and 8 after the foul, spanning one second temporally with a spatial dimension re-scaled to  $224 \times 224$  pixels. This approach was chosen because of the high computational cost associated with using a larger number of frames. Future research could explore whether an increase in frame rate or a larger temporal context enhances performance.

**Training details.** The encoder  $\mathbf{E}$  is pre-trained as detailed in the methodology, and the classifier  $\mathbf{C}$  is trained from scratch, both being trained in an end-to-end fashion. We use a cross-entropy loss, optimized with Adam on a batch size of 6 samples.

The learning rate starts at  $5e^{-5}$  and is multiplied by 0.3 every 3 steps. To artificially increase the dataset size, we use data augmentation and a random temporal shift to have a flexible number of frames used before and after the foul frame annotation during training. The model begins to overfit after 7 epochs and requires approximately 8 hours of training time on a single NVIDIA V100 GPU.

**Evaluations metrics.** To evaluate the performance of the VARS, SoccerNet-MVFouls uses the classification accuracy, which is the ratio of correctly classified actions regarding the total number of actions. As SoccerNet-MVFouls [35] is unbalanced, the authors also suggest a balanced accuracy, which is defined as follows:

$$\text{Balanced Accuracy (BA)} = \frac{1}{N} \sum_{i=1}^N \frac{TP_i}{P_i}, \quad (7)$$

with  $N$  being the number of classes,  $TP_i$  the number of True Positives and  $P_i$  the number of Positives for class  $i$ . To ensure a fair comparison, we use the same training, validation, and test sets as those used in the original paper [35].

### 3.2 Main results

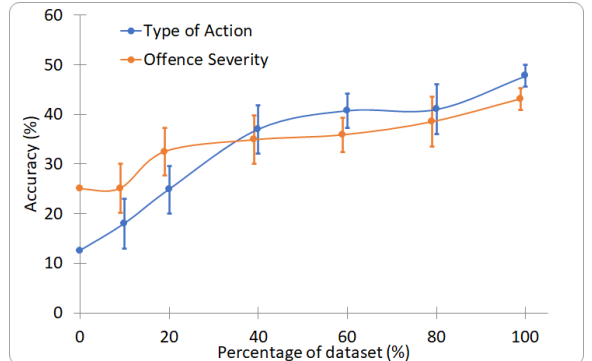
Table 1 shows the results obtained for the fine-grained foul and the offense severity classification task. Compared to the fixed combination of the feature vectors (mean or max pooling), our novel attention mechanism enhances the model’s ability to identify and classify the type of foul by 5% and the balanced accuracy by 1%. This demonstrates the effectiveness of combining the feature vectors of the different views based on their importance compared to max or mean pooling. Similarly, the attention mechanism improves the model’s performance to determine if an action is a foul and the corresponding severity by 3% and the balanced accuracy remains the same compared to max pooling. One might argue that the performance increase is based on the supplementary parameters introduced by the attention mechanism. However, the attention mechanism only adds an extra 0.1% of parameters to the model compared to when using max and mean pooling. This suggests that the performance increase derives from the use of the attention mechanism rather than the introduction of additional parameters.

| Feat. extr. | Pooling   | Type of Foul |             | Offence Severity |             |
|-------------|-----------|--------------|-------------|------------------|-------------|
|             |           | Acc.         | BA.         | Acc.             | BA.         |
| ResNet      | Mean      | 0.30         | 0.27        | 0.34             | 0.25        |
| ResNet      | Max       | 0.32         | 0.27        | 0.32             | 0.24        |
| R(2+1)D     | Mean      | 0.32         | 0.34        | 0.34             | 0.30        |
| R(2+1)D     | Max       | 0.34         | 0.33        | 0.39             | 0.31        |
| MViT        | Mean      | 0.44         | 0.40        | 0.38             | 0.31        |
| MViT        | Max       | 0.45         | 0.39        | 0.43             | <b>0.34</b> |
| MViT        | Attention | <b>0.50</b>  | <b>0.41</b> | <b>0.46</b>      | <b>0.34</b> |

**Table 1: Multi-task classification.** Attention pooling sets a new benchmark on the *SoccerNet-MVFoul* dataset for all the evaluation metrics and tasks. Type of foul classification accuracy increased by 5% while the balanced accuracy (BA) increased by 1%. We have an increment of 3% for the offense severity classification, while the balanced accuracy stays the same.

### 3.3 Detailed analysis

**Sensitivity analysis.** We first investigate the impact of the training dataset size on the performance of our two classification tasks. Figure 3 shows the evolution of the accuracy regarding different training dataset sizes. For each dataset size, we independently trained and tested the model 10 times to avoid any epistemic uncertainty bias. The tests were all performed on the same test set. As expected, we observe that increasing the

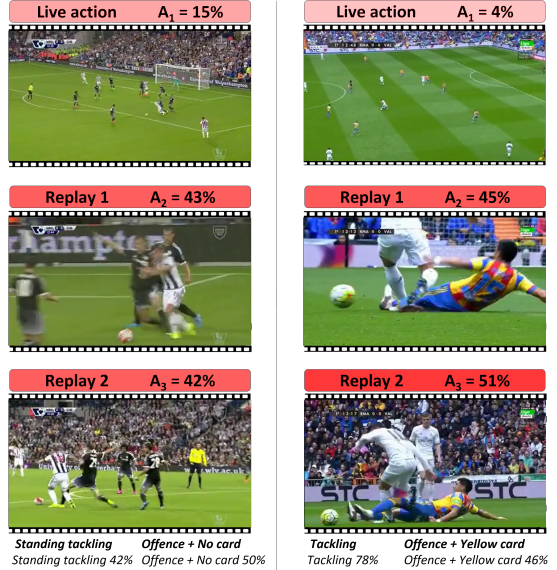


**Fig. 3: Performance evaluation for different dataset sizes.** 100% of the dataset corresponds to 2,319 actions. For each dataset size, we independently trained and tested the model 10 times. The tests were all performed on the same test set. The error bar corresponds to the standard deviation. For 0% of the dataset, we indicate the accuracy by taking a random decision.

dataset size improves the accuracy of our VARS. For the type of foul classification, we notice a significant improvement in accuracy with increasing dataset size, especially at the beginning. However, the accuracy reached a plateau between 40% and 80% of the data. Interestingly, we observed a sudden increase in accuracy when we increased the dataset size from 80% to 100%. This may be attributable to our unbalanced dataset. The dataset contains numerous “Standing tacklings” and “Tacklings”, while many of the other labels are underrepresented. Increasing the dataset size from 40% to 80% may not have improved accuracy if the model still struggles to generalize to certain actions due to a limited number of training samples. However, increasing the dataset size to 100% could have provided the model with the additional data necessary to better generalize actions. Moreover, Figure 3 reveals that our VARS is significantly more prone to epistemic uncertainty for smaller datasets, as indicated by the high standard deviation.

In contrast, the offense severity curve in Figure 3 initially shows a sharp increase, but later demonstrates a slower growth. Yet, with each increase in the dataset size, the accuracy improves, which confirms that more data would further improve the performance. The reason for this lies in the significant variability in the visual appearance of an offense with “No card”, “Yellow card”, or “Red card”. For instance, a yellow card can be the outcome of a tackle, or it can be the result of a player holding an opponent’s shirt. Although both instances may result in a yellow card, their visual representations differ significantly. To accurately determine whether an action is an offense or not and the corresponding severity, the model needs plenty of examples to learn the underlying distribution.

**Qualitative results.** Figure 4 shows the prediction of our VARS on two examples with a 3-view setup. In both examples, the VARS correctly determines the type of foul and correctly classifies both actions as a foul with the correct severity. Furthermore, the attention scores offer valuable insights into the contribution of different views or camera angles to the decision-making process of the model. In both cases, the “live action clips” have the lowest attention score, confirming our intuition that they were filmed from too far away



**Fig. 4: Qualitative results.** VARS prediction on two examples where the attention score of each view is given in percentage. The ground truth is given in bold and the model prediction with the confidence is given in italic.

to make an accurate decision. Both replays have a similar attention score, as they both offer a lot of information to the model. However, we can see that the most informative view has a slightly higher attention score. The attention score provides insight on which views contribute the most to classifications and helps us better understand how the model processes the visual data. This interpretability is especially important when the VARS is used in practice, as it is essential for fans, players, and referees to understand the reasoning behind decisions and feel confident that the technology is improving the fairness and integrity of the sport. Finally, the attention scores assigned to each view can assist broadcasters in automatically selecting the optimal camera angle for broadcasting purposes. Furthermore, it can support the VAR and help speed up the review process by automatically proposing the most informative camera perspective. This is particularly useful at a professional level, where the VAR can have up to 30 different camera perspectives at their disposal, making finding the optimal camera a challenge on its own. The attention scores would provide valuable information by highlighting the views that

are more likely to provide crucial details, to accelerate the decision-making process during the VAR review.

## 4 Human study

In contrast to classical classification tasks that involve well-defined and easily separable classes, determining whether an action in football constitutes a foul may be subjective. Despite the definitions and regulations provided by the *Laws of the Game* [47], the rule book published by the IFAB regarding when an action in football is considered a foul and its corresponding severity, these guidelines are still open to interpretation, leading to differing opinions about the same action. In practice, many actions fall into this gray area where both interpretations, foul or no foul, could be considered correct. In this study, we first analyze whether and how the performance of our VARS aligns with human performance (*i.e.*, referees and football players) by comparing the accuracy of the type of foul and offense severity classifications between VARS and our human participants. Secondly, we conduct an inter-rater agreement analysis of human decisions to quantify the extent of agreement among our human participants.

**Experimental setup.** The study involves two distinct groups of participants with different expertise in football: “Players” and “Referees”. The first group consisted of 15 male individuals aged 18 or older (with a mean  $M = 23.06$  and a standard deviation  $SD = 3.49$  years), who had been playing football for a minimum of three years ( $M = 8.71$  and  $SD = 3.32$  years). The second group consisted of 15 male individuals aged 18 or older ( $M = 25.33$  and  $SD = 4.51$  years), who are certified football referees and have officiated in at least 200 official games (from 223 to 1150 games). Both groups analyzed 77 actions, each presented with three different camera perspectives simultaneously. The participants could review the clips several times and watch the actions in slow motion or frame-by-frame, without any time restriction. To reduce bias, the actions were shown in a different random order to each participant. For each action, we measured the time taken by the participants to make their decision. This time was measured from the moment the participants started the video until they clicked on the ‘Next

|          | Type of Foul |       | Offence Severity |       | Time        |
|----------|--------------|-------|------------------|-------|-------------|
|          | Acc.         | Conf. | Acc.             | Conf. |             |
| Players  | <b>75%</b>   | 3.6   | 58%              | 3.3   | 41.53       |
| Referees | 70%          | 3.7   | <b>60%</b>       | 3.6   | 38.01       |
| VARS     | 60%          | -     | 51%              | -     | <b>0.12</b> |

**Table 2: Accuracy comparison between referees, players, and our VARS.** The survey was performed on a subset of the test set of size 77. The time is given in seconds and represents the average time needed to make a decision. Acc. stands for accuracy and conf. for confidence. A rating of 5 indicates high confidence, while a rating of 1 indicates low confidence.

video’ button. For each action, the participants had the same classification task as presented in Section 3.1. Specifically, they had to determine the type of foul, if the action was a foul or not, and the corresponding severity. For each action, we use the annotations from the SoccerNet-MVFoul dataset as ground truth to determine the accuracy for each participant. An important note is that the participants have a clear advantage over our VARS as they view clips lasting 5 seconds, with a frame rate of 25 fps, while our model gets a 1-second clip at 16 fps as input. Finally, let us note that our study was approved by the local university’s ethics committee (2223-080/5624). All analyses were performed using the JASP software.

### 4.1 Comparison to human performance

Table 2 shows the average accuracy compared to the ground truth of players, referees, and our VARS, respectively. These results align with similar studies [52–54], where the referees had an overall decision accuracy ranging from 45% to 80%.

In terms of the type of foul categorization, players ( $M = 0.752$ ,  $SD = 0.055$ ) were numerically more accurate than referees ( $M = 0.704$ ,  $SD = 0.120$ ), but this difference was not statistically significant, as shown by an independent samples Student t-test,  $t(28) = 1.421$ ,  $p = 0.166$ ,  $d = 0.519$ , 95% CI =  $[-0.214 - 1.243]$ . Mean confidence levels in these categorizations were comparable between players ( $M = 3.64$ ,  $SD = 0.28$ ) and referees ( $M = 3.71$ ,  $SD = 0.32$ ),  $t(28) < 1$ .



As for determining if an action corresponds to a foul and the corresponding severity, referees were slightly more accurate ( $M = 0.594$ ,  $SD = 0.091$ ) than players ( $M = 0.582$ ,  $SD = 0.061$ ). However, this difference was not statistically significant,  $t(28) = -0.401$ ,  $p = 0.691$ ,  $d = -0.147$ , 95% CI =  $[-0.862 - 0.571]$ . Although the accuracy of players and referees was comparable, referees were more confident in their severity judgments ( $M = 3.67$ ,  $SD = 0.36$ ) than players ( $M = 3.33$ ,  $SD = 0.39$ ),  $t(28) = -2.3$ ,  $p = 0.029$ ,  $d = -0.839$ , 95% CI =  $[-1.581 - -0.084]$ . Referees’ higher confidence might be due to their specific experience in assessing fouls and their severity on the field.

Overall, our results suggest that the accuracy of players and referees is comparable. The Bayesian version of the Student t-test provides support for this null hypothesis with Bayes factors BF10 of 0.732 and 0.366 for the type of foul and offense severity task, respectively. There is a possibility that this lack of difference between groups is due to power issues, *i.e.*, the sample size being too small. Replication studies conducted on larger groups would be valuable in revealing potential differences between the two human groups.

As we do not have a standard deviation for the VARS, we conducted two One-Sample t-tests to compare its performance against humans (players and referees were grouped as their accuracy was comparable). For action categorization, humans ( $M = 0.728$ ,  $SD = 0.095$ ) were significantly more accurate than our VARS ( $M = 0.597$ ),  $t(29) = 7.556$ ,  $p < .001$ ,  $d = 1.379$ , 95% CI =  $[0.870 - 1.876]$ . Humans were also more accurate ( $M = 0.588$ ,  $SD = 0.081$ ) than our VARS ( $M = 0.508$ ) for offense severity judgments,  $t(29) = 5.492$ ,  $p < .001$ ,  $d = 1.003$ , 95% CI =  $[0.556 - 1.437]$ . This difference in performance might be due to differences in training between our VARS and humans. Players and referees have accumulated an extensive amount of experience in football, through officiating, playing, and watching the game for countless hours. In contrast, our VARS has only been trained on an unbalanced training set of 2,916 actions, where some types of labels only occur a few times. For example, there are only 27 fouls with a red card in the training set, making it difficult for the model to precisely learn the difference between a foul with a yellow card and one with a red card. Considering the difficulty of the task and the significant experience disadvantage

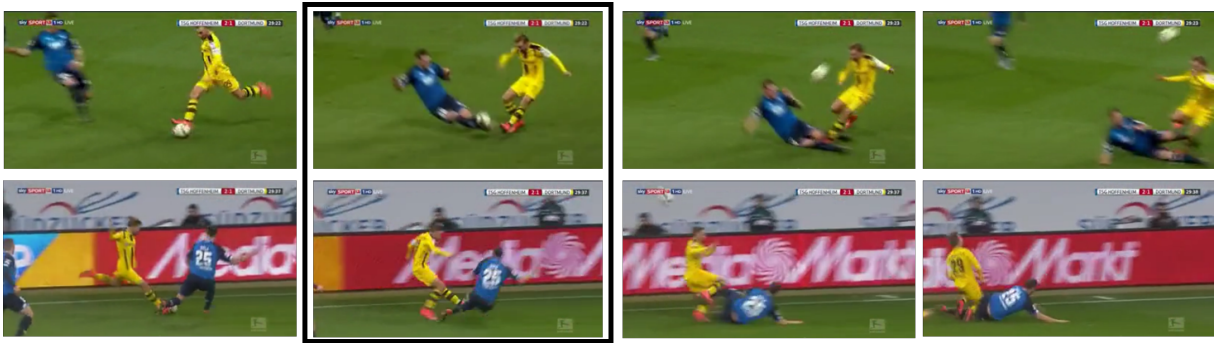
| Nb. of different decisions | 1   | 2   | 3   | 4  |
|----------------------------|-----|-----|-----|----|
| High-level referees        | 16% | 56% | 28% | 0% |
| Referee talents            | 2%  | 60% | 38% | 0% |

**Table 3: Similarity analysis of the results for Offense Severity classification.** Among high-level referees, 28% of cases result in three different decisions being made for the same action. For referee talents, this percentage even increases to 38%. These results show the significant challenge involved in determining whether an action should be classified as a foul and assessing its corresponding severity.

of our VARS compared to humans, the current results are promising. Further, it is notable that our VARS only requires 120ms to reach a decision, which is more than 300 times faster than humans. Both referees and players require a similar amount of time to make a decision. On average, players take around 41.53 seconds and referees 38.01 seconds, which is similar to the average time of 46 seconds taken for the VAR to make a decision as reported by López [55].

## 4.2 Inter-rater agreement

In this subsection, we investigate the reliability and consistency of humans in determining whether an action constitutes a foul and its severity. To assess the level of consensus among humans, we calculated inter-rater agreement in each group for the severity classification task. Since determining if an action is a foul and assessing its severity is the most important task, we only focus on evaluating inter-rater agreement for this aspect. To quantify the inter-rater agreement, we calculated the unweighted average Cohen’s kappa, which measures the agreement between multiple individuals. The referees achieved an unweighted average Cohen’s kappa of 0.213, indicating weak agreement. Similarly, players’ agreement was weak, with a score of 0.223. This suggests limited consistency among both groups in their assessments. Among our 15 referees, 7 are officiating at a high level (in the highest league of their country). These referees are called “high-level referees” in the following. All other referees are called “referee talents”. Table 3 shows the consensus in each subgroup for the offense severity classification task. As can be seen, high-level and referee



**Fig. 5: Example of the subjectivity of human choices.** Decisions taken by our participants: “No offense”, “Offense + No card”, and “Offense + Yellow card”.

talents reached a consensus between themselves for only 16% and 2% of the actions, respectively. In the majority of cases, multiple decisions were made for the same action, indicating the difficulty in determining whether an action should be classified as a foul and assessing its severity. Particularly among referee talents, 38% of actions resulted in three different decisions (out of four possible decisions to take) for the same action. Figure 5 shows an example of an action where all three decisions “No offense”, “Offense + No card” and “Offense + Yellow card” were taken among the referees. For certain referees, the fact that the defender plays the ball is considered enough to not award a free-kick in this situation. However, other referees believe that even if the defender plays the ball, he disregards the danger to, or consequences for, an opponent and awards a yellow card. These findings underscore the complexity and subjectivity inherent in refereeing decisions, highlighting the potential for further research to improve consistency and fairness in officiating.

## 5 Conclusion

Distinguishing between a foul and no foul and determining its severity is a complex and subjective task that relies entirely on the interpretation of the *Laws of the Game* [47] by each individual. Despite the challenges posed by this complex task and an unbalanced training dataset, our solution demonstrates promising results. While we have not reached human-level performance yet, we believe that VARS holds the potential to assist and support referees across all levels of professionalism in the future.

**Acknowledgement** This work was partly supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding and the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI). J. Held and A. Cioppa are funded by the F.R.S.-FNRS. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.

## 6 Declarations

**Availability of data and code.** The data and code are available at these addresses <https://github.com/SoccerNet/sn-mvfoul>

**Conflict of interest.** The authors declare no conflict of interest.

**Open access.**

## References

- [1] Cioppa, A., Deliège, A., Ul Huda, N., Gade, R., Van Droogenbroeck, M., Moeslund, T.B.: Multimodal and multiview distillation for real-time player detection on a football field. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, Seattle, WA, USA, pp. 3846–3855 (2020). <https://doi.org/10.1109/CVPRW50498.2020.00448>. <https://doi.org/10.1109/CVPRW50498.2020.00448>

- [2] Maglo, A., Orcesi, A., Pham, Q.-C.: Efficient tracking of team sport players with few game-specific annotations. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pp. 3460–3470. Inst. Electr. Electron. Eng. (IEEE), New Orleans, LA, USA (2022). <https://doi.org/10.1109/cvprw56347.2022.00390>. <https://doi.org/10.1109/CVPRW56347.2022.00390>
- [3] Vandeghen, R., Cioppa, A., Van Droogenbroeck, M.: Semi-supervised training to improve player and ball detection in soccer. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, pp. 3480–3489. Inst. Electr. Electron. Eng. (IEEE), New Orleans, LA, USA (2022). <https://doi.org/10.1109/cvprw56347.2022.00392>. <https://doi.org/10.1109/CVPRW56347.2022.00392>
- [4] Somers, V., Joos, V., Giancola, S., Cioppa, A., Ghasemzadeh, S.A., Magera, F., Standaert, B., Mansourian, A.M., Zhou, X., Kasaei, S., Ghanem, B., Alahi, A., Van Droogenbroeck, M., De Vleeschouwer, C.: SoccerNet game state reconstruction: End-to-end athlete tracking and identification on a minimap. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, Seattle, WA, USA (2024)
- [5] Cioppa, A., Delière, A., Giancola, S., Ghanem, B., Van Droogenbroeck, M., Gade, R., Moeslund, T.B.: A context-aware loss function for action spotting in soccer videos. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 13123–13133. Inst. Electr. Electron. Eng. (IEEE), Seattle, WA, USA (2020). <https://doi.org/10.1109/cvpr42600.2020.01314>. <https://doi.org/10.1109/CVPR42600.2020.01314>
- [6] Giancola, S., Ghanem, B.: Temporally-aware feature pooling for action spotting in soccer broadcasts. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), Nashville, TN, USA, pp. 4490–4499 (2021). <https://doi.org/10.1109/CVPRW53098.2021.00506>
- [7] Hong, J., Zhang, H., Gharbi, M., Fisher, M., Fatahalian, K.: Spotting temporally precise, fine-grained events in video. In: Eur. Conf. Comput. Vis. (ECCV). Lect. Notes Comput. Sci., vol. 13695, pp. 33–51. Springer, Tel Aviv, Israël (2022). [https://doi.org/10.1007/978-3-031-19833-5\\_3](https://doi.org/10.1007/978-3-031-19833-5_3)
- [8] Soares, J.V.B., Shah, A., Biswas, T.: Temporally precise action spotting in soccer videos using dense detection anchors. In: IEEE Int. Conf. Image Process. (ICIP), pp. 2796–2800. Inst. Electr. Electron. Eng. (IEEE), Bordeaux, France (2022). <https://doi.org/10.1109/icip46576.2022.9897256>. <https://doi.org/10.1109/ICIP46576.2022.9897256>
- [9] Soares, J.V.B., Shah, A.: Action spotting using dense detection anchors revisited: Submission to the SoccerNet challenge 2022. arXiv **abs/2206.07846** (2022) <https://arxiv.org/abs/2206.07846>. <https://doi.org/10.48550/arXiv.2206.07846>
- [10] Giancola, S., Cioppa, A., Georgieva, J., Billingham, J., Serner, A., Peek, K., Ghanem, B., Van Droogenbroeck, M.: Towards active learning for action spotting in association football videos. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pp. 5098–5108. Inst. Electr. Electron. Eng. (IEEE), Vancouver, Can. (2023). <https://doi.org/10.1109/cvprw59228.2023.00538>. <https://doi.org/10.1109/CVPRW59228.2023.00538>
- [11] Cabado, B., Cioppa, A., Giancola, S., Villa, A., Guijarro-Berdiñas, B., Padrón, E., Ghanem, B., Van Droogenbroeck, M.: Beyond the Premier: Assessing action spotting transfer capability across diverse domains. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, Seattle, WA, USA (2024)
- [12] Kassab, E.J., Solberg, H.M., Gautam, S., Sabet, S.S., Torjusén, T., Riegler, M., Halvorsen, P., Midoglu, C.: Tacdec. Proceedings of the ACM Multimedia Systems Conference 2024 on ZZZ (2024). <https://doi.org/10.1145/3625468.3652166>

- [13] Arbués Sangüesa, A., Martín, A., Fernández, J., Ballester, C., Haro, G.: Using player's body-orientation to model pass feasibility in soccer. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pp. 3875–3884. Inst. Electr. Electron. Eng. (IEEE), Seattle, WA, USA (2020). <https://doi.org/10.1109/cvprw50498.2020.00451>. <https://doi.org/10.1109/CVPRW50498.2020.00451>
- [14] Honda, Y., Kawakami, R., Yoshihashi, R., Kato, K., Naemura, T.: Pass receiver prediction in soccer using video and players' trajectories. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pp. 3502–3511. Inst. Electr. Electron. Eng. (IEEE), New Orleans, LA, USA (2022). <https://doi.org/10.1109/cvprw56347.2022.00394>. <https://doi.org/10.1109/CVPRW56347.2022.00394>
- [15] Gautam, S., Midoglu, C., Shafiee Sabet, S., Kshatri, D.B., Halvorsen, P.: Soccer game summarization using audio commentary, metadata, and captions. Proceedings of the 1st Workshop on User-centric Narrative Summarization of Long Videos (2022). <https://doi.org/10.1145/3552463.3557019>
- [16] Midoglu, C., Sabet, S.S., Sarkhoosh, M.H., Majidi, M., Gautam, S., Solberg, H.M., Kupka, T., Halvorsen, P.: Ai-based sports highlight generation for social media. Proceedings of the 3rd Mile-High Video Conference on zzz (2024). <https://doi.org/10.1145/3638036.3640799>
- [17] Gautam, S., Midoglu, C., Sabet, S.S., Kshatri, D.B., Halvorsen, P.: Assisting soccer game summarization via audio intensity analysis of game highlights. Unpublished (2022). <https://doi.org/10.13140/RG.2.2.34457.70240/1>
- [18] Midoglu, C., Hicks, S., Thambawita, V., Kupka, T., Halvorsen, P.: MMSys'22 grand challenge on AI-based video production for soccer. In: ACM Multimedia Systems Conference (MMSys), Athlone, Ireland, pp. 1–6 (2022). <https://doi.org/>
- [19] Magera, F., Hoyoux, T., Barnich, O., Van Droogenbroeck, M.: A universal protocol to benchmark camera calibration for sports. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, Seattle, WA, USA (2024)
- [20] Somers, V., De Vleeschouwer, C., Alahi, A.: Body part-based representation learning for occluded person Re-Identification. In: IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV), pp. 1613–1623. Inst. Electr. Electron. Eng. (IEEE), Waikoloa, HI, USA (2023). <https://doi.org/10.1109/wacv56688.2023.00166>. <https://doi.org/10.1109/WACV56688.2023.00166>
- [21] Mkhallati, H., Cioppa, A., Giancola, S., Ghanem, B., Van Droogenbroeck, M.: SoccerNet-caption: Dense video captioning for soccer broadcasts commentaries. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), pp. 5074–5085. Inst. Electr. Electron. Eng. (IEEE), Vancouver, Can. (2023). <https://doi.org/10.1109/cvprw59228.2023.00536>. <https://doi.org/10.1109/CVPRW59228.2023.00536>
- [22] Andrews, P., Nordberg, O.E., Zubicueta Portales, S., Borch, N., Guribye, F., Fujita, K., Fjeld, M.: AiCommentator: A multimodal conversational agent for embedded visualization in football viewing. In: Int. Conf. Intell. User Interfaces, pp. 14–34. ACM, Greenville, SC, USA (2024). <https://doi.org/10.1145/3640543.3645197>. <https://doi.org/10.1145/3640543.3645197>
- [23] Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: IEEE Int. Conf. Comput. Vis. (ICCV), pp. 945–953. Inst. Electr. Electron. Eng. (IEEE), Santiago, Chile (2015). <https://doi.org/10.1109/iccv.2015.114>
- [24] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by

- jointly learning to align and translate. arXiv **abs/1409.0473** (2014). <https://doi.org/10.48550/arXiv.1409.0473>
- [25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv **abs/1706.03762** (2017). <https://doi.org/10.48550/arXiv.1706.03762>
- [26] Pappalardo, L., Cintia, P., Rossi, A., Masuccio, E., Ferragina, P., Pedreschi, D., Giannotti, F.: A public data set of spatio-temporal match events in soccer competitions. *Sci. Data* **6**(1), 1–15 (2019). <https://doi.org/10.1038/s41597-019-0247-7>
- [27] Yu, J., Lei, A., Song, Z., Wang, T., Cai, H., Feng, N.: Comprehensive dataset of broadcast soccer videos. In: *IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, pp. 418–423. Inst. Electr. Electron. Eng. (IEEE), Miami, FL, USA (2018). <https://doi.org/10.1109/MIPR.2018.00090>
- [28] Scott, A., Uchida, I., Onishi, M., Kameda, Y., Fukui, K., Fujii, K.: SoccerTrack: A dataset and tracking algorithm for soccer with fish-eye and drone videos. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pp. 3568–3578. Inst. Electr. Electron. Eng. (IEEE), New Orleans, LA, USA (2022). <https://doi.org/10.1109/cvprw56347.2022.00401>. <https://doi.org/10.1109/CVPRW56347.2022.00401>
- [29] Jiang, Y., Cui, K., Chen, L., Wang, C., Xu, C.: SoccerDB: A large-scale database for comprehensive video understanding. In: *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pp. 1–8. ACM, Seattle, WA, USA (2020). <https://doi.org/10.1145/3422844.3423051>
- [30] Van Zandycke, G., Somers, V., Istasse, M., Don, C.D., Zambrano, D.: DeepSportradar-v1: Computer vision dataset for sports understanding with high quality annotations. In: *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pp. 1–8. ACM, Lisbon, Port. (2022). <https://doi.org/10.1145/3552437.3555699>
- [31] Giancola, S., Amine, M., Dghaily, T., Ghanem, B.: SoccerNet: A scalable dataset for action spotting in soccer videos. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pp. 1792–179210. Inst. Electr. Electron. Eng. (IEEE), Salt Lake City, UT, USA (2018). <https://doi.org/10.1109/cvprw.2018.00223>
- [32] Delière, A., Cioppa, A., Giancola, S., Seikavandi, M.J., Dueholm, J.V., Nasrollahi, K., Ghanem, B., Moeslund, T.B., Van Droogenbroeck, M.: SoccerNet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. In: *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, CVsports, Nashville, TN, USA, pp. 4508–4519 (2021). <https://doi.org/10.1109/CVPRW53098.2021.00508>. <http://hdl.handle.net/2268/253781>
- [33] Cioppa, A., Delière, A., Giancola, S., Ghanem, B., Van Droogenbroeck, M.: Scaling up SoccerNet with multi-view spatial localization and re-identification. *Sci. Data* **9**(1), 1–9 (2022). <https://doi.org/10.1038/s41597-022-01469-1>
- [34] Cioppa, A., Giancola, S., Deliege, A., Kang, L., Zhou, X., Cheng, Z., Ghanem, B., Van Droogenbroeck, M.: SoccerNet-tracking: Multiple object tracking dataset and benchmark in soccer videos. In: *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, CVsports, pp. 3490–3501. Inst. Electr. Electron. Eng. (IEEE), New Orleans, LA, USA (2022). <https://doi.org/10.1109/cvprw56347.2022.00393>. <https://doi.org/10.1109/CVPRW56347.2022.00393>
- [35] Held, J., Cioppa, A., Giancola, S., Hamdi, A., Ghanem, B., Van Droogenbroeck, M.: VARS: Video assistant referee system for automated soccer decision making from multiple views. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pp. 5086–5097. Inst. Electr. Electron. Eng. (IEEE), Vancouver, Can. (2023). <https://doi.org/10.1109/cvprw59228.2023.00537>. <https://doi.org/10.1109/cvprw59228.2023.00537>



1109/CVPRW59228.2023.00537

- [36] Cioppa, A., Giancola, S., Somers, V., Magera, F., Zhou, X., Mkhallati, H., Delière, A., Held, J., Hinojosa, C., Mansourian, A.M., Miralles, P., Barnich, O., De Vleeschouwer, C., Alahi, A., Ghanem, B., Van Droogenbroeck, M., Kamal, A., Maglo, A., Clapés, A., Abdelaziz, A., Xarles, A., Orcesi, A., Scott, A., Liu, B., Lim, B., Chen, C., Deuser, F., Yan, F., Yu, F., Shitrit, G., Wang, G., Choi, G., Kim, H., Guo, H., Fahrudin, H., Koguchi, H., Ardö, H., Salah, I., Yerushalmy, I., Muhammad, I., Uchida, I., Be'ery, I., Rabarisoa, J., Lee, J., Fu, J., Yin, J., Xu, J., Nang, J., Denize, J., Li, J., Zhang, J., Kim, J., Synowiec, K., Kobayashi, K., Zhang, K., Habel, K., Nakajima, K., Jiao, L., Ma, L., Wang, L., Wang, L., Li, M., Zhou, M., Nasr, M., Abdelwahed, M., Liashuha, M., Falaleev, N., Oswald, N., Jia, Q., Pham, Q.-C., Song, R., Hérault, R., Peng, R., Chen, R., Liu, R., Baikulov, R., Fukushima, R., Escalera, S., Lee, S., Chen, S., Ding, S., Someya, T., Moeslund, T.B., Li, T., Shen, W., Zhang, W., Li, W., Dai, W., Luo, W., Zhao, W., Zhang, W., Yang, X., Ma, Y., Joo, Y., Zeng, Y., Gan, Y., Zhu, Y., Zhong, Y., Ruan, Z., Li, Z., Huang, Z., Meng, Z.: SoccerNet 2023 challenges results. arXiv **abs/2309.06006** (2023) <https://arxiv.org/abs/2309.06006> <https://doi.org/10.48550/arXiv.2309.06006>
- [37] Leduc, A., Cioppa, A., Giancola, S., Ghanem, B., Van Droogenbroeck, M.: SoccerNet-Depth: a scalable dataset for monocular depth estimation in sports videos. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, Seattle, WA, USA (2024)
- [38] Held, J., Itani, H., Cioppa, A., Giancola, S., Ghanem, B., Van Droogenbroeck, M.: X-vars: Introducing explainability in football refereeing with multi-modal large language models. In: IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports, Seattle, WA, USA (2024)
- [39] Gautam, S., Sarkhoosh, M.H., Held, J., Midoglu, C., Cioppa, A., Giancola, S., Thambawita, V., Riegler, M.A., Halvorsen, P., Shah, M.: SoccerNet-echoes: A soccer game audio commentary dataset. arXiv **abs/2405.07354** (2024) <https://arxiv.org/abs/2405.07354> <https://doi.org/10.48550/arXiv.2405.07354>
- [40] Spitz, J., Wagemans, J., Memmert, D., Williams, A.M., Helsen, W.F.: Video assistant referees (var): The impact of technology on decision making in association football referees. *Journal of Sports Sciences* **39**(2), 147–153 (2020). <https://doi.org/10.1080/02640414.2020.1809163>
- [41] De Dios Crespo, J.: 2. The Contribution of VARs to Fairness in Sport, pp. 23–35. Routledge, New York City, NY, USA (2021). <https://doi.org/10.4324/9780429455551-2>
- [42] Holder, U., Ehrmann, T., König, A.: Monitoring experts: insights from the introduction of video assistant referee (VAR) in elite football. *Journal of Business Economics* **92**(2), 285–308 (2021). <https://doi.org/10.1007/s11573-021-01058-5>
- [43] Dufner, A.-L., Schütz, L.-M., Hill, Y.: The introduction of the video assistant referee supports the fairness of the game — an analysis of the home advantage in the german bundesliga. *Psychology of Sport and Exercise* **66**, 1–5 (2023). <https://doi.org/10.1016/j.psychsport.2023.102386>
- [44] Armenteros, M., Webb, T.: Educating International Football Referees: The importance of Uniformity, pp. 301–327. Routledge, New York City, NY, USA (2021). Chap. 16. <https://doi.org/10.4324/9780429455551-16>
- [45] Deutscher Fußball-Bund (DFB): Anzahl aktiver Schiedsrichter/-innen bis 2022. <https://www.dfb.de/verbandsstruktur/dfb-zentrale/> (2022)
- [46] Zeppenfeld, B.: Anzahl aktiver Schiedsrichter / Schiedsrichterinnen des Deutschen Fußball Bundes (DFB) von 2018/2019 bis 2022/2023. <https://de.statista.com/statistik/daten/studie/1243626/umfrage/dfb-anzahl-aktiver-schiedsrichter/> (2023)

- [47] IFAB: Laws of the game. Technical report, The International Football Association Board, Zurich, Switzerland (2022)
- [48] Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. In: IEEE Int. Conf. Comput. Vis. (ICCV), pp. 6804–6815. Inst. Electr. Electron. Eng. (IEEE), Montréal, Can. (2021). <https://doi.org/10.1109/iccv48922.2021.00675>
- [49] Li, Y., Wu, C.-Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., Feichtenhofer, C.: MViTv2: Improved multiscale vision transformers for classification and detection. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 4794–4804. Inst. Electr. Electron. Eng. (IEEE), New Orleans, LA, USA (2022). <https://doi.org/10.1109/cvpr52688.2022.00476>
- [50] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. *arXiv abs/1705.06950* (2017). <https://doi.org/10.48550/arXiv.1705.06950>
- [51] Yang, Z., Wang, L.: Learning relationships for multi-view 3D object recognition. In: IEEE Int. Conf. Comput. Vis. (ICCV), pp. 7504–7513. Inst. Electr. Electron. Eng. (IEEE), Seoul, South Korea (2019). <https://doi.org/10.1109/iccv.2019.00760>
- [52] MacMahon, C., Helsen, W.F., Starkes, J.L., Weston, M.: Decision-making skills and deliberate practice in elite association football referees. *Journal of Sports Sciences* **25**(1), 65–78 (2007). <https://doi.org/10.1080/02640410600718640>
- [53] Spitz, J., Put, K., Wagemans, J., Williams, A.M., Helsen, W.F.: Visual search behaviors of association football referees during assessment of foul play situations. *Cognitive Research: Principles and Implications* **1**(1) (2016). <https://doi.org/10.1186/s41235-016-0013-8>
- [54] Pizzera, A., Marrable, J., Raab, M.: The video review system in association football: implementation and effectiveness for match officials and referee education. *Managing Sport and Leisure*, 1–17 (2022). <https://doi.org/10.1080/23750472.2022.2147856>
- [55] López, A.M.: Average time needed for a video assistant referee (VAR) intervention in Brazil in 2019 and 2020. <https://www.statista.com/statistics/1010093/average-time-video-assistant-referee-checking-brazil/> (2023)

# X-VARS: Introducing Explainability in Football Refereeing with Multi-Modal Large Language Models

Jan Held<sup>1</sup> Hani Itani<sup>2</sup> Anthony Cioppa<sup>1</sup> Silvio Giancola<sup>2</sup>  
Bernard Ghanem<sup>2</sup> Marc Van Droogenbroeck<sup>1</sup>  
<sup>1</sup> University of Liège <sup>2</sup> KAUST

## Abstract

The rapid advancement of artificial intelligence has led to significant improvements in automated decision-making. However, the increased performance of models often comes at the cost of explainability and transparency of their decision-making processes. In this paper, we investigate the capabilities of large language models to explain decisions, using football refereeing as a testing ground, given its decision complexity and subjectivity. We introduce the *EX*plainable *VI*deo *AS*ssistant *RE*feree *S*ystem, X-VARS, a multi-modal large language model designed for understanding football videos from the point of view of a referee. X-VARS can perform a multitude of tasks, including video description, question answering, action recognition, and conducting meaningful conversations based on video content and in accordance with the Laws of the Game for football referees. We validate X-VARS on our novel dataset, SoccerNet-XFoul, which consists of more than 22k video-question-answer triplets annotated by over 70 experienced football referees. Our experiments and human study illustrate the impressive capabilities of X-VARS in interpreting complex football clips. Furthermore, we highlight the potential of X-VARS to reach human performance and support football referees in the future. We will provide code, model, dataset, and demo upon publication.

## 1. Introduction

Recently, the field of Artificial Intelligence (AI) has witnessed remarkable progress in the development of large language models [3, 6, 16, 25, 64]. These models have acquired a strong language understanding, enabling them to tackle a broad range of linguistic tasks, ranging from text generation and conversation to zero-shot question answering. The development has further progressed with multi-modal language models, going beyond the constraints of text-based inputs, but also including images, videos, and audio [1, 41, 69, 71]. However, as the capabilities of these

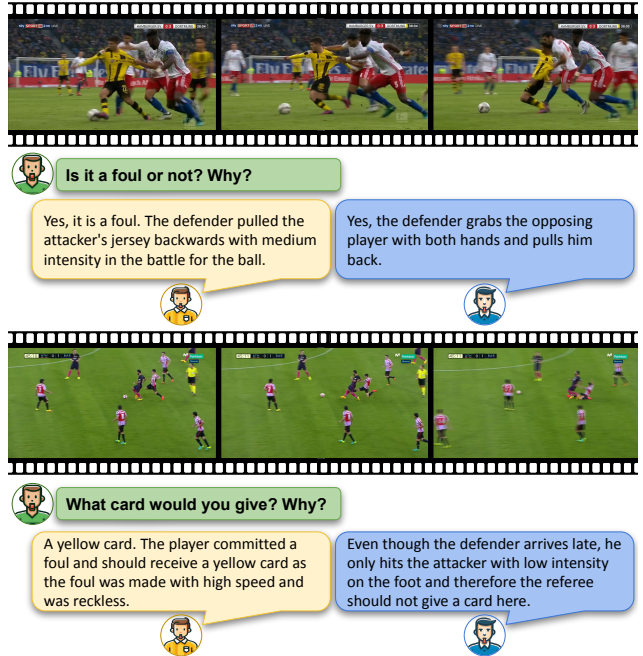


Figure 1. **SoccerNet-XFoul dataset.** Examples of annotations from two different referees for the same foul. The second example illustrates the complexity and subjectivity of refereeing decisions.

models continue to advance, the increase in model performance often comes at the cost of reduced explainability and transparency. This trend poses several challenges for users and developers who seek to understand the *why* and *how* behind a model's decision-making process. Explaining the reasoning process of AI models is particularly crucial in domains requiring high levels of trust, such as healthcare, autonomous driving, or sports. In the context of football, referee decisions can significantly impact the financial future and existence of clubs, making it essential for AI models to transparently explain their decision-making process. Such transparency is key to building trust and facilitating the acceptance and integration of AI in sports. For instance, during the Qatar World Cup 2022, FIFA employed an AI sys-

tem for semi-automated offside detection [19]. To enhance the system’s transparency and explainability, it generates a 3D representation of the game to allow referees and spectators to visually verify offside positions with undeniable clarity, bridging the gap between AI decision-making and human understanding.

In this paper, we explore the use of large language models to enhance transparency and explainability of automated referee decision-making. Football refereeing offers an ideal environment, as many of the decisions that referees make are subjective and reliant on each individual’s interpretation of the rules of the game. Particularly, we introduce X-VARS, the first multi-modal large language model designed to explain football refereeing decisions. X-VARS is based on a Vision Language Model (VLM) [41] that adapts the design of the LLaVa [37] model for spatio-temporal video modeling. We train X-VARS using a new training paradigm where we input the visual features and the multi-task predictions of our fine-tuned visual encoder CLIP ViT-L/14 [50] to the language model. We validate X-VARS on our novel *SoccerNet-XFoul* dataset containing more than 22k video-question-answer triplets about the most fundamental refereeing questions. More than 70 professional referees annotated our dataset and provided, for each question, detailed explanations about their decisions. X-VARS achieves state-of-the-art performance on the *SoccerNet-MVFoul* dataset, and our human study demonstrates that X-VARS generates explanations for its decisions at a level comparable to human referees. Finally, X-VARS can analyze and understand complex football duels and provide accurate decision explanations, opening doors for future applications to support referees in their decision-making processes.

**Contributions.** We summarize our contributions as follows: (i) We publicly release *SoccerNet-XFoul*, a new multi-modal dataset containing more than 22k video-question-answer triplets about refereeing questions. (ii) We introduce *X-VARS*, a new vision language model that can perform multiple multi-modal tasks such as visual captioning, question-answering, video action recognition, and can generate explanations of its decisions on-par with human level. (iii) We perform a thorough evaluation of our model, including analyses of our new training paradigm, the influence of the CLIP text predictions, and a human study that compares X-VARS to human referees.

## 2. Related work

**Sports video understanding.** The field of sports video understanding has gained a lot of interest lately [63]. It encompasses a wide range of tasks such as player segmentation, detection, and tracking [10, 42, 43, 56, 66], keypoint detection [38], summarizing [20, 45], player re-identification [43, 60], action spotting in untrimmed videos [8, 9, 14, 23, 27,

57–59], pass prediction and feasibility [2, 26], foul recognition [18, 24] or dense video captioning for football broadcasts commentaries [47]. Such tasks can be formulated and solved by leveraging recent advances in deep learning for general video understanding. However, progress in sports video understanding heavily relies on the availability of sports-centric large-scale annotated datasets [31, 32, 44, 49, 54, 65, 70]. SoccerNet [7, 11–13, 21, 22] stands among the largest and most comprehensive dataset for video understanding in soccer. *SoccerNet-MVFoul* [24] further extended *SoccerNet* by proposing a novel multi-view football dataset designed for foul classification annotated by professional referees. In this work, we further extend *SoccerNet-MVFoul* into a visual-question-answering dataset focused on football refereeing questions, named *SoccerNet-XFoul*.

**Vision language models.** Natural Language Processing (NLP) has witnessed remarkable advancements with the emergence of open-source Large Language Models (LLMs) [3, 6, 16, 25, 48, 52, 64]. These models have demonstrated exceptional capabilities in language understanding and generation tasks. LLMs have also served as the basis for the success of many vision-language models that are based on projecting the visual features of an image [5, 29, 37, 68, 71] or a video [1, 69] encoder onto the input embedding space of an LLM. This idea and its variants allowed leveraging the power of LLMs for multi-modal understanding. In this work, we propose X-VARS, a vision language model for visual captioning, question-answering, action video recognition, and conducting meaningful conversations based on video content.

**Explainability.** Recently, explainability in machine learning has received lots of attention, leading to the development of various techniques to demystify complex models. LIME (Local Interpretable Model-agnostic Explanations) [51] explains the predictions of any machine learning classifier by approximating any classifier locally with an interpretable model. Meanwhile, SHAP (SHapley Additive exPlanations) [39] offers a unified perspective on feature importance by averaging all feature combinations, ensuring consistent attributions. Grad-CAM (Gradient-weighted Class Activation Mapping) [55] employs gradient information from the final convolutional layer to generate a heat map highlighting crucial input image regions. Counterfactual Explanations [67] identify the least number of changes required in the input data to alter the model’s prediction, offering insights into decision boundaries and feature importance. Lastly, Explanation via Language [33] emphasizes natural language dialogues for enhanced interaction between experts and models, underscoring the importance of interactive systems tailored to user requirements. In this paper, we investigate how large language models can explain decisions, using football refereeing as a testing environment given its decision complexity and subjectivity.

| Dataset                        | Type     | #Instances | #Questions | #Context       | VQA | Captioning | AR |
|--------------------------------|----------|------------|------------|----------------|-----|------------|----|
| Conceptual 12M [4]             | Images   | 12M        | 12M        | Various        | -   | ✓          | -  |
| LAION-5B [53]                  | Images   | 3B         | 3B         | Various        | -   | ✓          | -  |
| LLava dataset [37]             | Images   | 158k       | 158k       | Various        | ✓   | ✓          | -  |
| MovieQA [62]                   | Videos   | 408        | 15k        | Movies         | ✓   | -          | -  |
| TVQA [34]                      | Videos   | 21k        | 150k       | TV shows       | ✓   | -          | -  |
| Video Instruction Dataset [41] | Videos   | 100k       | 100k       | Various        | ✓   | ✓          | -  |
| HowTo100M [46]                 | Videos   | 136M       | 136M       | Youtube        | -   | ✓          | -  |
| GOAL [61]                      | Videos   | 1k         | 53k        | Football       | -   | ✓          | -  |
| Sports-QA [35]                 | Videos   | 6k         | 94k        | Football       | ✓   | -          | ✓  |
| SoccerNet-caption [47]         | Videos   | 942        | -          | Football games | -   | ✓          | -  |
| <b>SoccerNet-XFoul (Ours)</b>  | Football | 10k        | 22k        | Football fouls | ✓   | ✓          | ✓  |

Table 1. **Comparative overview of relevant datasets.** *SoccerNet-XFoul* contains high-quality answers annotated by more than 70 experienced referees. Our dataset is the largest dataset in sports with complex questions and the only one focusing on refereeing questions. VQA stands for Visual Question Answering, AR stands for Action Recognition.

### 3. SoccerNet-XFoul dataset

The performance of supervised models mostly relies on the quality and quantity of available annotated datasets. Multi-modal datasets are generally harder to curate and annotate, which explains their usually smaller size compared to uncurated datasets. Table 1 shows a comparative overview of multi-modal datasets in the literature, specifically highlighting those that contain combinations of text and image, as well as text and video pairs. We introduce *SoccerNet-XFoul*, a dataset specifically designed for foul video recognition and explanation. It consists of high-quality video-text pairs with more than 10k video clips and 22k questions, annotated by more than 70 experienced referees. Compared to the other sports datasets, *SoccerNet-XFoul* has the most video clips and much more complex questions. In the following, we present our approach to creating this high-quality human-annotated dataset.

**Questions.** We identify 4 key questions on the most foundational, complex, and game-impacting decisions a referee must confront during a game. To answer the two first questions, “*Is it a foul or not? Why?*” and “*What card would you give? Why?*”, the model requires an in-depth understanding of the rules of the game [30] as well as an understanding of the context in which an action occurred. Factors such as the intent, the foul location, the game dynamic and the intensity of the contact must all be considered. The two last questions, if “*the defender stops a promising attack or a goal-scoring opportunity?*” and if “*the referee could have given an advantage?*” add a new layer of difficulty and prediction analysis. The answers to the questions are not only visual, since the model has to make predictions about potential future outcomes. For instance, in assessing whether the referee should have given an advantage, the model needs

to evaluate whether the attacking team would benefit more from continuing play rather than being granted a free-kick.

**Annotators.** As no public dataset is available that provides detailed answers and explanations to these refereeing questions, we conducted an annotation campaign with over 70 referees over a three-month period. To ensure high-quality answers, only experienced referees were selected for the annotations. The referees have officiated between 140 and 2,279 official games, with an average of 655 games. They were allowed to assess as many video clips as they wished, with the flexibility to pause at any time to avoid fatigue. Each annotator had the option to provide answers in German, French, English, or Spanish to prevent any linguistic barriers. The answers were translated from the original language to English by ChatGPT-3.5 [3] and then reviewed by another human referee to ensure accurate translation.

**Subjectivity.** Figure 1 displays an example of the subjectivity of the annotations. While one referee annotator would not give a card because he thought the foul was made with low intensity, the other annotator would give a yellow card because he believed the tackling was made with high speed and was reckless. Due to this inherent subjectivity in refereeing, our objective was to gather multiple answers for the same action, rather than collecting a single decision and explanation for each question. Therefore, the multiple decisions and explanations actually help the model to learn a range of valid interpretations and reasoning strategies employed by human referees. All in all, this can improve the robustness of the AI model, enabling it to make informed decisions even in ambiguous or subjective situations. Practically, the annotators were randomly assigned different video clips, ensuring that the same action might be evaluated multiple times. In the end, for each action, we have, on average, 1.5 answers for the same question.



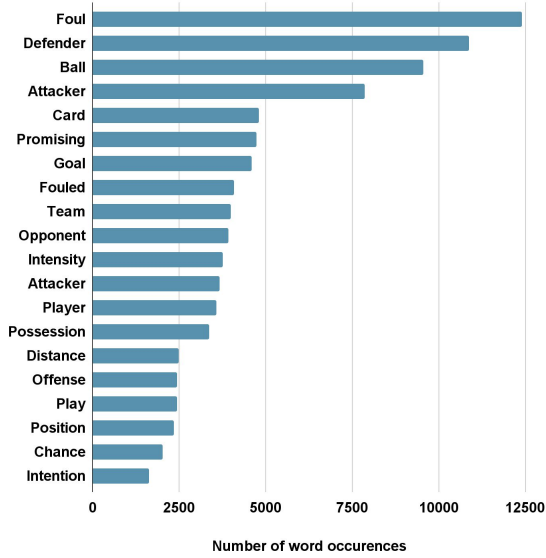


Figure 2. **Distribution of the most common words.** The most frequent words are “foul” and “defender,” followed by semantically related words related to football and referee actions and terms. There is thus a significant imbalance in the distribution.

**Statistics.** Our dataset contains 10k video clips with over 22k referee-generated questions and answers. Figure 2, shows the distribution of the most common words in the explanations of the referee annotators. The most frequently used words are specific terminologies for describing a duel between two players, ranging from descriptive terms such as *defender* or *card* to key terms to consider for evaluating fouls such as *intention* or *intensity*. The number of words per answer ranges from 1 to 66, with a total of more than 540k words and an average of almost 25 words per answer, with a significant imbalance in the distribution of words.

**Novelty.** Compared to traditional visual-question answering datasets, the *SoccerNet-XFoul* dataset is the first to answer refereeing questions, with detailed explanations of why a particular decision is correct. This explainability is a unique approach that enhances the dataset’s complexity and ensures a deeper understanding and representation of real-world scenarios where AI models must make and explain their decisions. Furthermore, the interpretation of situations in our dataset is context-dependent. The interpretation of a foul might differ depending on whether it occurs in the middle of the field or in the penalty area. To correctly answer questions, the model must have an in-depth understanding of the game. Finally, the model needs a level of predictive analysis. For instance, to determine if a defender stopped a promising attack, the model must understand what happened at the moment of the foul and what could have happened in the immediate future. This involves making complex predictions about potential future outcomes, a task that

is far more advanced compared to traditional VQA datasets. Hence, our *SoccerNet-XFoul* dataset is the first and largest visual question-answering dataset for referees in football, offering many new challenges to be explored.

## 4. Methodology

In this section, we provide a comprehensive description of our novel **EX**plainable **VI**deo **A**ssistant **R**eferee **S**ystem, “X-VARS”, for foul and severity recognition, providing explanations on its decision-making process. We begin by presenting the architecture with a detailed description of each individual component of X-VARS. Then, we provide an in-depth explanation of its training process.

### 4.1. Architecture

Figure 3 illustrates the key architectural components of X-VARS. We use Video-ChatGPT [41], a multi-modal model capable of understanding and generating detailed conversations about videos, as our foundation model. We make several changes to the architecture to adapt it to our needs. Formally, we input a video clip  $\mathbf{v} \in \mathbb{R}^{T \times H \times W \times C}$ , with  $T$ ,  $H$ ,  $W$  and  $C$  being respectively the number of frames, height, width, and channel dimension of the video, to CLIP ViT-L/14 [50],

$$\mathbf{f}_i, \mathbf{h}_i = \text{CLIP}(\mathbf{v}) , \quad (1)$$

and obtain the corresponding frame feature vector  $\mathbf{f}_i \in \mathbb{R}^{T \times D_1}$  and the hidden states  $\mathbf{h}_i \in \mathbb{R}^{T \times S \times D_2}$ , with  $S$  being the number of tokens obtained by multiplying  $w = W/p$  and  $h = H/p$ , where  $p$  is the patch size of CLIP,  $D_1$  the dimension of the output layer and  $D_2$  the dimension of the hidden states. We then average-pool the hidden states across the temporal dimension to obtain temporal features  $\mathbf{t} \in \mathbb{R}^{S \times D_2}$  and along the spatial dimension to get the video-level spatial representation  $\mathbf{s} \in \mathbb{R}^{T \times D_2}$ . Finally, we concatenate both to obtain spatio-temporal features.

$$\mathbf{z} = [\mathbf{t} \quad \mathbf{s}] \in \mathbb{R}^{(S+T) \times D_2} . \quad (2)$$

Before feeding the video features  $\mathbf{z}$  into the LLM, we project them into the same feature space as the text embeddings by applying a linear projection layer:

$$\mathbf{w} = \text{Linear}(\mathbf{z}) \in \mathbb{R}^{(S+T) \times D_2} . \quad (3)$$

with  $\mathbf{w}$  being a sequence of visual tokens. The feature vectors  $\mathbf{f}_i$  are also average-pooled along the temporal dimension to obtain a single video-level representation  $\mathbf{f} \in \mathbb{R}^{D_1}$ . The video-level feature representation  $\mathbf{f}$  is passed through two classification heads  $\mathbf{C}_{\text{foul}}$  and  $\mathbf{C}_{\text{sev}}$  to obtain the type of foul (*i.e.* Tackling, Holding, Pushing, Standing tackling, Elbowing, Dive, Challenge, or High leg) and to determine whether it is a foul or not, and the corresponding severity

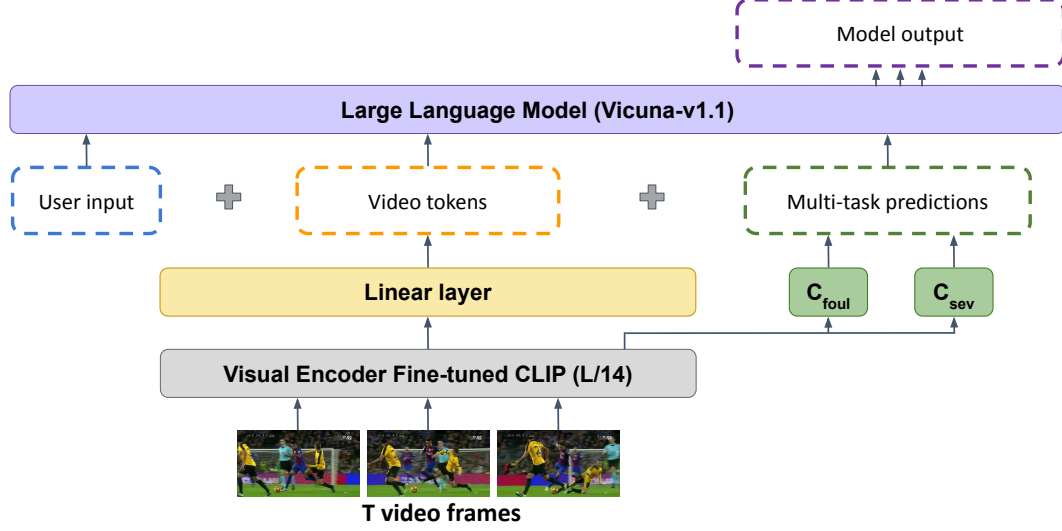


Figure 3. **Architecture of X-VARS.** X-VARS is a visual language model based on a fine-tuned CLIP visual encoder to extract spatio-temporal video features and to obtain multi-task predictions regarding the type and severity of fouls. The linear layer connects the vision encoder to the language model by projecting the video features in the text embedding dimension. We input the projected spatio-temporal features alongside the text predictions obtained by the two classification heads  $C_{foul}$  and  $C_{sev}$  (for the task of determining the type of foul and the task of determining if it is a foul and the corresponding severity) into the Vicuna-v1.1 model, initialized with weights from LLaVA.

(i.e. No offence, Offence + No card, Offence + Yellow card or Offence + Red card), with the predictions being:

$$P_{foul} = \arg \max C_{foul} , \quad (4)$$

$$P_{sev} = \arg \max C_{sev} . \quad (5)$$

These predicted labels with the highest confidence are injected as a textual prompt to the LLM. Hence, this multi-task classification enables the model to utilize acquired information to enhance the performance of the explanation.

To obtain high performance with LLMs, a crucial part consists in determining a prompt which is understandable by the LLM. As we use the Video-ChatGPT backbone, we design our prompt with the following query:

$$USER : < Question > < P_{foul} > < P_{sev} > < w > \quad (6)$$

Assistant :

where  $< Question >$  represents one of our questions randomly sampled from the training set of video-question-answer triplets,  $< P_{foul} >$  and  $< P_{sev} >$  are the two predictions on the foul type and severity recognition task obtained from the fine-tuned CLIP, and  $< w >$  are the projected spatio-temporal features. X-VARS is trained to predict the answers of the assistant as an auto-regressive model.

## 4.2. Training Paradigm

We propose a two-stage training approach. The first stage fine-tunes CLIP on a multi-task classification to learn prior knowledge about football and refereeing. The second

|         | CLIP | $C_{foul}$ & $C_{sev}$ | Linear Layer | LLM |
|---------|------|------------------------|--------------|-----|
| Stage 1 | 🔥    | 🔥                      | -            | -   |
| Stage 2 | ❄️   | ❄️                     | 🔥            | ❄️  |

Table 2. **Overview of the training stages.** In stage 1, we fine-tune CLIP and the classification heads  $C_{foul}$  and  $C_{sev}$  to give X-VARS some prior knowledge about refereeing. In stage 2, we keep them frozen and fine-tune the linear layer and partially the LLM.

step consists in fine-tuning the projection layer and several layers of the LLM to enhance the model’s generation abilities in the sport-specific domain.

**Stage 1: Fine-tuning to inject football knowledge.** While CLIP is excellent at generalizing across various image tasks, it lacks the ability to recognize fine-grained actions or events. These actions are mostly recognizable by considering the temporal dimension rather than images alone. For instance, assessing the severity of a foul requires considering factors such as the intensity and the speed, which cannot be determined by simply examining images. Since CLIP was not trained specifically on football data, the feature representation between two football clips would be very similar, despite the videos depicting different scenarios. Hence, since all our videos are related to football, the output features will share similarities. This proximity between features actually poses a challenge for the LLM, making it difficult to effectively distinguish between different actions. To avoid these issues, we fine-tune CLIP on the *SoccerNet*

*MVFoul* dataset [24] to learn prior knowledge about football. The training consists of minimizing the summed cross-entropy loss of both tasks. Given the similar magnitudes of both losses, we sum them without scaling or weighting.

**Stage 2: Feature alignment and end-to-end training.** We freeze the fine-tuned CLIPs weights and continue training the linear projection layer and the LLM. Training a projection layer from scratch requires many high-quality video-text pairs and computational resources. To alleviate this, we use the pre-trained weights of the projection layer from Video-ChatGPT [41], which was trained on a dataset of 100k video-text pairs. We further fine-tune this projection layer to map the spatio-temporal features of our football clips into the same dimensional space as the word embeddings. As demonstrated in [37, 40, 41, 71], a simple linear layer is sufficient for connecting video features with the word embedding. During training, we replace the predictions of CLIP  $\langle P_{foul} \rangle$  and  $\langle P_{sev} \rangle$  by the ground truth  $\langle G_{foul} \rangle$  and  $\langle G_{sev} \rangle$  as CLIPs predictions might be noisy, which could lead to confusions. Since determining foul type and severity is subjective, alignment between the ground truth of the *SoccerNet-MVFoul* dataset and the referee responses from our *SoccerNet-XFoul* dataset may vary. Consequently, even when giving  $\langle G_{foul} \rangle$  and  $\langle G_{sev} \rangle$  during training, the model may not only use this information without using the video tokens to produce the text.

## 5. Experiments

In this section, we analyze the performance of X-VARS on the two most important refereeing questions: “*Is it a foul or not? Why?*” and “*What card would you give? Why?*”. Given the importance of these questions, we conduct a comprehensive and detailed analysis, providing insights into the improvements in the video recognition performance, a human study to assess the model’s explanations, some qualitative results, and a thorough ablation study.

### 5.1. Implementation details

We fine-tune CLIP-L/14 on the *SoccerNet-XFoul* dataset for 14 epochs with a learning rate of  $5 \times 10^{-6}$  on a single Nvidia V100 GPU with a batch size of 64, using gradient accumulation to overcome memory limitation. The fine-tuning of the model takes about 9 hours. We use 16 frames in 224p resolution per clip, with 8 frames before and 8 frames after the foul. For the second stage, we employ QLORA [15, 28] to enhance memory efficiency and enable training on a single GPU. We only fine-tune 1% of the layers for 3 epochs using a learning rate of  $2 \times 10^{-4}$  and an overall batch size of 32. The training on 2 A100 40GB GPUs takes about 2 hours. Table 2 provides an overview of the state of the various key components during training.

|          |      | Distribution |     |    |     |     |
|----------|------|--------------|-----|----|-----|-----|
|          | Mean | 1            | 2   | 3  | 4   | 5   |
| Referees | 4.0  | 3%           | 10% | 8% | 46% | 33% |
| X-VARS   | 3.8  | 3%           | 17% | 4% | 46% | 30% |

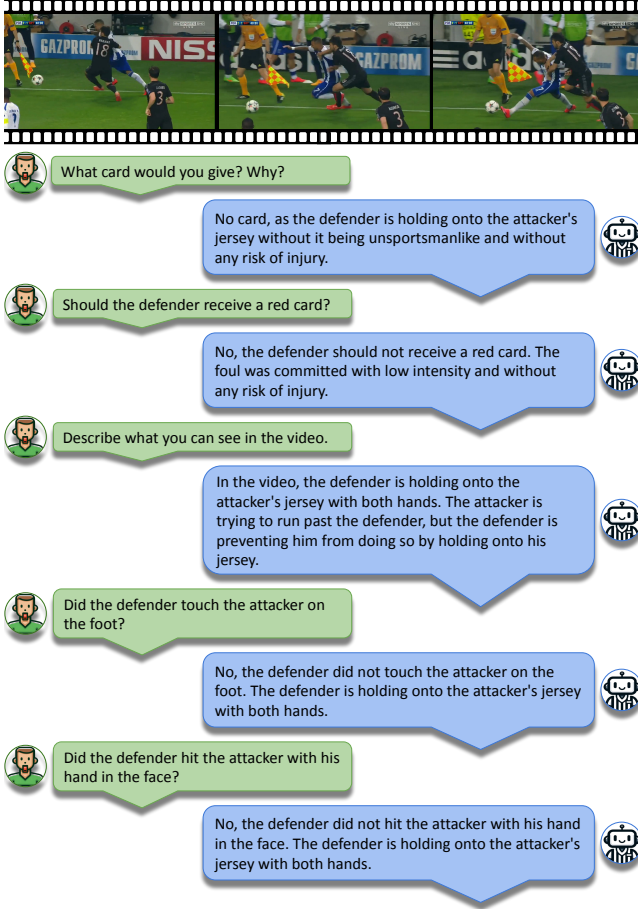
Table 3. **Score and distributions obtained during our human study comparing the quality of referees and X-VARS generated explanations.** The mean scores of X-VARS closely match those of human referees. In 46% of the video clips, X-VARS achieved higher scores for its explanations than the human referees. The distribution of the results is very similar for human referees and X-VARS. A score of 5 is the highest and represents *strongly agree* while 1 is *strongly disagree*.

### 5.2. Human study on explanation performance

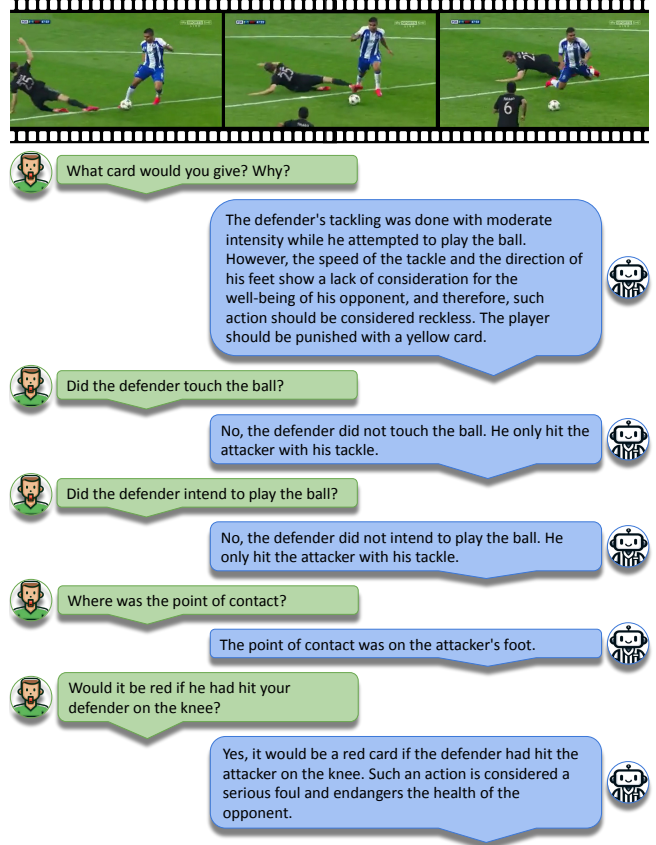
Evaluating generative tasks, such as text, image, or video generation, remains a significant challenge due to their subjective nature and the absence of proper evaluation metrics. Traditional language metrics are not very informative for our purpose, as two sentences can be linguistically very similar, yet have entirely different meanings. To achieve quantitative results, we conducted a human study with 20 football referees who evaluated the quality of responses without knowing if they were generated by a human referee or by X-VARS. The referee officiated between 85 and 850 official games, with an average of 490 games. Each participant assessed 20 random video clips, each lasting 5 seconds, with no time restrictions. They evaluated the quality of the explanation, considering whether the evaluation was consistent with the video and if the decision and explanation aligned with the Laws of the Game [30]. They rated each explanation on a scale of 1 to 5, with 5 indicating *strongly agree* and 1 indicating *strongly disagree*. Table 3 shows the results, with X-VARS performing similarly to the human referees, with only minimal score differences. X-VARS’s explanations were more convincing in 46% of the cases than the referee’s. While both show similar results for *strongly agree* and *agree*, X-VARS obtains more *disagree* responses than human referees. The majority of videos where participants disagreed with X-VARS involve types of fouls that are rare in our dataset, *i.e.* when the defender uses his arms illegally by pushing his opponent or hitting him with the elbow in the face. Overall, the human study highlights X-VARS’s impressive ability to understand football videos and explain its decisions at a level comparable to human referees.

### 5.3. Qualitative results

Figure 4 showcases two examples of conversations generated by our proposed X-VARS. Particularly, we illustrate its remarkable ability to understand and generate decisions with explanations related to visual content and the Laws of the Game [30]. Although X-VARS was only fine-tuned on



(a) **Q:** What card would you give? Why? **GT:** No card because the defender briefly held onto the attacker's arm during the fight for the ball, without it being unsportsmanlike.



(b) **Q:** What card would you give? Why? **GT:** No card. Even though the defender had no chance to play the ball, he touched the attacker with low intensity on the foot.

Figure 4. **Qualitative results.** Although X-VARS has never been specifically fine-tuned for conversation, it has inherited its conversational capabilities from the pre-trained model. X-VARS demonstrates impressive discussion skills while being aligned with the video content and the Laws of the Game. (a) X-VARS is close to the ground truth and is able to accurately answer the user's question. (b) This example shows the subjectivity of foul situations. X-VARS interprets the foul as medium intensity, while the human referee interprets it as low intensity with no chance to play the ball.

two questions, Figure 4 illustrates that X-VARS can generalize and accurately answer or describe video content without any specific fine-tuning. Furthermore, X-VARS was not fine-tuned for conversation, but we inherited these capabilities from the pre-trained conversational model VideoChatGPT [41], which serves as the foundation for X-VARS. Hence, throughout our two fine-tuning stages, we retained the conversation capabilities of the foundation model and can generate meaningful conversations with X-VARS about football and refereeing. Another interesting fact is the typical characteristic of LLMs to consistently agree with human users. Surprisingly, X-VARS mostly maintains its decision and offers comprehensive explanations for it, even when asked questions such as “Should the defender receive a red card?”, when the specific foul would not require any.

However, similarly to other LLMs, X-VARS has also inherited typical issues such as hallucinations, in which it recognizes actions in the video that are not present. Future work could investigate if more high-quality data or more advanced LLMs would limit this hallucination effect.

#### 5.4. Ablation study

**Video action recognition performance.** Table 4 shows the performance of CLIP-L/14 after the fine-tuning process in foul classification tasks. We compare it to the previous state-of-the-art (SOTA) achieved by Held *et al.* [24] by using the same number of frames and video quality to have a fair comparison. Held *et al.* used the MVit [17, 36] video encoder to extract spatio-temporal features. As SoccerNet-MVFoul contains multiple views for each action, they ag-



| Feat. extr.    | Pooling     | Type of Foul |             | Offence Severity |             |
|----------------|-------------|--------------|-------------|------------------|-------------|
|                |             | Acc.         | BA.         | Acc.             | BA.         |
| ResNet [24]    | Mean        | 0.30         | 0.27        | 0.34             | 0.25        |
| ResNet [24]    | Max         | 0.32         | 0.27        | 0.32             | 0.24        |
| R(2+1)D [24]   | Mean        | 0.32         | 0.34        | 0.34             | 0.30        |
| R(2+1)D [24]   | Max         | 0.34         | 0.33        | 0.39             | 0.31        |
| VARS-MViT [24] | Mean        | 0.44         | <b>0.40</b> | 0.38             | 0.31        |
| VARS-MViT [24] | Max         | 0.45         | 0.39        | 0.43             | 0.34        |
| CLIP-L/14      | Single-view | <b>0.51</b>  | 0.39        | 0.52             | <b>0.35</b> |
| X-VARS         | Single-view | /            | /           | <b>0.62</b>      | <b>0.35</b> |

Table 4. **Multi-task classification.** We compare the multi-task classification accuracy of the fine-tuned CLIP-L/14 and X-VARS (fine-tuned CLIP + LLM) to the performance obtained by Held *et al.* [24]. We obtain state-of-the-art performances for three of the four metrics while using a single view instead of multi-views. X-VARS enhances the classification accuracy of offence and severity by 19% compared to the previous SOTA. **Acc.** stands for Accuracy and **BA.** stands for balanced accuracy.

gregate the features of the different views by mean or max pooling. In this work, we fine-tune CLIP on a single view and evaluate it on the same actions, using only a single view instead of the multi-views. Despite fewer views, CLIP outperforms the previous SOTA in three of the four metrics, especially enhancing foul and severity classification, with an increase of 9% in accuracy. To determine the recognition performance of X-VARS, we asked X-VARS for each video clip, if it was a foul or not, and its corresponding severity. We then asked ChatGPT-3.5 to extract the classification predictions from the generated explanations. Finally, comparing these predictions of X-VARS to the ground truth, we observe a significant performance increase to 62% accuracy in determining whether a foul occurred and its severity compared to CLIPs predictions. Hence, X-VARS outperforms the previous state-of-the-art VARS (MVIT+Max Pool) system [24] by 19%. However, since most of the explanations of X-VARS do not explicitly indicate the type of foul, it is not possible to accurately extract it from the explanations. For this reason, we were not able to evaluate the accuracy of X-VARS in determining the type of foul.

**Influence of the video tokens.** We investigated if the LLM simply generates its answers based on the multi-task predictions that we give as input to the LLM or if it also considers the video tokens. To test this hypothesis, we use the X-VARS prediction obtained in the previous section, and we compare it to the CLIPs prediction provided as input. Interestingly, X-VARS did not simply replicate the CLIP prediction as it only agreed on 76% of the cases. This result shows that X-VARS, throughout its training, developed the ability to re-evaluate the multi-task predictions and understand that they are not always reliable. Consequently, X-VARS does not only rely on the text predictions for its answers but also incorporates information from the video tokens.

## CLIPs classification predictions vs. no text predictions.

To validate our new training paradigm, we compared the quality of our X-VARS trained with classification prediction as additional text input against training it only with video features. To compare the two models, we randomly selected a set of 40 video clips with a uniform distribution of various types of actions and severities. By qualitatively analyzing the results on the selected set, both models generate similar outcomes for most of the video clips. The main difference occurs for less frequent types of actions and severities. For instance, X-VARS without predictions fails to predict a single “No foul” instance and achieves a balanced accuracy of only 29%, while X-VARS obtains a 6% higher balanced accuracy. Figure 4a shows a clip of a defender holding his opponent, an underrepresented action in our dataset. X-VARS without predictions incorrectly predicts: “No card, as the defender pushed the attacker in the back with low intensity during the fight for the ball, without any risk of injury”. On the other hand, X-VARS with predictions provides an accurate explanation: “No card, as the defender is holding onto the attacker’s jersey without it being unsportsmanlike and without any risk of injury.”. Throughout our testing, we encountered several instances where X-VARS with prediction tokens aligned more closely with the ground truth, especially for underrepresented actions. These results show the effectiveness of our new training paradigm in achieving higher accuracy and more accurate explanations.

## 6. Conclusion

In this work, we investigated the potential of using LLMs to enhance transparency and explainability within decision-making processes. We proposed X-VARS, a multi-modal language model, which can perform a multitude of tasks, including video description, question answering, video action recognition, and conducting meaningful conversations based on video content. X-VARS achieves state-of-the-art performance in determining whether a duel between players constitutes a foul and in assessing the corresponding severity. The qualitative results and human study underscore the exceptional capabilities of X-VARS in explaining its decision, indicating its potential to enhance football refereeing by providing accurate decisions and explanations.

**Acknowledgement.** This work was partly supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding and the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI). J. Held and A. Cioppa are funded by the F.R.S.-FNRS. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.



## References

- [1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Monteiro Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv*, abs/2204.14198, 2022. 1, 2
- [2] Adrià Arbués Sangüesa, Adrià Martín, Javier Fernández, Coloma Ballester, and Gloria Haro. Using player’s body-orientation to model pass feasibility in soccer. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3875–3884, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). 2
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Sutskever Ilya, and Dario Amodei. Language models are few-shot learners. *arXiv*, abs/2005.14165, 2020. 1, 2, 3
- [4] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. *arXiv*, abs/2102.08981, 2021. 3
- [5] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. ShareGPT4V: Improving large multi-modal models with better captions. *arXiv*, abs/2311.12793, 2023. 2
- [6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and et al. PaLM: Scaling language modeling with pathways. *arXiv*, abs/2204.02311, 2022. 1, 2
- [7] Anthony Cioppa, Adrien Delière, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. Scaling up SoccerNet with multi-view spatial localization and re-identification. *Sci. Data*, 9(1):1–9, Jun. 2022. 2
- [8] Anthony Cioppa, Adrien Delière, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. A context-aware loss function for action spotting in soccer videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 13123–13133, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). 2
- [9] Anthony Cioppa, Adrien Delière, Silvio Giancola, Florian Magera, Olivier Barnich, Bernard Ghanem, and Marc Van Droogenbroeck. Camera calibration and player localization in SoccerNet-v2 and investigation of their representations for action spotting. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 4532–4541, Nashville, TN, USA, Jun. 2021. 2
- [10] Anthony Cioppa, Adrien Delière, Maxime Istasse, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 2505–2514, Long Beach, CA, USA, Jun. 2019. Inst. Electr. Electron. Eng. (IEEE). 2
- [11] Anthony Cioppa, Silvio Giancola, Adrien Delière, Le Kang, Xin Zhou, Zhiyu Cheng, Bernard Ghanem, and Marc Van Droogenbroeck. SoccerNet-tracking: Multiple object tracking dataset and benchmark in soccer videos. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 3490–3501, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [12] Anthony Cioppa, Silvio Giancola, Vladimir Somers, Florian Magera, Xin Zhou, Hassan Mkhallati, Adrien Delière, Jan Held, Carlos Hinojosa, Amir M. Mansourian, Pierre Miralles, Olivier Barnich, Christophe De Vleeschouwer, Alexandre Alahi, Bernard Ghanem, Marc Van Droogenbroeck, Abdullah Kamal, Adrien Maglo, Albert Clapés, Amr Abdelaziz, Artur Xarles, Astrid Orcesi, Atom Scott, Bin Liu, Byoungkwon Lim, Chen Chen, Fabian Deuser, Feng Yan, Fufu Yu, Gal Shitrit, Guanshuo Wang, Gyusik Choi, Hankyul Kim, Hao Guo, Hasby Fahrudin, Hidenari Koguchi, Håkan Ardö, Ibrahim Salah, Ido Yerushalmy, Iftikar Muhammad, Ikuma Uchida, Ishay Be’ery, Jaonary Rabarisoa, Jeongae Lee, Jiajun Fu, Jianqin Yin, Jinghang Xu, Jongho Nang, Julien Denize, Junjie Li, Junpei Zhang, Juntae Kim, Kamil Synowiec, Kenji Kobayashi, Kexin Zhang, Konrad Habel, Kota Nakajima, Licheng Jiao, Lin Ma, Lizhi Wang, Luping Wang, Menglong Li, Mengying Zhou, Mohamed Nasr, Mohamed Abdelwahed, Mykola Liashuha, Nikolay Falaleev, Norbert Oswald, Qiong Jia, Quoc-Cuong Pham, Ran Song, Romain Hérault, Rui Peng, Ruilong Chen, Ruixuan Liu, Ruslan Baikulov, Ryuto Fukushima, Sergio Escalera, Seungcheon Lee, Shimin Chen, Shouhong Ding, Taiga Someya, Thomas B. Moeslund, Tianjiao Li, Wei Shen, Wei Zhang, Wei Li, Wei Dai, Weixin Luo, Wending Zhao, Wenjie Zhang, Xinquan Yang, Yanbiao Ma, Yeeun Joo, Yingsen Zeng, Yiyang Gan, Yongqiang Zhu, Yujie Zhong, Zheng Ruan, Zhiheng Li, Zhijian Huang, and Ziyu Meng. SoccerNet 2023 challenges results. *arXiv*, abs/2309.06006, 2023. 2
- [13] Adrien Delière, Anthony Cioppa, Silvio Giancola, Meisam J. Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. SoccerNet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, *CVsports*, pages 4508–4519, Nashville, TN, USA, Jun. 2021. 2
- [14] Julien Denize, Mykola Liashuha, Jaonary Rabarisoa, Astrid Orcesi, and Romain Hérault. COMEDIAN: Self-supervised learning and knowledge distillation for action spotting using transformers. *arXiv*, abs/2309.01270, 2023. 2

- [15] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv*, abs/2305.14314, 2023. **6**
- [16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, abs/1810.04805, 2018. **1, 2**
- [17] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 6804–6815, Montréal, Can., Oct. 2021. Inst. Electr. Electron. Eng. (IEEE). **7**
- [18] Jiale Fang, Calvin Yeung, and Keisuke Fujii. Foul prediction with estimated poses from soccer broadcast video. *arXiv*, abs/2402.09650, 2024. **2**
- [19] FIFA. Semi automated offside technology. <https://www.fifa.com/technical/football-technology/football-technologies-and-innovations-at-the-fifa-world-cup-2022/semi-automated-offside-technology>, 2023. **2**
- [20] Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Dinesh Baniya Kshatri, and Pål Halvorsen. Soccer game summarization using audio commentary, metadata, and captions. *Proceedings of the 1st Workshop on User-centric Narrative Summarization of Long Videos*, Oct. 2022. **2**
- [21] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. SoccerNet: A scalable dataset for action spotting in soccer videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 1792–179210, Salt Lake City, UT, USA, Jun. 2018. Inst. Electr. Electron. Eng. (IEEE). **2**
- [22] Silvio Giancola, Anthony Cioppa, Adrien Delière, Floriane Magera, Vladimir Somers, Le Kang, Xin Zhou, Olivier Barnich, Christophe De Vleeschouwer, Alexandre Alahi, Bernard Ghanem, Marc Van Droogenbroeck, Abdulrahman Darwish, Adrien Maglo, Albert Clapés, Andreas Luyts, Andrei Boiarov, Artur Xarles, Astrid Orcesi, Avijit Shah, Baoyu Fan, Bharath Comandur, Chen Chen, Chen Zhang, Chen Zhao, Chengzhi Lin, Cheuk-Yiu Chan, Chun Chuen Hui, Dengjie Li, Fan Yang, Fan Liang, Fang Da, Feng Yan, Fufu Yu, Guanshuo Wang, H. Anthony Chan, He Zhu, Hongwei Kan, Jiaming Chu, Jianming Hu, Jianyang Gu, Jin Chen, João V. B. Soares, Jonas Theiner, Jorge De Corte, José Henrique Brito, Jun Zhang, Junjie Li, Junwei Liang, Leqi Shen, Lin Ma, Lingchi Chen, Miguel Santos Marques, Mike Azatov, Nikita Kasatkin, Ning Wang, Qiong Jia, Quoc Cuong Pham, Ralph Ewerth, Ran Song, Rengang Li, Rikke Gade, Ruben Debie, Runze Zhang, Sangrok Lee, Sergio Escalera, Shan Jiang, Shigeyuki Odashima, Shimin Chen, Shoichi Masui, Shouhong Ding, Sin-wai Chan, Siyu Chen, Tallal El-Shabrawy, Tao He, Thomas B. Moeslund, Wan-Chi Siu, Wei Zhang, Wei Li, Xiangwei Wang, Xiao Tan, Xiaochuan Li, Xiaolin Wei, Xiaoqing Ye, Xing Liu, Xinying Wang, Yandong Guo, Yaqian Zhao, Yi Yu, Yingying Li, Yue He, Yujie Zhong, Zhenhua Guo, and Zhiheng Li. SoccerNet 2022 challenges results. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 75–86, Lisbon, Port., Oct. 2022. ACM. **2**
- [23] Silvio Giancola, Anthony Cioppa, Julia Georgieva, Johsan Billingham, Andreas Serner, Kerry Peek, Bernard Ghanem, and Marc Van Droogenbroeck. Towards active learning for action spotting in association football videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 5098–5108, Vancouver, Can., Jun. 2023. Inst. Electr. Electron. Eng. (IEEE). **2**
- [24] Jan Held, Anthony Cioppa, Silvio Giancola, Abdullah Hamdi, Bernard Ghanem, and Marc Van Droogenbroeck. VARS: Video assistant referee system for automated soccer decision making from multiple views. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 5086–5097, Vancouver, Can., Jun. 2023. Inst. Electr. Electron. Eng. (IEEE). **2, 6, 7, 8**
- [25] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv*, abs/2203.15556, 2022. **1, 2**
- [26] Yutaro Honda, Rei Kawakami, Ryota Yoshihashi, Kenta Kato, and Takeshi Naemura. Pass receiver prediction in soccer using video and players’ trajectories. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3502–3511, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). **2**
- [27] James Hong, Haotian Zhang, Michaël Gharbi, Matthew Fisher, and Kayvon Fatahalian. Spotting temporally precise, fine-grained events in video. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 13695 of *Lect. Notes Comput. Sci.*, pages 33–51, Tel Aviv, Israël, 2022. Springer Nat. Switz. **2**
- [28] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv*, abs/2106.09685, 2021. **6**
- [29] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, Qiang Liui, Kriti Agarwal, Zewen Chi, Johan Bjorck, Vishrav Chaudhary, Subhojit Som, Xia Song, and Furu Wei. Language is not all you need: Aligning perception with language models. *arXiv*, abs/2302.14045, 2023. **2**
- [30] IFAB. Laws of the game. Technical report, The International Football Association Board, Zurich, Switzerland, 2022. **3, 6, 7**
- [31] Maxime Istasse, Vladimir Somers, Pratheeban Elancheliyan, Jaydeep De, and Davide Zambrano. DeepSportradar-v2: A multi-sport computer vision dataset for sport understandings. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 23–29, Ottawa, Ontario, Can., Oct. 2023. ACM. **2**
- [32] Yudong Jiang, Kaixu Cui, Leilei Chen, Canjin Wang, and Changliang Xu. SoccerDB: A large-scale database for com-

- prehensive video understanding. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, page 1–8, Seattle, WA, USA, Oct. 2020. ACM. 2
- [33] Himabindu Lakkaraju, Dylan Slack, Yuxin Chen, Chenhao Tan, and Sameer Singh. Rethinking explainability as a dialogue: A practitioner’s perspective. *arXiv*, abs/2202.01875, 2022. 2
- [34] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. TVQA: Localized, compositional video question answering. *arXiv*, abs/1809.01696, 2018. 3
- [35] Haopeng Li, Andong Deng, QiuHong Ke, Jun Liu, Hossein Rahmani, Yulan Guo, Bernt Schiele, and Chen Chen. SportsQA: A large-scale video question answering benchmark for complex and professional sports. *arXiv*, abs/2401.01505, 2024. 3
- [36] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MViTv2: Improved multiscale vision transformers for classification and detection. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4794–4804, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 7
- [37] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv*, abs/2304.08485, 2023. 2, 3, 6
- [38] Katja Ludwig, Julian Lorenz, Robin Schön, and Rainer Lienhart. All keypoints you need: Detecting arbitrary keypoints on the body of triple, high, and long jump athletes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 5179–5187, Vancouver, Can., Jun. 2023. Inst. Electr. Electron. Eng. (IEEE). 2
- [39] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv*, abs/1705.07874, 2017. 2
- [40] Chenyang Lyu, Minghao Wu, Longyue Wang, Xinting Huang, Bingshuai Liu, Zefeng Du, Shuming Shi, and Zhaopeng Tu. Macaw-LLM: Multi-modal language modeling with image, audio, video, and text integration. *arXiv*, abs/2306.09093, 2023. 6
- [41] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-ChatGPT: Towards detailed video understanding via large vision and language models. *arXiv*, abs/2306.05424, 2023. 1, 2, 3, 4, 6, 7
- [42] Adrien Maglo, Astrid Orcesi, and Quoc-Cuong Pham. Efficient tracking of team sport players with few game-specific annotations. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3460–3470, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [43] Amir M. Mansourian, Vladimir Somers, Christophe De Vleeschouwer, and Shohreh Kasaei. Multi-task learning for joint re-identification, team affiliation, and role classification for sports visual tracking. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, page 103–112, Ottawa, Ontario, Can., Oct. 2023. ACM. 2
- [44] Cise Midoglu, Steven Hicks, Vajira Thambawita, Tomas Kupka, and Pål Halvorsen. MMSys’22 grand challenge on AI-based video production for soccer. In *ACM Multimedia Systems Conference (MMSys)*, pages 1–6, Athlone, Ireland, Jun. 2022. 2
- [45] Cise Midoglu, Saeed Shafiee Sabet, Mehdi Houshmand Sarkhoosh, Mohammad Majidi, Sushant Gautam, Håkon Maric Solberg, Tomas Kupka, and Pål Halvorsen. Ai-based sports highlight generation for social media. *Proceedings of the 3rd Mile-High Video Conference on zzz*, Feb. 2024. 2
- [46] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. *arXiv*, abs/1906.03327, 2019. 3
- [47] Hassan Mkhallati, Anthony Cioppa, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. SoccerNet-caption: Dense video captioning for soccer broadcasts commentaries. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 5074–5085, Vancouver, Can., Jun. 2023. Inst. Electr. Electron. Eng. (IEEE). 2, 3
- [48] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and et al. Training language models to follow instructions with human feedback. *arXiv*, abs/2203.02155, 2022. 2
- [49] Luca Pappalardo, Paolo Cintia, Alessio Rossi, Emanuele Massucco, Paolo Ferragina, Dino Pedreschi, and Fosca Giannotti. A public data set of spatio-temporal match events in soccer competitions. *Sci. Data*, 6(1):1–15, Oct. 2019. 2
- [50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Int. Conf. Mach. Learn. (ICML)*, pages 8748–8763, Jul. 2021. 2, 4
- [51] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “why should i trust you?”: Explaining the predictions of any classifier. *arXiv*, abs/1602.04938, 2016. 2
- [52] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim,

- Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Froberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, and et al. BLOOM: A 176B-parameter open-access multi-lingual language model. *arXiv*, abs/2211.05100, 2022. 2
- [53] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Workman, and et al. LAION-5b: An open large-scale dataset for training next generation image-text models. *arXiv*, abs/2210.08402, 2022. 3
- [54] Atom Scott, Ikuma Uchida, Masaki Onishi, Yoshinari Kameda, Kazuhiro Fukui, and Keisuke Fujii. SoccerTrack: A dataset and tracking algorithm for soccer with fish-eye and drone videos. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, pages 3568–3578, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [55] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *arXiv*, abs/1610.02391, 2016. 2
- [56] Karolina Seweryn, Gabriel Cheć, Szymon Łukasik, and Anna Wróblewska. Improving object detection quality in football through super-resolution techniques. *arXiv*, abs/2402.00163, 2024. 2
- [57] Karolina Seweryn, Anna Wróblewska, and Szymon Łukasik. Survey of action recognition, spotting and spatio-temporal localization in soccer – current trends and research perspectives. *arXiv*, abs/2309.12067, 2023. 2
- [58] João V. B. Soares and Avijit Shah. Action spotting using dense detection anchors revisited: Submission to the SoccerNet challenge 2022. *arXiv*, abs/2206.07846, 2022. 2
- [59] João V. B. Soares, Avijit Shah, and Topojoy Biswas. Temporally precise action spotting in soccer videos using dense detection anchors. In *IEEE Int. Conf. Image Process. (ICIP)*, pages 2796–2800, Bordeaux, France, Oct. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [60] Vladimir Somers, Christophe De Vleeschouwer, and Alexandre Alahi. Body part-based representation learning for occluded person Re-Identification. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, pages 1613–1623, Waikoloa, HI, USA, Jan. 2023. Inst. Electr. Electron. Eng. (IEEE). 2
- [61] Alessandro Suglia, José Lopes, Emanuele Bastianelli, Andrea Vanzo, Shubham Agarwal, Malvina Nikandrou, Lu Yu, Ioannis Konstas, and Verena Rieser. Going for GOAL: A resource for grounded football commentaries. *arXiv*, abs/2211.04534, 2022. 3
- [62] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhausen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. MovieQA: Understanding stories in movies through question-answering. *arXiv*, abs/1512.02902, 2015. 3
- [63] Graham Thomas, Rikke Gade, Thomas B. Moeslund, Peter Carr, and Adrian Hilton. Computer vision for sports: current applications and research topics. *Comput. Vis. Image Underst.*, 159:3–18, Jun. 2017. 2
- [64] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv*, abs/2302.13971, 2023. 1, 2
- [65] Gabriel Van Zandycke, Vladimir Somers, Maxime Istasse, Carlo Del Don, and Davide Zambrano. DeepSportradar-v1: Computer vision dataset for sports understanding with high quality annotations. In *Int. ACM Work. Multimedia Content Anal. Sports (MMSports)*, pages 1–8, Lisbon, Port., Oct. 2022. ACM. 2
- [66] Renaud Vandeghen, Anthony Cioppa, and Marc Van Droogenbroeck. Semi-supervised training to improve player and ball detection in soccer. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, CVsports, pages 3480–3489, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2
- [67] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *arXiv*, abs/1711.00399, 2017. 2
- [68] Xiao Xu, Chenfei Wu, Shachar Rosenman, Vasudev Lal, Wanxiang Che, and Nan Duan. BridgeTower: Building bridges between encoders in vision-language representation learning. *arXiv*, abs/2206.08657, 2022. 2
- [69] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. mPLUG-Owl2: Revolutionizing multi-modal large language model with modality collaboration. *arXiv*, abs/2311.04257, 2023. 1, 2
- [70] Junqing Yu, Aiping Lei, Zikai Song, Tingting Wang, Hengyou Cai, and Na Feng. Comprehensive dataset of broadcast soccer videos. In *IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, pages 418–423, Miami, FL, USA, Apr. 2018. Inst. Electr. Electron. Eng. (IEEE). 2
- [71] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. *arXiv*, abs/2304.10592, 2023. 1, 2, 6