D4.2

XR and Media Transformation Services, API and Authoring Tools v2

Project Title	XR mEdia eCOsystem
Contract No.	101070250
Instrument	Innovation Action
Thematic Priority	HORIZON-CL4-2021-HUMAN-01-06
Start of Project	1 September 2022
Duration	38 months

Deliverable title	XR and Media Transformation Services, API and Authoring Tools v2
Deliverable number	D4.2
Deliverable version	V1.0
Previous version(s)	D4.1
Contractual Date of delivery	30.06.2025
Actual Date of delivery	30.06.2025
Nature of deliverable	Report
Dissemination level	Public
Partner Responsible	CERTH
Author(s)	Antonis Karakottas [CERTH], Vangelis Chatzis [CERTH], Angelos Kanlis [CERTH], Tasos Tsalakopoulos [CERTH], Roberto Iacoviello [RAI], Maurizio Montagnuolo [RAI], Daniel Berjón Diez [UPM], Julián Cabrera Quesada [UPM], Jesús Gutiérrez Sánchez [UPM], Francisco Morán Burgos [UPM], Victoria Muñoz Murillo [UPM], Isabel María Rodríguez Márquez [UPM], Javier Usón Peirón [UPM], Cláudia Marinho [MOG], Jesus Luque Muriel [MEDIAPRO]
Reviewer(s)	Cláudia Marinho [MOG], Maurizio Montagnuolo [RAI]
EC Project Officer	Andreea Popescu

Abstract	This report provides the second version of XReco's transformation services as well as authoring tool and APIs development.
Keywords	Large Language Model, Content tagging, Neural rendering, Video Upscale, Super Resolution, Neural Radiance Fields, NeRF, Structure from Motion, SfM, Volumetric Capturing, Neural reconstruction, 3D reconstruction, 4D reconstruction, Unity3D

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Revision History

Version	Date	Modified By	Comments
V0.1	08/05/2025	Antonis Karakottas (CERTH)	First table of contents
V0.2	15/05/2025	Antonis Karakottas (CERTH)	Updated table of contents
V0.3	20/06/2025	CERTH, UPM, i2CAT, JRS, CAPGEMINI, RAI, UNIBAS, NVIDIA	First Draft
V0.4	23/06/2025	CERTH, JRS	Second Draft
V0.5	25/06/2025	Cláudia Marinho (MOG), Maurizio Montagnuolo (RAI)	Final Second Draft
V0.6	29/06/2025	Nico Patz (DW)	Final Second Draft Review
V0.7	30/06/2025	Antonis Karakottas (CERTH)	Review
V1.0	30/06/2025	Nico Patz (DW)	Final Document

Glossary

Abbreviation	Meaning
AdaIN	Adaptive Instance Normalisation
API	Application Programming Interface
BRISQUE	Blind Referenceless Image Spatial Quality Evaluator
CLI	Command Line Interface
CNN	Convolutional Neural Network
FR	Full Reference
FVV	Free Viewpoint Video
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GFP-GAN	Generative Facial Prior GAN
HD	High Resolution
HDR	High Dynamic Range
HMD	Head-Mounted Display
HR	High Resolution
IoU	Intersection over Union
LD	Low Resolution
LPIPS	Learned Patched Image Patch Similarity
LLM	Large Language Model
LoRA	Low Rank Adaptation
LR	Low Resolution
MOS	Mean opinion score
MLP	Multi-Layer Perceptron
MP	Megapixels

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

ABBREVIATION	MEANING
MR	Mixed Reality
MSE	Mean Square Error
MVS	Multi-View Stereo
NeRF	Neural Radiance Fields
NICs	Network Interface Cards
NIQE	Natural Image Quality Evaluator
NLP	Natural Language Processing
NMR	Neural Media Repository
NN	Neural Network
PSNR	Peak Signal-to-Noise Ratio
QA	Question Answering
QEM	Quadric Error Metric
RBVSR	Real-Basic Video Super Resolution
RDMA	Remote Direct Memory Access
RGB+D	Red, Green and Blue + Depth
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SDF	Signed Distance Function
SfM	Structure from Motion
SISR	Single Image Super Resolution
SR	Single Image Super-Resolution
SSIM	Structural Similarity Index
TSV	Tab-separated values file format
T2T	Text to Text
UCF	Unified Computing Framework
UDP	User Datagram Protocol
VMAF	Video Multi-method Assessment Fusion
VQA	Video Quality Assessment
VSR	Video Super Resolution
V2T	Video to Text
WebRTC	Web Real-Time Communication
WP	Work Package
XR	eXtended Reality

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Table of Contents

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Index of Figures

Index of Tables

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

1 Executive Summary

This deliverable represents a significant maturation milestone in the XReco project's WP4, transitioning from the foundational technologies presented in D4.1 to integrated, production-ready services within a distributed microservices architecture. This second version deliverable demonstrates substantial progress in validation, implementation and platform optimisation, addressing the critical evolution from prototype to production-ready systems.

The deliverable highlights the successful validation and deployment of all major XReco services included in the MVP, while also showcasing additional algorithms that extend beyond the MVP's core scope. This achievement enables seamless integration across the platform's ecosystem, maintaining modularity and scalability essential for broad adoption. Through this evolution, the project demonstrates its commitment not only to technological innovation but also delivering practical, implementable solutions that address real-world challenges in content search and creation.

The deliverable showcases the successful validation and deployment of all major XReco services that are part of the XReco MVP, enabling seamless integration across the platform's ecosystem while maintaining the modularity and scalability essential for broad adoption. This evolution reflects the project's commitment to delivering not just technological innovation, but practical, implementable solutions that address real-world content search and content creation challenges.

In a nutshell, this deliverable provides a description regarding the activities within WP4 until M34, encompassing five tasks:

- **Content sourcing and filtering (T4.1)**: Services for content search, incoming content monitoring and filtering according to a particular topic or production, interactive retrieval of content, location-centric retrieval, MR interfaces for content retrieval.
- **Neural rendering services (T4.2)**: Services for reconstructing 3D scenes from 2D image data utilizing Neural Radiance Field (NeRF) approaches, as well as Network Acceleration infrastructure.
- **3D asset aggregation and optimisation services (T4.3)**: Services for 3D reconstruction from 2D image data employing computer vision as well as machine learning pipelines. Additionally, services for media content (video, 3D point cloud data) enhancement.
- **XR volumetric and Free Viewpoint Video services (T4.4)**: Services for producing human-centred volumetric and free-viewpoint video, utilising RGB-D data.
- **APIs and authoring tool development (T4.5)**: The development of authoring tools, as well as of appropriate communication APIs between authoring tools and XReco services.

These tasks collectively contribute to the goal of advancing the capabilities of XR applications, with a focus on content creation, and integration. This progress is pivotal for fostering innovation and modularity within the XR landscape, ultimately enhancing the quality and diversity of immersive experiences and content creation.

2 Introduction

This deliverable reports on the maturation and integration of the vertical technologies that enable XR application realisation according to the use cases and requirements specified in WP5 and WP2 respectively. Building upon the foundational work presented in D4.1, this deliverable documents the successful transition from prototype technologies to production-ready services deployed within a distributed microservices architecture.

The containerisation and integration efforts represent a significant milestone in the XReco project timeline, showcasing not just technological innovation but the practical implementation of scalable, enterprise-grade solutions addressing real-world content search and content creation challenges. This evolution has been carefully orchestrated to maintain modularity while enabling seamless interoperability between services, reflecting modern practices and cloud-native design principles.

Since the initial development phase, the XReco platform has undergone substantial architectural refinements to ensure optimal performance, reliability, and scalability. The adoption of containerisation technologies has enabled consistent deployment across diverse environments while facilitating rapid iteration based on user feedback and changing requirements.

The implementation follows a microservices approach where each component operates independently yet integrated coherently within the overall ecosystem. This design supports horizontal scaling of high-demand services without impacting the entire platform, a critical consideration for processing-intensive operation like neural rendering.

The work package continues to be organised into five interconnected tasks, each now reaching production readiness for most of the developed technologies with validated implementations. The current status reflects substantial progress in validation, implementation, and platform optimisation since D4.1.

2.1 Document Organisation

Considering the diverse technical domains covered within WP4 and their advanced implementation status, this deliverable is organised to highlight both the technical achievements and the integration aspects of most components. In contrast with D4.1 which was organised according to the Task structure within WP4, this updated version is structured in consistent technological thematic areas, in order to provide a coherent flow and readability, as well as an overview of the achievements. The document structure is as follows:

Section 3: **Content Search, Monitoring, and Filtering** presents the intelligent content discovery and management services that operate on top of the Neural Media Repository (NMR). This section encompasses the Mixed Reality multimedia retrieval system, news content tagging using fined-tuned Large Language Models (LLMs), and novel content detection methodologies. These services form the foundation for content curation and discovery workflows that enable efficient asset management across the XReco platform.

Section 4: **Scene Reconstruction Services** addresses the core 3D reconstruction capabilities, encompassing both neural rendering approaches and more traditional computer vision methodologies. Neural rendering services include the enhanced Fast NeRF in-the-wild implementation with octree optimisation, 3D Gaussian Splatting (3DGS), and fast neural reconstruction. Computer vision services focus on Structure from Motion (SfM) with

Multi-View Stereo densification. This section demonstrates the platform's ability to transform 2D visual content into 3D representations suitable for XR applications.

Section 5: **Scene Reconstruction Services Validation** provides comprehensive quality evaluation methodologies and results for the reconstruction services described in Section 4. This unified validation approach represents a significant advancement over D4.1, demonstrating the maturation of evaluation frameworks and the establishment of quality benchmarks across different reconstruction methodologies. The section includes both objective metrics and subjective evaluation protocols, ensuring robust assessment of reconstruction fidelity.

Section 0: **Human-Centred Reconstruction, Volumetric and Free-Viewpoint Video** focuses on specialised services for capturing and rendering human subjects in XR environments. This includes GDNeRF for sparse-view human reconstruction, human-centred Gaussian Splatting approaches, RGB-D based free-viewpoint video (FVV) systems, holoportation capabilities, and automated 3D face reconstruction services. These technologies address the specific challenges of human representation in immersive media productions.

Section 7: Asset Aggregation and Optimisation Services presents enhancement and optimisation technologies that improve the quality and usability of multimedia assets. This encompasses 2D video upscaling services, blind face restoration capabilities, human-centred point cloud super-resolution, comprehensive 3D data enhancement methodologies, and 3D content generation services. These services ensure that assets within the XReco ecosystem meet quality standards regardless of their original capture conditions.

Section 8: **Network Acceleration Infrastructure** describes the underlying computational and networking optimisations that enable efficient deployment of XReco services. This includes direct memory management approaches, unified cloud services tools, and performance optimisation strategies that support the real-time processing requirements of XR content creation.

Section 9: **Authoring Tool Development and Service Communication** presents the integrated authoring environments that enable content creators to leverage XReco services effectively. This encompasses Unity-based authoring, XR Capsules for template-driven content creation, the Orchestrator dashboard for service coordination, and a CMS-based authoring solution for location-specific content development.

Supporting Documentation and Technical Details

Annex I: **Extended Information for Components Context** provides detailed implementation specifications and technical documentation that support the main content. This includes comprehensive details on sample structures for RSS items in news content tagging, complete API implementation for the GDNeRF service, and other technical specifications that enable developers to integrate XReco services.

Annex II: **RGB-D Cameras Information** offers comprehensive information on RGB-D sensor technologies, including detailed comparisons of capture devices and depth estimation algorithms. This technical reference supports the implementation decisions described throughout the document and provides guidance for hardware selection in different deployment scenarios.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

3 Content search, monitoring, and filtering

3.1 Overview

This section describes tools and frontends building on top of the NMR described in D3.1 and D3.2. As these components are part of the authoring workflow, they are addressed in WP4.

Most of these components are backend components that filter incoming content to select items relevant for stories being worked on, a component for tagging news content and interfaces for media search, including novel search paradigms such as search in mixed reality.

3.2 Content sourcing and filtering

The content sourcing and filtering component aims to automatically select content items from those ingested into the NMR, based on the relevance of those items for stories being worked on. These are represented by the content collected for them, materialised as content baskets (see D3.1 for a description of the concept and implementation of content baskets). The content involved may be multimodal, i.e., include text, 2D, and 3D content. The aim is to select candidate items that may be of relevance for the work of the journalist or content creator.

D4.1 has described the adaptation of the USTORY[1] framework to make it compatible with a newer deep learning framework version, and to exchange the backbone network with CLIP[2], in order to support multimodal content. This section focuses on the novel content detection aspect, describing first a video-to-text service, and then an LLM-based novel content detection approach on the resulting texts.

3.2.1 Video to text

The video-to-text component is responsible for analysing video content and determining whether it introduces novel elements relevant to a story currently in development, in terms of content baskets. In order to perform downstream tasks such as novelty detection, text is a more sustainable and interoperable representation than feature embeddings. This allows to apply the current state of the art of language models, without lock-in on a specific feature embedding space. Unlike image captioning, which focuses on visual information within single frames (spatial information), video to text (V2T) must also account for the semantics over time (temporal information), making it a more complex challenge[3].

[1] Yoon, Susik, et al. "Unsupervised Story Discovery from Continuous News Streams via Scalable Thematic Embedding." arXiv preprint arXiv:2304.04099 (2023).
[2] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PMLR, 2021.
[3] Moloud Abdar et al., "A Review of Deep Learning for Video Captioning," 2023, http://arxiv.org/abs/2304.11431.

For several years, video captioning was tackled using task specific models, often based on LSTM[4] architectures. Early approaches were inspired by image captioning techniques and typically generated single-sentence descriptions.

Subsequently, ''dense video captioning'' methods emerged, aiming to produce longer and more detailed descriptions. These models included segmenting videos into actions[5] or events[6], which were then translated into sentences. With multimodal embeddings, models like VideoBERT[7] improved representation.

More recently, transformer-based models, such as mPLUG-2[8] and VAST[9], have advanced the field by leveraging vision-language integration. However, keeping pace with the capabilities of general-purpose multimodal large language models (MLLMs) remains a significant challenge.

MLLMs have accelerated video captioning progress[10], offering flexible solutions through fine-tuning or prompting. Many are vision/video variants of broader LLM families, including VideoLLaMA[11], InternVideo2[12], Deepseek-VL[13], and Pixtral[14]. Table 1 summarizes recent open-source methods.

Table 1: A selection of recent video to text methods with permissive licenses.

Method	Type	Year	License	Comment
mPLUG-2	Specific	2023	Apache-2.0	Combines pre-trained vision and language transformers for video to text tasks.
VAST	Specific	2023	MIT	Fuses video, audio, and subtitle information via a transformer architecture.

[4] Anna Rohrbach, Marcus Rohrbach, and Bernt Schiele, “The Long-Short Story of Movie Description,” in *Pattern Recognition: 37th German Conference, GCPR 2015, Aachen, Germany, October 7-10, 2015, Proceedings 37* (Springer, 2015), 209–21.
[5] Andrew Shin, Katsunori Ohnishi, and Tatsuya Harada, “Beyond Caption to Narrative: Video Captioning with Multiple Sentences,” in *2016 IEEE International Conference on Image Processing (ICIP)* (IEEE, 2016), 3364–68.
[6] Ranjay Krishna et al., “Dense-Captioning Events in Videos,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, 706–15.
[7] Chen Sun et al., “Videobert: A Joint Model for Video and Language Representation Learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, 7464–73.
[8] Haiyang Xu et al., “mPLUG-2: A Modularized Multi-Modal Foundation Model Across Text, Image and Video,” in *Proceedings of the International Conference on Machine Learning* (PMLR, 2023).
[9] Sihan Chen et al., “VAST: A Vision-Audio-Subtitle-Text Omni-Modality Foundation Model and Dataset,” in *Conference on Neural Information Processing Systems*, 2023.
[10] Yunlong Tang et al., “Video Understanding with Large Language Models: A Survey,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.
[11] Zesen Cheng et al., “VideoLLaMA 2: Advancing Spatial-Temporal Modeling and Audio Understanding in Video-LLMs” (arXiv, 2024), https://doi.org/10.48550/arXiv.2406.07476.
[12] Yi Wang et al., “InternVideo2: Scaling Foundation Models for Multimodal Video Understanding,” in *Proceedings of the European Conference on Computer Vision*, 2024, http://arxiv.org/abs/2403.15377.
[13] Zhiyu Wu et al., “Deepseek-Vl2: Mixture-of-Experts Vision-Language Models for Advanced Multimodal Understanding,” *arXiv Preprint arXiv:2412.10302*, 2024.
[14] Pravesh Agrawal et al., “Pixtral 12B,” *arXiv Preprint arXiv:2410.07073*, 2024.

Method	Type	Year	License	Comment
Tarsier[15]	MLLM	2024	Apache-2.0	Employs temporal modelling through LLMs with the DREAM evaluation benchmark.
VideoLLaMA 2	MLLM	2024	Apache-2.0	Enhances spatio-temporal understanding with a convolutional connector.
InternLM-XC-2.5[16]	MLLM	2023	Apache-2.0	Supports long-context inputs/outputs and high-resolution inputs.
VideoChat2[17]	MLLM	2024	Apache-2.0	MVBench benchmark for spatio-temporal understanding with VideoChat2 as a baseline.
InternVideo2	MLLM	2024	Apache-2.0	Leader on MVBench for fine-grained action description, supports long-context inputs.
Deepseek-VL2	MLLM	2024	MIT & DsML	Efficient MLLM with competitive performance.
Pixtral 12B	MLLM	2024	Apache-2.0	Efficient model from the mistral family with image support.

3.2.2 Service architecture

Video captioning presents challenges across the entire processing pipeline, beginning with variations in input video properties and extending to the requirements of the textual output. Differences in video length, resolution, orientation, and frame rate can complicate practical implementation. Simultaneously, output expectations may vary in terms of language, structural format (e.g., natural language vs. keywords), level of detail, and length, adding further complexity.

To ensure robustness and flexibility, we adopted a modular client-server architecture with polling mechanisms to efficiently handle large volumes of video data. The architecture comprises the following components:

- **V2T Server**: Performs shot boundary detection and caption generation.
- **Client Scripts**: Manage job submission, polling for completion, and error handling.
- **Text-to-Text (T2T) Server**: Enables LLM-based post-processing of captions, including translation, fact extraction, keyword identification, and more.

Effective coordination across these components is essential for maintaining consistency and performance throughout the pipeline.

[15] Jiawei Wang, Liping Yuan, and Yuchen Zhang, "Tarsier: Recipes for Training and Evaluating Large Video Description Models" (arXiv, 2024), https://doi.org/10.48550/arXiv.2407.00634.
[16] Pan Zhang et al., "InternLM-XComposer-2.5: A Versatile Large Vision Language Model Supporting Long-Contextual Input and Output," 2024, http://arxiv.org/abs/2407.03320.
[17] Kunchang Li et al., "MVBench: A Comprehensive Multi-Modal Video Understanding Benchmark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

3.2.3 V2T server

The V2T server provides two primary functions: shot boundary detection and video captioning. To address hardware and context length constraints, long videos are segmented into shots. The resulting clips are then individually captioned using a V2T model with a specific prompt.

For captioning, we selected the open-source InternLM-XComposer 2.5 model over Tarsier and VideoLLaMA2, based on experiments done in the FAIRmedia project[18]. These advantages include better accuracy, responsiveness to prompts, support for long contexts and high-resolution video, and reasonable in-video text recognition.

The recommended default prompt is "Describe the following in detail with a word limit of 50 words." This generates natural-language descriptions in English for each clip. Alternatively, the model can be prompted to produce concise single-sentence captions or to focus on specific aspects of the video, depending on the desired output. The audio track may optionally be removed if the aim is to focus on visual content only.

The dockerized service accepts video input via direct upload or URL, including MinIO support, and manages separate queues for shot detection and captioning. Configuration options such as logging are available via an environment file. An example response is shown below:

```
{
  "success": true,
  "data": {
    "collective_id": "CID8b82e4a96aecffe2fb28ffcfb5735c10",
    "status": "complete",
    "is_complete": true,
    "file_name": "test.mp4",
    "file_size": "2.63 MB",
    "video_duration_s": 19.52,
    "timestamp": "2025-06-23T15:07:47.938618+00:00",
    "jobs": [
      {
        "status": "complete",
        "job_id": "4d145276365ed9beb3b6fc73ac5fb388",
        "collective_id": "CID8b82e4a96aecffe2fb28ffcfb5735c10",
        "file_name": "test.mp4",
        "use_audio": false,
        "video_duration_s": 19.52,
        "file_size": "2.63 MB",
        "prompt": "Describe the following in detail with a word limit of 50 words.",
        "model_name": "internlm",
        "caption": "A group of tourists ... exploration.",
        "inference_time_s": 24.522,
        "clip_duration_s": 16.48,
        "clip_pos": 0,
        "clip_start_s": 0.0,
        "clip_end_s": 16.48,
        "timestamp": "2025-06-23T15:08:59.858970+00:00"
      },
      {
        "status": "complete",
        "job_id": "30ad1f3c03d0a47f0418d606f53d7595",
```

[18] https://www.joanneum.at/digital/en/projects/fairmedia/

```
        "collective_id": "CID8b82e4a96aecffe2fb28ffcfb5735c10",
        "file_name": "test.mp4",
        "use_audio": false,
        "video_duration_s": 19.52,
        "file_size": "2.63 MB",
        "prompt": "Describe the following in detail with a word limit of 50 words.",
        "model_name": "internlm",
        "caption": "A serene lake ... landscape.",
        "inference_time_s": 8.647,
        "clip_duration_s": 3.04,
        "clip_pos": 1,
        "clip_start_s": 16.48,
        "clip_end_s": 19.52,
        "timestamp": "2025-06-23T15:09:09.244173+00:00"
      }
    ]
  },
  "message": "Collective status retrieved."
}
```

3.2.4 V2T client

The client script for the V2T server manages the submission and polling of video processing jobs. It supports batch processing of videos provided either as local files or URL lists. To ensure robust operation, the script detects and handles potential issues arising from variations in video size, format, or orientation, which could otherwise lead to server-side errors such as memory overflows.

Upon completion, the script generates a tab-separated values (TSV) file for each truncated video clip. Each entry includes the generated caption, clip start and end times, and the clip's position within the original video.

A TSV example for multiple videos is shown below:

```
"file_name"  "clip_pos" "clip_start_s"  "clip_end_s"    "caption"   ...
"test.mp4"  "0" "0.0"   "16.48" "A group of tourists ... exploration."   ...
"test.mp4"  "1" "16.48" "19.52" "A serene lake ... landscape."   ...
...
"test2.mp4"    "0" "3.72"  "125.731" "The next video ... ends."   ...
"test2.mp4"    "..." "..."  "..." " ... "   ...
...
```

3.2.5 T2T server

Post-processing of video captions enables the generation of textual descriptions in different languages or formats after the initial extraction of relevant information in English via optimized prompts to the V2T server. To support this, we developed a separate text-to-text (T2T) service—a general-purpose module built on large language models (LLMs), such as LLaMA 3.1–8B-Instruct. The T2T service facilitates various tasks, including translation (see Table 2), fact or keyword extraction, the combination of multiple text sources, and more.

The service supports higher-level abstractions through LLM Chains, where outputs from one step can serve as inputs for subsequent steps. This enables complex applications, such as a two-step LLM-based text correspondence measure.

Following the V2T architecture, the T2T service employs a server-client model with polling. Model configurations, system/user prompts, and parameters are fully customizable. For LLM Chains, step dependencies and parsing for specific textual formats (e.g., JSON, Python dictionaries) can be configured using conventional methods or prompt-based syntax correction. Consistent with the V2T service, each atomic LLM call is logged and registered, enabling full traceability of dependent operations.

An example T2T response for fact extraction is shown below:

```
{
  "success": true,
  "data": {
    "job": {
      "status": "complete",
      "job_id": "c338b8eea5ea80d0260660ce79e021b8",
      "collective_id": "VAL54c47d590104e58e727c1dd2e7af0700",
      "model_key": "llama31_8B",
      "prompt_key": "facts-extraction",
      "text_inputs": {
        "text1": "A group of tourists ... exploration."
      },
      "output_text": "{\"facts\": [{\"fact_index\": 1, \"fact\": \"A group of tourists gathers at the entrance
of MUTITJULU.\"}, ... ]}",
      "timestamp": "2025-06-23T15:31:28.946128+00:00",
      "inference_time_s": 26.804
    }
  },
  "message": "Job status retrieved."
}
```

The corresponding prompt used for this task is:

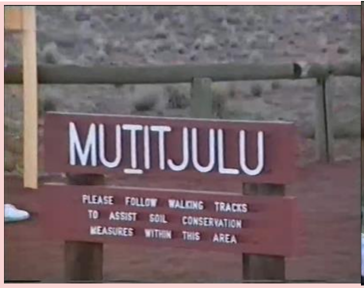
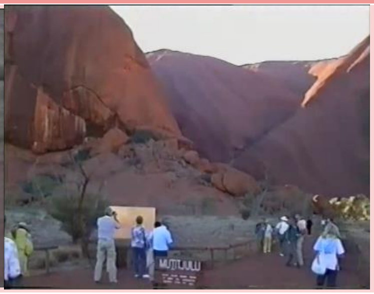
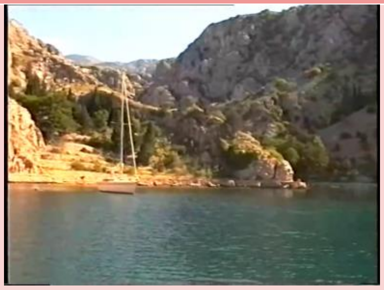
```
prompt_key: "facts-extraction",
system_prompt: "You are a helpful AI Assistant.",
user_prompt: "Bellow is a description of a video clip:\nVideo Description: {text1}\n\nExtract key facts from
the above video description paragraph.\n \
   - Each fact should contain a subject, predicate, and object. Optionally include context such as time,
place, and causality when relevant (e.g., where, when, why an action occurs).\n \
   - Ensure facts capture modalities (e.g., 'could', 'might'), negations, and quantifiers (e.g., 'all',
'some') if present in the video description.\n \
   - Facts should be atomic, meaning that they cannot be further split into multiple simpler facts.\n \
   - Pronouns must be substituted by the corresponding nouns.\n \
   - Avoid redundant or overlapping facts, ensuring each fact is unique.\n \
   - Scene cuts, camera motions, and purely technical details are NOT facts, but gestures, facial expressions,
and actions that carry meaning ARE facts.\n \
   - Implicit aspects such as 'The video starts.' or 'The video ends.' are NOT facts.\n \
   - The number of facts extracted should dynamically depend on the complexity of the video description.
Extract as many key facts as necessary to fully capture the relevant information.\n\n \
   Please generate the response in the form of a Python dictionary string with keys \"facts\". Each fact must
include its index in the format:\n \
   {\n \
   \"fact_index\": index_number, \"fact\": \"copy the fact here\"\n \
   }\n\n \
   DO NOT PROVIDE ANY OTHER OUTPUT TEXT OR EXPLANATION. Only provide the VALID Python dictionary string.\nFor
example, your response should look like this: {{'facts': [{{'fact_index': 1, 'fact': 'Fact 1 description.'}},
{{'fact_index': 2, 'fact': 'Fact 2 description.'}}, ...]}}",
```

3.2.6 Example results

The following presents the results of applying the V2T pipeline to a test case involving the artificial concatenation of two independent video shots from the Wiener Video Rekorder[19] collection. The pipeline correctly identifies and segments them and subsequently generates detailed descriptions for each.

To illustrate the translation capabilities of the T2T service, an example output is also provided in Spanish. Additionally, extracted facts are presented to demonstrate post-processing functionality.

Table 2: Example test video with shot keyframes, V2T generated captions, and corresponding Spanish translations.

Video	SHOT1 (16.48 seconds)	SHOT2 (3.04 seconds)
Keyframes	 MUTITJULU PLEASE FOLLOW WALKING TRACKS TO ASSIST SOIL CONSERVATION MEASURES WITHIN THIS AREA 	
Prompt	`Describe the following in detail with a word limit of 50 words.`	
V2T output	`A group of tourists gathers around a signpost at the entrance of MUTITJULU, a natural landmark. They are dressed in casual attire suitable for outdoor activities, with some wearing hats for sun protection. The signpost, made of wood, bears the name of the location and advises visitors to follow walking tracks to aid soil conservation. The surrounding landscape is arid, with sparse vegetation and rocky terrain, indicative of a desert or semi-desert environment. The sky is clear, suggesting favorable weather conditions for outdoor exploration. (84 words)`	`A serene lake nestled in a rugged mountainous area, with a single sailboat gently floating on the calm waters. The surrounding cliffs are adorned with patches of greenery, and the sunlight bathes the scene in a warm glow, highlighting the natural beauty of the landscape. (19 words)`
T2T output: Spanish translation	`Un grupo de turistas se reúne alrededor de un poste indicador en la entrada de MUTITJULU, un lugar natural. Están vestidos con ropa casual adecuada para actividades al aire libre, y algunos llevan sombreros para protegerse del sol. El poste indicador, hecho de madera, lleva el nombre del lugar y aconseja a los visitantes que sigan los senderos para ayudar a la conservación del suelo. El paisaje circundante es árido, con una vegetación escasa y un terreno rocoso, indicativo de un desierto o un entorno semi-desértico. El cielo está despejado, lo que sugiere condiciones climáticas favorables para la exploración al aire libre.`	`Un lago sereno se encuentra en una zona montañosa abrupta, con un solo barco a vela flotando suavemente en las aguas tranquilas. Las rocas circundantes están adornadas con parches de vegetación, y la luz solar baña la escena con una cálida luz, resaltando la belleza natural del paisaje.`

[19] https://www.mediathek.at/wiener-videorekorder/english-information

Extracted facts for SHOT1:

Fact Index	Fact
1	A group of tourists gathers at the entrance of MUTITJULU.
2	The tourists are dressed in casual attire suitable for outdoor activities.
3	Some tourists are wearing hats for sun protection.
4	The signpost is made of wood.
5	The signpost bears the name of the location MUTITJULU.
6	The signpost advises visitors to follow walking tracks.
7	The signpost advises visitors to follow walking tracks to aid soil conservation.
8	The surrounding landscape is arid.
9	The surrounding landscape has sparse vegetation.
10	The surrounding landscape has rocky terrain.
11	The surrounding landscape is indicative of a desert or semi-desert environment.
12	The sky is clear.
13	The clear sky suggests favourable weather conditions for outdoor exploration.

3.3 Novel content detection in news

Determining whether a document is novel or contains some novel parts w.r.t. another document can be seen as a problem of comparing embeddings, as question answering between the documents, or as checking claims extracted from one document vs. another or a collection. NN-based novelty detection on document level can be approached using sentence embeddings in a vector space. As this may not fully capture semantics of the whole documents, this is complemented with attention to determine document-level novelty[20]. A dataset for testing entailment of propositions has been provided[21], and the authors test on baseline approaches, including seq2seq, BERT embeddings and question answering.

Framing the novelty detection problem within a question-answering paradigm effectively leverages recent advancement in LLMs and QA systems. The Peak across[22] methodology exemplifies this approach, utilising question extraction from a source document to identify cross-document relationships within a collection. This technique builds upon two established foundations: The Qasem method for extracting question-answer pairs from unstructured text[23] and advanced contextualisation techniques[24] for refined question formulation. The

[20] Ghosal, Tirthankar, et al. "Is your document novel? Let attention guide you. An attention-based model for document-level novelty detection." Natural Language Engineering 27.4 (2021): 427-454.
[21] S. Chen, H. Li, Q. Wang, Z. Zhao, M. Sun, X. Zhu, and J. Liu, "Vast: A vision-audio-subtitle-text omni-modality foundation model and dataset," Advances in Neural Information Processing Systems, vol. 36, pp. 72 842–72 866, 2023.
[22] Caciularu, Avi, et al. "Peek Across: Improving Multi-Document Modeling via Cross-Document Question-Answering." The 61st Annual Meeting of the Association for Computational Linguistics (2023).
[23] Klein, Ayal, et al. "QASem Parsing: Text-to-text Modeling of QA-based Semantics." *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 2022.
[24] Pyatkin, Valentina, et al. "Asking It All: Generating Contextualized Questions for any Semantic Role." *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021.

implementation involves pretraining a multi-document model (QAMDen) on the NewsHead dataset[25], and task-specific fine-tuning for targeted question-answering applications. Beyond this core methodology, supplementary QA frameworks demonstrate versatility by enabling multi-step reasoning chains for comprehensive verification[26] and conducting gap analysis between generated answers and reference materials to identify information deficiencies[27].

A claim-based methodology identifies novel content by extracting assertions from a document and verifying them against reference sources. Current LLM-driven verification strategies – surveyed comprehensively[28] – encompass evidence retrieval from corpora, prompt engineering for optimised verification, transfer learning for domain adaptation, or evidence sentence generation. Notable implementations include Kao et al.'s end-to-end framework[29], multi-step prompting for enhanced verification accuracy[30], and DistilFEVERen, RAI's fine-tuned DistlBERT[31] model for claim refutation (English/Italian). A critical limitation persists: these approaches presuppose pre-extracted claims, lacking integrated methods for automated claim extraction from source documents.

Due to advances in LLMs and synergies with other tasks, we changed our original plan to base the approach on question answering but decided to post it as a problem of analysing overlapping facts.

The basic idea of the metric is to extract single sentence statements contained in a text T^* ("facts") and check them against another text T_k. We then investigate two approaches for comparing them: one is similar to the approach used in claim checking/refutation, while the other directly applies an LLM for labelling facts as novel. The fact extraction is model $\boldsymbol{M}_{fex}(T) \rightarrow F$, where F is the set of facts, $\boldsymbol{F} = \{f_1, \dots, f_l\}$ is implemented as a prompt for an LLM. The LLM is prompted to identify atomic factual *statements* from paragraph-length input. Extracted facts must be self-contained, including subject, predicate, and object, and must retain linguistic modifiers such as modality, negation, and quantification.

The system uses the Llama 3.1 8B[32, 33] model with deterministic hyperparameters and a high token limit to preserve context integrity. Smaller models (e.g., 3B variants) showed deficiencies in meaningful fact extraction.

[25] Gu, Xiaotao, et al. "Generating representative headlines for news stories." *Proceedings of The Web Conference 2020*. 2020.
[26] R. Aly, M. Strong, and A. Vlachos, "Qa-natver: Question answering for natural logic-based fact verification," in The 2023 Conference on Empirical Methods in Natural Language Processing, 2023.
[27] R. Rabin, A. Djerbetian, R. Engelberg, L. Hackmon, G. Elidan, R. Tsarfaty, and A. Globerson, "Covering uncommon ground: Gap-focused question generation for answer assessment," in The 61st Annual Meeting Of The Association For Computational Linguistics, 2023.
[28] A. Dmonte, R. Oruche, M. Zampieri, P. Calyam, and I. Augenstein, "Claim verification in the age of large language models: A survey," arXiv preprint arXiv:2408.14317, 2024.
[29] W.-Y. Kao and A.-Z. Yen, "How we refute claims: Automatic fact-checking through flaw identification and explanation," in Companion Proceedings of the ACM Web Conference 2024, 2024, pp. 758–761.
[30] X. Zhang and W. Gao, "Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method," arXiv preprint arXiv:2310.00305, 2023.
[31] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.
[32] A. Grattafiori and et al., "The llama 3 herd of models," 2024. Available: https://arxiv.org/abs/2407.21783
[33] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

Prompts were iteratively refined for robustness; the final versions are documented in the GitHub repository of the dataset.

3.3.1 Fact entailment or contradiction

Given the facts extracted F^*, text T_k may support these facts (*entailment*), be in conflict with them (*contradiction*), or not contain information related to this statement (*neutral*). The checking model $M_{chk}(T, F) \rightarrow (E, C, N)$, where E, C and N are binary vectors of size l, encodes for each fact whether it is considered entailed, contradicting or neutral $|E| + |C| + |N| = I$, where $|\cdot|$ denotes the L_1 norm. We thus obtain $(E_{T \rightarrow T_k}, C_{T \rightarrow T_k} N_{T \rightarrow T_k}) = M_{chk}(T_k, M_{fex}(T^*))$.

Facts that are entailed in either matching direction are clearly not novel. Facts contained in the text T_k from the collection and found to be neutral are prior knowledge not related to the new text, and thus also not novel. Novelty is thus based on the neutral facts in the new text (as no related information could be found in the text from the collection), and contradicting facts in both directions (as a contradiction implies new information):

$$v_k = \frac{\left|N_{T^* \rightarrow T_k}\right| + \left|C_{T^* \rightarrow T_k}\right| + \left|C_{T_k \rightarrow T^*}\right|}{l^* + \left|C_{T_k \rightarrow T^*}\right|}$$

The checking model is also implemented as a prompt for Llama 3.1 8B. Each fact is evaluated for entailment, contradiction, or neutrality with respect to a comparison paragraph. This step is performed collectively: the entire fact set is assessed against the full reference or hypothesis text, rather than individually. If parsing fails due to syntactic inconsistencies, an auxiliary LLM call attempts structural correction. Manual inspection was occasionally needed for edge cases, such as removing spurious quotation marks around named entities.

In order to obtain an aggregated score for the collection of text, we consider facts that are either supported by all texts in the collection, or that contradict at least one text within it.

$$N_{\mathcal{T}} = \frac{-1}{n} \sum_{k=1}^{n} N_{T^* \rightarrow T_k}$$

$$C_{\mathcal{T}} = \frac{1}{n} \sum_{k=1}^{n} C_{T^* \rightarrow T_k}$$

$$v_{\mathcal{T}} = \frac{\sum_{j=1}^{l} - \left\lceil N_{T^* \rightarrow T_{k,j}} \right\rceil + \sum_{j=1}^{l} \left\lceil C_{T^* \rightarrow T_{k,j}} \right\rceil + \frac{1}{n} \sum_{k=1}^{n} \left|C_{T_k \rightarrow T^*}\right|}{l + \frac{1}{n} \sum_{k=1}^{n} \left|C_{T_k \rightarrow T^*}\right|}$$

3.3.2 Fact novelty

The alternative approach is to define a novelty model $M_{nov}(T,F) \rightarrow (K,U)$ where K and U are binary vectors of size l, encoding for each fact whether it is considered known or unknown ($|K| + |U| = l$). We thus obtain $\left(K_{T\rightarrow T_k}, U_{T\rightarrow T_k}\right) = M_{nov}(T_k, M_{fex}(T^*))$. The checking model is also implemented as a prompt for Llama 3.1 8B.

We determine the pairwise novelty as the maximum of the fraction of novel facts in both directions:

$$v_k = \max\left(\frac{|U^*|}{l^*}, \frac{U_k}{l_k}\right)$$

The aggregation over the collection considers whether a fact is found to be novel to *all* texts in the collection, i.e.

$$U_{\mathcal{T}} = \frac{-1}{n}\sum_{k=1}^{n} U_{T^*\rightarrow T_k}$$

$$v_{\mathcal{T}} = \frac{\sum_{j=1}^{n} - \left[N_{T^*\rightarrow T_{k,j}}\right]}{l}$$

We first provide results on the stability of the fact extraction and matching approach, based on work and a dataset (FM-V2T) from the FAIRmedia project[34]. In order to assess the reliability of the metric, we test the metric on self-matching the references in the FM-V2T dataset. This should result in entailment close to 1, and contradictions close to 0. We perform the experiment twice, matching forward and backward, in order to test the reproducibility of the models. In addition, we repeat the same experiment with self-matching the output of InternLM-XComposer-2.5[35] on the FM-V2T dataset. Figure 1 shows the results, indicating that the metric is very reliable in terms of contradictions, which are almost 0. The rate of entailments is very high, though with more outliers, i.e. captions resulting in lower entailment rates. This means that support for some extracted statements could not be verified, and they are thus considered neutral. We also match the

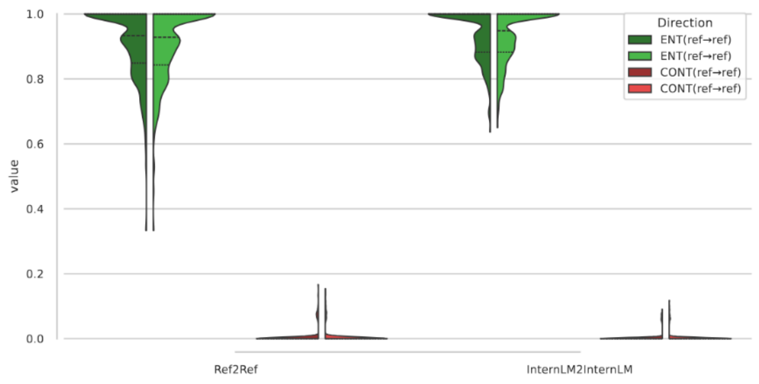


Figure 1: Bidirectional validation of LLM metric for entailment (ENT, green) and contradiction (CONT, red): Across matching direction (left/right part of each subplot), as well as across the (a) reference-against-reference and (b) InternLM-to-InternLM6 output.

[34] https://www.joanneum.at/digital/en/projects/fairmedia/

[35] Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, et al. 2024. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. arXiv preprint arXiv:2407.03320 (2024).

InternLM[35] video-to-text model output against itself, which turns out to be even more reliable, with a lower number of outliers. This is probably due to simpler and shorter nature of the outputs compared to the references.

There are few multimedia datasets which are suitable for the task. We identified the FIVR200K dataset[36], created by the H2020 InVID project[37], as a relevant one. It is a collection of YouTube videos related to different news events. For parts of the dataset, annotations of four categories of relations between videos are provided: near duplicate video(i.e., at least one scene overlaps), , duplicate scene videos (i.e., all scenes overlap) complementary scene videos (i.e., capturing the same scene of the event, but with complementary content), and incident scene videos (i.e., related to the same scene, but not capturing overlapping content). Unfortunately, not all of the videos are still available on Youtube, and if one video used in an annotated pair is not available, this makes the pair unusable.

3.4 News content tagging

NewsTagger is a service developed for automatically tagging news content. It is a fine-tuned variant of Llama-3.1-8B-Instruct[38] (open-sourced and developed by Meta), adapted specifically for the task of assigning a set of semantically meaningful tags to news articles. It is useful for supporting downstream tasks such as content indexing, retrieval, and thematic categorization in a scalable and domain-adaptable manner.

This component relates to the requirements FR.4.2 ("The Neural Media Repository MUST enable efficient context-based search and retrieval"), FR.57.2 ("The Data Sharing Platform MUST provide the functionality to provide content-based and source-based filtering"), and FR.180.1 "The Data Sharing Platform MUST provide the possibility of applying a tag (or a label) to a newly created asset" of the XReco system.

Recent advances in NLP, especially with the rise of LLMs, have fundamentally changed how text classification and annotation tasks are approached. LLMs, pre-trained on vast multilingual datasets and fine-tuned for specific applications, now deliver strong results across tasks such as summarisation, headline generation, and tagging.

In the media and journalism domain, research shows that while large, resource-intensive models can achieve high performance for tasks like automatic headline generation[39], smaller models – when fine-tuned for a particular task – can match the results of their larger counterparts[40]. This insight underpins our NewsTagger approach, which leverages fine-tuned, smaller LLMs to automatically assign semantic tags to news articles. Tagging, much like headline generation, requires a deep semantic understanding of the text, but does not always need the full general-purpose capabilities of the largest models.

[36] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and I. Kompatsiaris, "FIVR: Fine-grained incident video retrieval," IEEE Transactions on Multimedia, vol. 21, no. 10, pp. 2638–2652, 2019.
[37] https://cordis.europa.eu/project/id/687786
[38] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct (last accessed May 29th, 2025)
[39] Gu, X., Mao, Y., Han, J., Liu, J., Wu, Y., Yu, C., Finnie, D., Yu, H., Zhai, J. and Zukoski, N. *Generating representative headlines for news stories.* In: *Proceedings of The Web Conference 2020*, 2020, pp. 1773–1784.
[40] Scotta, Stefano and Alberto Messina. *Experimenting task-specific LLMs.* In: *Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI 2023) co-located with the 22nd International Conference of the Italian Association for Artificial Intelligence (AI*IA 2023)*. 2023.

Building on prior work in headline generation, NewsTagger adapts these methods for multilingual semantic tagging, evaluating whether fine-tuned, compact LLMs can serve as efficient alternatives to massive, general-purpose models. By targeting real-world media content in languages such as Italian, English, Slovenian, and German, NewsTagger also contributes to the underexplored area of multilingual, domain-specific LLM applications. Thanks to the instruction-following and multilingual strengths of models like Llama-3.1-8B-Instruct, a single fine-tuning process can yield a tagging system that generalises across languages, reducing the need for separate models and making the approach both efficient and scalable.

3.4.1 Model fine-tuning procedure

To fine tune the base model Llama-3.1-8B-Instruct, we collected a dataset from publicly accessible RSS feeds in the four languages considered, filtering those with either no text or no tags, and spanning from January 2023 to July 2024. The resultant dataset totalled approximately 31,000 RSS items, with about 11,000, 9,000, 10,000, and 1,000 for Italian, German, English, and Slovenian, respectively.

The fine-tuning process of the base model Llama-3.1-8B-Instruct was performed after the 8-bit quantization of the model using the LoRA technique to enable efficient training of low-rank adapters, significantly reducing memory and computational requirements while preserving model performance.[41] The whole fine-tuning process took approximately 60 hours of training on an on-premises NVIDIA A40 40GB GPU. The prompts fed to the model are detailed in Annex I - Section 11.1.

3.4.2 Experimental results

In this section, we present an analysis of the performance of the fine-tuned NewsTagger model. We compare it against three different models: gpt4-o[42] from OpenAI, which is one of the best commercial models available, Llama-3.1-70B-Instruct[43], which is the best model of the LLama 3.1 generation, and Llama-3.1-8B-Instruct, which is the base model of the NewsTagger, to check the improvements made by the fine-tuning process.

All models, including the NewsTagger, were appropriately prompted to assign semantic tags to the RSS items. The quality of the tags generated by each model was then evaluated against the ground truth, i.e., tags assigned by professionals, as originally posted in published RSS feeds. The analysis was conducted on a test set of 5,069 RSS items, comprising 1,485 in Italian, 1,500 in German, 1,500 in English, and 584 in Slovenian. These RSS items were collected from the same sources that were used for the training, but they were published after July 2024. This allowed us to test the model's ability to generalise to unseen, real-world data outside of the fine-tuning dataset. The quality of the tags generated by the models was evaluated according to the following measures:

Intersection over Union (IoU): defined, for each RSS item and model, as the number of tags generated by the model that are also present in the ground truth, over the total number of unique tags from both the model and the ground truth. Formally, fixed a model, let M be the set of tags generated by the model and G the set of ground truth tags, the Intersection over Union is defined as:

[41] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W., 2022. Lora: Low-rank adaptation of large language models. ICLR, 1(2)
[42] https://openai.com/index/hello-gpt-4o/ (last accessed May 30th, 2025)
[43] https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct (last accessed May 30th, 2025)

$$IoU := \frac{|M \cap G|}{|M \cup G|}$$

For example, if the ground truth tags for an RSS item are $\{x, y, z\}$, and the model generates $\{x, y, w\}$, then $IoU = 0.5$.

Average Semantic Similarity (S): this quantity measures the average semantic similarity between the list of generated tags and the list of ground truth tags. For each RSS item and model, let M be the list of generated tags and G the list of ground truth tags. We build a similarity matrix Φ, where the rows correspond to the tags in M and the columns to the tags in G. The element on the i-th row and j-th column of Φ is given by the Cosine Similarity of the embeddings, computed using the sentence transformer model all-MiniLM-L6-v2[44], of the i-th tag in M and of the j-th in G. The average semantic similarity measure is defined as:

$$S = \frac{1}{2}(S_{row} + S_{col}),$$

where S_{row} and S_{col} are the mean value of the maximum values of each row and each column of Φ, respectively.

Figure 2 and Figure 3 show the average values of the previously defined metrics on the test set, respectively. For each model, the averages were computed both per language and over the entire test set, regardless of the language of the RSS items. The results clearly demonstrate the effectiveness of the NewsTagger model. Both in terms of S and IoU, the NewsTagger outperforms all the alternative models considered, demonstrating that domain-specific fine-tuning, even on smaller language models, is a viable alternative to relying on larger/expensive LLMs.

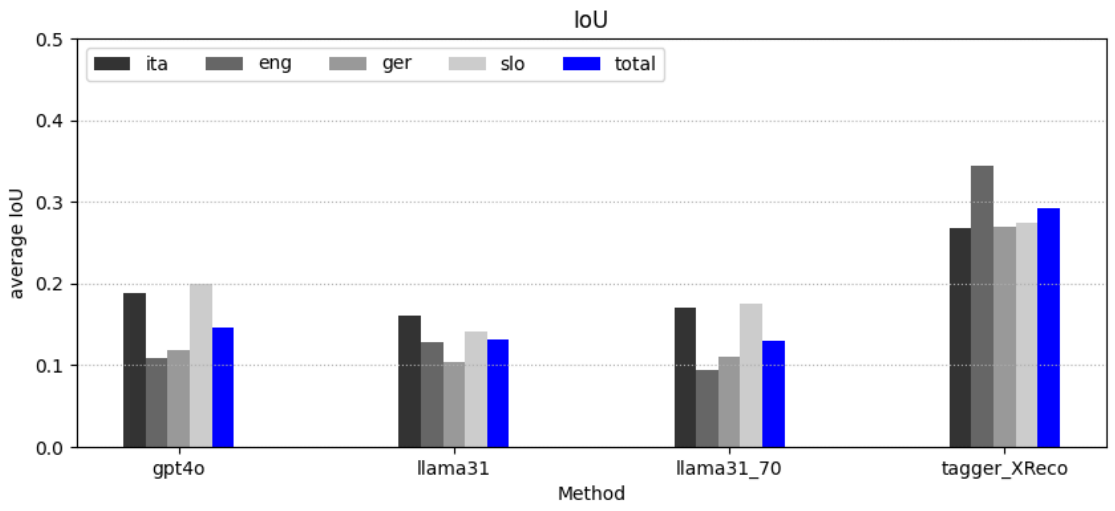


Figure 2: Average IoU over the tags generated by each model for the RSS items in the test set, for individual languages (ita, eng, ger, slo) and aggregated (total).

[44] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 (last accessed May 30th, 2025)

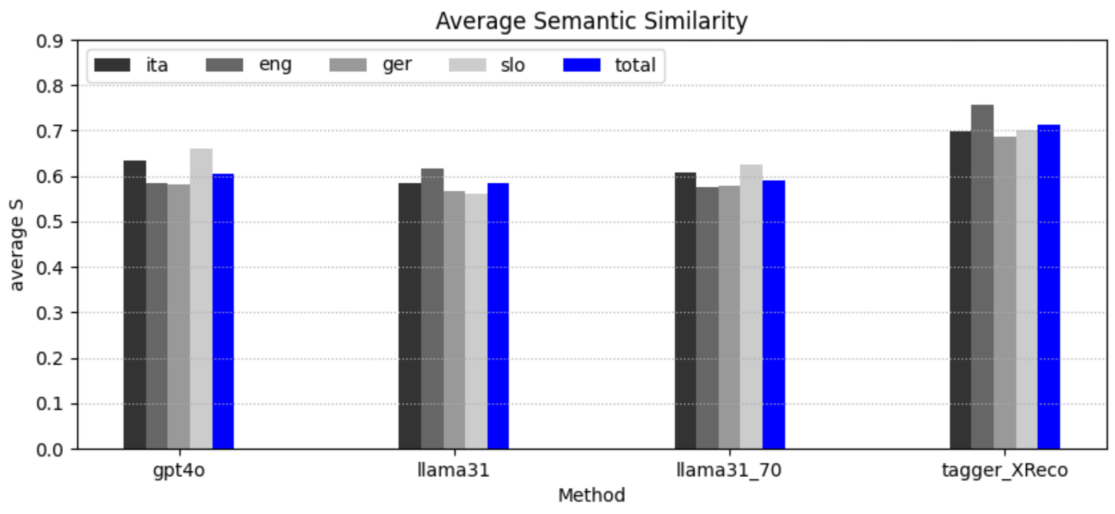


Figure 3: Average S value over the tags generated by each model for the RSS items in the test set for individual languages (ita, eng, ger, slo) and aggregated (total).

The main limitation of this analysis lies in the fact that the only ground truth available is the set of tags assigned by the publishers of the RSS feeds. This does not exclude the possibility that, in some cases, the tags generated by one of the models may be as good as - or even better (i.e., more informative) than - the original ones. Therefore, the conclusion should highlight not that NewsTagger's tags are necessarily better than those of the other models, but rather that they are more similar to the reference tags assigned by journalists.

3.4.3 Deployment and integration in the XReco platform

The NewsTagger service is packaged and distributed as a Docker image and runs as a container on a Docker host. The service is exposed through REST API implemented using the FastAPI framework[45] and served using Uvicorn.[46] The generation of tags is invoked by NMR by calling the */tags* endpoint of the NewsTagger service. Given an item uploaded to the platform, such as a news article, an image, a video or a 3D model, the tags are generated combining the title, the subtitle and the description of the uploaded item. It is not mandatory to specify all three, but at least one of them.

3.5 Mixed Reality User Interface

As the boundaries between physical and digital environments continue to blur, creating seamless workflows for multimedia retrieval and content authoring in Mixed Reality (MR) is becoming increasingly essential. To address this, we have developed a Mixed Reality Multimedia Retrieval Smartphone App that bridges intuitive, real-world interactions with content search on mobile devices.

This app is adapted from our earlier prototype (MR)² and has been extended and tailored to match the technological infrastructure and requirements of the XReco project[47]. It enables mobile users to capture their

[45] https://fastapi.tiangolo.com/ (last accessed May 30th, 2025)
[46] https://www.uvicorn.org/ (last accessed May 30th, 2025)
[47] https://xreco.eu/mixed-reality-user-interface/

environment, perform contextual queries, and retrieve reusable media assets in real time. The app is designed for tight integration with authoring environments, especially the Unity-based ZAUBAR Authoring Tool, enabling direct content flow from the physical world into XR experience design.

3.5.1 App Functionality

The Mixed Reality Multimedia Retrieval Smartphone App supports multiple query modalities and live interaction features:

- Live object detection using integrated YOLOv12 models[48] to enable context-aware, real-world object-based querying.
- Manual region selection for visual queries within the live camera feed that extend a single detected object.
- Text queries entered by the user are processed as full-text search queries, similar to those sent from the XReco Marketplace.
- Visual result presentation in a scrollable grid or as an augmented overlay, enabling instant browsing and reuse.

All queries are executed in real-time. Depending on the modality, features are extracted either directly on the device or through backend services and matched against collections utilising the NMR-backend.

3.5.2 Placement in XReco stack

The app enables multimodal queries based on object detection, region selection, or text input, which are sent to the NMR-backend for similarity-based multimedia retrieval. All query modalities provided in the Marketplace are available in the smartphone app. Query features are processed and matched using a PostgreSQL database with pgVector[49]. The results are returned to the mobile interface and can be exported directly to the Unity-based ZAUBAR Authoring Tool. The architecture supports that content ingested via the orchestrator also becomes directly available for the MR search.
The architecture and the connection to other XReco services is visible in Figure 4.

[48] Tian, Y., Ye, Q., & Doermann, D. (2025). Yolov12: Attention-centric real-time object detectors. arXiv preprint arXiv:2502.12524.
[49] https://github.com/pgvector/pgvector

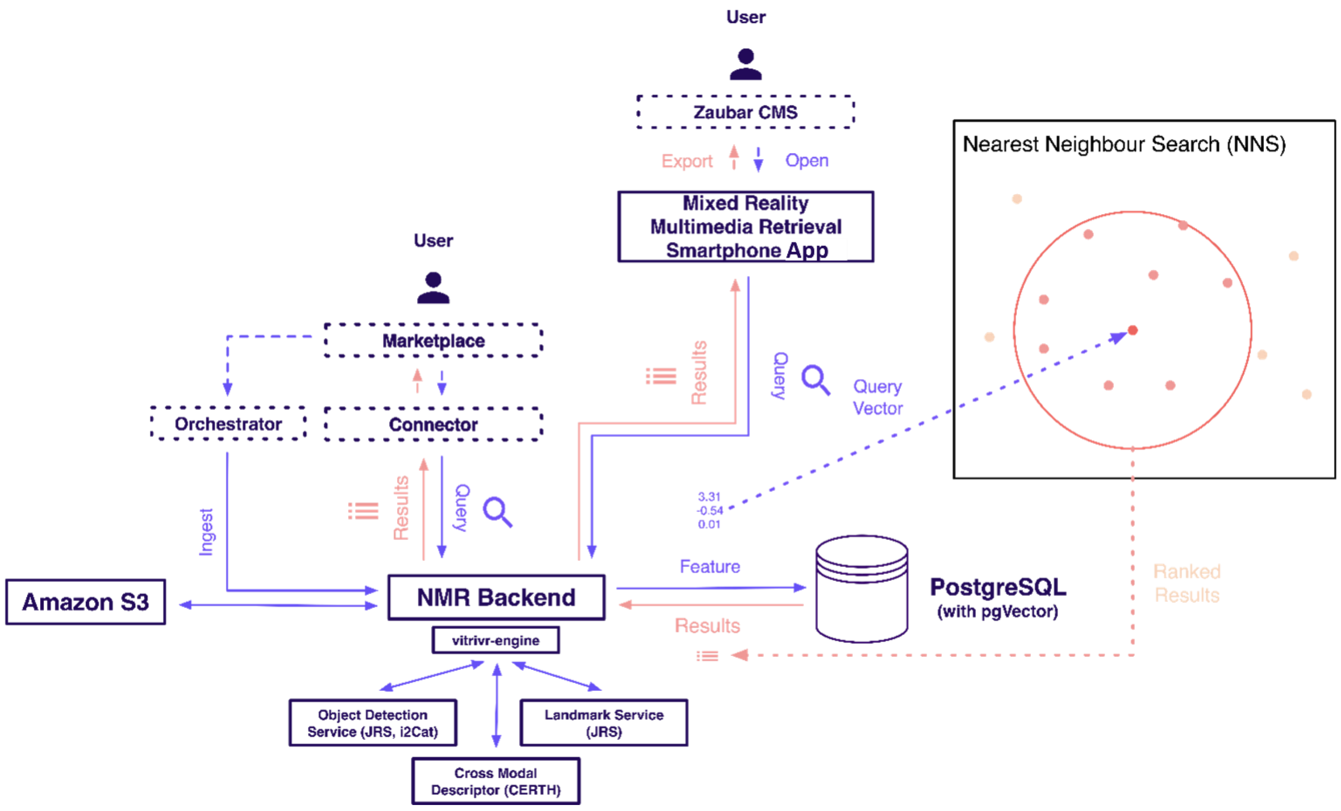


Figure 4: System architecture of the Mixed Reality Multimedia Retrieval smartphone app integrated into the XReco platform.

3.5.3 Backend Infrastructure

The app communicates with the NMR-backend, reusing the same API endpoints employed by the Marketplace within XReco, thereby ensuring full compatibility with existing pipelines.
Key backend features include:

- An additional endpoint for similarity search based on captured images, supporting object-based queries from the live retrieval interface.
- Feature extraction services powered by CLIP, enabling robust cross-modal queries across images and text.
- A pgVector-enhanced PostgreSQL database, allowing for efficient and scalable similarity search across large collections.
- Optional integration with third-party services, such as:
  - Object detection models (e.g., from JRS and i2Cat)
  - Landmark recognition systems
  - Cross-modal descriptor providers (e.g., CERTH)
- All descriptor handling and result ranking is managed via the NRM-backend built upon vitrivr-engine.

3.5.4 Integration into XReco via ZAUBAR Authoring Tool

To fully support content reuse and authoring workflows, the smartphone app is integrated with the ZAUBAR Authoring Tool, a Unity-based content creation platform.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

This integration enables a bidirectional workflow:

- From authoring to retrieval: The authoring tool can directly launch the smartphone app, allowing content creators to perform mobile retrieval sessions in the field.
- From retrieval to authoring: Users can select assets from search results and send them back to the Unity authoring interface, where they are embedded as linkable, reusable objects (via URL-based asset references).

This integration supports a mobile-first authoring approach: creators are no longer limited to curating content from office desktops but can instead collect, retrieve, and reuse material live on location, making XR experience creation faster and easily connected to the physical context.

4 Scene Reconstruction Services

This section presents the core 3D reconstruction technologies developed within XReco, encompassing both neural rendering approaches and more traditional computer vision methodologies. These services have evolved significantly from their initial implementations in D4.1, now achieving production-ready status with enhanced performance, quality and integration capabilities.

4.1 Neural Rendering

4.1.1 Fast NeRF in-the-wild

Building upon the foundational work established in D4.1, where an adapted version of NeRF-in-the-wild (NeRFw)[50] was implemented, this deliverable presents significant advancements designed to enhance both performance and reconstruction quality in unconstrained, real-world scenarios. These improvements directly address the slow training convergence limitations of NeRFw identified in the previous deliverable.

To overcome these challenges, D4.2 introduces two major architectural enhancements: a hierarchical octree feature volume representation for improved spatial efficiency, and an advanced appearance and transient object handling mechanism inspired by Hallucinated-NeRF (HA-NeRF)[51]. These innovations collectively enable faster convergence while maintaining high-quality reconstruction capabilities for practical deployment within the XReco ecosystem. An overview of our pipeline is presented in Figure 5: Fast NeRF in-the-wild pipeline.Figure 5.

[50] Martin-Brualla, R., Radwan, N., Sajjadi, M. S., Barron, J. T., Dosovitskiy, A., & Duckworth, D. (2021). Nerf in the wild: Neural radiance fields for unconstrained photo collections. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 7210-7219).

[51] Chen, X., Zhang, Q., Li, X., Chen, Y., Feng, Y., Wang, X., & Wang, J. (2022). Hallucinated neural radiance fields in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12943-12952).

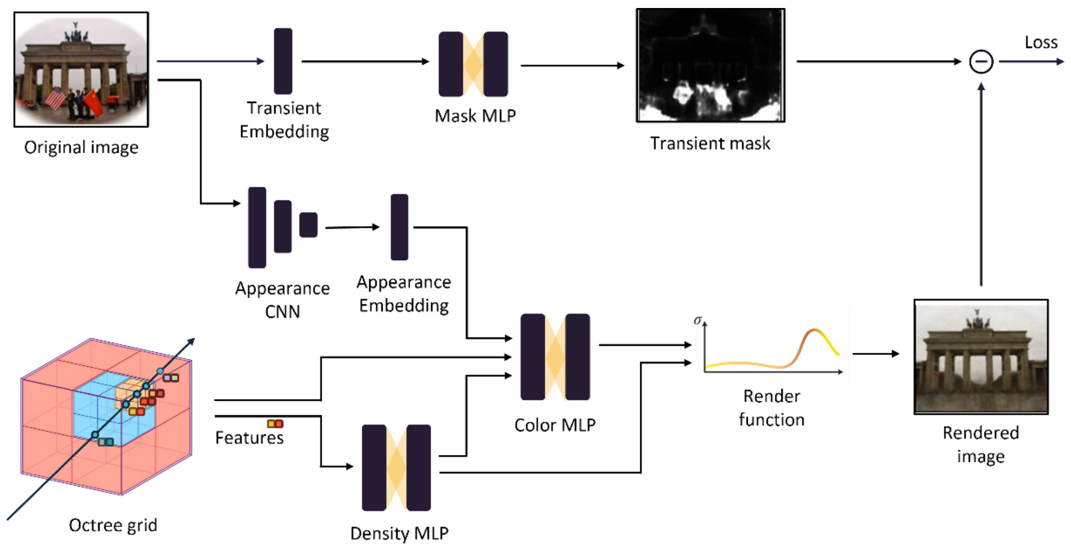


Figure 5: Fast NeRF in-the-wild pipeline.

Octree feature volume: To optimise spatial sampling and drastically reduce computational overhead, we implemented our approach utilising an octree-based spatial data structure. An octree efficiently represents sparse volumetric scenes by allocating high resolution information only where needed (i.e., in areas with significant scene detail). This approach reduces unnecessary computation in empty or uniform regions, leading to improved rendering and training performance. The use of the octree structure is particularly valuable in the XReco context, where large, aggregated datasets from multiple sources must be processed efficiently. Kaolin's[52] integration with PyTorch allowed for seamless integration with our previously developed pipelines, resulting in a system that significantly improves memory usage and rendering times while maintaining high reconstruction quality.

Integration of appearance and transient embeddings from Ha-NeRF: While D4.1 leveraged NeRFw-style appearance and transient embeddings, our current implementation adopts a more robust mechanism to better disentangle scene geometry from variable appearance and transient phenomena. The NeRFw approach is limited as it requires optimising embeddings for every training image, hindering its ability to generalise to novel, unseen image samples. Ha-NeRF overcomes this with two key components:

- A convolutional encoder which learns a disentangled appearance representation, mapping each input image to a latent vector that captures its appearance due to different illumination. This architecture allows the model to render the scene with appearances from images outside the original training set, enabling true appearance interpolation. To ensure the appearance is fully separated from scene geometry, a view-consistent loss prevents geometric details from being encoded into the appearance vector.
- Instead of modelling transient objects with a computationally expensive 3D field as in NeRFw, HA-NeRF utilises a streamlined 2D visibility map. This map, modelled by an implicit function (via an MLP), predicts the probability that each pixel belongs to the static background versus the transient foreground. It is

[52] https://developer.nvidia.com/kaolin

trained in an unsupervised manner, which effectively learns to separate the static scene from dynamic elements, resulting in cleaner reconstructions.

These advancements in appearance disentanglement and transient handling further align our NeRF pipeline with XReco's broader goals of unified, cross-repository content aggregation and semantic search in 3D environments.

4.1.1.1 Results and observations

To validate the performance of our enhanced NeRF pipeline, we conducted a series of experiments comparing our method against the mentioned baselines for in-the-wild scene reconstruction: NeRFw and Ha-NeRF. The evaluation focuses on two primary aspects: reconstruction quality, measured by standard image-based metrics, and computational efficiency, measured by total training time.

Experimental Setup: The evaluation was performed on the Phototourism dataset[53], which is used widely in the literature in relevant tasks. It includes four scenes, Brandenburg Gate, Trevi Fountain, Taj Mahal, and Sacre Coeur, which are characterised by significant variations in lighting, camera parameters, and transient occluders.

Metrics: We use three standard image-based metrics to assess reconstruction quality via rendering: PSNR, SSIM, and LPIPS.

Energy efficiency: For a training run of 300,000 iterations, our method consumed 2.59 kWh. In comparison, NeRFw and Ha-NeRF methods consumed 9.75 kWh and 7.71 kWh, respectively. Fast-NeRF in-the-wild demonstrates a significant improvement in energy efficiency, consuming roughly one-third of the GPU power required by comparable techniques.

Implementation details: Our model was trained on an NVIDIA A10G 24GB GPU running on AWS g5.4xlarge EC2 instances. The core architecture integrates the octree representation with the appearance and transient modelling components from HA-NeRF, as described above. The quantitative results, summarized in Table 3, demonstrate that our method achieves a significant improvement in computational efficiency without compromising reconstruction quality.

Table 3: Quantitative results comparing different NeRF-in-the-wild methodologies on the Phototourism dataset.

Methods	Brandenburg Gate			Sacre Coeur			Trevi Fountain			Taj Mahal		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ha-NeRF	20.04	0.887	**0.139**	20.02	0.801	0.171	20.18	0.69	0.222	20.84	0.826	0.249
NeRFw	**23.45**	0.811	0.247	**25.4**	**0.87**	**0.124**	22.15	0.69	**0.117**	22.72	0.767	0.301
Octree-NeRF (ours)	23.15	**0.902**	0.238	22.23	0.77	0.165	**23.39**	**0.77**	0.241	**25.73**	**0.849**	**0.163**

As shown Table 3, our method delivers PSNR, LPIPS and SSIM scores that are on par with, or superior to, both NeRF-w and HA-NeRF. This confirms that the integration of the octree structure does not negatively impact the model's ability to reconstruct scenes with high fidelity. The most significant result is the dramatic reduction in training time. By leveraging an octree to focus computation only on occupied space, our method converges **8 times faster** than the baselines. We ran these experiments in the same machine for 300k iterations each. While

[53] Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: exploring photo collections in 3D. In ACM siggraph 2006 papers (pp. 835-846).

both NeRFw and Ha-NeRF need **48 hours** for convergence, our method needs only **6 hours**. This rapid convergence is critical for the XReco project's goal of processing large-scale, multi-source datasets efficiently.

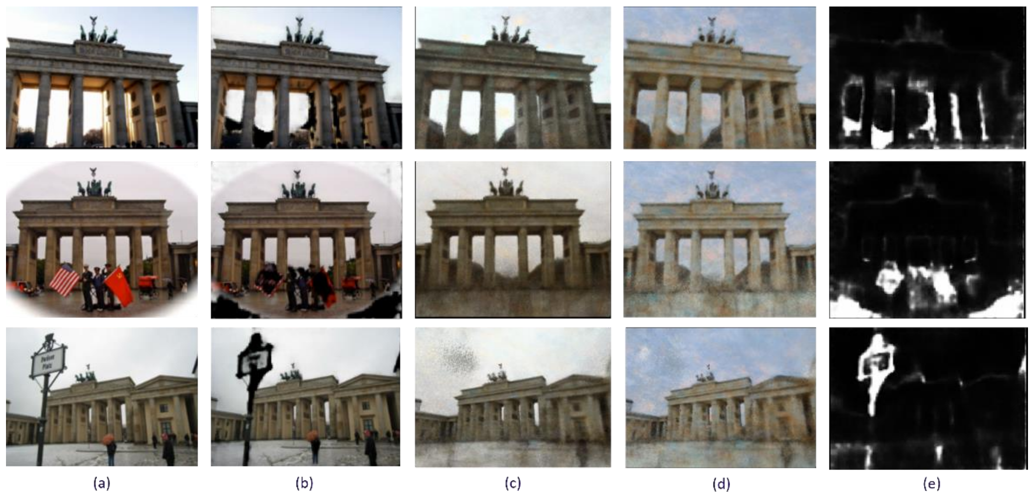


Figure 6: Qualitative results on Phototourism Brandenburg Gate scene. (a) The original image. (b) The original image with transiency subtracted. (c) The rendered Octree-based NeRF from the same viewpoint with same appearance. (d) The rendered Octree-based NeRF from the same viewpoint with a different appearance. (e) The estimated transiency mask.

Qualitative results also underscore the effectiveness of our approach. Figure 6 presents rendered novel views from the Branderburg Gate scene with different appearances, as well as the estimated transiency mask.

The experimental results validate that our proposed method successfully addresses the primary limitations of previous in-the-wild NeRF techniques. By combining the hierarchical octree representation for speed with an advanced architecture for appearance and transient modelling, we have developed a pipeline that is both fast and robust. The system delivers reconstruction quality on par with the state-of-the-art, while significantly reducing computational requirements, making it a highly effective and scalable solution for the demands of the XReco project.

4.1.2 3D Gaussian Splatting

Although Gaussian Splatting[54] (3DGS) does not constitute a purely neural reconstruction methodology – being fundamentally a gradient descent optimisation algorithm that optimises Gaussian parameters through multi-view RGB supervision – it is presented within this section due to its hybrid nature, bridging traditional computer vision techniques with neural rendering approaches. This classification reflects the method's integration of classical optimisation principles with modern differential rendering paradigms and novel 3D representations.

The developed service provides static scene reconstruction using 3DGS, a state-of-the-art technique for high-fidelity 3D reconstruction from multi-view images or video input. This service is implemented based on the open-

[54] Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph., 42(4), 139-1.

source gsplat[55] component from the NeRFStudio framework[56], which features a permissive license and delivers performance on par with the original 3DGS method. Users can upload images or video sequences of static scenes, which the service processes to produce photorealistic and spatially consistent 3D representations through 3D Gaussian primitives. Designed with a focus on efficiency, rendering quality, and broad compatibility with common input formats, this service offers a robust and scalable solution suitable for real-world applications and deployment.

The 3DGS reconstruction technology addresses diverse industry requirements for accurate 3D scene reconstruction. In digital heritage, institutions can preserve physical spaces and artefacts with high fidelity from standard photographs and videos (Figure 7, Figure 8). Architecture and real estate professionals can generate immersive 3D walkthroughs from visual captures, enhancing design validation and client engagement. The entertainment industry can reconstruct film sets and locations from footage, supporting virtual production and VFX workflows. For training applications, realistic 3D environments enable spatially accurate simulation experiences. Technical advantages include high rendering performance and compact representation, making this technology particularly valuable for AR/VR applications and on-device visualisation. The system's accessibility supports both professional deployment and research experimentation, facilitating rapid iteration across diverse use cases.

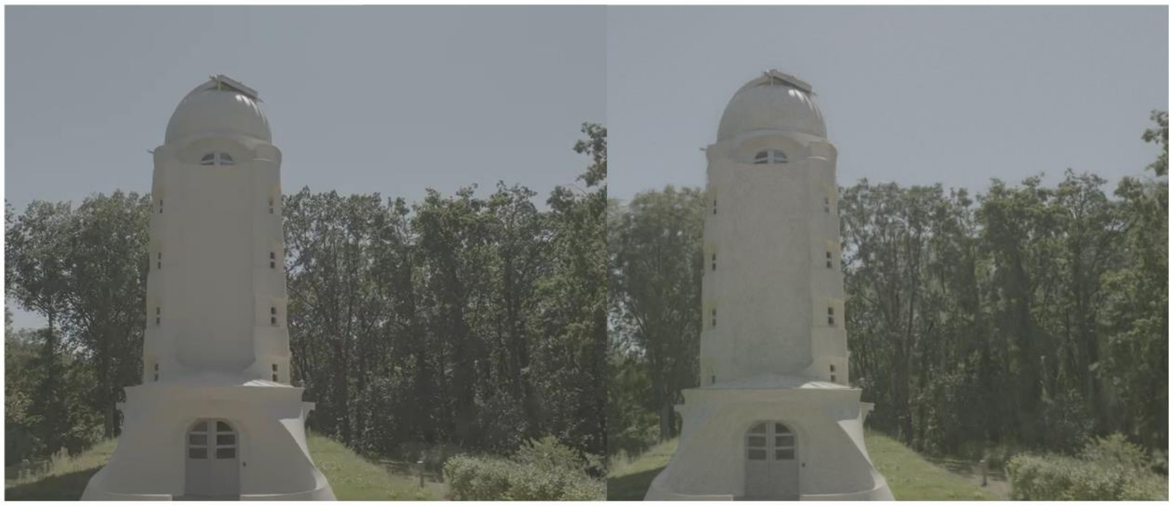


Figure 7: Qualitative results of our 3DGS pipeline on the Einstein Tower footage, captured by DW. (left) A frame from the original video footage. (right) The corresponding 3DGS render.

[55] https://github.com/nerfstudio-project/gsplat
[56] https://github.com/nerfstudio-project/nerfstudio

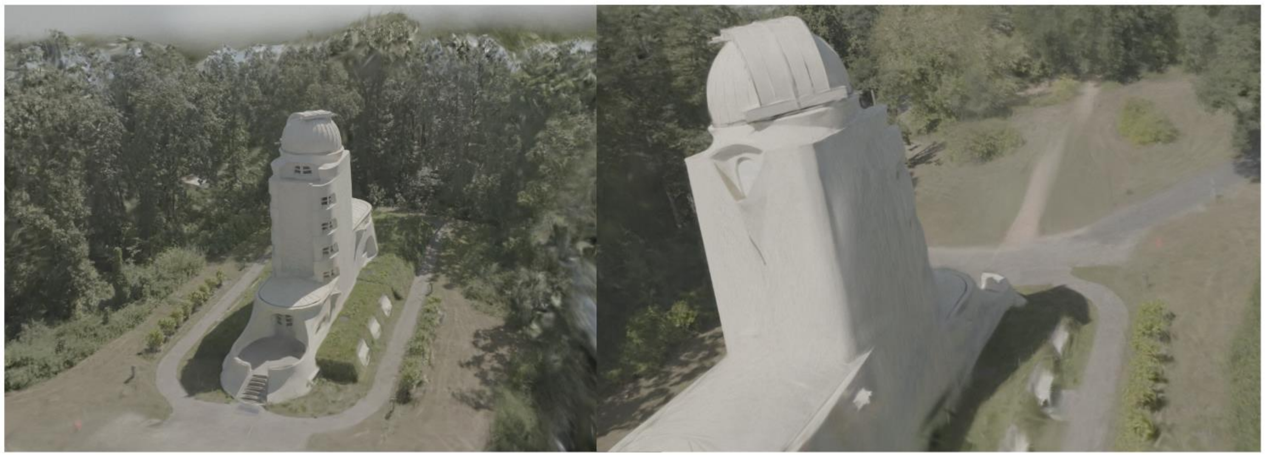


Figure 8: Qualitative results from 3DGS on the Einstein Tower footage captured by DW. Two renders from viewpoints outside the original video footage. Although some artifacts appear (especially at the horizon), the geometry and the level of detail remain consistent.

4.1.2.1 Implementation Overview

The delivered pipeline includes the following steps:

Input Handling: Supports image sequences and video files. Automated camera pose estimation (e.g., via COLMAP[57]) is included if external camera parameters are unavailable.

- **Data Preprocessing**: Frames are extracted, resized, and processed to ensure consistent quality. Camera intrinsics and extrinsics are validated or inferred.
- **Gaussian Splatting Reconstruction**: The core reconstruction engine generates a 3D point cloud with per-point anisotropic Gaussian distributions. These represent colour, opacity, and anisotropic scale, enabling high-quality rendering.
- **Output**: The result is a 3D representation viewable within a browser-based viewer or exportable for integration into downstream applications (e.g., Unity, Unreal Engine).

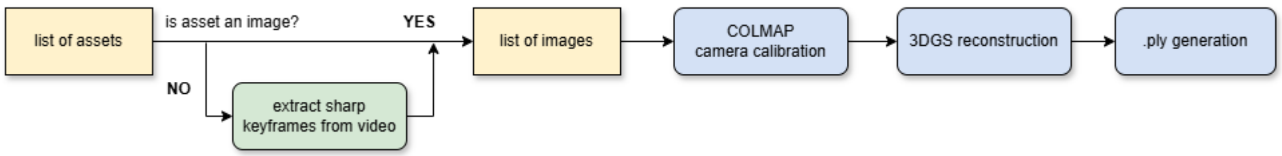


Figure 9: A schematic representation of the 3DGS reconstruction pipeline

The chosen implementation strikes a balance between licensing freedom (Apache License 2.0), performance, and practical usability, ensuring the reconstruction service can be deployed in a wide range of environments, including cloud-based and local setups.

[57] Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-Motion Revisited. Conference on Computer Vision and Pattern Recognition (CVPR).

The 3DGS reconstruction service is packaged as a Docker image and deployed as a container on a Docker-compatible host. It provides a RESTful API built with the FastAPI framework, offering a streamlined and high-performance interface for client interactions. A job tracking system, implemented using Celery[58], manages asynchronous reconstruction tasks, allowing clients to monitor job progress and query their status. Upon completion of a reconstruction task, the system can generate a download link to access the resulting data.

4.1.3 Fast Neural reconstruction in-the-wild

Extracting explicit 3D geometry as textured meshes from unstructured image collections represents a critical functionality within the XReco platform. While the Fast NeRF in-the-wild service (Section 4.1.1) successfully renders photorealistic novel views of 3D scenes, this service addresses a complementary requirement by generating texture mesh outputs that enable direct integration with standard 3D engines such as Unity3D.

A fundamental limitation of NeRF-based methods is that their underlying volume density representation makes it difficult to extract a precise and continuous surface. While meshes can be extracted from the learned density field, these often suffer from noise or lack fine detail. To overcome this, this algorithm employs a different neural representation specifically designed for high-fidelity surface recovery.

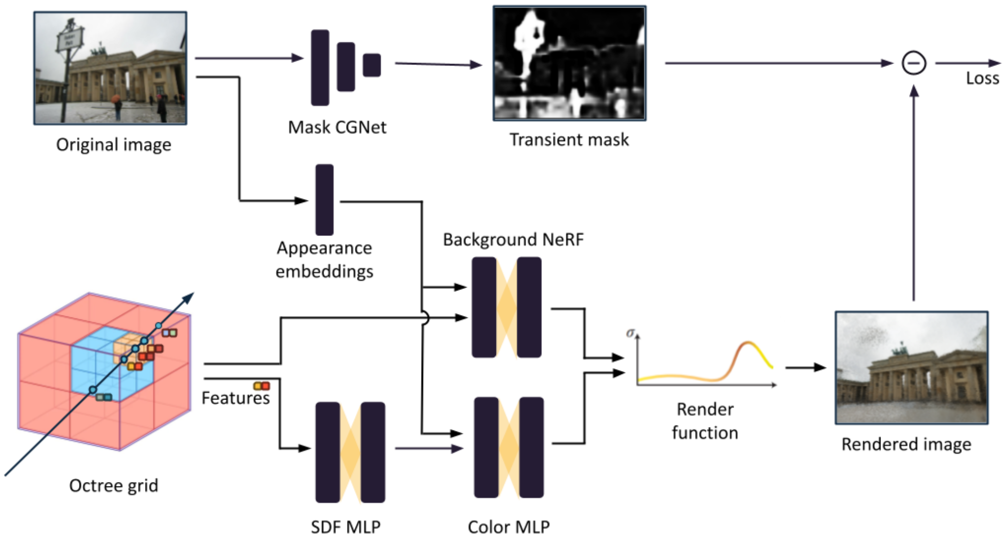


Figure 10: Octree based reconstruction in-the-wild pipeline. We ray-trace an octree-based feature volume to propagate features to an SDF and an RGB MLP. Additionally, the input image is fed into a CNN for estimating a transiency mask. Moreover, we optimise appearance embeddings to encode the appearance of each image which provide us with the ability to learn an appearance space.

Our Fast Neural Reconstruction in-the-wild pipeline builds upon the methodology introduced in NeuS[59]. As depicted in Figure 10, this approach employs a hierarchical network structure comprising three specialised MLPs. The primary MLP encodes a Signed Distance Function (SDF) representation of the complete neural field, with the object surface mathematically defined as the zero-level set of this function. The output from this SDF network is

[58] https://docs.celeryq.dev/en/latest/index.html
[59] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., & Wang, W. (2021). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689.

subsequently processed by a secondary MLP dedicated to encoding colour information across the neural field. A tertiary MLP, implementing a NeRF-based architecture, models background elements (non-object regions such as sky) to ensure complete scene representation.

To maintain photometric consistency, these networks undergo concurrent optimisation through volume rendering. This integrated training approach ensures that the rendered colour for each ray is precisely determined by both the surface location (defined by the SDF) and the surface gradient at the ray-surface intersection point, thereby preserving geometric accuracy.

A significant architectural enhancement, consistent with our Fast NeRF in-the-wild methodology (Section 4.1.1), incorporates an octree-based feature volume representation. This hierarchical spatial structure delivers substantial improvements in memory utilisation and computation efficiency. By storing optimised features within the octree grid as the primary reconstruction information source, the system enables deployment of considerably more compact MLPs for SDF and colour representation learning. This structural optimisation yields measurable reductions in training duration while simultaneously decreasing energy consumption requirements.

Appearance modelling: In this pipeline, we follow NeRFw, introducing per-image appearance embeddings to account for photometric variations in unstructured image collections. These embeddings are learned during optimisation and are provided as conditions to the RGB network, therefore, allowing the model to adjust the emitted radiance of the scene for a particular image, ensuring that the trained 3D geometry remains static across all images despite difference in illumination, exposure, and post-processing.

Occlusion handling: Transient objects such as pedestrians and vehicles often disrupt static scene reconstruction. To robustly filter these occluders, we generate a per-pixel visibility map using a learning-based CGNet segmentation network[60]. We train our pipeline in an end-to-end manner on Phototourism dataset scenes[53], without manual mask annotations (as in Neural Reconstruction in-the-wild[61]), by utilising a self-supervised loss. We multiply the predicted visibility map with the per-pixel squared difference between the rendered and observed images before computing the reconstruction loss. This encourages the network to downweight regions where transient objects appear, preserving only the static scene geometry for the SDF-based reconstruction.

Energy efficiency: Our method, running for 200,000 iterations consumes only 1,81kWh of GPU power. The only other alternative method to ours, which is Neural Reconstruction in-the-wild (using pre-trained masks)[62] is estimated to consume at least one order of magnitude more power.

The evaluation was performed on the same scenes as our Fast NeRF in-the-wild pipeline (Section 4.1.1) and we follow the same evaluation scheme as described in Section 4.1.1. Our model was trained on an NVIDIA RTX A6000 48GB GPU. Qualitative results on the Brandenburg Gate scene are presented in Figure 11, showcasing a textured mesh output. Additionally, interpolation on the learned appearance embedding space is presented for the same

[60] Wu, T., Tang, S., Zhang, R., Cao, J., & Zhang, Y. (2020). Cgnet: A light-weight context guided network for semantic segmentation. IEEE Transactions on Image Processing, 30, 1169-1179.
[61] Sun, J., Chen, X., Wang, Q., Li, Z., Averbuch-Elor, H., Zhou, X., & Snavely, N. (2022, July). Neural 3d reconstruction in the wild. In ACM SIGGRAPH 2022 conference proceedings (pp. 1-9).
[62] Sun, J., Chen, X., Wang, Q., Li, Z., Averbuch-Elor, H., Zhou, X., & Snavely, N. (2022, July). Neural 3d reconstruction in the wild. In ACM SIGGRAPH 2022 conference proceedings (pp. 1-9).

scene in Figure 12, in which we showcase that the integral parts of the reconstructed object remain structurally distinct while only the appearance of the scene is modified. Quantitative results are presented in Section 5.1.3.

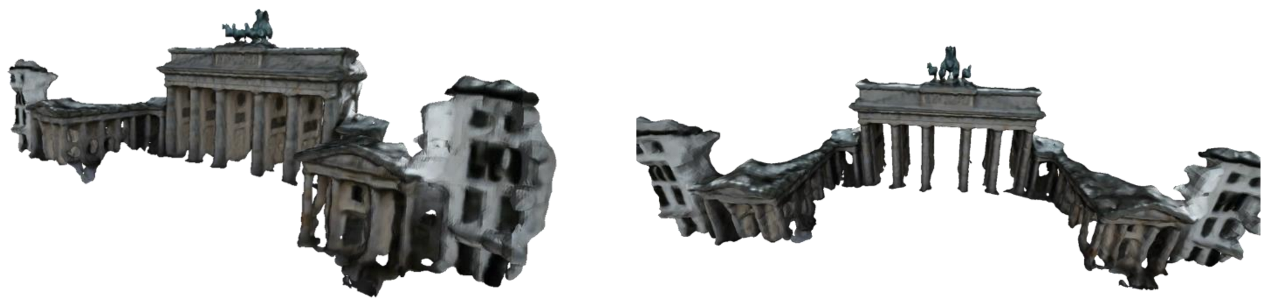


Figure 11: Textured mesh results on Phototourism's Brandendburg Gate scene from different viewpoints.

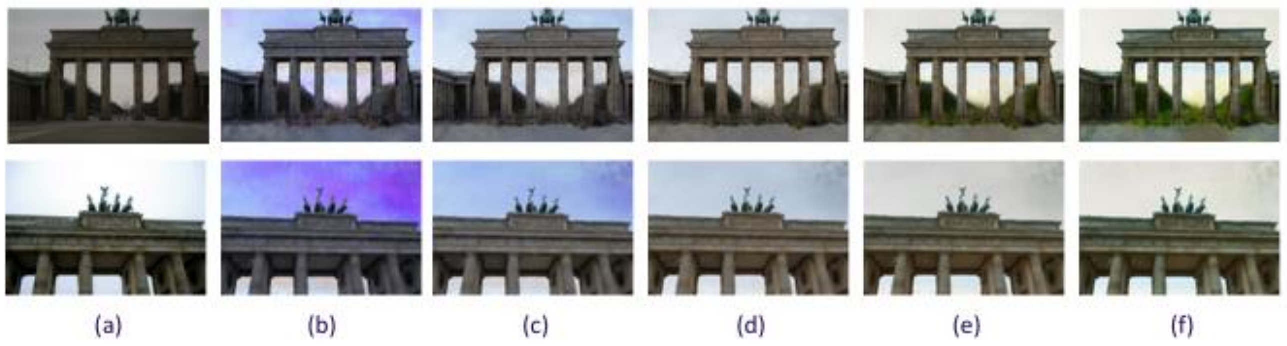


Figure 12: Qualitative results of appearance interpolation in the learned embedding space. (a) the original image, (b)-(f) model rendering output with different appearance embeddings.

Our method successfully reconstructs high-fidelity, coloured 3D meshes from in-the-wild images using an SDF-based representation for explicit surface extraction. Efficiency is central to the design, with the incorporation of the octree feature volume training and energy consumption are significantly accelerated by pruning empty space. The pipeline demonstrates robustness by using appearance embeddings to handle photometric variations and a self-supervised network to effectively filter transient occluders.

4.1.4 3DVista

In recent years, the advancement of artificial intelligence models, particularly techniques based on NeRFs, has revolutionised the way 3D content can be generated. NeRF technologies enable the creation of photorealistic 3D reconstructions from simple video sequences (like drone footage) or image collections captured from multiple viewpoints, significantly reducing the time and technical expertise required compared to traditional 3D modeling methods.

3D Vista is a user-friendly 3D reconstruction service powered by NerfStudio and NeRF technology. It allows users to easily generate high-quality 3D models from video footage. The system automatically configures optimal parameters—refined through extensive testing—based on a single user input: the desired model resolution.

With just one setting, 3D Vista generates a ready-to-run command string, streamlining the entire NeRF workflow for fast and accurate results.

The **3D Vista** service was developed to address the growing demand for accessible and efficient tools for creating 3D models from video footage, such as drone captures or mobile recordings. In fields such as audiovisual production, augmented and virtual reality, architecture, and cultural heritage documentation, there is a strong need to quickly and cost-effectively transform real-world environments into detailed digital replicas.

By integrating with NerfStudio, 3D Vista makes this process accessible even to non-expert users, automating the steps of preprocessing, training, and 3D model visualization. The use of optimized and pre-trained NeRF models allows for faster and more efficient training phases, supporting iterative, rapid, and scalable workflows.

4.1.4.1 Experimental results

In this section, we present the results generated by systematically combining key parameters, including texture resolution and vertex count, to produce low, medium, high, and super high resolution mesh variants. These settings simplify the user's workflow by automating the process of balancing individual parameters. Users may simply select the resolution tier that best fits their needs and requirements.
Table 4 shows the mesh profiles defined according to different texture resolution, and number of vertices, together with the corresponding output size and computation time using an NVIDIA RTX A5000 GPU. An example of a 3D model of the Basilica of Superga generated by applying the LOW profile and the SUPER HIGH profile starting from the same 2D video is illustrated in Figure 13.

Table 4: 3D Vista mesh profiles and corresponding parameters.

Profile	Texture Resolution	# Vertex	Size (MB)	Time
LOW	1,024	50,723	23	5 minutes
MID	2,048	152,450	72	50 minutes
HIGH	4,096	252,404	119	1 hour 45 minutes
SUPER HIGH	4,096	504,415	247	4 hours 30 minutes

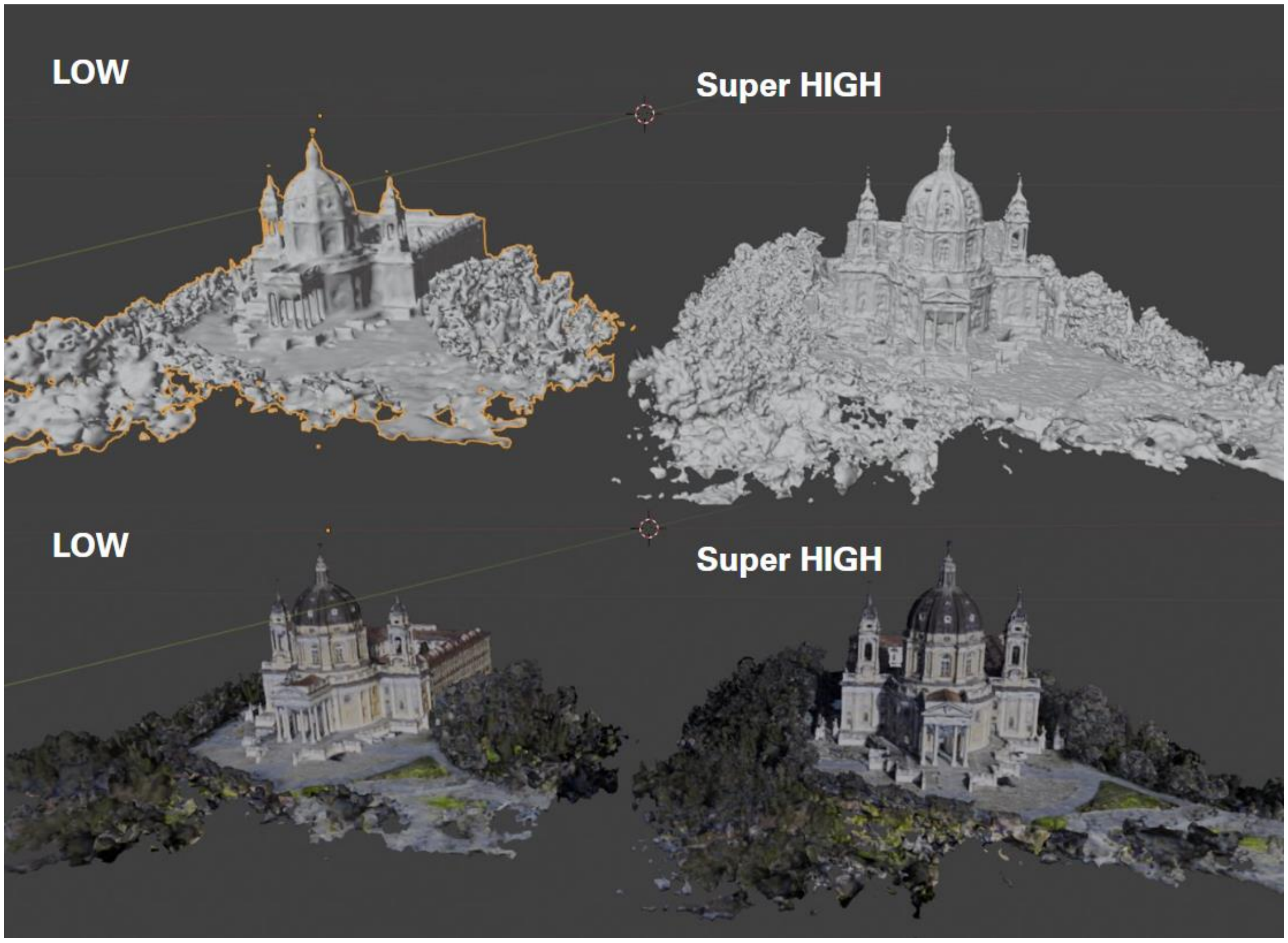


Figure 13: Examples of two 3D models of the Basilica di Superga, generated starting from the same 2D video source and applying two opposite mesh profiles of the 3D Vista service.

4.2 Vision based Reconstruction

4.2.1 Structure from Motion

4.2.1.1 Previous pipeline

The B1 version of the Structure from Motion (SfM)-based 3D reconstruction service offered by XReco relied on a traditional SfM approach using unordered 2D images as input to produce a dense, textured 3D mesh model.

The process begins with an initial SfM optimisation aimed at obtaining the camera parameters for the input images, which uses SIFT[63] to extract the feature points needed in this task. The results include a sparse point cloud, which is not suitable for 3D reconstruction. Once the camera parameters are available, contrast

[63] D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, Springer IJCV (Intl. Journal of Computer Vision), vol. 60, p. 91–110, Nov. 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

adjustment algorithms such as CLAHE[64] and MSRCR[65] are applied to the images, and feature points are once again extracted, this time using A-KAZE[66]. As shown in Figure 14, this approach significantly increases the number of feature points obtained, which can then be triangulated using the camera parameters to end up with a much denser point cloud.

Poisson surface reconstruction is applied to the point cloud to generate a triangular mesh. The number of triangles of this mesh is reduced to the number requested by the user using a combination of Laplacian smoothing and Quadric Error Metric (QEM)[67]-based decimation.

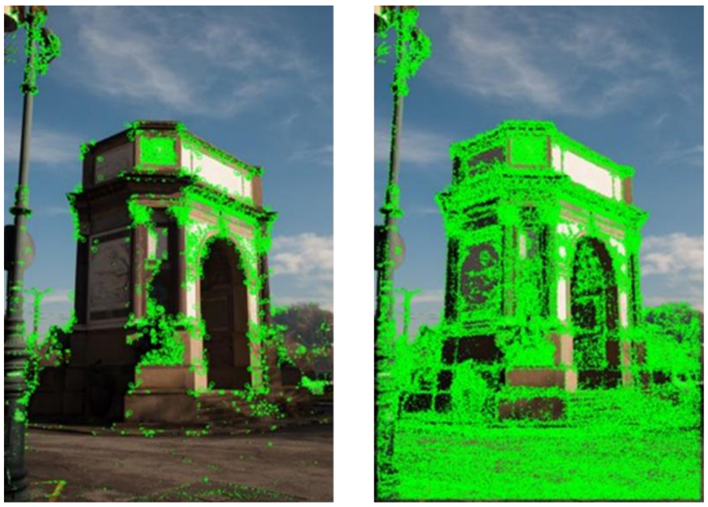


Figure 14: SIFT feature points vs pre-processing + A-KAZE feature points.

Finally, a multi-texturing[68] step is performed to compute a texture atlas for the mesh. This approach evaluates per-triangle visibility from each camera to interpolate texture contributions smoothly, reducing seams and lighting inconsistencies. Additionally, partial occlusions are handled by comparing color consistency across views, with inconsistent contributions being suppressed. Textures are efficiently packed into an atlas using block-based packing.

4.2.1.2 Densifying point clouds using Multi-View Stereo

One of the main improvements introduced in the updated pipeline is the modification of the method used for point cloud densification. In previous versions, densification was performed using feature points detected with A-KAZE. However, in the new workflow this approach has been replaced by an algorithm based on the PatchMatch Multi-View Stereo (MVS)[69] algorithm. This change not only allows more accurate results to be obtained, thanks to better exploitation of photometric consistency between multiple calibrated views, but also significantly improves the computational efficiency of the process. The PatchMatch algorithm structure is

[64] S. M. Pizer, E. P. Amburn, J. D. Austin, et al., "Adaptive histogram equalization and its variations", Elsevier Computer Vision, Graphics, and Image Processing, vol. 39, no. 3, p. 355–68, Sep. 1987. DOI: 10.1016/S0734-189X(87)80186-X.

[65] Z. Rahman, D. J. Jobson, G. A. Woodell, "Multiscale retinex for color rendition and dynamic range compression", Proc. SPIE, vol. 2847, Applications of Digital Image Processing XIX, Nov. 1996. DOI: 10.1117/12.258224.

[66] P. F. Alcantarilla, J. Nuevo, A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces", Proc. BMVC (British Machine Vision Conf.), p. 13.1–11, Sep. 2013. DOI: 10.5244/C.27.13 (https://bmva-archive.org.uk/bmvc/2013/Papers/paper0013/).

[67] M. Garland, P. S. Heckbert: "Surface simplification using quadric error metrics", Proc. ACM SIGGRAPH, p. 209–16, Aug. 1997. DOI: 10.1145/258734.258849.

[68] R. Pagés, D. Berjón, F. Morán, N. García, "Seamless, Static Multi-Texturing of 3D Meshes", EuroGraphics Computer Graphics Forum, vol. 34, no. 1, p. 228–38, Feb. 2015. DOI: 10.1111/cgf.12508.

[69] S. Shen: "Accurate Multiple View 3D Reconstruction Using Patch-Based Stereo for Large-Scale Scenes", *IEEE Transactions on Image Processing*, vol. 22, nº 5, pp. 1901–1914, May 2013. DOI: 10.1109/TIP.2013.2237921

designed for highly parallel execution, which considerably reduces processing times without compromising quality.

The method used is based on the MVS approach, which is widely used in computational photogrammetry because it allows highly accurate estimation of the geometry of a 3D scene from multiple views. The system implemented relies on an optimised version of this algorithm, which combines efficiency and robustness against occlusions and complex geometries. This process can be divided into three main blocks:

Preprocessing: First, a region of interest is determined that delimits the 3D volume where relevant geometry is expected to be found. This region is calculated from the initial cloud and expanded to ensure the inclusion of potentially visible peripheral areas. Next, inter-camera visibility is analysed, constructing a graph that indicates which image pairs share sufficient overlap and provide adequate parallax. Finally, multi-resolution versions of the input images are generated, allowing hierarchical processing from coarse scales to full resolutions, facilitating progressive and more stable estimation.

Depth Estimation: The core of the densification process consists of estimating, for each pixel in each reference image, a depth and surface (normal) orientation hypothesis. Initially, random hypotheses are assigned and refined by means of an adaptive spatial propagation scheme, where the best solutions are diffused to neighbouring pixels. This process is enhanced by stochastic refinement, which introduces small perturbations to escape local minima and explore alternative solutions in ambiguous or low-contrast regions.

Each hypothesis is validated by comparing its consistency with multiple neighbouring images, assessing both photometric similarity (based on texture patch matching) and geometric consistency (through epipolar consistency and normal orientation). The algorithm integrates all these metrics into a composite cost function that guides the selection of the optimal solution for each pixel.

Fusion and Post-processing: Once the per-pixel depths are estimated, multi-view fusion is performed to reconstruct a dense point cloud in three-dimensional space. This step combines the different validated hypotheses, resolving conflicts and eliminating ambiguities through voting and inter-view consistency mechanisms.

Subsequently, adaptive filtering is applied to improve the quality of the result by eliminating inconsistent points, with poor visibility or located in unreliable regions. Additional information such as average colour, surface normals and scale or local density measures are also calculated, attributes that enrich the point cloud and prepare it for the next stages of the pipeline, such as mesh reconstruction.

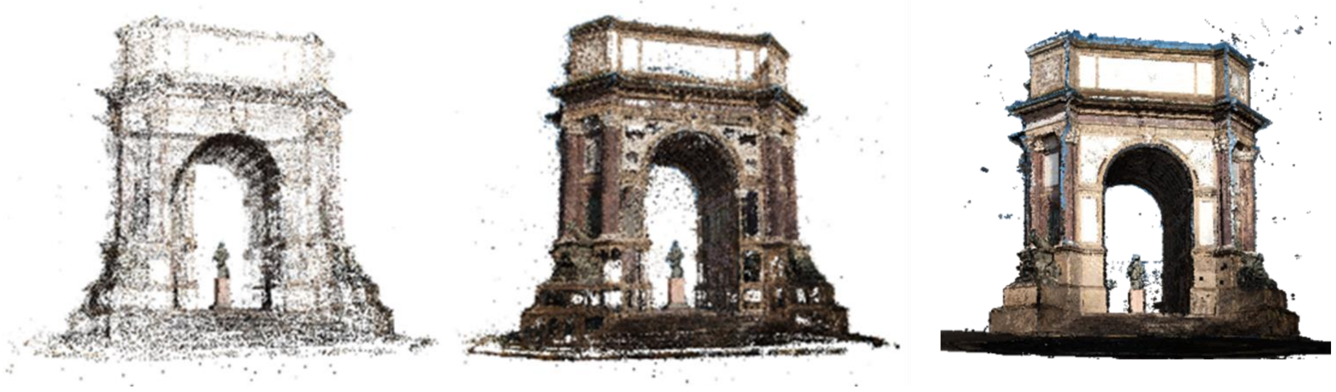


Figure 15: Comparison between SIFT (left), A-KAZE (middle), and MVS (right).

4.2.1.3 Mesh optimisation

Mesh cutting/trimming In certain cases, the primary goal is to obtain an accurate 3D representation of a specific object, without the need to reconstruct the surrounding environment. However, depending on how the images are captured, reconstruction algorithms often generate a significantly larger scene that includes much of the background visible to the cameras.

For this reason, when the object lies entirely within the perimeter defined by the camera positions, it is highly beneficial to include a functionality that automatically crops the generated 3D mesh. This automatic cropping is based on the estimated camera positions obtained during the SfM process.

It is important to note that cropping the input images to focus solely on the object is not a viable solution. Such preprocessing could alter the effective resolution and field of view of the images, potentially causing inconsistencies that negatively impact both the geometric reconstruction and the texturing quality. 3D reconstruction algorithms rely on accurate keypoint matches and consistent multi-view coverage, reducing overlap through image cropping can lower the number of shared features, resulting in a less accurate model.

The mesh cropping process begins by defining a vertical coordinate system. Reconstructed meshes are typically generated in arbitrary coordinates unrelated to the object's real-world orientation. Defining a consistent vertical axis allows for precise alignment and enables targeted mesh cropping, isolating the object and removing irrelevant parts of the scene, thereby improving efficiency and clarity.

This process starts with a set of *N* cameras, each associated with a calibration matrix that encodes its orientation in 3D space. In this context, only the rotation around the camera's local x-axis (*rotx*) is considered, as it corresponds to the tilt relative to the horizontal plane. For each camera i, its optical axis $\boldsymbol{r}_i \epsilon \boldsymbol{R}^3$ is extracted based solely on this rotational component. From these direction vectors $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \dots, \boldsymbol{r}_N\}$, a cumulative covariance matrix M is constructed.

$$M = \sum_{i=1}^{N} \boldsymbol{r}_i \boldsymbol{r}_i^T \ \epsilon \ \boldsymbol{R}^{3x3}$$

An eigen decomposition is then performed:

$$M = Q \Lambda Q^{-1}$$

where $Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3]$ contains the orthonormal eigenvectors and $\Lambda = diag(\lambda_1, \lambda_2, \lambda_3)$ the eigenvalues. The eigenvector associated with the smallest eigenvalue represents the direction of least variance across all camera viewing directions, in other words, the direction most orthogonal to the observation set. This vector is selected as the vertical axis of the custom coordinate system:

$$\boldsymbol{d} = \boldsymbol{q}_{min}$$

Using this axis, each camera position $\boldsymbol{p}$ is orthogonally projected onto a plane perpendicular to the axis $\boldsymbol{d}$ and centered at the origin. This projection is achieved by removing the component in the axis direction:

$$\boldsymbol{p}_\perp = \boldsymbol{p} - \frac{\boldsymbol{p} \cdot \boldsymbol{d}}{||\boldsymbol{d}||^2} \boldsymbol{d}$$

A second plane is defined at a positive offset $\delta > 0$ along the axis.

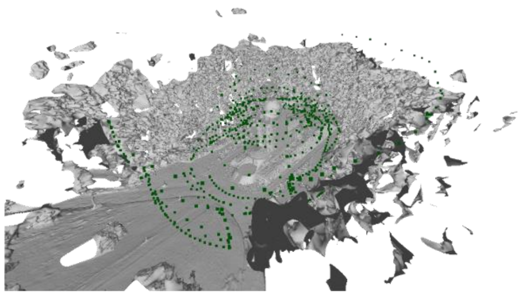
XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

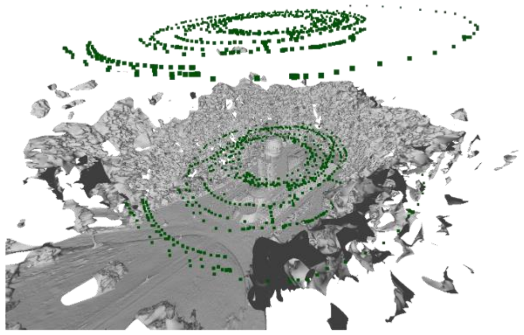
$$\boldsymbol{p}' = \boldsymbol{p}_{\perp} + \delta \cdot \boldsymbol{d}$$

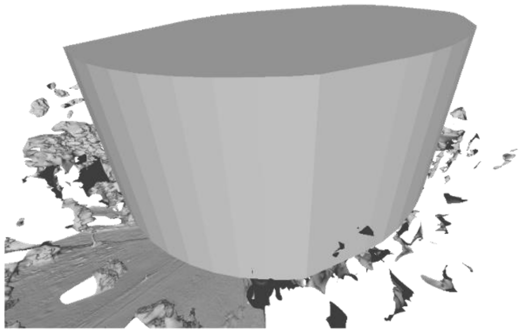
Resulting in two projections per original point. Using all the projected points, a 3D convex hull[70] *H* is constructed. This hull represents the smallest convex volume that encloses all camera positions and serves as a proxy volume for defining the region of interest. If $\{\boldsymbol{p}_i\}$ are the projected points, then:

$$H = Conv(\{\boldsymbol{p}_i\}) = \left\{\sum_i \lambda_i \boldsymbol{p}_i \mid \lambda_i \geq 0, \sum \lambda_i = 1\right\}$$

Next, a cutting plane perpendicular to the vertical axis is generated and centered at the mesh's centroid. The intersection between this plane and the mesh creates a triangulated surface representing the cutting region. This method ensures structural consistency, unlike computing a convex hull over a single projected plane, which may produce non-flat or incomplete geometries.


(a) The original mesh and the original camera positions.


(b) The original mesh and the projected camera positions.


(c) The original mesh and the convex hull obtained from the projected camera positions.

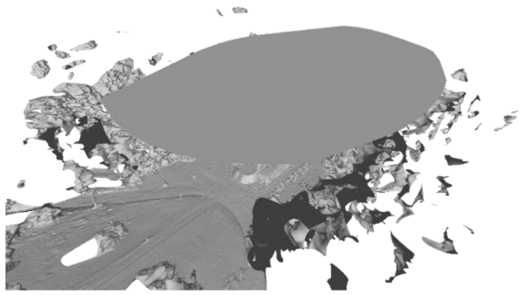

(d) The original mesh and the cutting plane obtained.

Figure 16: Process to obtain the cutting plane.

Once the cutting plane is defined, each vertex is orthogonally projected onto the plane using the formula:

$$\boldsymbol{v}_{proj} = \boldsymbol{v} - [(\boldsymbol{v} - \boldsymbol{p}_0) \cdot \widehat{\boldsymbol{d}}] \cdot \widehat{\boldsymbol{d}}$$

[70] C. B. Barber, D. P. Dobkin, H. Huhdanpää: "The Quickhull Algorithm for Convex Hulls", *ACM Transactions on Mathematical Software*, vol. 22, n.º 4, p. 469–483, Dec. 1996. DOI: 10.1145/235815.235821.

where p_0 is a point on the plane and $\hat{d}$ is the normalized direction vector perpendicular to the plane.

An orthonormal basis $\{e_1, e_2\}$ is then constructed in the plane, where $\boldsymbol{e}_1$ is a vector lying on the plane, and $\boldsymbol{e}_2 = \hat{\boldsymbol{n}} \times \boldsymbol{e}_1$, with $\hat{\boldsymbol{n}}$ being the normal vector to the plane. Each projected vertex is then mapped to 2D coordinates via dot products:

$$\boldsymbol{x}_i = (\boldsymbol{v}_{proj} - \boldsymbol{p}_0) \cdot \boldsymbol{e}_1$$

$$\boldsymbol{y}_i = (\boldsymbol{v}_{proj} - \boldsymbol{p}_0) \cdot \boldsymbol{e}_2$$

From the resulting 2D coordinates, a polygon $P \subset R^2$is built from the previously generated intersection. Each projected vertex is tested for inclusion in this polygon:

$$\boldsymbol{v}_i^{2D} \in P \implies vertex\ included$$

Only those faces whose three vertices are entirely contained within the polygon P are retained:

$$f = (i, j, k) \in F_{filtered} \Leftrightarrow \boldsymbol{v}_i^{2D}, \boldsymbol{v}_j^{2D}, \boldsymbol{v}_k^{2D} \in P$$

Finally, the remaining vertices and faces are remapped to ensure the mesh's topological continuity. Unused vertices are discarded, and face indices are updated accordingly, resulting in a clean and optimised representation of the target 3D object.

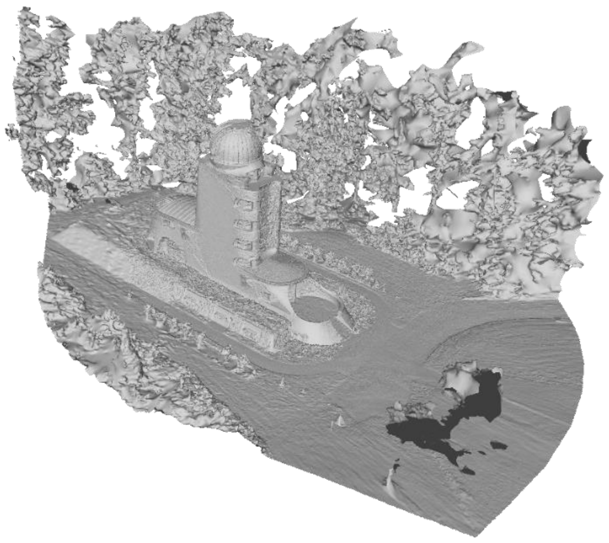


Figure 17: Result of applying the mesh cutting on the 3D mesh of the Einstein tower.

Mesh filtering: The geometry resulting from the reconstruction process may contain multiple disconnected components, also known as "islands." These can arise because the reconstruction method attempts to generate information in regions with insufficient data or because certain mesh regions partially intersect with the cutting plane when the clipping stage is applied. To enhance the robustness and quality of the model, components with significantly smaller surface areas compared to the main component are initially discarded.

Specifically, all components representing less than 10% of the total number of faces of the largest component are removed. This operation helps eliminate artifacts, structural noise, and degenerate regions, thereby improving both computational performance and the clarity and readability of the model.

From a topological standpoint, a search for connected components is performed on the mesh graph, and their relative size is evaluated by the condition:

$$\frac{|F_i|}{|F_{max}|} < 0.1 \Rightarrow component\ removed$$

where F_i denotes the set of faces of component i, and F_{max} is the set of faces of the largest component.

Once smaller islands have been removed, the mesh may still contain several disconnected components. To manage them individually, a complete partition of the mesh into connected components is performed, defined as maximal subsets of vertices and faces in which any pair of vertices is connected by a path of edges. This separation allows each component to be treated as an independent entity within the global mesh set. Formally, this corresponds to a decomposition into equivalence classes under the connectivity relation between vertices.

Although this partitioning step is theoretically sufficient, removing small islands beforehand has been shown to significantly improve computational performance by reducing the complexity of the connectivity graph and the overhead associated with processing numerous insignificant components. For instance, in the case of the Arco Valentino mesh, this pre-filtering step reduces computation time from 620 seconds to just 40, demonstrating its practical importance in optimizing mesh workflows.

Among all separated components, the one with the greatest number of vertices is automatically selected, regarded as the main component because it presumably contains the geometry of greatest interest.

Given the set $\{M_1, M_2, \dots, M_N\}$ of separated components, the main component is defined as:

$$M_{main} = \arg\max_{M_i} |V_i|$$

where V_i is the set of vertices of component M_i.

This criterion favours retaining the most structurally significant region, automatically discarding disconnected fragments of lesser importance without requiring manual intervention. Figure 18 compares the resulting mesh without any post-processing (left) and the same mesh after applying only the mesh filtering step (right).

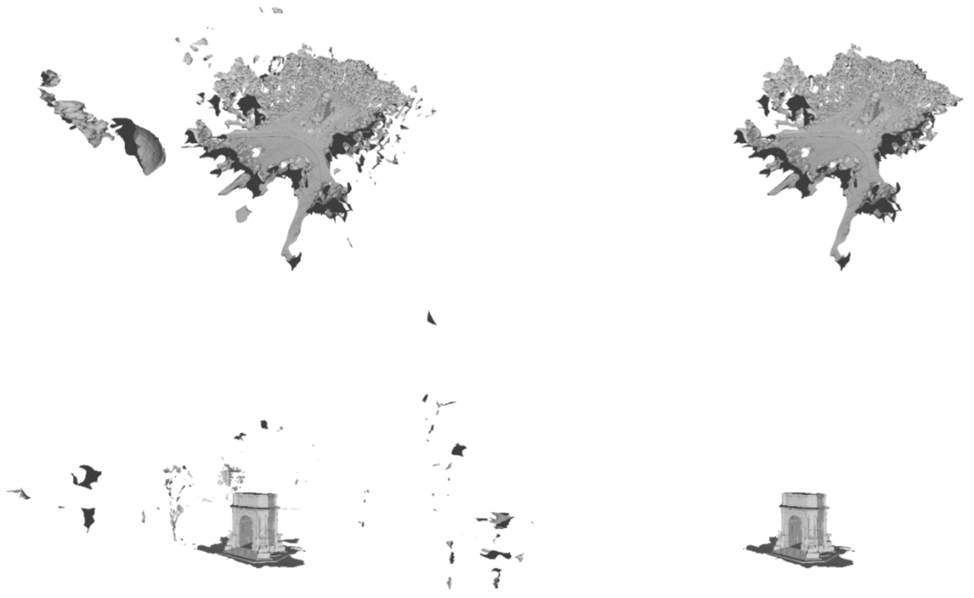


Figure 18: Comparison between the mesh without any post-processing (left) and the mesh after applying the mesh filtering step (right).

Mesh simplification: We apply again a QEM-based decimation process to make sure that simpler (i.e., flatter) mesh areas are represented with fewer triangles while mesh corners and details are preserved. However, this time, thanks to the prior optimisation (mesh cutting and mesh filtering), the quality of the final model increases: since the same number of triangles is now distributed over a smaller surface area, the level of detail on the object of interest increases.

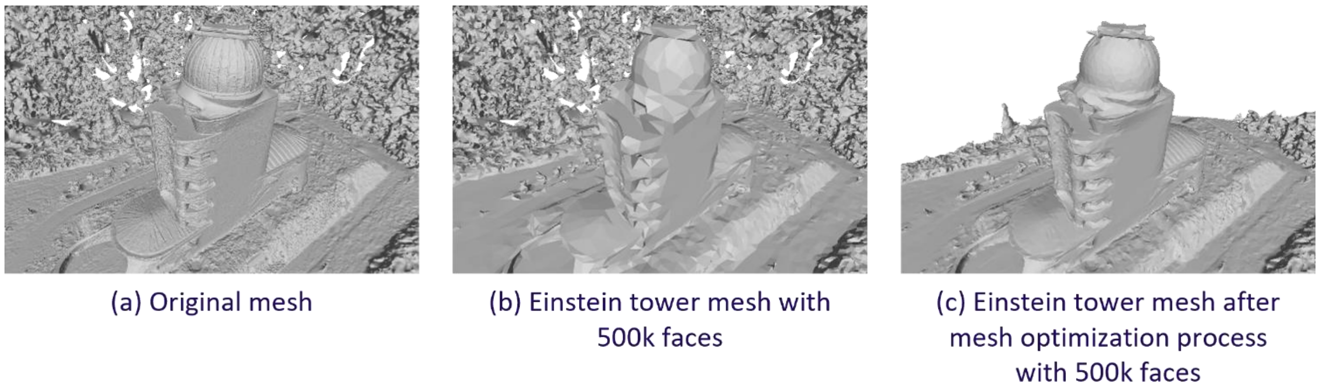

(a) Original mesh
(b) Einstein tower mesh with 500k faces
(c) Einstein tower mesh after mesh optimization process with 500k faces

Figure 19: Comparison between the original mesh of the Einstein tower (a), the mesh with 500k faces and the mesh after the complete optimisation process with 500k faces.

5 Scene Reconstruction Services Validation

This section presents a complete validation framework for most of the 3D reconstruction services described in Section 4, encompassing both objective (for all the services presented) and subjective (for SfM) quality assessment methodologies. The evaluation approach employs synthetic view rendering from reference models to enable systematic and repeatable quality analysis across different reconstruction parameters and input conditions.

5.1 Reconstruction quality evaluation

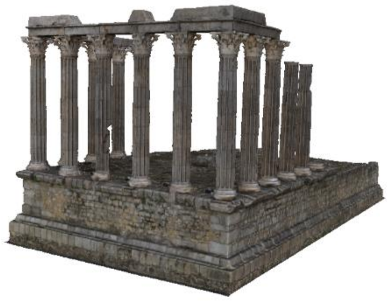
The methodology used to evaluate the quality of reconstruction services was based on the simulation of 3D modeling processes applied to archaeological sites, buildings, and antiquities. This methodology enables a systematic and repeatable evaluation of reconstruction quality under varied input conditions. To this end, six 3D meshes were selected from the BASICS[71] dataset and downloaded from Sketchfab[72] to serve as reference models. These models, representative of real-world cultural heritage objects, are shown in Figure 20.

a) Horn of Salt Diggers Brotherhood of Wieliczka.

b) Palace of Fine Arts.

c) Kriegerdenkmal.

d) Schwarzenbach - houses with interior.


e) Roman Temple of Evora.

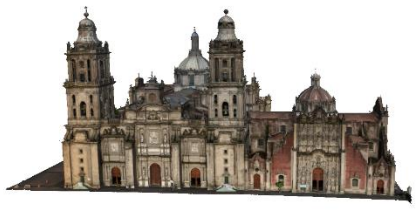

f) Mexico City Metropolitan Cathedral.

Figure 20: Selected models for the 3D reconstruction quality evaluation

[71] A. Ak, E. Zerman, M. Quach, A. Chetouani, A. Smolic, G. Valenzise, and P. Le Callet, “Basics: Broad quality assessment of static point clouds in a compression scenario,” IEEE Transactions on Multimedia, vol. 26, pp. 6730–6742, 2024.
[72] https://sketchfab.com/

The models were chosen to ensure variability while maintaining comparable characteristics. For example, objects a, b, and c in Figure 20 each contain fewer than 500,000 triangles, whereas objects d, e, and f have approximately 2 million triangles each.

Each reference model was used to generate 1,000 synthetic views from virtual camera positions uniformly distributed over a spherical surface, following a spiral trajectory from the base to the top. All images were rendered using a fixed field of view (42°) and a resolution of 3840 × 2160 pixels (~8.3 MP).

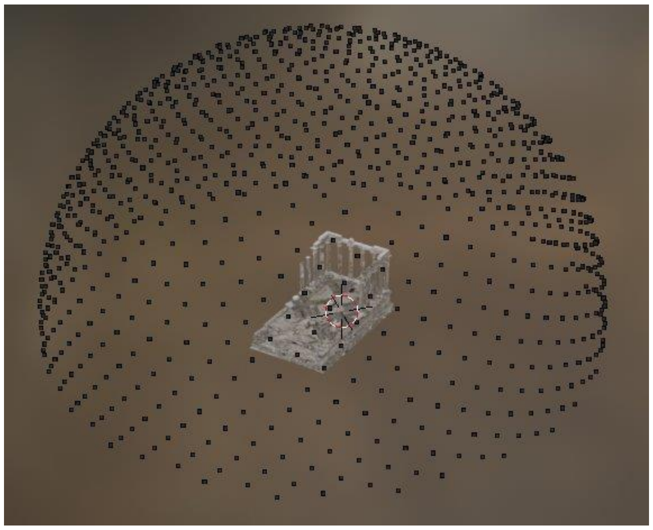


Figure 21: Example of one of the reference models along with the camera positions used to capture the synthetic views.

For the reconstruction process, different subsets of these images were used, containing 50, 125, 250, or 500 views, respectively. An independent set of 300 additional views, not included in the reconstruction process, was reserved exclusively for evaluation. This ensures that the assessment focuses on previously unseen viewpoints, thereby improving the robustness and fairness of the evaluation.

An example of one of the reference models along with the camera positions used to capture the 1000 synthetic views is shown in Figure 21.

To ensure accurate comparison, the reconstructed models were aligned with the reference models using transformation parameters (scale, rotation, and translation). This alignment facilitates the calculation of both 2D and 3D quality metrics.

For full-reference (FR) metrics, image comparisons were made from matching viewpoints in both the original and reconstructed models. Since only specific regions of each image were relevant, binary masks were created to isolate the object from the background.

5.1.1 Objective quality metrics

The evaluation focused on commonly used image-based quality metrics to assess the effects of reconstruction parameters in each service.

For FR metrics, comparisons were performed between images rendered from corresponding viewpoints in the original and reconstructed models. Since only specific regions of each image were relevant, binary masks were generated to isolate the object from the background, ensuring that quality assessments focused solely on the reconstructed content. The following metrics were employed:

- **FR metrics**: PSNR[73], SSIM[74] and LPIPS[75].
- **No-reference metrics**: BRISQUE[76] and Natural Image Quality Evaluator (NIQE)[77].

Together, these metrics provided a robust and automated framework for evaluating the visual fidelity of 3D reconstructions under varying input conditions.

5.1.2 SfM

We conducted objective and subjective experiments to evaluate the quality of the models reconstructed using SfM and how it varies depending on different configuration parameters. This was achieved by systematically varying three parameters: the number of input images, the number of triangles used in the mesh, and the texture resolution.

5.1.2.1 Objective evaluation

For each viewpoint, the union of the binary masks from the reference and reconstructed images was used to compute image-based metrics. This approach avoids the overly optimistic results that can arise when using the intersection of masks, particularly in cases where reconstructions significantly deviate from the ground truth. To validate the effectiveness of the masking strategy, the Intersection over Union (IoU) was also computed across all views, achieving an average IoU of 98.44% (±0.79%).

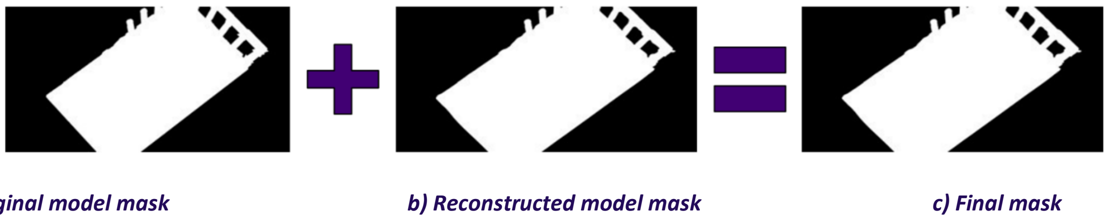


Figure 22: Visualisation of the union between the reference model's mask and the corresponding reconstructed model's mask, used to produce the final mask employed for evaluation in that view.

[73] https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio
[74] https://en.wikipedia.org/wiki/Structural_similarity_index_measure
[75] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 586-595).
[76] Mittal, A., A. K. Moorthy, and A. C. Bovik. "No-Reference Image Quality Assessment in the Spatial Domain." IEEE Transactions on Image Processing. Vol. 21, Number 12, December 2012, pp. 4695–4708.
[77] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "completely blind" image quality analyzer," IEEE Signal processing letters, vol. 20, no. 3, pp. 209–212, 2012.

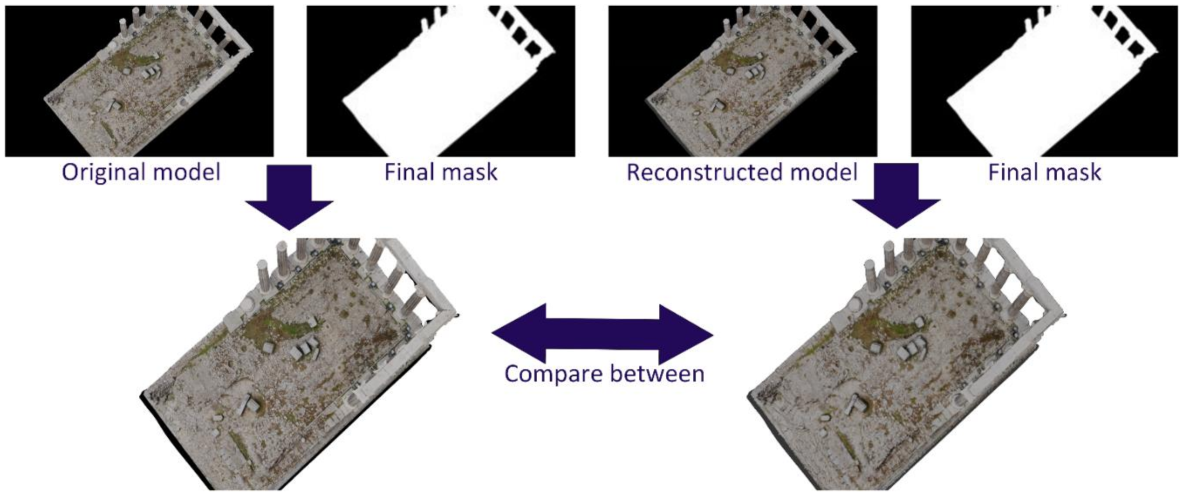


Figure 23: Workflow for generating the final comparison view between original and reconstructed models.

Additionally, because SfM produces mesh-based reconstructions, two geometry-based metrics were incorporated to complement the image-based evaluations: Hausdorff[78] distance and L^2 (Euclidean) distance.

These metrics offer a spatial analysis of reconstruction accuracy, allowing for a more comprehensive assessment that considers not only visual similarity but also geometric fidelity. Table 5 presents the mean and Standard Deviation (SD) of both image-based for the 300 test images and model-based quality metrics.

Table 5: Mean ± SD evaluated over the 300 test captured images for the selected 3D model reconstruction parameters. The best result is bold underlined, the second best in bold, the third in underlined, and the worst is unformatted. A ↑ indicates that higher values correspond to better quality, while ↓ signifies the opposite. Texture resolution is indicated in MP.

Parameter		Metrics						
		PSNR ↑	SSIM ↑	LPIPS ↓	BRISQUE ↓	NIQE ↓	Hausdorff ↓	L^2 ↓
		Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
No. images	50	24.037 ± 2.931	**0.634** ± 0.143	0.124 ± 0.075	70.974 ± 12.962	**6.710** ± 1.533	**0.011** ± 0.003	0.001 ± 0.000
	125	**24.105** ± 3.029	**0.636** ± 0.147	0.121 ± 0.077	69.708 ± 13.500	**6.759** ± 1.600	**0.011** ± 0.004	0.001 ± 0.000
	250	**24.088** ± 3.021	0.632 ± 0.147	**0.120** ± 0.077	**69.156** ± 13.656	6.789 ± 1.615	**0.013** ± 0.005	0.001 ± 0.000
	500	23.907 ± 2.909	0.618 ± 0.145	**0.119** ± 0.077	**68.701** ± 13.913	6.802 ± 1.621	0.014 ± 0.008	0.001 ± 0.000

[78] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the Hausdorff distance," in Proceedings. IEEE international conference on multimedia and expo, vol. 1. IEEE, 2002, pp. 705–708.

Parameter		Metrics						
		PSNR ↑	SSIM ↑	LPIPS ↓	BRISQUE ↓	NIQE ↓	Hausdorff ↓	L^2 ↓
		Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
No. triangles	5k	22.797 ± 2.458	0.587 ± 0.121	0.127 ± 0.078	71.682 ± 13.629	**6.759** ± 1.617	0.012 ± 0.005	0.001 ± 0.000
	25k	24.062 ± 3.002	0.634 ± 0.149	0.120 ± 0.077	69.789 ± 13.464	**6.767** ± 1.595	0.012 ± 0.006	0.001 ± 0.000
	50k	**24.518** ± 3.048	**0.647** ± 0.153	**0.117** ± 0.076	**68.917** ± 13.476	6.762 ± 1.585	0.012 ± 0.006	0.001 ± 0.000
	100k	**24.759** ± 2.958	**0.651** ± 0.149	**0.118** ± 0.074	**68.151** ± 13.329	6.771 ± 1.575	0.012 ± 0.006	0.001 ± 0.000
Texture resolution	1	22.132 ± 2.381	0.482 ± 0.108	0.181 ± 0.086	81.499 ± 9.242	7.002 ± 1.187	-	-
	4	23.748 ± 2.667	0.607 ± 0.115	0.133 ± 0.069	69.811 ± 10.605	6.750 ± 1.485	-	-
	16	**25.031** ± 2.845	**0.707** ± 0.108	**0.090** ± 0.051	**64.168** ± 12.433	**6.650** ± 1.773	-	-
	32	**25.226** ± 2.897	**0.724** ± 0.108	**0.079** ± 0.046	**63.060** ± 12.918	**6.657** ± 1.820	-	-

In terms of the number of input images, the results are inconsistent across metrics. While LPIPS and BRISQUE indicate improvements with more images, Hausdorff and NIQE both suggest the opposite. PSNR, SSIM, and L^2 show no clear trend.

Regarding mesh complexity, most metrics, such as PSNR, BRISQUE, and LPIPS, point to a quality increase with higher triangle counts. All metrics identify the 5k triangle models as the lowest quality. The best results are generally found with 100k triangles, LPIPS favours the 50k configuration though. In contrast, NIQE shows a reverse pattern, with quality decreasing as triangle count increases.

In terms of texture resolution, most metrics agree on improved quality at higher resolutions, with the exception of NIQE, which rates 16 MP slightly higher than 32 MP.

To further investigate these differences, statistical analyses were performed. Since the data did not follow a normal distribution (confirmed by multiple normality tests), non-parametric Kruskal–Wallis[79] tests were used, see Table 6. For parameters showing statistically significant results ($p < 0.05$), post-hoc pairwise comparisons were performed using Wilcoxon tests with Bonferroni correction, see Table 7. These analyses confirmed significant effects for triangle count and texture resolution across most metrics. For the number of images, only

[79] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," Journal of the American statistical Association, vol. 47, no. 260, pp. 583–621, 1952.

the Hausdorff distance showed statistically significant differences, especially between configurations with 500 images and those with 50 or 125. This suggests that geometry-based evaluations are more sensitive to image count than appearance-based metrics, potentially due to the uniform distribution of camera viewpoints in the dataset.

Table 6: P-values for the Kruskal-Wallis test, asterisks indicate statistical significance. A '-' indicates that the test was not performed.

Parameter	Metrics						
	PSNR	SSIM	LPIPS	BRISQUE	NIQE	Hausdorff	L^2
Number of images	0.9648	0.7447	0.9300	0.4001	0.8794	0.0349*	0.8122
Number of triangles	< 0.0001*	0.0036*	< 0.0001*	0.0666	0.9872	0.9824	< 0.0001*
Texture resolution	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	0.0024*	-	-

Table 7: P-values for the paired-Wilcoxon tests with Bonferroni correction, asterisks indicate statistical significance. N/A has been used to indicate the cases in which the Kruskal-Wallis test did not report statistical difference. A '-' indicates that the test was not applicable. Texture resolution is indicated in MP.

Parameter	Comparison	Metrics						
		PSNR	SSIM	LPIPS	BRISQUE	NIQE	Hausdorff	L^2
Number of images	50 vs 125	N/A	N/A	N/A	N/A	N/A	1.0000	N/A
	50 vs 250	N/A	N/A	N/A	N/A	N/A	0.2199	N/A
	50 vs 500	N/A	N/A	N/A	N/A	N/A	0.0135*	N/A
	125 vs 250	N/A	N/A	N/A	N/A	N/A	0.6443	N/A
	125 vs 500	N/A	N/A	N/A	N/A	N/A	0.0051*	N/A
	250 vs 500	N/A	N/A	N/A	N/A	N/A	0.7250	N/A
Number of triangles	5k vs 25k	< 0.0001*	< 0.0001*	< 0.0001*	N/A	N/A	N/A	< 0.0001*
	5k vs 50k	< 0.0001*	< 0.0001*	< 0.0001*	N/A	N/A	N/A	< 0.0001*
	5k vs 100k	< 0.0001*	< 0.0001*	< 0.0001*	N/A	N/A	N/A	< 0.0001*
	25k vs 50k	< 0.0001*	< 0.0001*	< 0.0001*	N/A	N/A	N/A	< 0.0001*
	25k vs 100k	< 0.0001*	< 0.0001*	0.0420*	N/A	N/A	N/A	< 0.0001*
	50k vs 100k	< 0.0001*	0.6424	1	N/A	N/A	N/A	< 0.0001*
Resolution	1 vs 4	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	-	-
	1 vs 16	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	-	-
	1 vs 32	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	-	-
	4 vs 16	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	0.0723	-	-
	4 vs 32	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	0.1748	-	-
	16 vs 32	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*	0.5398	-	-

Post-hoc analysis reveals that FR metrics (PSNR, SSIM, LPIPS) effectively detect changes in mesh detail, particularly at lower triangle counts. Likewise, texture resolution significantly impacts all image-based metrics, though NIQE only distinguishes between the lowest and higher resolutions, not between medium and high settings.

The results indicate that variations in the objective metrics are largely driven by differences in mesh complexity and texture resolution inherent to the models. In contrast, the number of input images has a limited impact on the metrics when the images are uniformly distributed, suggesting that model characteristics play a more significant role in influencing the evaluation outcomes under these acquisition conditions.

5.1.2.2 Subjective evaluation

The Double Stimulus Degradation Category Rating (DCR)[80] methodology has been employed to design the subjective evaluation of the models. This approach allows participants to view the original model alongside its reconstructed versions, each generated using different parameter settings and therefore exhibiting varying levels of distortion. By directly comparing the original and distorted models, users are better equipped to assess the visual quality of the 3D reconstructions with greater accuracy and consistency.

Prior to the evaluation, a selection process was conducted to determine which models would be used. Five individuals collaboratively divided and assessed all available models, ensuring that each model was evaluated during an initial round of voting. These preliminary assessments were used solely for model selection and were not included in the final results. Based on this initial evaluation, the **"**Schwarzenbach - houses with Interior" was excluded, as it consistently received high scores (mostly 4s and 5s), indicating limited potential for revealing perceptible quality differences across parameter variations.

Following this selection process, the decision was made to evaluate the remaining models using the parameter combinations shown in Table 8.

Table 8: Combinations of parameters evaluated during the subjective experiment.

Number of images	Number of triangles	Resolution (MP)
50	5k	1
		4
		16
		32
	12.5k	1
		4
		16
		32
	100k	32
125	25k	1
		4
		16
		32
	100k	32
250	5k	32
	50k	32
	100k	32
500	100k	32

[80] ITU, P. (1999). 910. Subjective video quality assessment methods for multimedia applications. International telecommunications union telecommunication sector.

These parameter combinations were selected with the goal of exploring a wide and representative spectrum of reconstruction quality levels. By systematically varying the number of input images, the mesh complexity (number of triangles), and the texture resolution (in megapixels), it was possible to simulate different real-world scenarios, from low-resource reconstructions to high-fidelity outputs.

The combinations were designed to isolate the impact of each parameter while maintaining control over the others. For example, in some tests the number of images varied while keeping resolution and mesh complexity constant, allowing for an analysis of how image coverage affects perceived quality. In other cases, triangle count, or resolution was varied independently. Additionally, certain combinations intentionally pushed parameters to extreme values (e.g., 5k triangles or 1MP textures) to examine perceptual thresholds for model degradation.

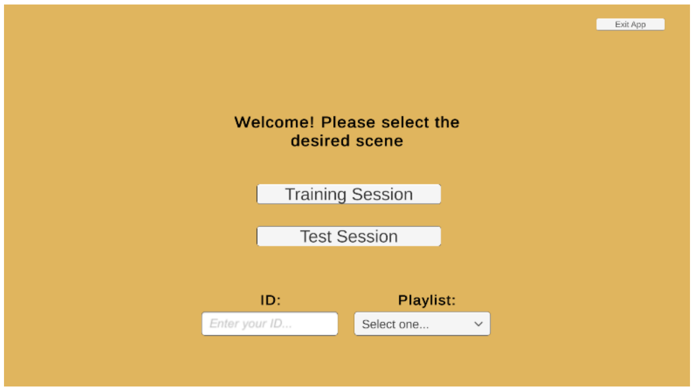
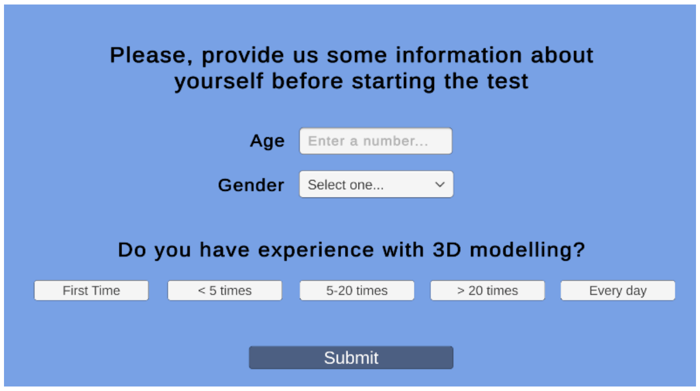


Figure 24: Illustration of the Main Menu of the subjective evaluation application.

This experimental design ensures a comprehensive evaluation of how each individual factor contributes to the subjective perception of quality in 3D reconstructions.

An interactive application was developed in order to conduct the tests using Unity and C#. This tool was specifically designed to streamline the evaluation process and is structured into several functional scenes. The central scene is the Main Menu, see Figure 24, from which users can access two primary modes: the Training Session and the main evaluation mode, referred to as the Test Session. Both scenes share a similar visual layout and interaction logic to ensure consistency throughout the user experience.

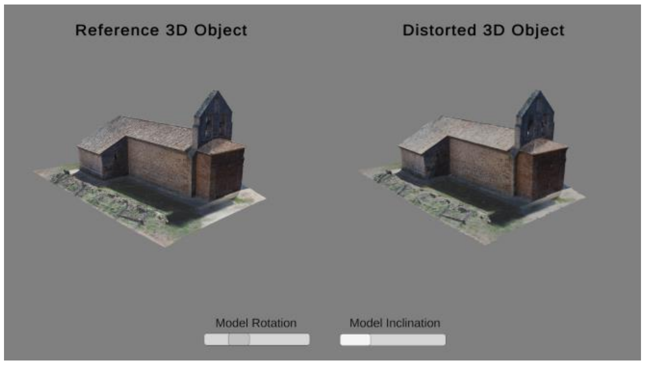

Please, provide us some information about yourself before starting the test
Age
Enter a number...
Gender
Select one...
Do you have experience with 3D modelling?
First Time
< 5 times
5-20 times
> 20 times
Every day
Submit
Figure 25: Illustration of the Tester data collection interface.

The Training Session serves as a preparatory environment in which users can become familiar with the visual exploration and rating mechanics of the application. In this mode, three distorted versions of a 3D model, distinct from those used in the Test Session, are presented.

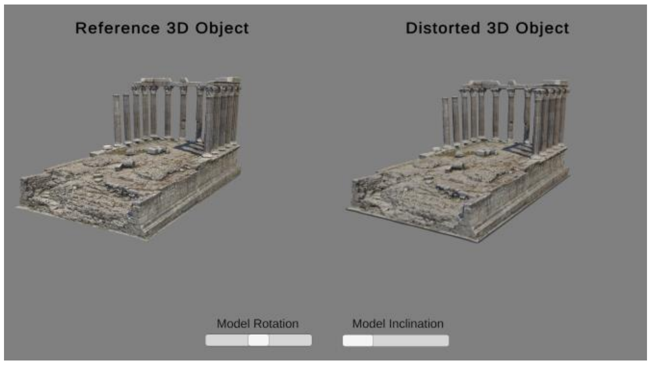
Before entering the Test Session, users are required to provide basic demographic and contextual information, including an identification code (ID), age, gender, the specific playlist to be evaluated, and their previous experience with 3D models, as shown in Figure 25.

In both the Training and Test Sessions, participants are presented with a pair of 3D models: a reconstructed original and its corresponding distorted version. These models are dynamically loaded from two separate folders, Originals and Distorted, using Unity's *Resources.Load()* function, following the structure defined by preconfigured playlists. The selection of model pairs is handled through a randomization algorithm that ensures no repeated combinations are shown and prevents consecutive evaluations of identical models. Additionally, the system automatically detects when all pairs in the playlist have been evaluated, triggering a transition to a final closing scene.

The application enables synchronized rotation and tilting of both models using a system of coupled cameras that orbit around each object. These camera movements are controlled via sliders, ensuring that the user observes both models from identical and symmetrical perspectives.



a) Illustration of the Training Session view.



b) Illustration of the Test Session view.

Figure 26: Illustration of the user interface during the visualisation of models in both the Training (a) and Test sessions (b).

Once the visual inspection is complete, the user proceeds to a voting scene where they rate the quality of the distorted model in comparison to the original, using a scale from 1 to 5, see Figure 27. All results are automatically recorded in a CSV file along with additional metadata such as user input and session configuration. Furthermore, the application logs all camera interactions (rotations and tilts), enabling subsequent analysis of user behaviour and visual exploration patterns.

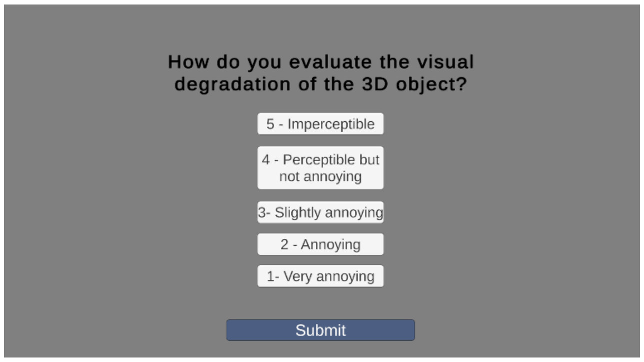

How do you evaluate the visual degradation of the 3D object?
5 - Imperceptible
4 - Perceptible but not annoying
3- Slightly annoying
2 - Annoying
1- Very annoying
Submit

Figure 27: Illustration of the voting interface.

The user testing phase involved 36 participants, 26 men and 10 women. 14 of them completed two sessions with different playlists on separate days, while the remaining 22 completed a single session.

In terms of prior experience with 3D models, 7 participants reported no previous exposure, 11 had interacted with 3D models fewer than five times, 6 had between 5 and 20 prior experiences, 7 had used 3D models on more than 20 occasions, and 5 indicated they use them on a daily basis.

Each testing session lasted between 15 and 35 minutes, depending on the individual participant.

At the start of each session, participants received a detailed explanation of the evaluation procedure, the type of data that would be collected, and the structure of the test. In addition, a short screening test was conducted to identify potential visual impairments, such as colour blindness or difficulty seeing at certain distances. After this initial briefing, users proceeded to interact with the application.

Each session comprised the evaluation of a training set followed by two distinct test playlists, each containing a different subset of 3D models. Participants were free to take as much time as needed to examine the models before submitting their ratings. Additionally, they were given the opportunity to take a short break between the evaluation of each playlist to reduce visual fatigue and maintain consistent performance throughout the session.

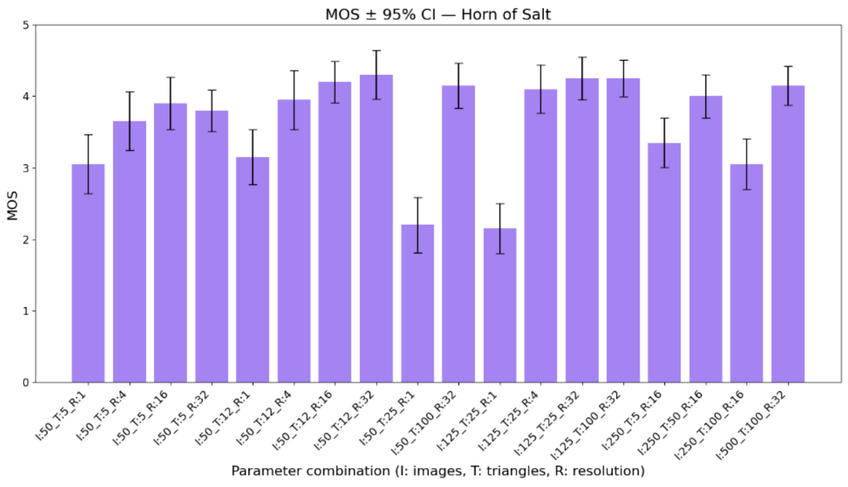


Figure 28: MOS ± 95% CI for the "Horn of Salts Diggers Brotherhood of Wieliczka" model across different parameter combinations in subjective tests.

The results obtained for each model are presented below.

Horn of Salt diggers brotherhood of Wieliczka: In the following graph, Figure 28, we can observe that the model generally received very good scores, consistently achieving an average Mean Opinion Score (MOS) above 3 in all cases, except when using both 50 and 125 images combined with 25,000 triangles and a 1MP resolution.

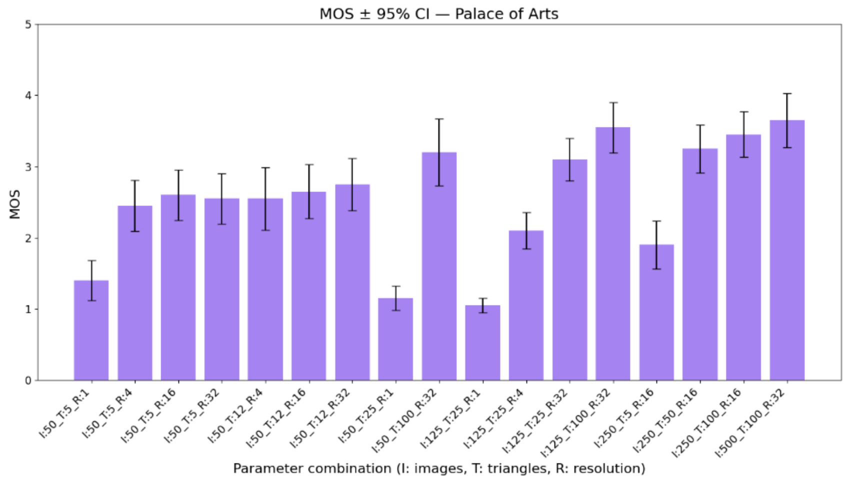


Figure 29: MOS ± 95% CI for the "Palace of Fine Arts" model across different paraemeter combinations in subjective tests.

Palace of Fine Arts: In Figure 29, the model's scores are noticeably lower compared to the previous case. This difference may be attributed to the fact that, although the original models have a similar number of triangles (fewer than 500,000), this model's geometry and texture are more complex. As a result, the quality could potentially improve by increasing the number of triangles. However, it is worth noting that when higher parameter settings are applied, the MOS score does surpass 3.

Kriegerdenkmal: Figure 30 shows that this model scored above 3 in nearly all parameter combinations. These favourable results might be due to the relatively simple geometry of the original model and the fairly uniform colouration, which likely makes reconstruction easier and more visually pleasing.

Roman Temple of Evora: In Figure 31, the model received very high scores. Despite the original model having a high triangle count, the uniform colour of the object helps mask geometric imperfections, leading to better subjective evaluations.

Mexico City Metropolitan Cathedral: In the case of Figure 32, the ratings drop again, with MOS scores exceeding 3 only in four instances. This decline can be explained by the fact that the original model is the most complex of all studied, both in terms of geometry and texture. Consequently, distortions become more apparent when comparing the original and reconstructed models.

Overall, across all cases, resolution emerges as the most influential parameter, followed by the number of triangles. The number of images appears to have the least impact in these scenarios, which aligns with the results observed in the objective tests.

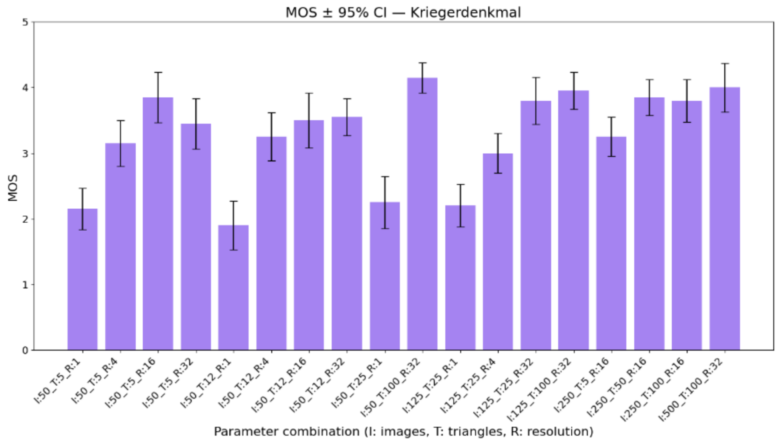


Figure 30: MOS ± 95% CI for the "Kriegerdenkmal" model across different parameter combinations in subjective tests.

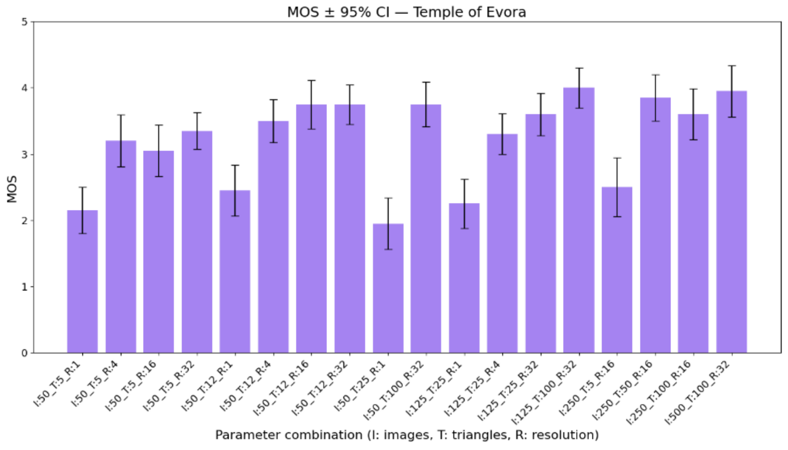


Figure 31: MOS ± 95% CI for the "Roman Temple of Evora" model across different parameter combinations in subjective tests.

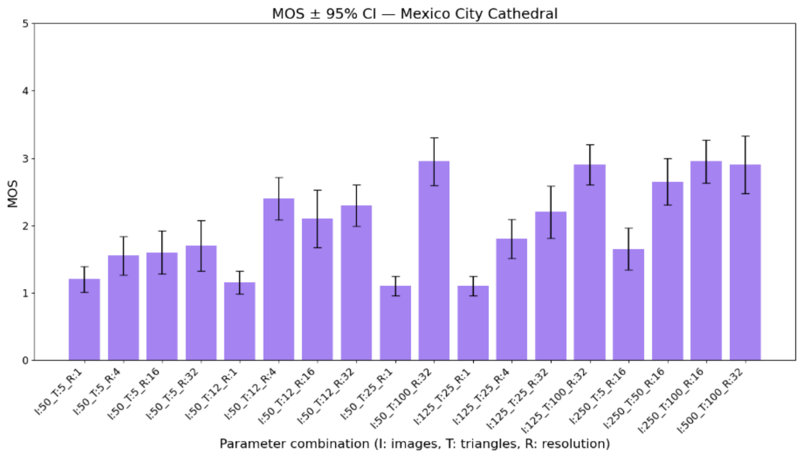


Figure 32: MOS ± 95% CI for the "Mexico City Cathedral" model across different parameter combinations in subjective tests.

5.1.3 Fast Neural Reconstruction in-the-wild

In a unified manner, we additionally tested our Fast Neural Reconstruction in-the-wild algorithm (Section 4.1.3) on the proposed dataset with the proposed evaluation pipeline. The quantitative results are presented in Table 9, while qualitative results are presented in Figure 33.

Table 9: Quantitative results for Fast Neural Reconstruction in-the-wild. Results on image-based reference and non-reference metrics are provided, along with geometry-based metrics.

Scene	No. images	PSNR	SSIM	LPIPS	BRISQUE	NIQUE	HAUSDORFF	L2
Dataset	50	24.402 ± 3.080	0.475 ± 0.235	0.264 ± 0.094	35.567 ± 23.12	20.06 ± 0.64	0.122 ± 0.048	0.009 ± 0.008
	125	27.993 ± 3.49	0.804 ± 0.093	0.199 ± 0.087	26.543 ± 15.31	20.31 ± 0.59	0.105 ± 0.057	0.008 ± 0.005
	250	**28.091 ± 3.59**	**0.808 ± 0.095**	**0.196 ± 0.087**	25.67 ± 15.64	20.34 ± 0.55	0.105 ± 0.1	**0.006 ± 0.003**
	500	27.17 ± 2.82	0.778 ± 0.073	0.215 ± 0.077	**23.55 ± 17.5**	**20.15 ± 0.47**	**0.097 ± 0.058**	0.007 ± 0.005

The experimental results demonstrate that model performance correlates positively with the quantity of input images. Specifically, image-based quality metrics achieve optimal performance on the 250-image subset, while geometric quality metrics indicate superior results with larger image sets, though the improvement over the 250-image subset remains marginal. These findings suggest that while rendering quality plateaus beyond a certain input threshold, geometric reconstruction accuracy continues to benefit from additional input data.

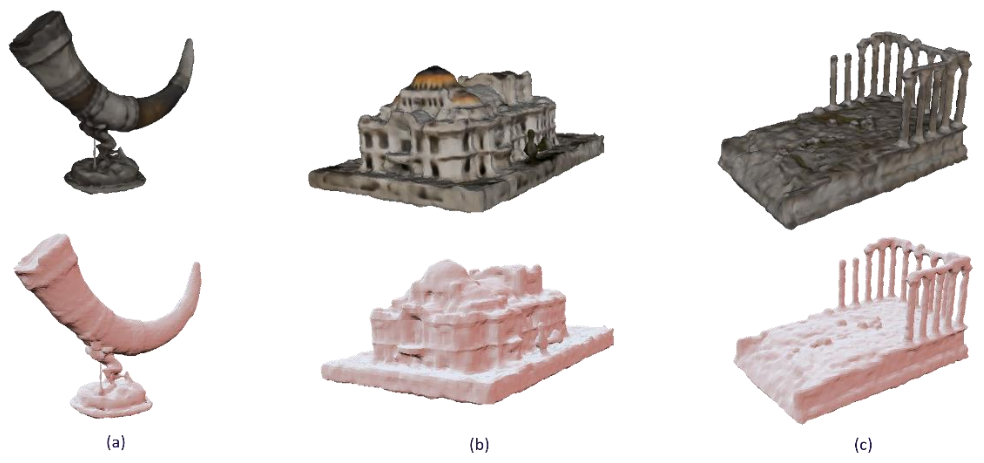


Figure 33: Fast Neural Reconstruction in-the-wild qualitative results. The top row presents textured, while the bottom row presents geometric results. (a) Horn-of-salt. (b) Palace of fine arts. (c) Roman temple of Evora.

This distinction has important implications for application-specific deployment strategies. Use cases requiring high geometric precision – such as architectural documentation, cultural heritage preservation, or engineering applications – will benefit from comprehensive image datasets to achieve optimal reconstruction accuracy. Conversely, applications prioritising visual fidelity over geometric precision – including XR experiences, entertainment content, and immersive visualisation – can achieve satisfactory results with reduced input requirements, thereby minimising data collection overhead and processing complexity.

5.1.4 Fast NeRF in-the-wild

We also evaluated our Fast NeRF in-the-wild pipeline against the image-based metrics, as this method does not produce explicit geometric outputs suitable for geometric evaluation. The quantitative results are presented in Table 10. Additionally, qualitative results on the six scenes of the dataset proposed by UPM are presented in Figure 34

Table 10: Quantitative results of Fast NeRF-in-the-wild. Here only the image-based metrics are considered, since our NeRF pipeline does not provide consistent geometry in the explicit sense.

Scene	No. images	PSNR	SSIM	LPIPS	BRISQUE	NIQUE
Dataset	50	20.716 ± 0.59	0.6431 ± 0.06	0.3169 ± 0.0606	48.012 ± 11.976	19.58 ± 0.52
	125	20.798 ± 1.049	0.679 ± 0.046	0.361 ± 0.081	60.302 ± 10.411	**19.256 ± 0.675**
	250	21.711 ± 0.647	0.674 ± 0.055	0.364 ± 0.085	61.482 ± 12.924	19.329 ± 0.894
	500	**21.991 ± 0.679**	**0.726 ± 0.051**	**0.302 ± 0.075**	**52.132 ± 12.125**	19.664 ± 0.704

The results demonstrate a consistent improvement in rendering quality with increased input data. PSNR values show steady enhancement from 20.716dB (50 images) to 21.991dB (500 images), indicating improved signal-to-noise ratio and overall image fidelity. SSIM scores exhibit the most significant improvement, reflecting enhanced structural preservation in the reconstructed views. Unlike previous geometric reconstruction analysis, these results clearly demonstrate that the pure NeRF implementation benefits consistently from additional input images across all evaluated metrics except from NIQUE, which exhibits a different behaviour, as it assesses intrinsic image properties that may not correlate directly with reconstruction improvements.

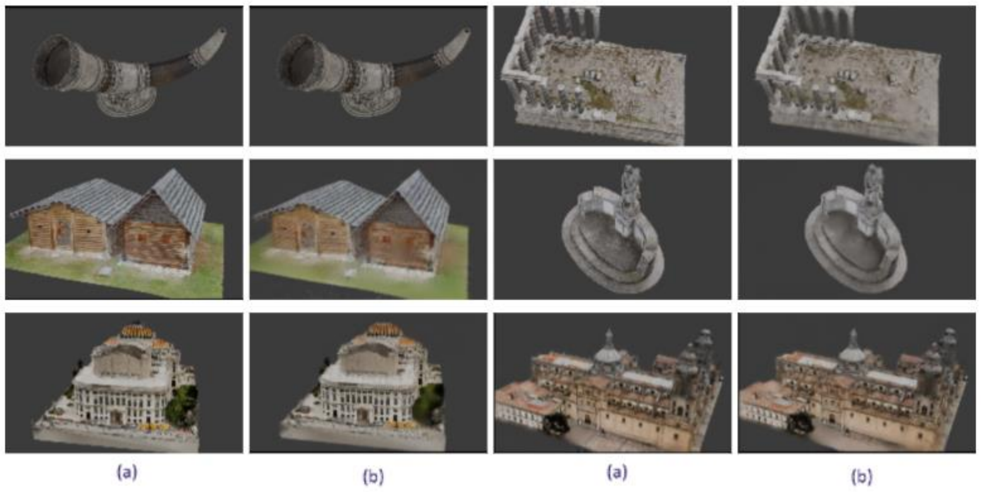


Figure 34: Qualitative results on the six scenes of the proposed dataset. (a) The original image. (b) The corresponding rendered image with Fast NeRF in-the-wild.

5.1.5 3D Gaussian Splatting

We conducted evaluation of our 3DGS pipeline on two representative scenes of the proposed dataset: Horn of Salt Diggers and Schwarzenbach houses. All experiments followed a consistent protocol: for each train/test split, camera poses where estimated via COLMAP from the training images, and the 3DGS model was trained for 10,000 iterations. Quantitative results are summarised in

Table 11, while qualitative comparisons are illustrated in Figure 35. Here we calculate only image-based metrics as geometry in the sense of a mesh is not represented in 3DGS.

Table 11: Quantitative results for 3DGS on different splits of the proposed dataset.

Dataset	**# Training images**	**Metrics**		
		PSNR	**SSIM**	**LPIPS**
Schwarzenbach houses	50	29.286	0.898	0.125
	125	32.070	0.922	0.096
	250	31.238	0.921	0.098
	500	31.759	0.928	0.092
Horn of salt	50	27.058	0.907	0.111
	125	38.161	0.989	0.016
	250	**40.266**	**0.990**	**0.015**
	500	40.125	0.990	0.015

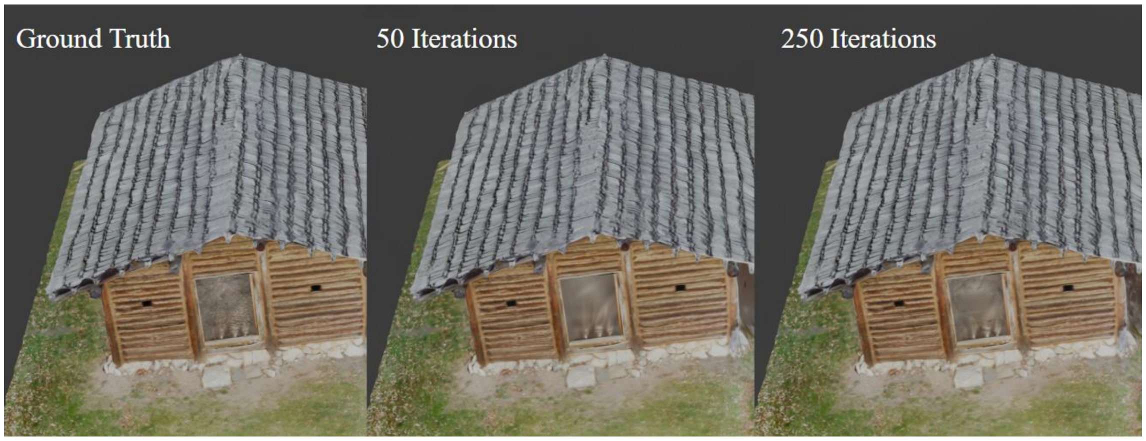


Figure 35: Visual comparison between the ground truth (left) and renderings from models trained with 50 and 250 images (middle and right respectively) on the Schwarzenback Houses dataset. While the model trained with 50 images produces a reasonably detailed

rendering, it lacks accuracy in regions with limited view coverage – such as the building's entrance – highlighting the importance of view diversity over sheer quantity.

6 Human-centred Reconstruction, Volumetric and Free-Viewpoint Video

This section presents specialised 3D reconstruction and volumetric video technologies specifically designed for capturing and rendering human subjects in XR environments. Unlike the general scene reconstruction services described in Section 4, these technologies address the unique challenges of human representation, including sparse viewpoint requirements, real-time performance constraints, and the complex dynamics of human movement and appearance.

6.1 Human-centred NeRF

While D4.1 demonstrated NeRF's effectiveness for realistic novel view synthesis in dynamic, human-centred scenarios, significant usability challenges remain. Traditional NeRF implementations require numerous camera viewpoints and extensive view overlap through SfM, creating complex, costly setups unsuitable for human-centred applications. Additionally, standard NeRFs lack generalisation capabilities across different scenes.

To address these limitations, we developed GDNeRF[81] (accepted at ICME 2025[82]), a method capable of high-quality view synthesis using sparse camera configurations with minimal viewpoints. The algorithm leverages depth map information to construct probabilistic feature volumes from limited source images. Key innovations include: a 3D CNN generator for processing ambiguous and occluded scene information; Style codes (similar to StyleGAN[83]) for gradual feature volume enhancement; A coarse-to-fine depth estimation strategy for improved rendering efficiency.

Evaluation on the CWI[84] (7 cameras) and ActorsHQ[85] datasets demonstrates superior performance in sparse camera settings where existing generalisable NeRF approaches produce artifacts and blurred renders. GDNeRF successfully synthesises target views using only the three closest source views, significantly reducing setup complexity while maintaining rendering quality. Figure 36 provides an overview of the method.

[81] Sergio Montoya, Ivan Huerta, Josep Escrig. GDNeRF: Generalizable Depth-based NeRF for sparse view synthesis. IEEE International Conference on Multimedia & Expo (ICME), June 2025
[82] https://2025.ieeeicme.org/
[83] Tero Karras, Samuli Laine, Timo Aila; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4401-4410
[84] Reimat et al., "Cwipc-sxr: Point cloud dynamic human dataset for social-xr," in Proceedings of the 12th ACM Multimedia Systems Conference,
2021, pp. 300–306
[85] Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, Matthias Nießner; ACM Transactions on Graphics (SIGGRAPH), 2023, 12 pp.

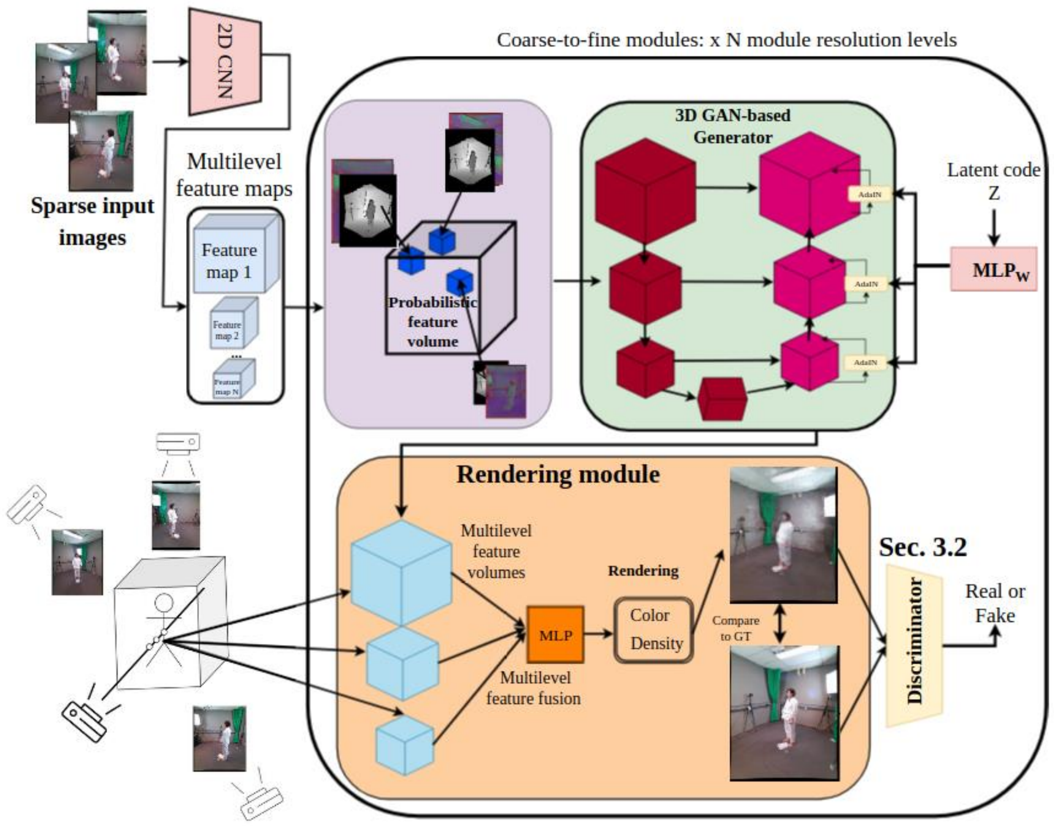


Figure 36: From sparse few views, our method extracts a set of feature maps which are projected into our probabilistic feature volume based on the source views depth maps. Our generator module processes the probabilistic feature volume to generate a feasible multilevel feature volume. These volumes contain features that the volumetric renderer uses to synthesize the target view in real time, without the need of per-scene training.

In addition to what was discussed in Section 4.4.6 of D4.1, we have incorporated a generative prior which improves visible features and generates occluded or missing content in the scene for the sparse view setting.

6.1.1 Generative rendering volume

When addressing sparse novel view synthesis, it is crucial to incorporate a generative prior to infer missing information based on the contextual source images. To achieve this within our framework, we adopt the generator from StyleGAN and the discriminator from StyleGANv2[86], which have been studied extensively in recent literature[87].

[86] Karras et al., "Analyzing and improving the image quality of StyleGAN," in Proc. CVPR, 2020
[87] Zhou et. al, "Point-stylegan: Multi-scale point cloud synthesis with style modulation," Computer Aided Geometric Design, vol. 111, pp. 102309, 2024.

Specifically, the generator $\mathcal{G}$ is a 3D CNN that processes the probabilistic feature volume F_k and produces multiple outputs at different resolutions. The generator follows a UNet-like architecture[88], incorporating AdaIN layers[89] to embed the generative prior. We employ the Non-Saturating GAN objective[90]. The generator is trained to produce feasible and realistic images given the sparse source views. To simplify notation, we encapsulate the generation and volumetric rendering steps using $\mathcal{G}(\cdot)$. Therefore, the function $\mathcal{G}(s, w)$ outputs a rendered image conditioned on the source views s and a latent code w.

The discriminator, besides distinguishing between real and generated images, also analyses whether the generated image is coherent with the context of the source views. This helps the generator produce outputs that are not only visually realistic but also geometrically and texturally consistent with the input. Additionally, we experiment with incorporating both synthesized depth information and ground truth depth as an auxiliary supervision term. This encourages the predicted depth to align with the ground truth depth maps, improving the spatial fidelity of the rendered views.

6.1.2 Experiments

We conduct both qualitative and quantitative evaluations, alongside comparisons with competing approaches. Our quantitative evaluation involves measuring three key metrics, PSNR, SSIM, and LPIPS[91]. We experimented with two datasets: CWI[84] and ActorsHQ[85], as introduced in Section 4.4.2 of Deliverable D4.1.

6.1.3 Ablation study

Our GDNeRF model is primarily compared to ENeRF[92], the current state-of-the-art method for generalizable NeRFs. Unlike ENeRF, which relies solely on image-based rendering (IBR), our method combines both IBR and model-based rendering (MBR). Specifically, we leverage a probabilistic feature volume in conjunction with a generative model to better handle ambiguous and occluded regions—scenarios where ENeRF tends to struggle, especially under sparse camera setups. This sparse-view limitation is one of the key challenges that GDNeRF aims to address. We present a detailed ablation study in Table 12, evaluating performance across three datasets: DTU[93], CWI[84], and ActorsHQ[85]. In this study: **GDNeRF-Multi** refers to the model variant using a multilevel feature volume representation. **GDNeRF-GANs** refers to the full model, incorporating both multilevel features and a generative rendering module.

[88] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer international publishing.
[89] Chen et al., "On self modulation for generative adversarial networks," arXiv preprint arXiv:1810.01365, 2018
[90] Goodfellow et al., "Generative adversarial networks," Communications of the ACM, vol. 63, no. 11, pp. 139–144, 2020.
[91] Zhang et al., "The unreasonable effectiveness of deep features as a perceptual metric," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 586–595.
[92] Lin et al., "Efficient neural radiance fields for interactive free-viewpoint video," in SIGGRAPH Asia Conference Proceedings, 2022.
[93] Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., & Aanæs, H. (2014). Large scale multi-view stereopsis evaluation. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 406–413. IEEE.

The results show that even the base GDNeRF model, which uses only the probabilistic feature volume, significantly outperforms ENeRF across all datasets. Furthermore, adding the multilevel render volume—inspired by ZipNeRF's[94] rendering technique—enhances results on the CWI and ActorsHQ datasets. Finally, the inclusion of the generative rendering volume leads to further improvements in rendering quality and consistency.

The results on the ActorsHQ dataset fulfil the requirements of codes NF.156.1, NF.157.1 and NF.158.1 of the 3D Reconstruction Objective KPIs.

Table 12: Results on DTU, CWI, and ActorsHQ for the different components of GDNeRF and ENeRF. Our model significantly outperforms ENeRF. Even the baseline GDNeRF, utilising only the probabilistic feature volume, already surpasses its performance.

Methods	CWI			ActorsHQ		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
ENeRF	14.523	0.471	0.446	21.046	0.807	0.177
GDNeRF	18.504	0.665	0.427	26.445	0.900	0.116
GDNeRF-Multi	18.328	**0.666**	0.433	**26.568**	0.901	0.112
GDNeRF-GANs	**18.897**	0.648	**0.381**	25.446	**0.911**	**0.097**

Our model consistently outperforms ENeRF in sparse setting scenarios, particularly on CWI. Qualitatively, this difference is evident in Figure 37. While ENeRF struggles to render the person adequately due to the sparse input views, GDNeRF produces high-quality renders. Figure 38 depicts the qualitative results on the ActorsHQ dataset. As can be seen, GDNeRF outperforms ENeRF, exhibiting greater consistency, higher quality and far less artifacts. Finally, we provide the detailed integration API in Annex I – Section 11.2.

[94] Barron, Jonathan T., et al. "Zip-nerf: Anti-aliased grid-based neural radiance fields." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023.

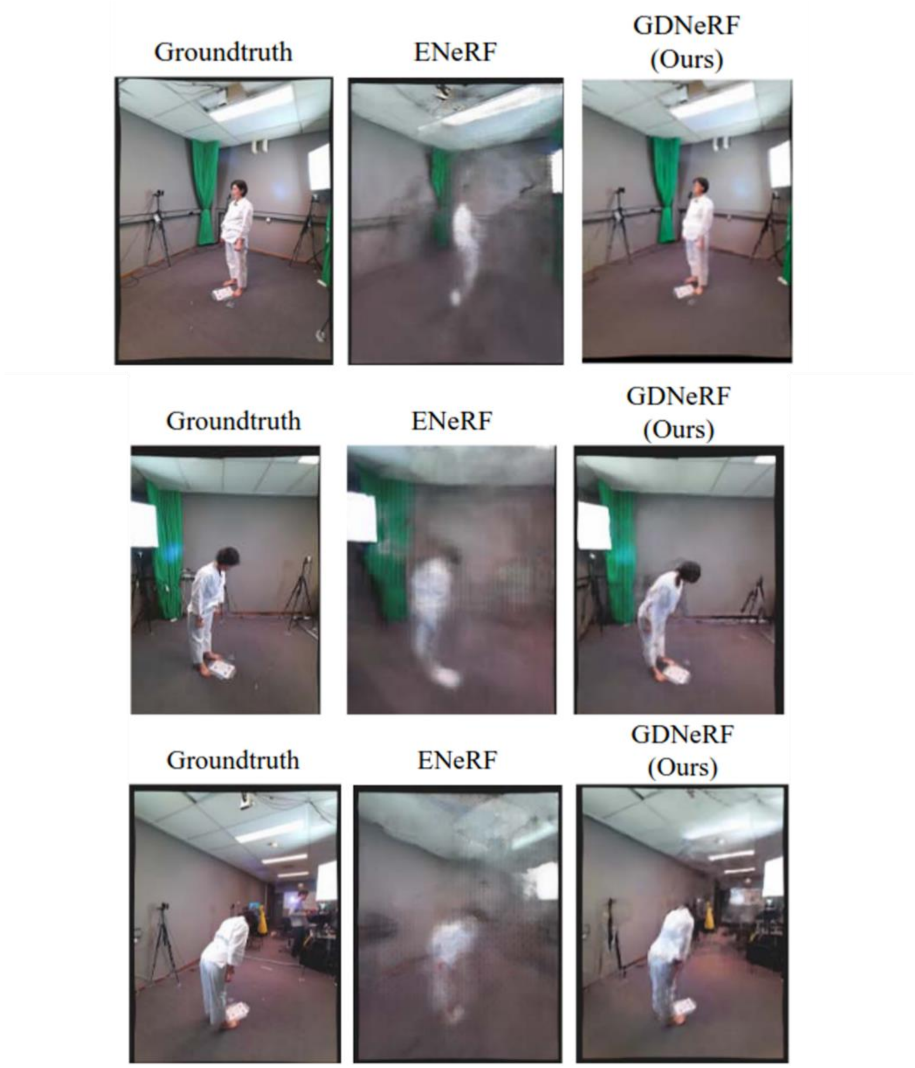


Figure 37: Comparison between ENeRF and GDNeRF on CWI. ENeRF generates many artefacts and holes compared to our GDNeRF

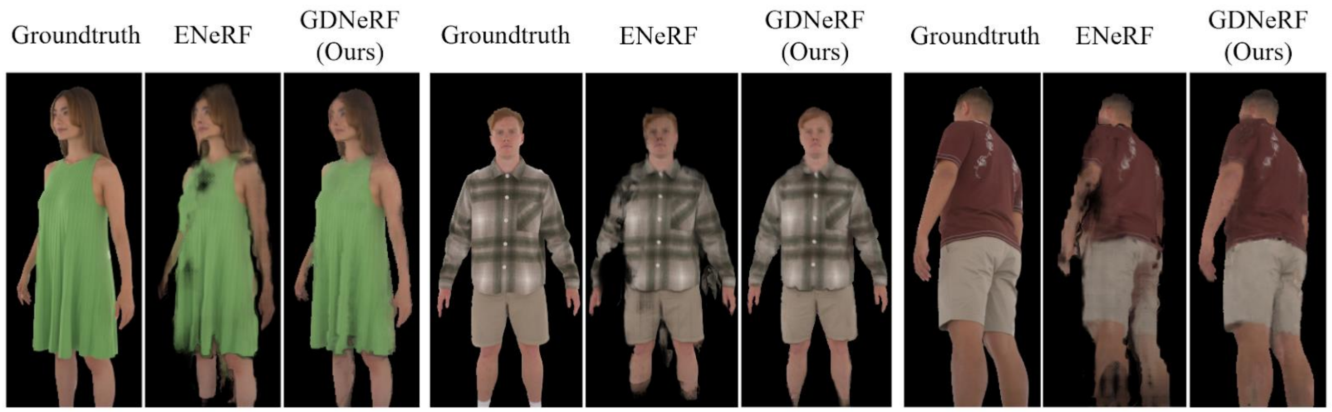


Figure 38: Comparison between ENeRF and GDNeRF in ActorsHQ. ENeRF generates many artifacts and holes compared to our GDNeRF.

6.2 Human-centred Gaussian Splatting GDGS

3DGS has emerged as a compelling alternative to NeRFs due to several distinct advantages. Unlike NeRF's computationally expensive volumetric rendering through neural networks, 3DGS enables real-time rendering on standard GPUs by representing scenes as Gaussian primitives. This computational efficiency makes 3DGS particularly suitable for interactive applications including gaming, VR, and robotics. The explicit point-based representation employed by 3DGS offers superior interoperability, manipulation capabilities, and optimisation potential compared to NeRF's implicit volumetric approach. Recent research demonstrates that 3DGS excels in preserving detailed geometry and texture fidelity[95]. These advantages have motivated our focus on 3DGS-based algorithms, leading to the development of a novel method for generalisable sparse view synthesis.

Dynamic reconstruction captures 3D scenes with moving objects and deformations. HyperReel[96] uses geometric primitives with memory-efficient volume representation, while 4DGS[97] extends 3DGS to spatio-temporal 4D volumes. LongVolCap[98] handles extended sequences by modelling temporal redundancy. However, these methods require substantial camera arrays – approximately 60 cameras for complete 360^{o} coverage.

Human-oriented reconstruction leverages parametric templates like SMPL[99] as priors. HumanNeRF[100] reconstructs humans from single-camera videos using canonical poses and deformation models. SIFU[101] enhances textures through cross-attention and diffusion, while HumanSplat[102] predicts Gaussian properties from single images using human priors. SplattingAvatar[103] combines mesh-based shape modelling with 3DGS. Despite these advances, achieving real-time, generalisable sparse-view synthesis remains challenging.

[95] Huang, Binbin, et al. "2d gaussian splatting for geometrically accurate radiance fields." *ACM SIGGRAPH 2024 conference papers*. 2024.

[96] Attal, Benjamin, et al. "HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

[97] Duan, Yuanxing, et al. "4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes." *arXiv e-prints* (2024): arXiv-2402.

[98] Xu, Zhen, et al. "Representing long volumetric video with temporal gaussian hierarchy." *ACM Transactions on Graphics (TOG)* 43.6 (2024): 1-18.

[99] Loper, Matthew, et al. "SMPL: A skinned multi-person linear model." *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023. 851-866.

[100] Weng, Chung-Yi, et al. "Humannerf: Free-viewpoint rendering of moving people from monocular video." *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*. 2022.

[101] Zhang, Zechuan, Zongxin Yang, and Yi Yang. "Sifu: Side-view conditioned implicit function for real-world usable clothed human reconstruction." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

[102] Pan, Panwang, et al. "Humansplat: Generalizable single-image human gaussian splatting with structure priors." *Advances in Neural Information Processing Systems* 37 (2024): 74383-74410.

[103] Shao, Zhijing, et al. "Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

6.2.1 Methodology

With the intention of improving our GDNeRF, we substitute the probabilistic feature volume construction and ZipNeRF[104] rendering scheme with a per-pixel Gaussian splatting prediction. In Figure 39, we show the main components of GDGS. Specifically, GDGS consists of the following components:

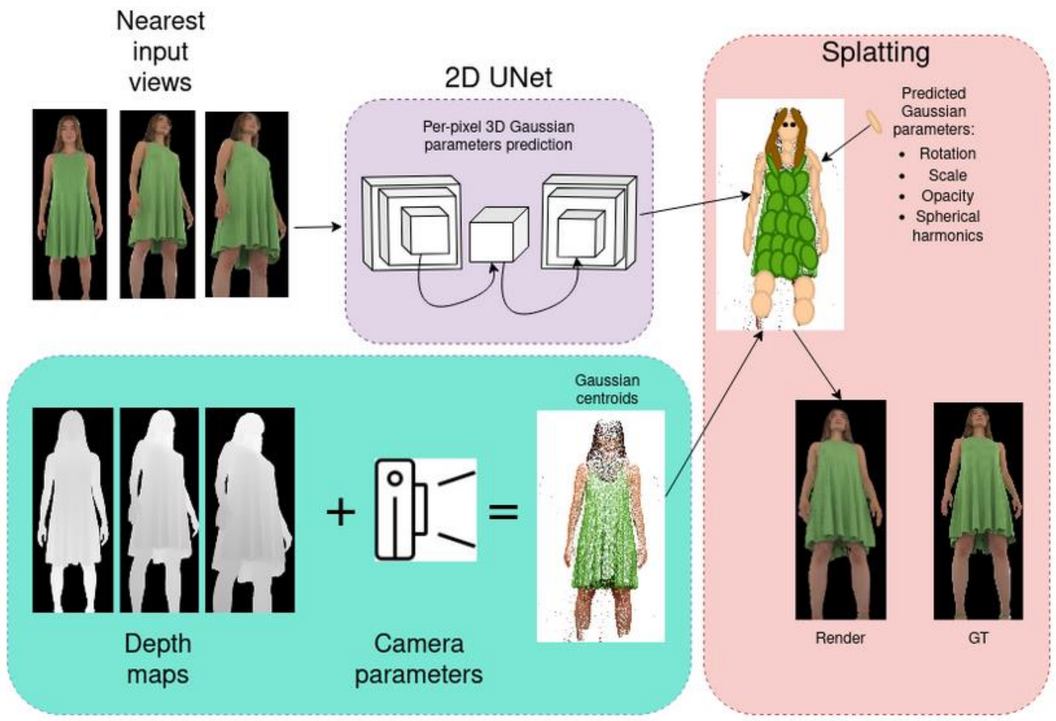


Figure 39: From sparse views, our method extracts a set of feature maps which are transformed into per-pixel 3D Gaussian primitives. A Gaussian rasterizer is used to synthesise the target view in real time, without the need of per-scene training.

- **Nearest Input View Selection:** Selects the input view that is closest to the target direction to be synthesised.
- **2D UNet with Cross-View Attention:** This module includes convolutional layers that extract features from each input view independently, along with cross-view attention modules that enable information sharing across views. The cross-view attention mechanism helps determine which features are important in each view and how they can complement each other.
- **Gaussian Prediction Layer:** Based on the feature maps generated by the 2D UNet, a final layer predicts per-pixel 3D Gaussian attributes, including position offset, rotation, scale, opacity, and spherical harmonics. The final 3D positions of the Gaussians are obtained by computing 3D coordinates from depth maps and camera parameters, then adding the predicted position offsets.
- **Splatting:** After applying the input mask to filter the per-pixel Gaussians, all remaining Gaussians are merged into the scene and rendered using a Gaussian rasterizer.

[104] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., & Hedman, P. (2023). Zip-nerf: Anti-aliased grid-based neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 19697-19705).

6.2.2 Results

Our GDGS model is compared to GDNeRF and ENeRF, the current state-of-the-art method for generalizable NeRFs. We present a detailed ablation study of the key components of GDGS and a comparison to previous approaches in Table 13, evaluating performance in the ActorsHQ dataset. The results on the ActorsHQ dataset fulfil the requirements of codes NF.156.1, NF.157.1 and NF.158.1 of the 3D Reconstruction Objective KPIs.

Table 13: Results on ActorsHQ for the different components of GDGS. Our model significantly outperforms ENeRF and GDNeRF. Even the baseline GDGS already surpasses the performance of previous methods.

Methods	ActorsHQ		
	PSNR	SSIM	LPIPS
ENeRF	21.046	0.807	0.177
GDNeRF	26.568	0.911	0.097
GDGS	27.597	0.917	0.078
+ Completion	27.627	0.908	0.081
+ Position offset	29.746	0.923	0.041
+ Camera embeddings	**31.805**	**0.938**	**0.011**

The experiments of adding different components to GDGS are the following:

- **Completion**: We experiment with incorporating a completion module that processes the predicted Gaussians to generate a new set of Gaussians aimed at filling regions not visible from the input views. To achieve this, we adapt the architecture from PoinTr[105] to handle the attributes of the 3D Gaussians and produce an additional set of points. However, this approach yielded limited results: the completed regions appear unrealistic and tend to form smooth blobs, as illustrated in Figure 40.
- **Position offset**: Due to potential noise in camera calibration and/or input depth maps, we augment the RGB input by concatenating the 3D positions of each point in world coordinates. This allows the model to learn to adjust the positions of individual 3D Gaussians when necessary. The impact of this addition on rendering quality is shown in Table 13 and Figure 41.
- **Camera embeddings**: Some scenes exhibit camera-dependent illumination conditions that are difficult to model without incorporating environment- or camera-specific latent embeddings. To address this, we concatenate Plücker camera embeddings to the input, enabling the model to account for per-camera illumination effects. These embeddings help the model correct illumination inconsistencies and fuse features in a way that is independent of the specific camera's lighting conditions.

In Figure 42, we show an example of new view synthesis from 5 views for the Actor01 of ActorsHQ. In Figure 43, we show the results of some free viewpoint frames from a video that we have visualized at 7 FPS with a dynamic 3DGS visualizer.

[105] Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., & Zhou, J. (2021). Pointr: Diverse point cloud completion with geometry-aware transformers. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 12498-12507).

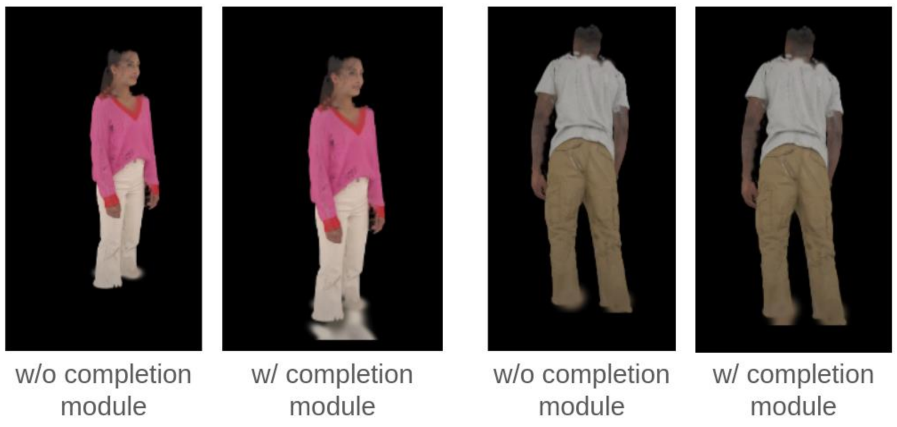


Figure 40: Comparison of a rendered view with GDGS with and without the completion module. We use just 3 views spanning the scene, in order to generate more occlusions. As can be seen, the completion module did not achieve what was initially intended.

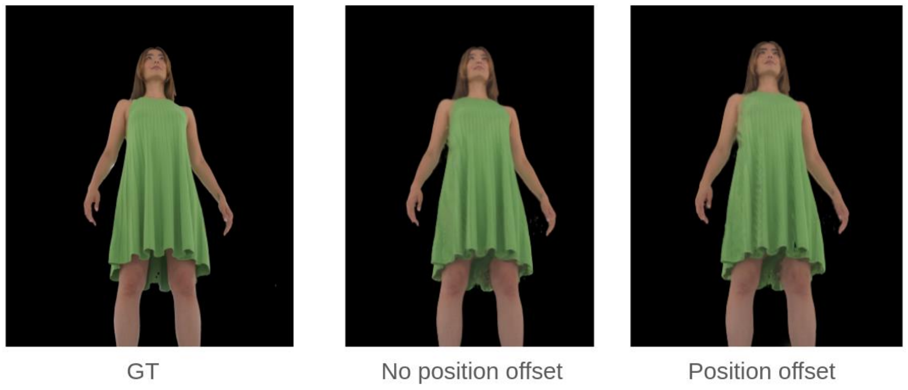


Figure 41: Comparison of a rendered view with GDGS with and without the position offset.

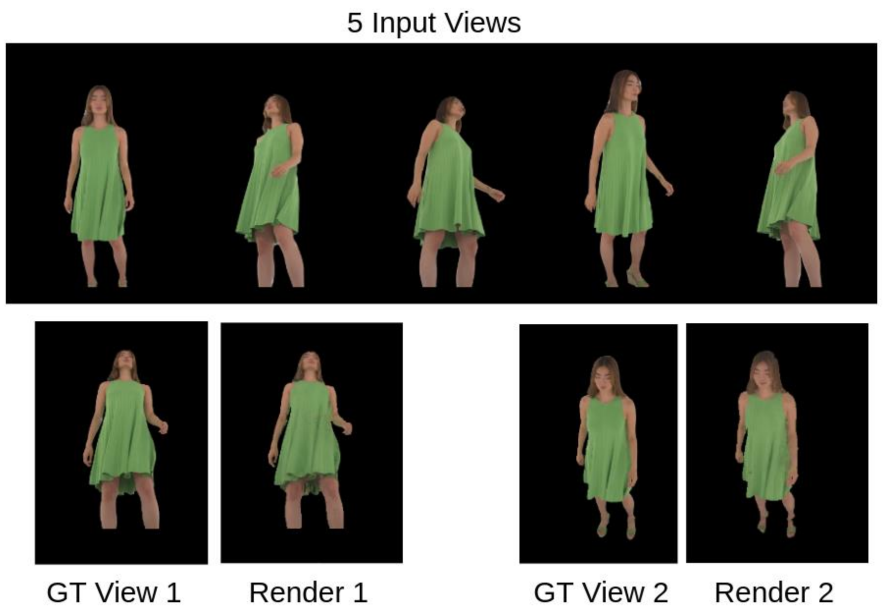


Figure 42: Given 5 input views, GDGS renders a new view in a generalisable manner in real-time.

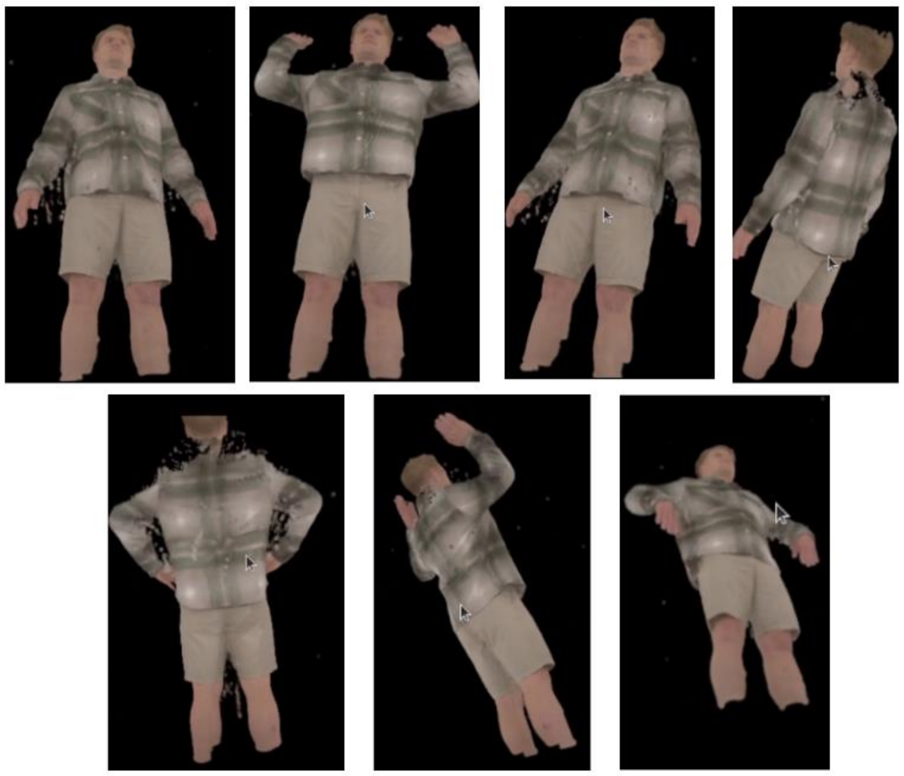


Figure 43: Frames from an online dynamic 3DGS visualiser given 5 input views.

Additionally, we experiment with low-cost manual recordings with a few synchronized ORBBEC cameras. One example of such a setup can be seen in Figure 44, for which we show some results in Figure 45.

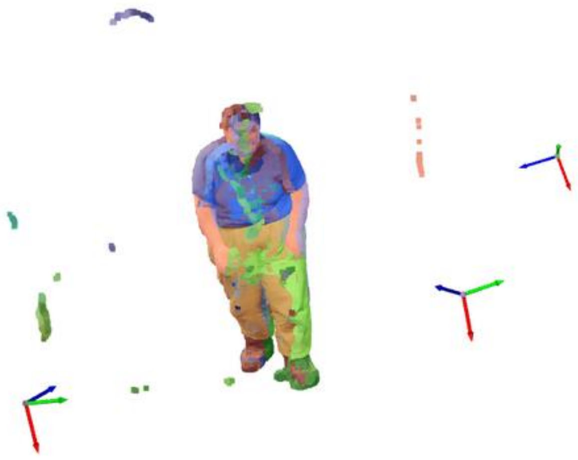


Figure 44: ORBBEC cameras example setup with projected point-clouds from the depth estimation. As can be seen, the algorithm needs to model noise that comes from the depth sensor and the calibration.

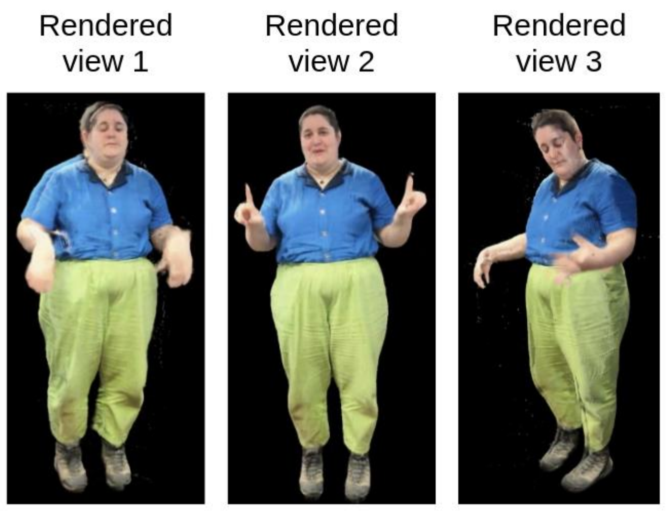


Figure 45: Real-time rendering results from GDGS in our few camera setup.

6.2.2.1 API Implementation

The GDGS API provides an interface for training and running inference on GDGS-based models. It is built using **FastAPI**, with asynchronous job handling through **Celery** and **Redis**. The endpoints and workflow are the same as those in the GDNeRF API (see Annex I – Section 11.2).

6.3 RGB-D based real-time free-viewpoint-video

The Free Viewpoint Video (FVV) functionality provided by XReco is based on the FVV Live system[106, 107]. This technology allows its users to navigate freely around a scene controlling the position of a virtual camera. It is an end-to-end system, covering capture of the scene, video transmission, virtual view synthesis and delivery to the user. It can work in real-time with minimum latency.

The system is divided into 3 main modules: the capture, formed by the cameras and the Capture Servers; the rendering module, which is in charge of rendering the virtual views requested by users; and a WebRTC[108] based interface for users to connect to the system. It also offers a replay module that allows the playback of pre-recorded FVV content. Figure 46 provides a schematic representation of the connections between components. An in-detail explanation of these components was provided in D4.1.

[106] Pablo Carballeira, Carlos Carmona, César Díaz, Daniel Berjón, Daniel Corregidor, Julián Cabrera, Francisco Morán, Carmen Doblado, Sergio Arnaldo, María del Mar Martín, and Narciso Garcíaa, “FVV Live: A Real-Time Free-Viewpoint Video System with Consumer Electronics Hardware,” IEEE Transactions on Multimedia, vol. 24, pp. 2378–2391, 2022.
[107] Pablo Pérez, Daniel Corregidor, Emilio Garrido, Ignacio Benito, Ester González-Sosa, Julián Cabrera, Daniel Berjón, César Díaz, Francisco Morán, Narciso García, Josué Igual, and Jaime Ruiz, “Live Free-Viewpoint Video in Immersive Media Production Over 5G Networks,” IEEE Transactions on Broadcasting, vol. 68, no. 2, pp. 439–450, 2022.
[108] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <https://www.rfc-editor.org/info/rfc7478>.

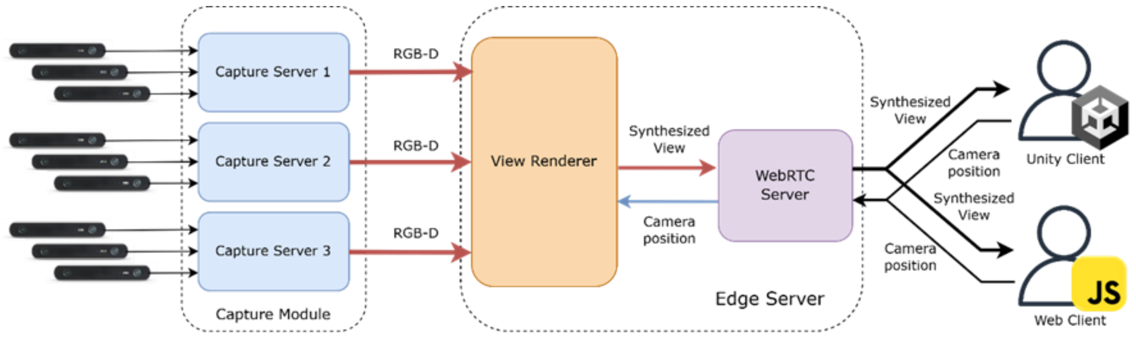


Figure 46: General architecture of the FVV Live system using WebRTC to deliver FVV to the user.

6.3.1 Virtual viewpoint selection for virtual scenario integration

The virtual view rendering pipeline of the FVV Live system has been updated to enable the integration of real-time captured FVV content into virtual scenes[109]. This approach enables inserting an external volumetric video feed, that can be either live or pre-recorded, in real-time and requiring limited complexity to the users' terminals.

The system follows a remote rendering approach, where the user transmits their point of view to the system and a coherent virtual view is then synthesized. The synthesized view is displayed inside the virtual scene on a plane that rotates to always face the user, which is an entity usually called billboard. While this integration is explained in detail in the next subsection, Figure 47 shows an example of a virtual view inserted into a virtual scene.

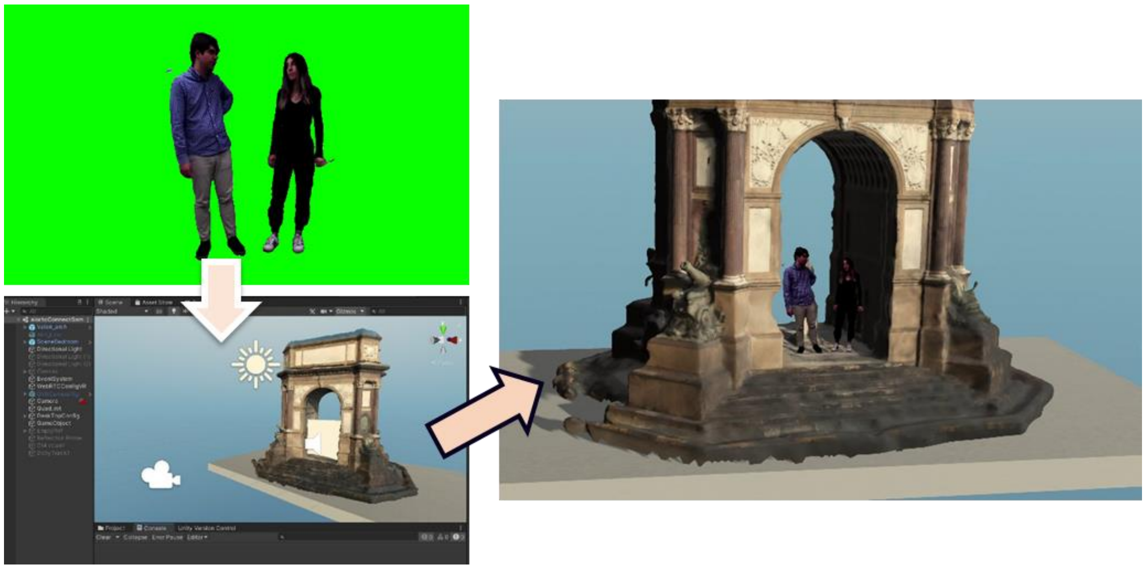


Figure 47: FVV Live synthesised views (left-up) are sent to the user application containing the virtual scene (left-down). They are displayed on a billboard (2D plane) inside the virtual scene (right).

[109] Javier Usón, Victoria Muñoz, Carlos Cortés, Daniel Berjón, Francisco Morán, César Díaz, Jesús Gutierrez, Fernando Jaureguizar, Narciso García, and Julián Cabrera, "Real-time free viewpoint video for immersive videoconferencing," in 2024 16th International Conference on Quality of Multimedia Experience (QoMEX), 2024, pp. 171–174.

The decision of the viewpoint that the FVV Live Virtual View Renderer must choose to make the synthetic view coherent with the virtual scene is made following the diagram in Figure 48. The client application communicates the position of the camera and the billboard (X, Y and Z coordinates) and the horizontal field of view encoded in a JSON message. The View Renderer then uses that information to compute the direction from where the client application camera is watching the billboard and places the FVV virtual camera on the surface of a sphere around the scene.

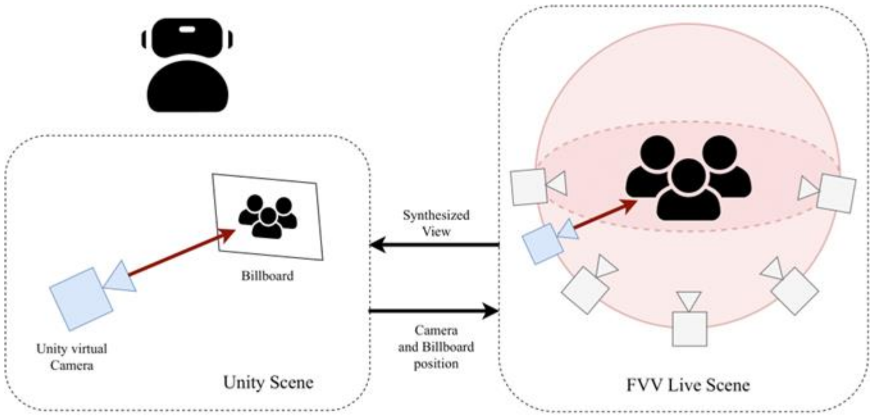


Figure 48: Graphic scheme of the communication pipeline between the client application camera and the virtual camera from FVV Live synthesised by the View Renderer.

The JSON message used to communicate the user position contains the following information:

- Position of the camera in coordinates of the virtual scene (X, Y, Z).
- Euler rotation angles of the camera in coordinates of the virtual scene (yaw, pitch, roll) in degrees.
- Camera field of view in degrees.
- Position of the billboard in coordinates of the virtual scene (X, Y, Z).
- Extra parameters for functionality, such as communicating the radius of the sphere and pausing the video when consuming pre-recorded content.

The complete process, presented in Figure 49, involves the following steps:

1. Given the position of the camera in the virtual scene ($\boldsymbol{c}_{Unity}$), as well as of the billboard ($\boldsymbol{c}_{avatar}$), the Virtual View Renderer computes the direction from where the user is visualizing the avatars as simply:

$$\boldsymbol{d}_{\text{Unity}} = \frac{\boldsymbol{c}_{\text{Unity}} - \boldsymbol{c}_{\text{avatar}}}{|\boldsymbol{c}_{\text{Unity}} - \boldsymbol{c}_{\text{avatar}}|_2}$$

   with $|\cdot|_2$ representing the Euclidean norm of the vector used to normalize it.

2. The sphere around the FVV Live scene is defined with a parameter r (defined by the user) representing the radius. The center of this scene ($\boldsymbol{c}_{\text{Scene}}$) is then defined as the point at r distance from the central reference camera:

$$\boldsymbol{c}_{\text{Scene}} = \boldsymbol{c}_{\text{Cam4}} + r\,\boldsymbol{d}_{\text{Cam4}}$$

with $\boldsymbol{c}_{\text{Cam4}}$ being the position of the centre reference camera and $\boldsymbol{d}_{\text{Cam4}}$ its forward vector.

3. To place the virtual camera on the surface of the defined sphere, its centre ($\boldsymbol{c}_{Virtual}$) is defined as:

$$\boldsymbol{c}_{\text{Virtual}} = \boldsymbol{c}_{\text{Scene}} + r(-\boldsymbol{d}_{\text{Unity}})$$

4. Only the orientation of the virtual camera remains to be defined. FVV Live represents orientation using rotation matrices, which can be built from an orthonormal basis formed by the direction vectors: forward ($\boldsymbol{f}_{\text{Virtual}}$), up ($\boldsymbol{u}_{\text{Virtual}}$) and right ($\boldsymbol{r}_{\text{Virtual}}$). To define this basis, $\boldsymbol{d}_{\text{Unity}}$ is used as a starting point:

$$\boldsymbol{f}_{\text{Virtual}} = \boldsymbol{d}_{\text{Unity}}$$

$$\boldsymbol{r}_{\text{Virtual}} = \frac{\boldsymbol{f}_{\text{Virtual}} \times \boldsymbol{up}}{|\boldsymbol{f}_{\text{Virtual}} \times \boldsymbol{up}|_2}, \quad \text{with } \boldsymbol{up} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\boldsymbol{u}_{\text{Virtual}} = \frac{\boldsymbol{f}_{\text{Virtual}} \times \boldsymbol{r}_{\text{Virtual}}}{|\boldsymbol{f}_{\text{Virtual}} \times \boldsymbol{r}_{\text{Virtual}}|_2}$$

The vector $\boldsymbol{r}_{\text{Virtual}}$ is computed using the positive upward direction (named $\boldsymbol{up}$ here) to ensure that avatars are always rendered standing in place.

This approach was designed to obtain the following advantages:

- The location of the avatars in the virtual scene is independent of the FVV Live scene. This means that the avatars can be placed anywhere on the virtual scene by selecting where the billboard should appear.
- Similarly, the FVV viewpoint is independent of the camera orientation (rotation). FVV always renders the avatars standing in place, the client application engine is the one in charge of handling their location and orientation based on where the billboard is.
- Having the avatars fixed in place greatly reduces the effect of motion-to-photon (M2P) latency. Since the location of the avatars does not depend on the transmitted video, the effects of delay are much less noticeable.

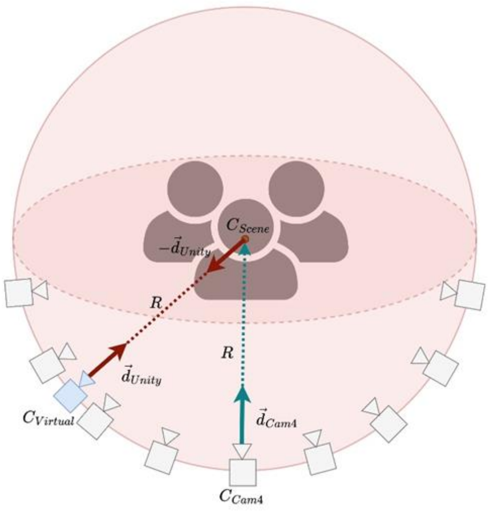


Figure 49: Detailed diagram of the selection of the viewpoint for FVV Live integration in virtual scenarios.

6.3.2 Integration of FVV Live feed with virtual scenarios

To develop experiences where FVV Live is integrated into virtual scenes crafted using 3D content retrieved from the XReco platform, or generated with XReco tools, two communication pipelines have been developed which allow the visualization of FVV Live real-time-captured scenes from applications built with Unity and Unreal.

Both Unity and Unreal applications display the video received from the FVV system onto a billboard object in the engine's scene. This billboard uses a green-screen shader for seamless avatar integration and always faces the camera through dynamic rotation. As users move around, the engine transmits updated camera and billboard positions to ensure the correct synthetic view is received from the FVV. Both engines' applications share a set of different scenarios for the user to choose from dynamically and freely move around them, as well as a laboratory scene (Figure 50) where the user can interact with the displayed objects while viewing the FVV video.

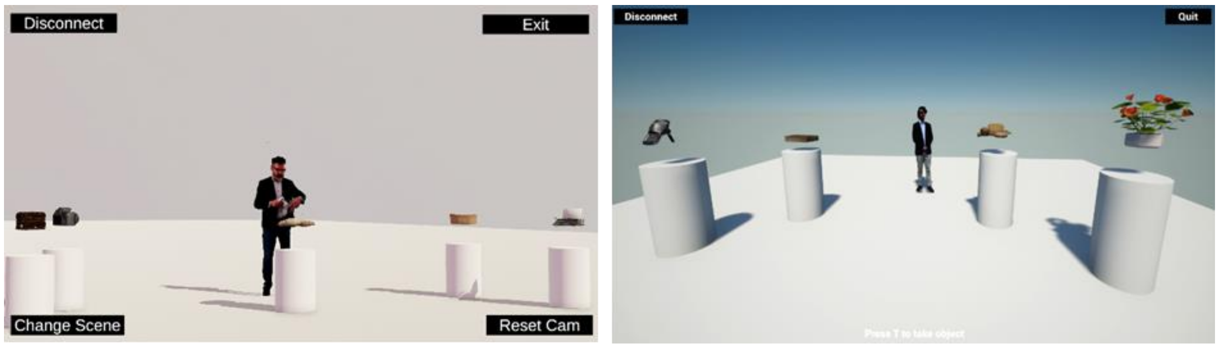


Figure 50: Interactive lab scene in Unity3D (left) and in Unreal Engine (right).

6.3.2.1 Unity implementation

The main elements of this solution are the *WebRTCConfiguration* module, the camera movement script and the billboard, named *Quad_rot* in Unity's hierarchy (Figure 51) and responsible for rendering the FVV Live video.

The *WebRTCConfiguration* contains the script in charge of the connection establishment with the WebRTC server. This module also manages the media reception: when a video or audio track is received, it is added to a *Mediastream*. If it is a video track, its texture is applied onto the billboard of the scene. If it is an audio track, it will be added to an *Audiosource* in the scene and played. This script is also responsible for sending JSON messages to the FVV Live system with the Unity camera position, rotation, field of view and a few more parameters. These messages are only sent when the camera is moving around the scene and also include a few parameters for extra functionalities like "pause" and the FVV Live scene sphere radius (Figure 49) so it can be dynamically updated from the Unity application.

The billboard has two main components: one that manages the texture rendering of the billboard and camera-dependent billboard rotation; and a Chroma-Key shader that controls the Greenscreen effect, the mask colour can be adjusted as well as its threshold sensitivity.

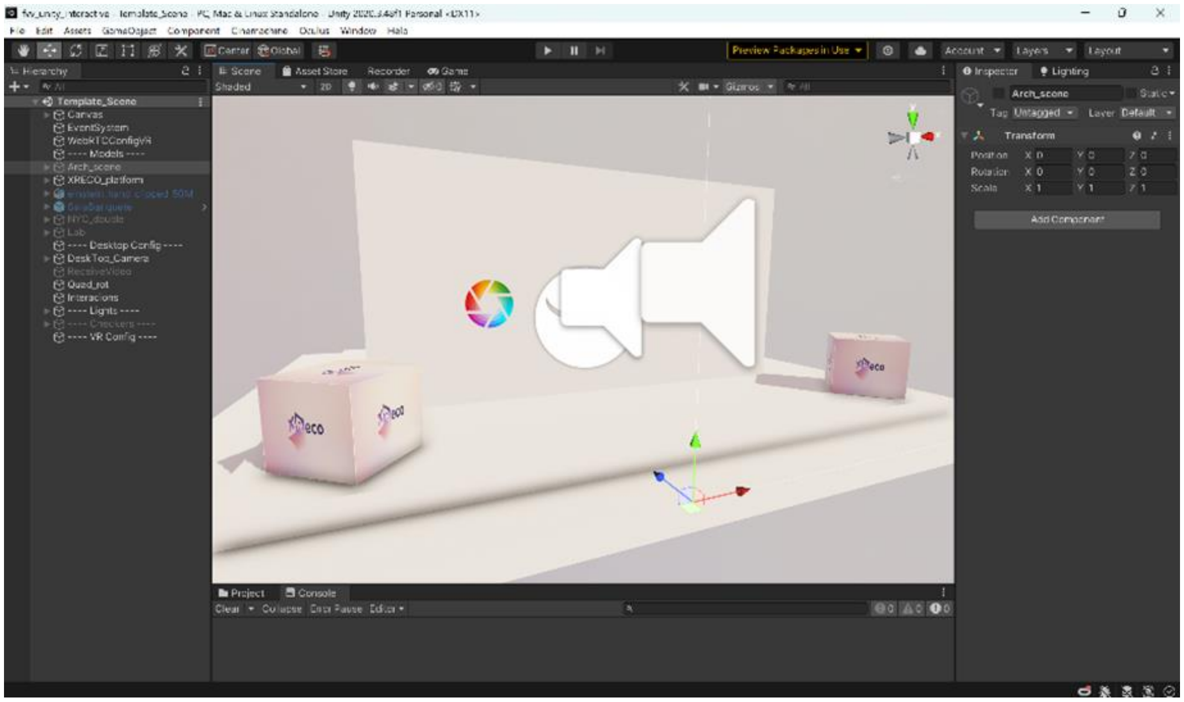


Figure 51: Example of an FVV Live Unity project.

Regarding the communication architecture, the Unity application follows the main approach covered in previous sections. As illustrated in Figure 52 the Unity application runs a WebRTC client, establishing a connection to the WebRTC Server that relays the Free Viewpoint Video (FVV) to the user. To receive the appropriate synthetic view from the FVV via the WebRTC, the Unity client transmits its camera and billboard positions as JSON-formatted WebRTC data channel messages to the server, which transmits this information to the FVV system via UDP. To successfully achieve this implementation, the Unity WebRTC plugin is used on the client side and *aiortc*, a Python library for WebRTC, is deployed on the server.

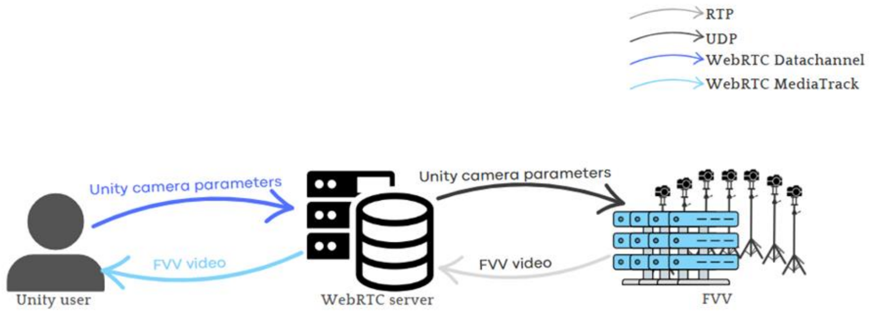


Figure 52: Unity3D implementation architecture.

Desktop (2D screen) and HMD Unity applications were developed to visualize FVV Live scenes. In these applications, the user can choose dynamically between a set of different scenarios such as the Arco de Valentino di Torino, a New York City square, the astrophysical observatory of Albert Einstein from Potsdam, a medieval banquet hall, a stage or the interactive laboratory just by clicking on the "Change Scene" button or pressing the "B" button on the HMD controller.

In the desktop application, the user can freely move around the Unity scene by pressing the right mouse button and using the WASD keys. An in-game screenshot is included in Figure 53. In the laboratory room the user can interact with different displayed objects, just by getting close to the object and left-clicking over it, then the object can be rotated using the left mouse button or be zoomed in/out using the mouse wheel.

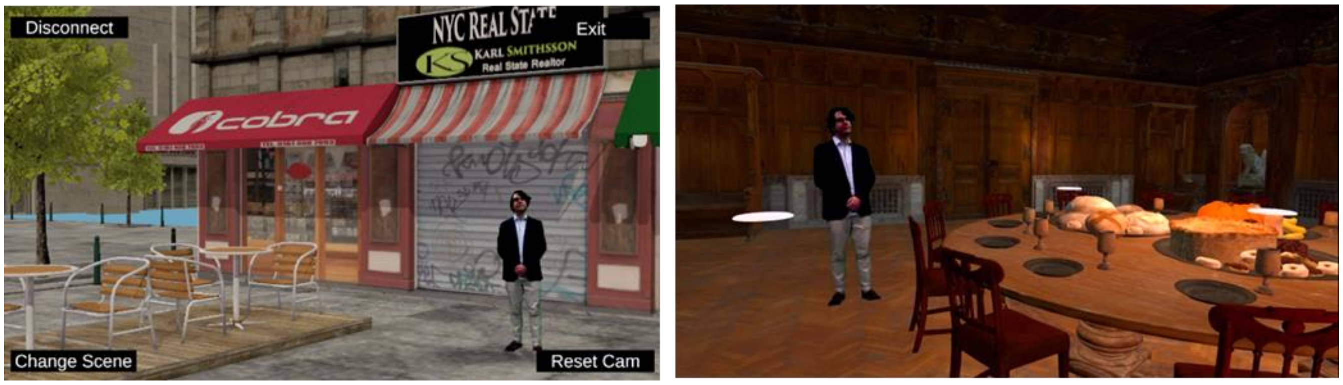


Figure 53: In-game screenshot of an FVV Live Unity-based application (left) and an HMD Unity-based application (right).

In the Unity HMD application, the user can freely move around the Unity scene either by physical movement around the actual space where the user is placed or via teleportation. An in-game screenshot is included in Figure 53. In the laboratory room the user can interact with different displayed objects, just by getting close to the object and pressing the "Secondary Hand Trigger" button on the HMD controller over it, once the user has taken the object, it can be freely manipulated by the user.

6.3.2.2 Unreal implementation

The main elements of this solution are the *Billboard blueprint*, the *Camera blueprint* and the *Stream Media Source* (*MediaStream* in Unreal's content folder), which connects to the server and receives the FVV Live video.

The *Billboard blueprint* is responsible for sending JSON messages to the FVV system with the Unreal camera information. These messages have the same format as the ones used by the Unity implementation. This blueprint also manages the billboard rotation, so it always faces the camera.

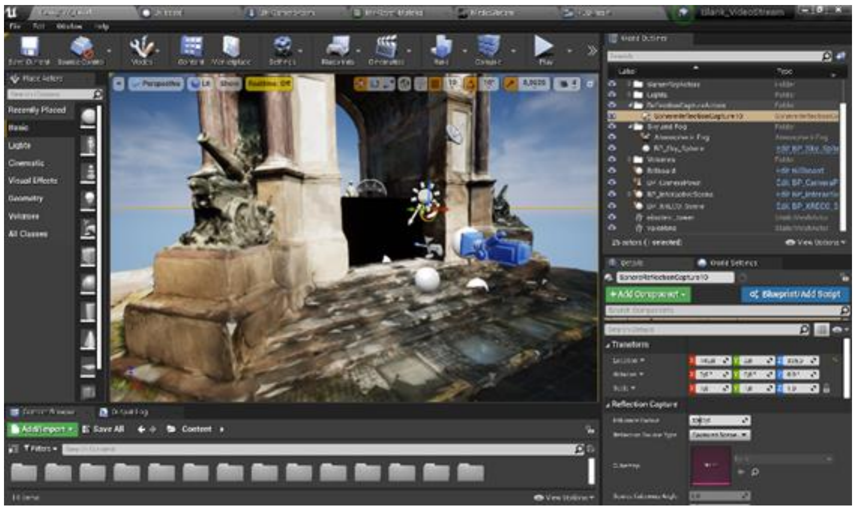


Figure 55: Example of an FVV Live Unreal project.

Regarding communication architecture, a different approach was developed since Unreal does not provide a straightforward implementation of WebRTC. As illustrated in Figure 54, the Unreal application runs an RTSP client, establishing a connection to an RTSP[110] server that relays the FVV Live synthesized view to the user. To achieve this, an *ffmpeg* module is used to receive an RTP[111] stream from FVV Live and send it to the RTSP server in the correct format. To receive the appropriate synthetic view from FVV Live, the Unreal client transmits its camera and billboard positions as JSON-formatted UDP[112] messages directly to the FVV system.

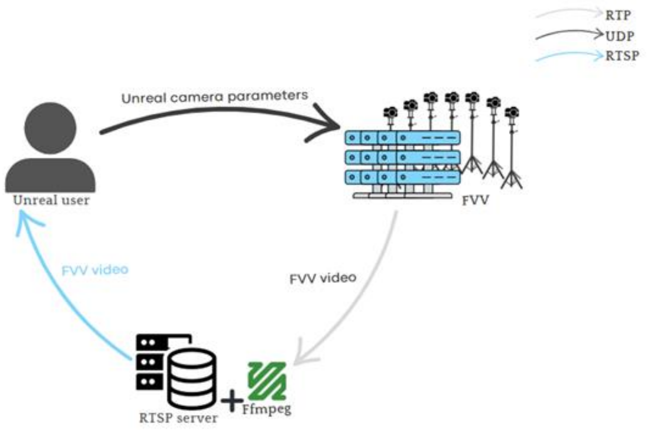


Figure 54: Unreal implementation architecture.

The Unreal Desktop application functionalities are similar to those in the Unity application: the user can choose dynamically between a set of different scenarios such as the Arco de Valentino di Torino, the astrophysical observatory of Albert Einstein from Potsdam, a stage or the interactive laboratory just by pressing the “Space” key. An in-game screenshot is included in Figure 56. The user can also freely move around the Unreal scene by pressing the right click of the mouse and using the WASD keys.

[110] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, DOI 10.17487/RFC2326, April 1998, <https://www.rfc-editor.org/info/rfc2326>.
[111] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.
[112] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

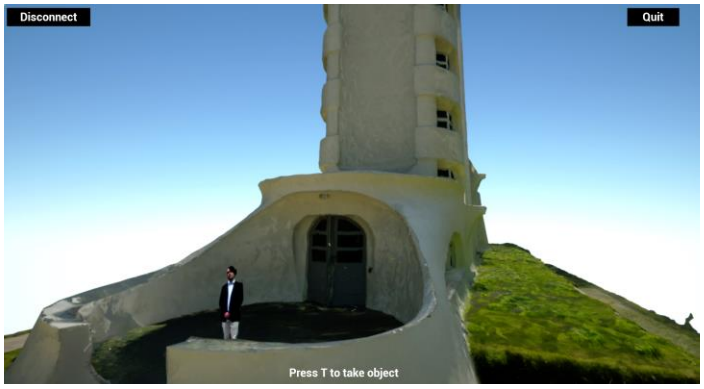


Figure 56: In-game screenshot of the Unreal application.

6.3.3 Virtualized deployment of FVV Live components

The Virtual View Synthesis components (Stream selector, View Renderer and WebRTC Server modules) have been virtualized as Docker images, enabling their deployment in cloud computing environments and edge computing scenarios. This allows the management of several simultaneous users and/or contents by deploying the necessary number of instances of each module. Additionally, a transmission simulation (FVV Replay) module is available to deliver the same pre-recorded content simultaneously to several users. On the other hand, the client applications are not provided as virtualized components since they are meant to run in user terminals.

6.3.3.1 Multiple user management

FVV Live follows a remote rendering approach, where each user requests their specific point of view to an instance of the View Renderer. Three different approaches have been developed to manage multiple users, with all of them involving the deployment of one instance of View Renderer per user:

- Live transmission: to manage several users visualizing a real-time capture feed simultaneously, the RGB+D content is transmitted to the Stream Selector component, which is in charge of its distribution among all the View Renderer instances. Users can communicate with their View Renderer through the WebRTC Server. Figure 57 presents a diagram of this approach.

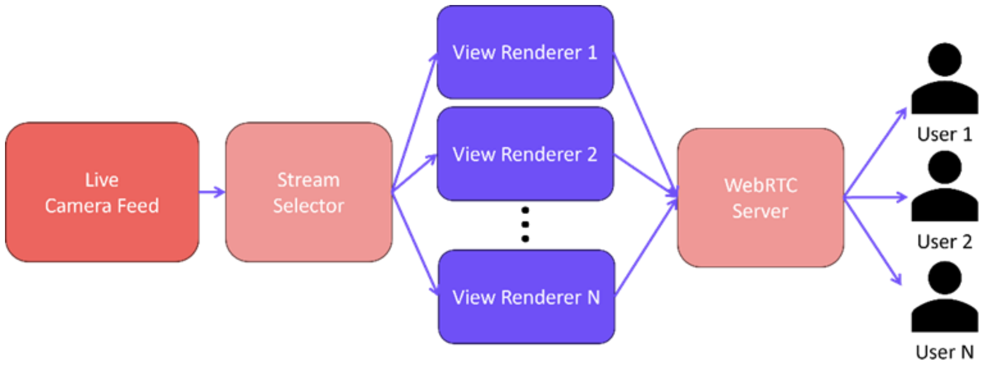


Figure 57: Diagram of the configuration designed to deliver real-time FVV content to multiple simultaneous users.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

- Pre-recorded content transmission (simulation): using the same configuration as in the Live scenario, the FVV Replay module can be used to deliver pre-recorded content to multiple users. In this configuration, all users would watch the same content in a synchronized way. Figure 58 shows this configuration.

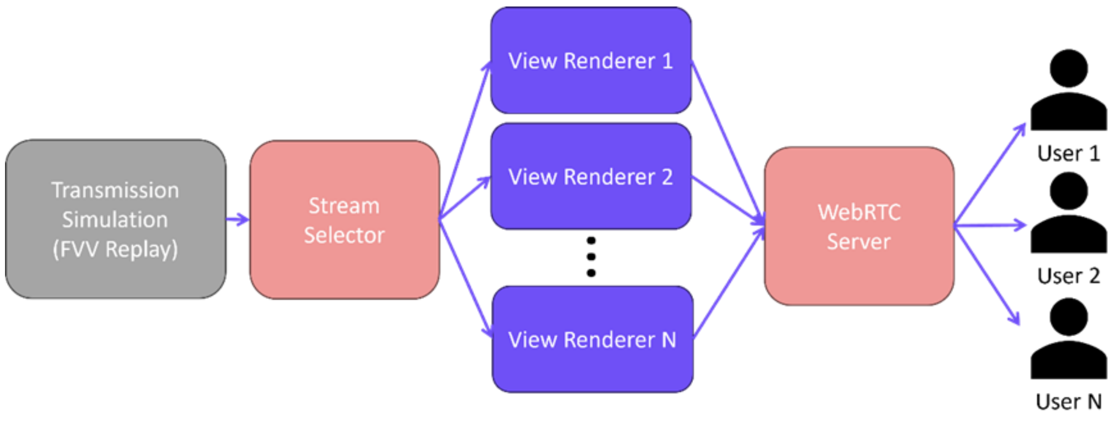
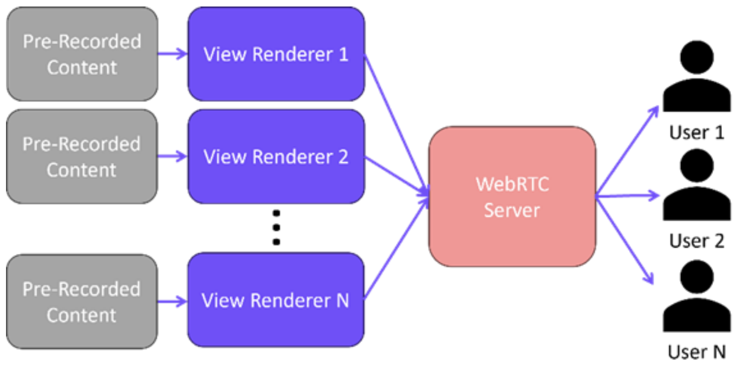


Figure 58: Diagram of the configuration designed to deliver the same pre-recorded FVV content to multiple simultaneous users.

- Pre-recorded content without transmission: View Renderer instances can function in "offline" mode, where the RGB-D content is available in the host machine storage and can be accessed without network transmission. This configuration allows users to consume content in a "video on demand" way, where each user watches it at their own pace (Figure 59).



-

Figure 59: Diagram of the configuration designed to deliver pre-recorded FVV content as VoD to multiple users.

6.3.3.2 Multiple concurrent contents

In a similar way to the management of several users, several View Renderer instances can be deployed to simultaneously deliver multiple different contents to one user. The system is flexible, allowing the combination of both live and pre-recorded content. Figure 60 offers an example of a configuration to deliver live content and two different pre-recorded contents to one user.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

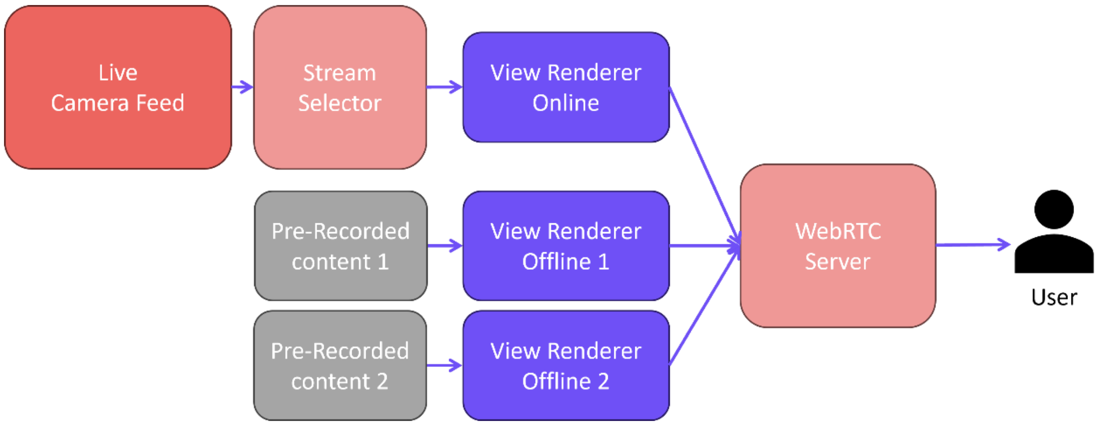


Figure 60: Example of configuration capable of delivering several live and pre-recorded FVV streams to one user.

If enough computing resources are available, these two approaches can be combined to deliver multiple contents to multiple simultaneous users.

6.3.4 Deep learning-based RGB-D content generation

The FVV Live capture module is heavily influenced by the RGB+D information captured by Stereolab ZED cameras. They are consumer-grade stereo cameras with important limitations in their quality and flexibility, such as providing fixed camera lenses and pre-determined resolution configurations (1080p or 720p). The aim to improve the system quality and flexibility motivated the study of depth estimation techniques that could be applied on any kind of camera setup.

Thanks to modern deep learning approaches, monocular depth estimation is now possible. State-of-the-art models can estimate accurate depth information from only one RGB image, greatly reducing the complexity of the setup and the amount of information to process per frame. We performed a study where we tested the performance of three monocular depth estimation models applied to the FVV use case: Depth Anything V2[113], Depth-Pro[114] and UniDepth V2[115].

To achieve this goal, a depth scale adjustment procedure based on multicamera calibration was designed to make the estimation performed on every camera match the scale of the estimation on the rest of the cameras. Figure 61 shows a diagram of the complete process.

[113] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao, “Depth anything v2,” arXiv:2406.09414, 2024.

[114] Aleksei Bochkovskii, Ama¨el Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun, “Depth pro: Sharp monocular metric depth in less than a second,” arXiv, 2024.

[115] Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool, “UniDepthV2: Universal monocular metric depth estimation made simpler,” 2025.

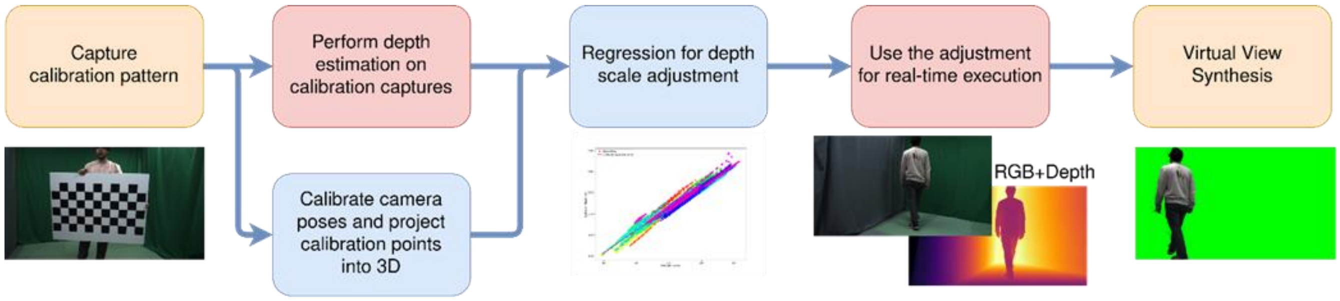

Capture calibration pattern
Perform depth estimation on calibration captures
Calibrate camera poses and project calibration points into 3D
Regression for depth scale adjustment
Use the adjustment for real-time execution
RGB+Depth
Virtual View Synthesis

Figure 61: Monocular depth for volumetric capture pipeline diagram.

The first step involves a synchronized recording of several instances of a pattern (in this case the OpenCV checkerboard) with different positions and orientations. A large number of visual features are extracted from these captures, which are then fed into the OpenMVG Structure from Motion algorithm to perform camera calibration. The results include a precise calibration of the intrinsic and extrinsic parameters of all the cameras in the setup and a point cloud of the 3D position of every feature triangulated during the calibration process. Additionally, the resulting calibration can be scaled making the distance between features match the square size of the calibration pattern, obtaining a calibration with a scale similar to the real scene (in millimetres).

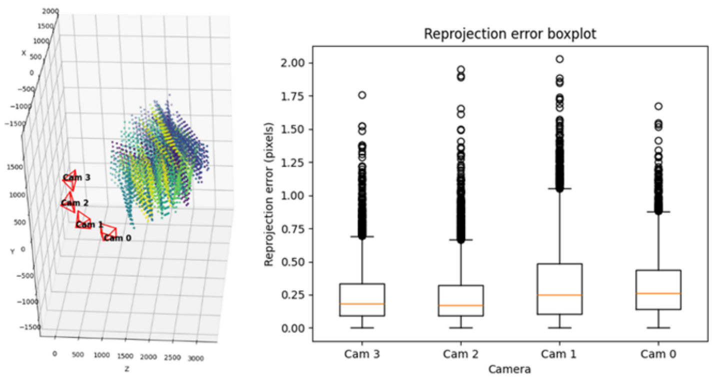
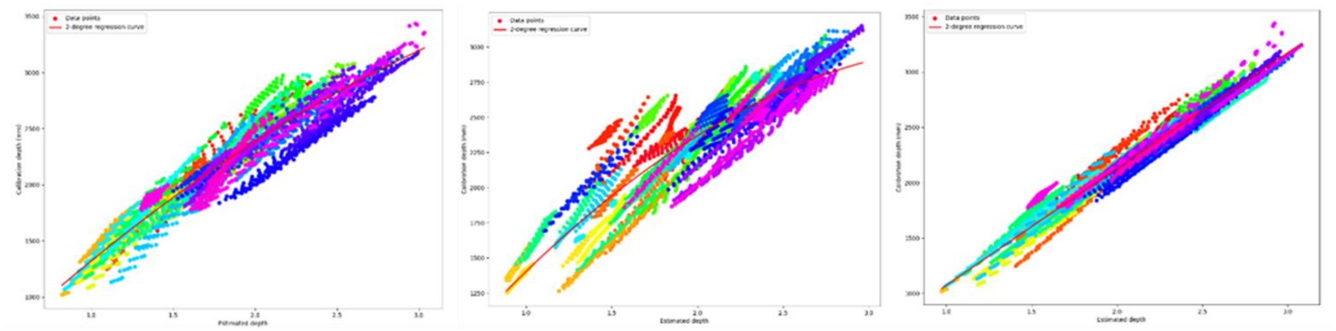


Figure 62: Calibration Results, 3D plot on (left). Boxplot representation of the reprojection error distribution (right).

To analyse the quality of the calibration obtained, the reprojection error is studied. This error is computed by reprojecting the 3D features back into their original images and comparing this new 2D position to their original 2D position. Figure 62 shows an example of camera calibration for a setup with 4 cameras, with the reprojection error being mainly under 1 pixel.

Using the 3D position of the features and the camera calibration parameters, the depth values for each feature are computed. These depth values can be used as a ground truth to fit a quadratic regression that yields a per-camera depth scale adjustment that corrects inconsistencies in the estimation between cameras. The scale adjustment can be leveraged in future captures to obtain RGB+D content coherent among all the cameras in the setup. Figure 63 presents examples of scale adjustments performed with the three proposed models.



(a) Depth Anything V2 (b) Depth Pro (c) UniDepth V2

Figure 63: Depth scale adjustment results for one camera using the three proposed models.

To evaluate the quality of the RGB-D information generated, the FVV live render algorithm was used to synthesize virtual views corresponding to the reference cameras in the setup. As shown in Figure 63, for each camera and frame, the reference viewpoint is rendered using the information from the closest 3 cameras, yielding a reconstruction of said reference view. This reconstruction is then compared to its reference using 2D quality metrics: PSN, SSIM and LPIPS. Figure 65 shows results of the rendering process.

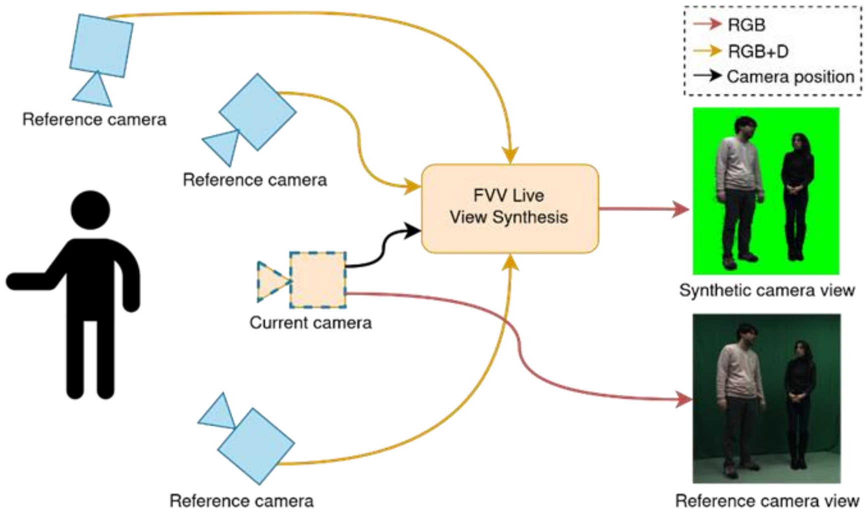


Figure 64: Diagram illustrating the rendering process performed for evaluation. The results are two corresponding images which can be compared to obtain objective metrics.

For these experiments, the focus is on the reconstruction of the foreground elements (the people in the scene). Thanks to FVV Live layered image synthesis, segmentation can be applied to the RGB-D material to only render foreground elements. In this case, the segmentation was performed using the method proposed in D4.1 based on Depth Anything[116]. An additional segmentation mask is generated from the virtual views, indicating the

[116] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick, "Segment anything," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023, pp. 4015–4026.

rendered regions in the image. This mask is combined with the foreground mask of the reference view through the union operation, and the result is applied to be able to consider only the foreground when computing the metrics. Furthermore, we propose using the Intersection over Union (IoU) between these masks as a metric, as it reflects the degree of alignment between the reference view and the synthetic view.

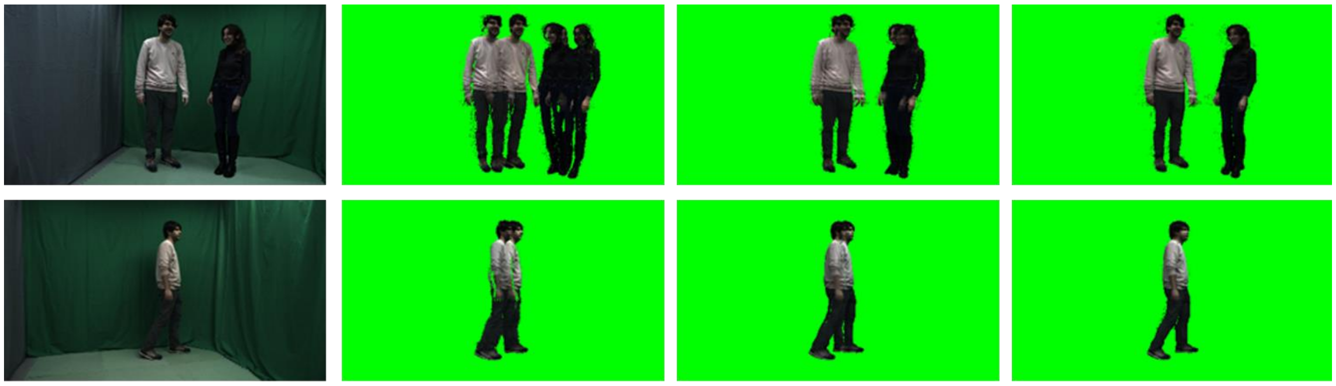


Figure 65: Images rendered using FVV Live and depth information obtained with the proposed pipeline.

The average results of the experiments are shown in Table 14. For all the metrics, the best performance is obtained by UniDepth, which matches the quality shown by the examples on Figure 65.

Table 14: Monocular depth estimation for FVV experiments results.

Model	PSNR (dB) ↑	SSIM (%) ↑	LPIPS ↓	Mask IoU (%) ↑
Depth Anything	13.39 ± 2.84	40.81 ± 7.50	0.13 ± 0.07	56.22 ± 15.86
Depth-Pro	16.16 ± 2.57	54.25 ± 8.65	0.08 ± 0.04	76.73 ± 11.22
Uni-Depth	**18.27 ± 2.40**	**62.96 ± 8.98**	**0.06 ± 0.03**	**86.97 ± 6.28**

Additionally, Table 15 shows the result for the timing experiments performed using an NVIDIA RTX 4090. The best performance again comes from UniDepth, being able to reach almost 20 fps for both resolutions. In the case of Depth-pro and UniDepth, there are no significant differences when changing the resolution, since both models perform a resizing operation before processing the input images.

Table 15: Monocular depth estimation timing experiment results.

Model	1280x720		1920x1080	
	Time (ms)	**FPS**	**Time (ms)**	**FPS**
Depth Anything	292	3.34	1133	0.88
Depth-Pro	170	5.89	174	5.74
UniDepth	**51**	**19.59**	**54**	**18.66**

Based on the obtained results, UniDepth demonstrates compatibility with the FVV Live system and shows potential for integration in specific use cases. However, the implementation of such integration lies beyond the scope of the current project.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

6.3.5 End-to-End Tests

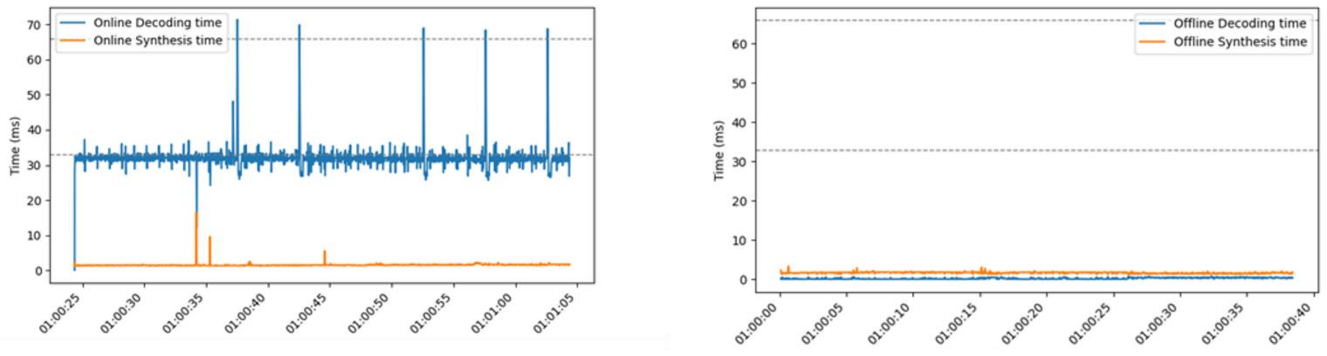
System performance was evaluated through a series of tests designed to meet the proposed requirements. During these executions, performance indicators such as processing time per-frame (framerate) and transmitted bitrate, were captured as log data. Three specific configurations were considered for this assessment:

- Live transmission of real-time RGB-D content.
- Simulation of transmission with pre-recorded RGB-D content.
- Pre-recorded content delivery using the VoD (offline) mode.

For the live transmission, the cameras were placed covering a range of approximately **100 degrees** (NF.161.1), capturing at **30 fps** (NF.120.2 and NF.122.2) with resolution of **1080p** (NF.160.1). In those cases where pre-recorded contents were used, they were post-processed to enhance the segmentation quality with the deep learning-based approach proposed in D4.1. The enhanced segmentation greatly reduces the transmission bitrate, since only the essential depth information is transmitted.

The sequences used for evaluation involved a person standing in the middle of the scene talking. In the live transmission experiments, the peak recorded transmission bandwidth was **55 Mbps** per camera, using real time green screen segmentation. For the simulated transmission, the peak bandwidth was reduced to **15 Mbps** per camera thanks to the enhanced segmentation. Both results fulfil requirement NF.162.1.

Regarding rendering resolution and framerate, Figure 66 shows the rendering time results for the live and VoD tests rendering at a resolution of 1080p (NF.164.1) using an NVIDIA RTX 4090. In the VoD case where there is no transmission, the renderer does not need to wait to receive, so the rendering is only limited by the computing capabilities and reaches up to **500 fps** (NF.163.1). For the Live transmission, the rendering framerate is limited by the transmission to 30 fps. In this case and since cameras can only be synchronized by software, time misalignments between video streams can produce frame-drops that slightly reduce the average framerate of a scene. In this experiment, the average framerate was **29.1 fps**.



(a) Live transmission (online) *(b) VoD visualization (offline)*

Figure 66: Rendering time measurements for the FVV Live end-to-end tests.

Finally, sequence visualisation was tested using the FVV Live web console and a Unity scene, both of which allow for **6 Degrees of Freedom** (DoF) navigation (NF.165.1).

6.3.5.1 Several users and contents tests

In order to fulfil requirements 131.1 and 132.1, FVV Live rendering modules were virtualised, and their deployment can be orchestrated as explained in Section 6.3.3. These approaches were tested on 4 servers each one with an NVIDIA RTX 4090. Each server could support the rendering capabilities needed by 3 users, with 12 concurrent users in total. Combination of real-time capture and pre-recorded content was also tested, with 3 servers managing the volumetric capture and one the rendering of both the live transmission and one pre-recorded content. Figure 67 shows an example of two simultaneous web consoles controlling their own virtual camera to visualize the same content from different points of view.

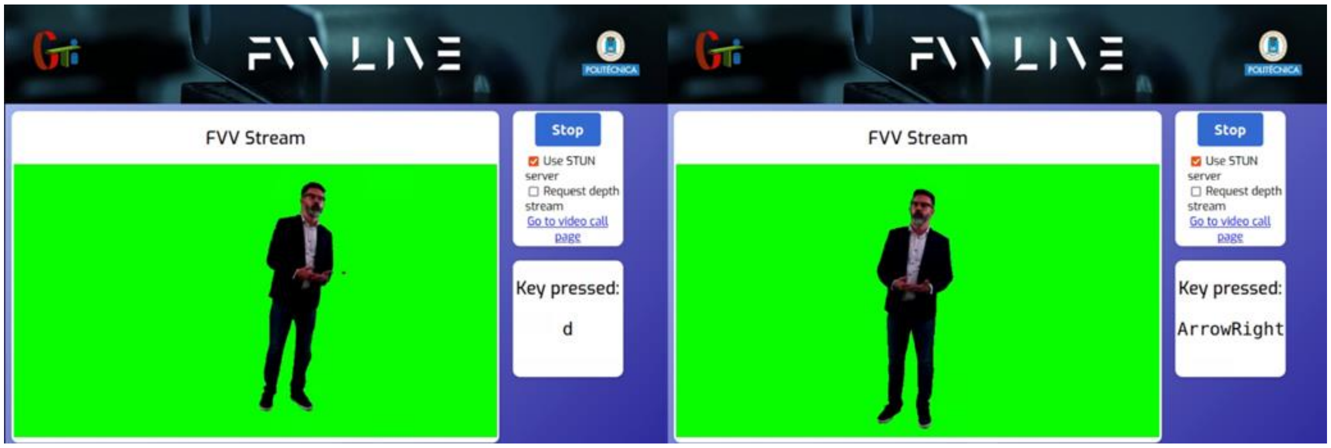


Figure 67: Two concurrent FVV Live web console clients visualizing the same content from two different points of view.

6.3.5.2 Latency analysis

Latency measurements have been carried out to get the most significant metrics about these implementations. These tests' results are shown in Table 15, where the Motion-to-Photon latency main metrics are illustrated. The requirements NF.119.2 and NF.121.2 are satisfied in all cases.

Table 3: FVV Live M2P latency analysis

Implementation	Average M2P (ms)	M2P St. Deviation (ms)
Unity Desktop	240	33
Unity HMD	217	45
Unreal (desktop)	244	29

The process to obtain these metrics is as follows: a variation of the original Unity/Unreal application was developed so the camera position messages are only sent when a specific button is pressed. The timestamp of the button-pressing is registered and compared to the one corresponding to the new view. An example of this process is illustrated on Figure 68.

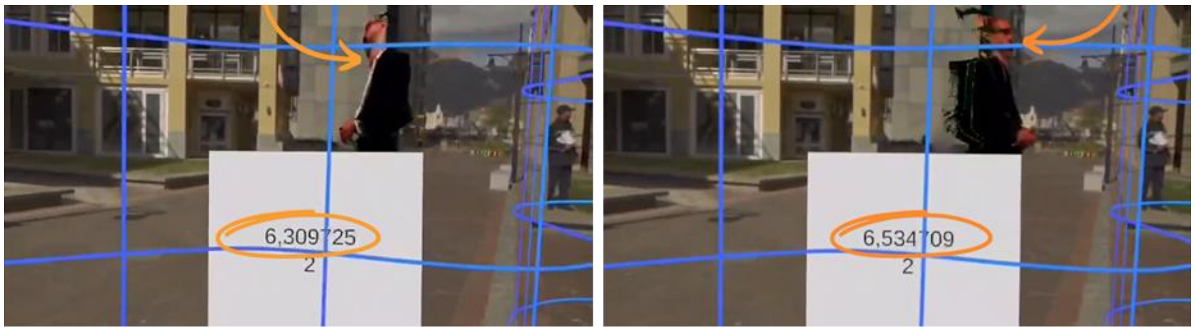


Figure 68: Screenshot of M2P measurements carried out on the HMD.

The Unity and Unreal Desktop measurements were run on an Asus Expertbook laptop with the following main technical specifications:

- Processor: 13th Gen Intel(R) Core(TM) i7-1355U, 1700 Mhz, 10 Core(s), 12 Logical Processor(s)
- 32 GB RAM
- Intel(R) Iris(R) Xe Graphics Card

While the HMD measurements were run on a Meta Quest 3 Headset:

- Processor: Octa-core Kryo (1 x 3.19 GHz, 4 x 2.8 GHz, 3 x 2.0 GHz)
- 8 GB RAM
- GPU: Adreno 740

6.4 RGB-D based real-time holoportation

Volumetric capture systems utilize multiple cameras positioned around the capture area to record images from various viewpoints. This setup enables comprehensive coverage of a human subject, resulting in a true 360º video. It also helps compensate for occluded or missing areas that may not be visible from a single camera's perspective, as illustrated in Figure 69. This section describes multi-view camera calibration pipelines in the context of real-time holoporation (see D4.1 – Section 6.3).

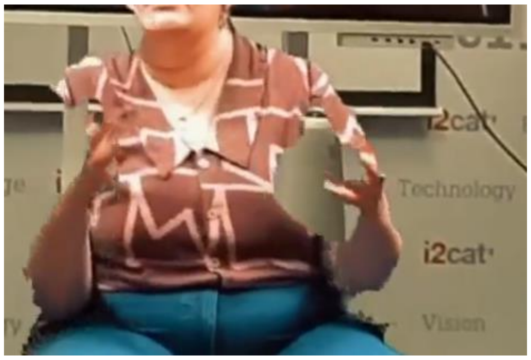


Figure 69: Effect of shadows from a single view reconstruction.

The system developed by i2CAT leverages cameras capable of capturing dense depth information in real time, along with colour data, to reconstruct a 3D subject using input from a single camera. A transformation is then applied to each geometric dataset to align and integrate them into a cohesive 3D reconstruction. This

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

reconstruction is subsequently compressed and transmitted over the network to the client side, where it is decompressed and rendered, enabling remote viewing of the captured scene.

Two key aspects of the capture side of the pipeline we are tackling and improving as part of this project are the cameras used and the calibration procedure.

As part of the development effort, we implemented a recording pipeline enabling the generation of datasets that allow the reproduction of the whole reconstruction pipeline without the cameras or reproduce pre-recorded videos converting this pipeline into a possible new source of 3D assets. Moreover, this allows to record datasets with more cameras than what is supported for the real-time pipeline, but that can be tested and used as input for other off-line 3D reconstruction pipelines.

6.4.1 Automatic calibration

RGB-D sensors compute the spatial coordinates of each visible point relative to their own local coordinate system. Typically, the origin is located at the centre of the camera lens, with the z-axis extending outward from the camera and the xy-plane aligned with the camera's image plane. However, to generate a unified 3D reconstruction from multiple viewpoints, the individual 3D representations must be transformed into a common, coherent coordinate system. This transformation process is relevant even in single-camera setups, as the coordinate system used for visualisation (e.g. in software environments like Unity3D) may differ. For instance, Unity assumes the origin is located on the floor with the y-axis oriented vertically.

This process known as extrinsic calibration, or simply calibration involves determining the transformation matrices for each camera that align their respective point clouds into a unified 3D reconstruction. Broadly, calibration requires solving two main problems: identifying corresponding points across different views and computing the transformations that best align them. One of the most widely used algorithms for aligning point clouds is the Iterative Closest Point (ICP)[117] algorithm. ICP works by iteratively minimizing the distance between points in overlapping point clouds to achieve alignment. In the context of this project, our focus has been on improving the point selection process for calibration.

Previously, reference points were selected manually on a known object placed at the centre of the capture area. For each camera's point cloud, five points were manually identified and used for alignment. This approach was slow, labour-intensive, and prone to human error, as the selection was subjective and performed on inherently noisy data (see Figure 71-left). To improve this method, we explored automatic point detection approaches. These methods rely on placing a known object - commonly referred to as a calibrator - within the capture area, ensuring it is visible to all cameras. The calibrator must be easily reproducible, interpretable by software, and generally consists of high-contrast geometric patterns in black and white. Automatic detection not only reduces human error but also allows for a greater number of reference points per view, improving the robustness of the resulting alignment. A classic example is the use of a chessboard pattern: it is high-contrast, symmetrical, and contains numerous identifiable corners. However, its repetitive pattern makes it difficult to distinguish orientation, and its flat, single-sided nature limits visibility when cameras are placed at opposing angles.

[117] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In 3-D Digital Imaging and Modeling, 2001.

A more advanced alternative involves using ArUco[118] markers. These are square fiducial markers with a black border and a binary-coded interior arranged in a grid. Each marker contains an ID and error-checking bits, allowing for unique and orientation-invariant detection. ArUco markers can be used as standalone features or integrated into larger calibration patterns, such as ArUco boards or even 3D objects covered in ArUco markers. The latter approach is particularly effective for complex multi-camera configurations, as different cameras may detect different markers while still contributing to a unified coordinate system (see Figure 70).

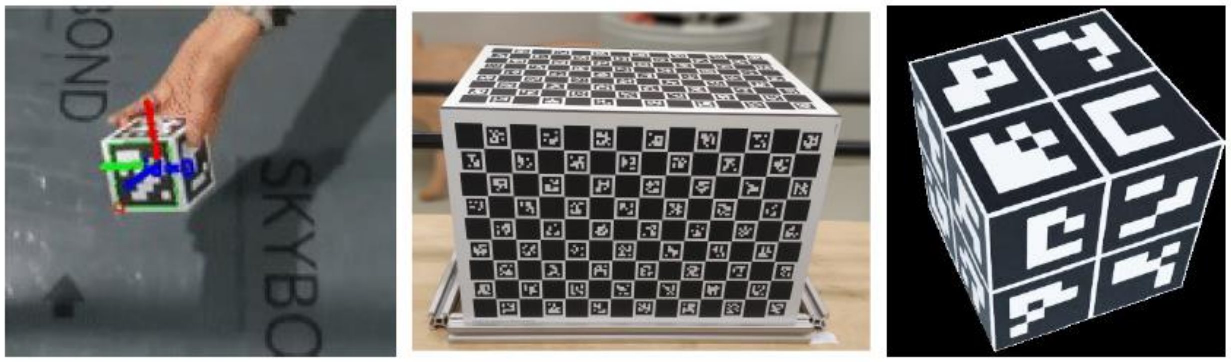


Figure 70: Cube with ArUcos detected with OpenCV (left), cuboid used in Medeiros et al. (centre), and cube used in Moreira et al. (right).

The specific object we ended up implementing is the one seen on Figure 71-right. It uses 6 markers per face and has a front and back face pasted to a rigid box. The markers are printed on a standard A3 sheet, which is easy to reproduce everywhere, and the width of the ground reference object is easy to adjust by running a simple script with the actual width of the box used. Figure 72 presents calibration results on the final merged point cloud.

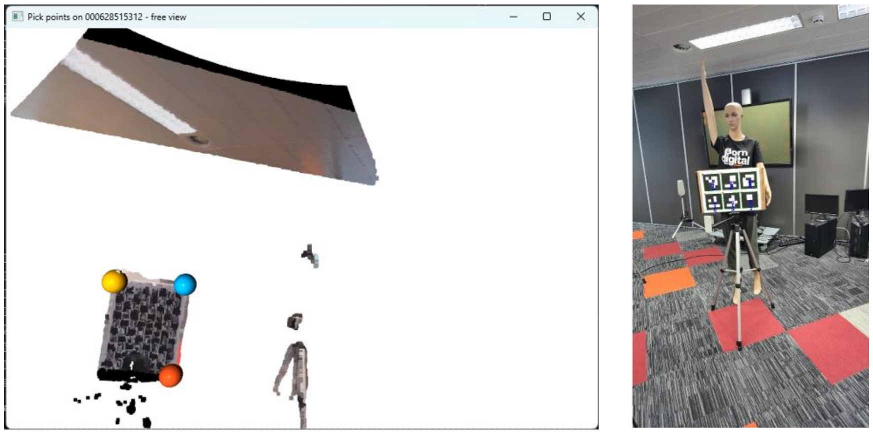
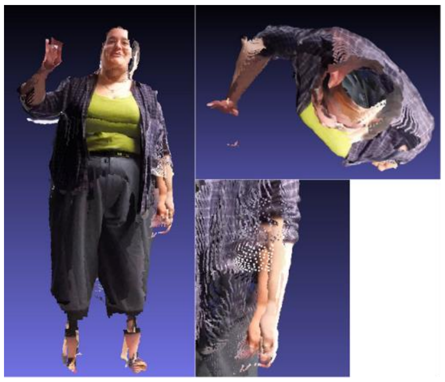


Figure 71: Example of manually selected points for calibration (left), calibration object and detected markers in the automatic calibration (right); The four corners of each marker are detected and used as reference points.

[118] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6):2280–2292, 2014.


(a) Manual procedure

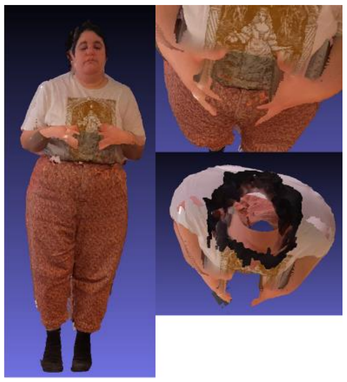

(a) Automatic procedure

Figure 72: Calibration results with (a) manual and (b) automatic multi-view calibration pipelines.

6.5 3D Face Reconstruction

The following workflow was established for the generation of Synthetic Humans, aimed at minimizing human intervention and ensuring an efficient and flexible process exploiting RAI's extensive archive of images and videos. A practical application of this methodology was demonstrated within the RAI News Media Demonstrator (D5.1 - Section 2), which featured the 3D reconstruction of Guglielmo Marconi as an Unreal Metahuman.

6.5.1 Automated 3D Facial Model Reconstruction from 2D Images

The creation of realistic, high-fidelity 3D models of human faces is a complex task that traditionally requires significant manual intervention, specialized artistic skills, and expert knowledge of sophisticated modelling software. A key objective within WP4 was the automation of this process, focusing specifically on generating a 3D model of an individual's face from a limited set of 2D reference images. To streamline and automate this procedure, we have engineered a solution built upon FaceBuilder[119], a plugin for Blender[120]. Our approach leverages the core capabilities of FaceBuilder to produce a high-quality base mesh of the face while abstracting away the manual steps typically required. By utilising artificial intelligence, the plugin can automatically deduce the correct configuration and parameters for the virtual cameras corresponding to each input photograph. The selection of FaceBuilder as the foundational technology for our reconstruction pipeline was motivated by several key features, most notably its highly accurate, automatic alignment of facial key-points (fiducial markers) across different photographs.

6.5.2 Methodology: Leveraging FaceBuilder for Facial Reconstruction

To maximize efficiency and eliminate the need for manual interaction with the Blender interface, we developed a custom Python script that automates the entire FaceBuilder workflow. This script serves as a command-line

[119] https://keentools.io/products/facebuilder-for-blender
[120] https://www.blender.org/

tool that programmatically executes the necessary steps for mesh creation, significantly simplifying the process for the end-user.

The script's core functionalities include:

- **Head Creation:** Programmatically initializes the base 3D head model within the environment.
- **Image Ingestion:** Automatically imports the provided reference photographs.
- **Camera and Pin Alignment:** Leverages the FaceBuilder API to automatically place and align the virtual cameras for each image and configure the facial pins, which is the most critical step for ensuring an accurate likeness.
- **Texture Generation:** Creates and bakes the final facial texture map from the aligned source images.

The primary outcome of this development is a fully automated process for generating high-quality 3D facial meshes. This scripted approach minimizes human intervention, ensuring an efficient, flexible, and scalable creation process, making it accessible to operators without prior experience in 3D modelling or Blender, thereby greatly enhancing the efficiency of our overall character creation pipeline.

6.5.3 Experimental results

This section details the comprehensive evaluation conducted to assess the reliability, robustness, and fidelity of the synthetic human generation workflow. The primary focus was on quantifying and qualifying the similarity between the generated Synthetic Humans and their corresponding reference characters, utilizing both objective computational metrics and subjective human perception.

6.5.3.1 Test Material Preparation

To ensure a robust validation, a representative dataset of 9 individuals was curated (Figure 73). This dataset was explicitly designed to encompass a broad spectrum of demographic attributes, including diverse ethnicities, age groups, and genders. This diversity aims to demonstrate the workflow's capability to accurately reflect traits representative of the global population. For each character, 80-100 original images were collected from the RAI archive, depicting a wide range of poses, expressions, and camera distances (close-ups to long shots). This image set was used to fine-tune a Stable Diffusion model, enabling the generation of novel AI images for each reference character.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

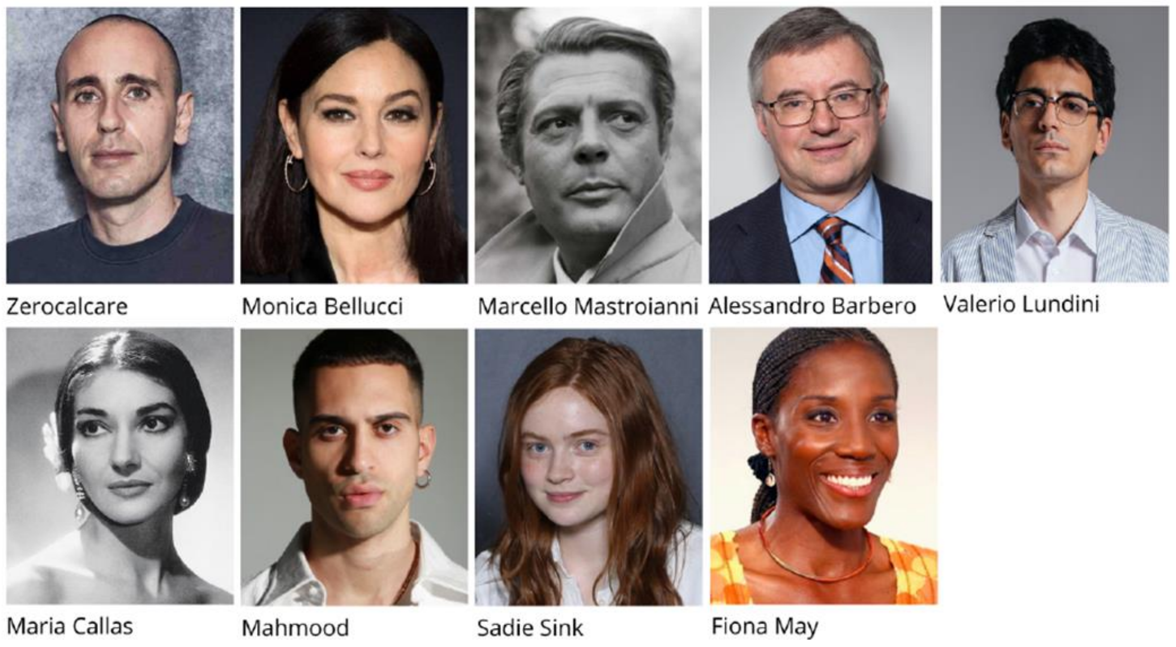


Figure 73: Reference dataset of 9 individuals used for 3D model and texture validation.

6.5.3.2 Test Material Preparation

The automated script generated a 3D head mesh for each individual using both original images and AI-generated images. This mesh can be used in any 3D software, but in our case, it served as the basis for the creation of a MetaHuman (MH) in Unreal Engine.

Evaluation Metrics:

- **Objective:** Cosine similarity of 512-dimensional face embeddings extracted using ArcFace121.
- **Subjective:** An online survey with 46 participants assessed perceived realism and similarity. Participants rated resemblance (1-5 scale) and selected the most realistic textures from visual comparisons.
- **Comparisons were made between:**
  - Original images vs. AI-generated images.
  - Original images vs. rendered 3D head meshes.

6.5.3.3 Test Results

Stage 1: AI-Generated Image Fidelity

- **Objective (Cosine Similarity):** Embeddings from AI-generated images consistently showed Cosine similarity > 0.5 (Figure 74) when compared to original images. This threshold indicates a high probability that the images represent the same individual.

[121] Jing Yang Niannan Xue Irene Kotsia Jiankang Deng, Jia Guo and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. https://arxiv.org/pdf/1801.07698v4.pdf.

- **Subjective (Survey):** Participants rated AI image resemblance to the original character on a 1-5 scale. 68% of responses assigned the highest score (5), demonstrating strong perceived visual fidelity.

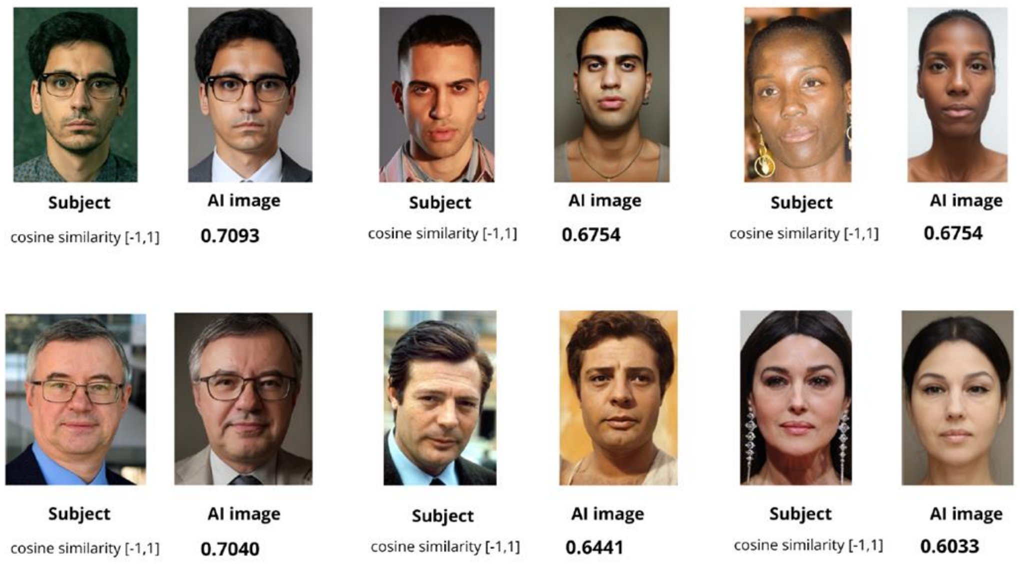


Figure 74: Cosine similarity scores between ArcFace embeddings of original images and corresponding Stable Diffusion generated images.

Stage 2: 3D Head Mesh Fidelity

- **Objective (Cosine Similarity):** Comparisons between original images and rendered 2D views of the reconstructed 3D mesh yielded cosine similarity scores approaching 0.5 (Figure 75). A slight decrease compared to Stage 1 was observed, attributable to inherent differences between 2D photos and 3D renders, and variations in pose within the original images.
- **Subjective (Survey):** Participants rated the 3D mesh resemblance to the original character. 33.82% assigned the highest score (5) and 29.95% assigned a score of 4, indicating a strong majority (63.77%) perceived high or very high similarity.

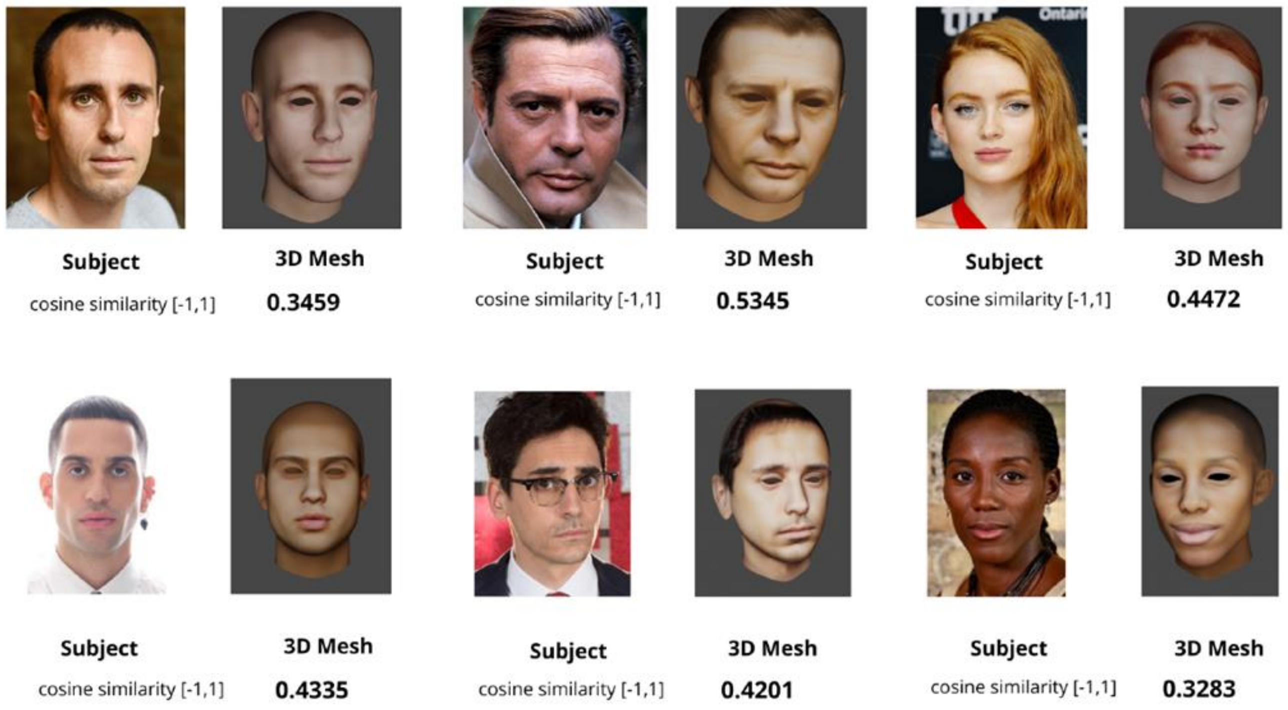


Figure 75: Cosine similarity scores between original images and rendered views of the reconstructed 3D head meshes.

6.5.3.4 Conclusion on the evaluation

The evaluation demonstrates the robustness of the synthetic human generation workflow:

1. **High-Fidelity AI Generation:** Stable Diffusion, fine-tuned on diverse character images, produces 2D outputs with high objective (Cosine Sim. > 0.5) and subjective (68% top rating) similarity to reference characters.
2. **Accurate 3D Reconstruction:** The automated pipeline generates 3D head meshes that retain significant fidelity to the original subjects, confirmed by objective metrics nearing the recognition threshold and strong subjective ratings (63.77% scores 4 or 5).
3. **Workflow Robustness:** The process proved effective across a demographically diverse dataset, indicating its potential for broad applicability.

7 Asset Aggregation and Optimisation services

This section presents enhancement and optimisation technologies that improve the quality and usability of multimedia assets within XReco. These services address the challenges of transforming content of varying quality into professional-grade assets suitable for XR applications and broadcast production.

The services encompass 2D video upscaling for resolution enhancement of legacy content, blind face restoration for improving degraded facial imagery, human-centred point cloud super-resolution with gap-filling capabilities, and 3D content generation enabling text-to-3D asset creation.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

7.1 2D Video Upscale

The 2D Video Upscale component is a super resolution technique for improving the visual quality of 2D urban video content, i.e., videos depicting scenes of buildings, city landscapes, and monuments. This component relates to the requirement NF88.2 ("It MUST be possible to enhance the 2D Multimedia Shared Material using AI-based techniques") of the XReco system.

7.1.1 Motivations and related work

Recent advances in Computer Vision, driven by AI research, have enabled innovative applications including generative AI tools for video and image generation. However, challenges remain in broadcast environments, particularly regarding video resolution and temporal coherence for professional use.

Traditional video enhancement relies on classical upsampling filters (bilateral, bicubic), which struggle to preserve details at high magnification factors (4x or more), often producing artifacts or blurred results. AI-based Video Super Resolution (VSR) addresses these limitations by leveraging spatio-temporal information to generate high-resolution videos from low-resolution inputs while maintaining perceptual quality.

Key challenges:

- **Generalisation:** VSR algorithms perform inconsistently across diverse content types (nature, sports, cartoons).
- **Training complexity:** Performance depends heavily on datasets, architectures, batch size, augmentation strategies, and sequence length.
- **Computational demands:** Execution time and memory consumption limit real-time deployment in business context.

Comparing VSR algorithms like EDVR[122] BasicVSR++[123] or RVRT[124] requires standardised datasets and evaluation protocols. Video Quality Assessment (VQA) employs two approaches:

- **Subjective assessment:** Human observers rate quality against ground truth following ITU BT-500 guidelines (0-10 scale)125, 126. Though reliable, this method is expensive and time-consuming.
- **Objective metrics:** Automated quality assessment using Full-Reference metrics (PSNR, SSIM, VMAF) comparing against originals, or No-Reference metrics (LPIPS, BRISQUE) analysing statistical features. MS-SSIM and VMAF are widely considered most reliable.

[122] EDVR: Video Restoration with Enhanced Deformable Convolutional Networks - https://arxiv.org/abs/1905.02716 (last accessed June 06th, 2025)
[123] BasicVSR++: Improving Video Super-Resolution with Enhanced Propagation and Alignment - https://arxiv.org/abs/2104.13371 (last accessed June 06th, 2025)
[124] Recurrent Video Restoration Transformer with Guided Deformable Attention - https://arxiv.org/abs/2206.02146 (last accessed June 06th, 2025))
[125] ITU BT500 Recommendations - https://www.itu.int/rec/R-REC-BT.500 (last accessed June 06th, 2025)
[126] ITU BT500 Recommendations - https://www.itu.int/rec/R-REC-BT.500 (last accessed June 06th, 2025)

While objective metrics offer speed and repeatability crucial for development, they may not fully capture human perception. Optimal evaluation combines both subjective and objective approaches to ensure comprehensive quality assessment.

7.1.2 Model fine-tuning procedure

This section briefly introduces Real-Basic Video Super Resolution (RBVSR)[127] the AI-based VSR solution of this experimentation[128] followed by a description of our approach.

VSR algorithms could be designed following different architectures and network topologies (e.g. CNN, Transformer, RNN, etc.). RBVSR is a deep neural network composed of a first denoising stage for suppressing artifacts and visible defects of LR frames at the beginning of the process to avoid their temporal propagation before being introduced to the VSR module. In this case, the VSR core network is called BasicVSR. The public model and the network's structure were designed and developed only for 4x upscaling. For example, an input video sequence having a native resolution of 960x540 can be enhanced and upscaled to a final resolution of 3840x2160. RBVSR was trained on the REDS dataset129 and evaluated on a custom dataset, called VideoLQ, following a two-stage strategy. In the first stage, the network was pre-trained for 300K iterations using the Charbonnier loss (a mix between the L1 and L2 loss) for both fidelity loss and image cleaning loss. In the second stage, the "base" model is finetuned adding perceptual loss and adversarial loss for 150K iterations following a GAN-based approach.

The goal of our study is twofold: trying to improve the performance of the public 4x model for our use case and expanding the application perimeter with a new solution for the 2x upscale case that in our opinion could be the most appreciated and requested by users within the upscaling solutions offered by the XReco platform.

Our approach consists of two steps. In the first, we studied the algorithm, the scripts and ran a bunch of tests to check the reliability of the starting point. Then, after the definition of a custom dataset, we tried to understand if and how each tuneable parameter of the training script could influence the behaviour of the network. Loss functions, batch size, sequence length, optimiser configuration were considered and, last but not least, the data for both training and evaluation. Obviously, data plays a key role during the development of an AI-based algorithm. As already mentioned, we created our own 4K video dataset consisting of 50 clips with a resolution of 3840x2160, 5 seconds each and a total of over 5,5K frames. Videos were carefully selected and extracted from our drone footage database mainly representing urban scenes and landscapes with a wide variety of information and details that are essential for these applications. For the 4x case, there were 2 options: starting a training process from scratch or resuming the existing model and working on its refinement. Once 3 video sequences and 4 objective metrics for the models' evaluation had been selected, we decided to proceed directly with the second option because of the time constraints imposed by the project. We conducted several tests with different configurations until we reached our scope. In our experience, the most impactful training variables are

[127] Chan, Kelvin C.K. and Zhou, Shangchen and Xu, Xiangyu and Loy, Chen Change, "Investigating Tradeoffs in Real-World Video Super-Resolution", IEEE Conference on Computer Vision and Pattern Recognition, 2022.
[128] The source code is available at https://github.com/ckkelvinchan/RealBasicVSR/tree/master (last accessed June 06th, 2025)
[129] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee, "NTIRE 2019 challenge on video deblurring and super resolution: Dataset and study", In CVPRW, 2019. 2, 7

represented by sequence length, crop shapes and loss functions. Unlike the Single Image Super Resolution (SISR) application, where there is only one image to elaborate, VSR algorithms take advantage of multiple data sources represented by a chain of subsequent frames. In other words, the more information correlated is available for training (i.e., sequence length), the better the algorithm's performance is. Furthermore, the variation of loss functions has been evaluated during our experiments and, as we will see in the next section, this approach in some cases could be useful to improve the numerical accuracy of this kind of algorithms.

Alongside the study done in the first phase, we decided to move a step forward by adding a new service based on the same network. As said before, a 4x upsample could be excessive in many situations where users may need a simple 2x upscale of a given video before the invocation of subsequent processing pipelines (like for example, another service offered by the XReco platform). The adaptation for the 2x case required a light lifting of the network's architecture mainly consisting in the drop of an upsample layer. The operation is the same as before: for example, an input video sequence having a resolution of 960x540 is now upscaled by a factor of 2 to obtain a final video resolution of 1920x1080. We have not mentioned the video framerate as it remains untouched like the input video sequence. The only variation between input and output is the resolution. In this second part of the study, as in the previous stage, we started from the public 4x model and refined it with our settings and dataset. The main difference is that during the evaluation phase, our 2x model is compared with the bicubic filter, one of the most used non-learnable filters, while in the first step we decided to double check both the original 4x model and the bicubic filter too.

Finally, we apported some improvements to the original inference script in order to optimize the whole procedure and obtain better memory management.

7.1.3 Experimental results

Table 16, Table 17, and Table 18 present the experimental results achieved for the different settings described in the previous section. The metrics used for evaluation are PSNR, SSIM, MS-SSIM and VMAF. For PSNR and SSIM we assessed the quality on the luminance (Y) channel.

PSNR is a pixel-wise metric based on the Mean Square Error (MSE) and the absolute difference between images. The higher the PSNR value, the better the image quality. This is one of the most used metrics even if it does not correlate well with the human visual system. SSIM relies on structural similarity and considers other features, such as contrast and brightness. MS-SSIM is an extension of SSIM, where MS means Multi Scale. Here the similarity is considered across different scales and resolutions of images. For both SSIM and MS-SSIM the higher the final score, the better the image quality. VMAF tries to predict the subjective quality of a video by combining several assessment methods based on machine learning and fusing multiple metrics. The power of this metric is both the ability to emulate the perception of the human visual system and to support different viewing conditions, such as mobile devices or 4K TVs. The score is represented on a scale 0-100, where 100 means perfect correlation between the reconstructed image and the original one.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Table 16: Comparison between original model and our finetuning for the 4x case.

METRIC	ORIGINAL MODEL	4X FINETUNING
Y-PSNR	29,50	30,19
Y-SSIM	0,84	0,86
MS-SSIM	0,94	0,95
VMAF 4K	83,92	83,45

Table 17: Comparison between classical non-learnable filter and our finetuning for the 4x case.

METRIC	BICUBIC FILTER	4X FINETUNING
Y-PSNR	30,23	30,19
Y-SSIM	0,83	0,86
MS-SSIM	0,94	0,95
VMAF 4K	62,69	83,45

Table 18: Comparison between classical non-learnable filter and our finetuning for the 2x case.

METRIC	BICUBIC FILTER	2X FINETUNING
Y-PSNR	36,65	34,61
Y-SSIM	0,95	0,95
MS-SSIM	0,99	0,99
VMAF 4K	95,55	99,80

Among the 4 presented metrics, the one with the strongest correlation with the human evaluations is VMAF, followed by SSIM/MS-SSIM and PSNR. Table 18, for example, clearly shows the "conflict" between VMAF and PSNR scores. Obviously, both are relevant but as stated before we consider more reliable VMAF than others. The 4-points range of Table 18 for the VMAF scores (95,55 vs 99,80) probably reflects the humans' eyes quality perception of the 2 upscaled videos, while if it were small or too small, such as the slight differences shown in Table 16, (83,92 vs 83,45), subjective tests probably struggle to find any variation.

The average scores shown on the tables above are represented with three different colours: RED is the worst value, BLUE is the best value, and GREY is the case of equivalence.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Table 16 collects the results related to the comparison between the original RBVSR model and our finetuned model for the 4x super resolution case confirming that the finetuning process could often lead to better performance, as expected, even if the difference is not so high in terms of scores. Our finetuning improves the baseline in all cases except for the VMAF 4K where the difference is very small (83,92 vs 83,45). Figure 76 reflects the average scores collected from Table 16 and highlights the differences that could be detected also by non-expert eyes. The finetuned model outperforms the baseline even if only slightly, and the image clearly shows this similarity. The quality of both 4x models is awesome. The image is a crop of a selected frame rich in details, representing a square from the drone's point of view. The higher performance of the finetuned model may be appreciated in the central part of the image where a lot of details almost disappear during the upscaling process with the original 4x model. Also, the statues are better represented but, as said before, the final upscaled images look similar. In this first case, the winner is the optimized model because of the slight improvements shown, as confirmed by 3 out of 4 metrics.

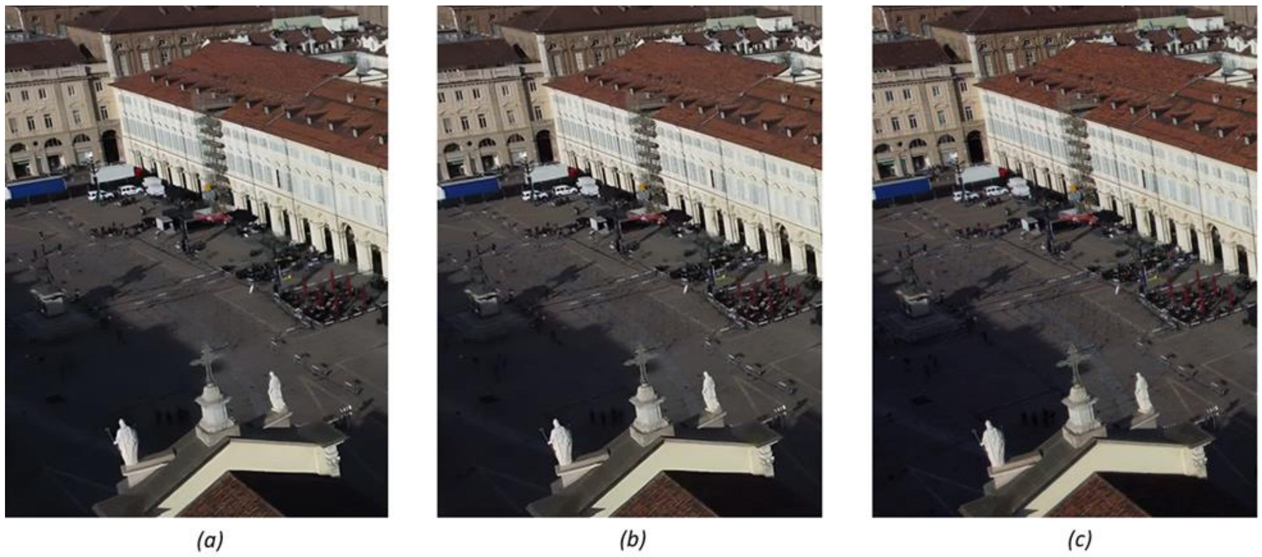


Figure 76: Comparison between original model (a), our finetuning (b), and original frame (c) on 4x upscale case (zoom in for better view).

Figure 77 shows a comparison between our optimized 4x model and the bicubic filter. The power of AI-based VSR tools lies in their better details' handling and this image, supported by Table 17, clearly shows this large difference. Despite the scores' correlation for both PSNR and SSIM/MS-SSIM, the most impactful difference is represented by VMAF scores which confirm again that our model seems to be much more accurate in terms of image quality and details when the upscale factor is high, as in this case. Theoretically, the slight difference in PSNR or SSIM values is a plus if we consider merely the numbers as one would do in a challenge, but in practice, the perceived quality should be basically the same and people watching their comparison probably won't notice any difference. However, this is not the case because of the large gap between the VMAF scores. In fact, the sharpness across the whole frame generated by our model is incredible. Every element of the image is represented with greater fidelity. As in the previous experiment, in this second comparison the winner is our finetuned model.

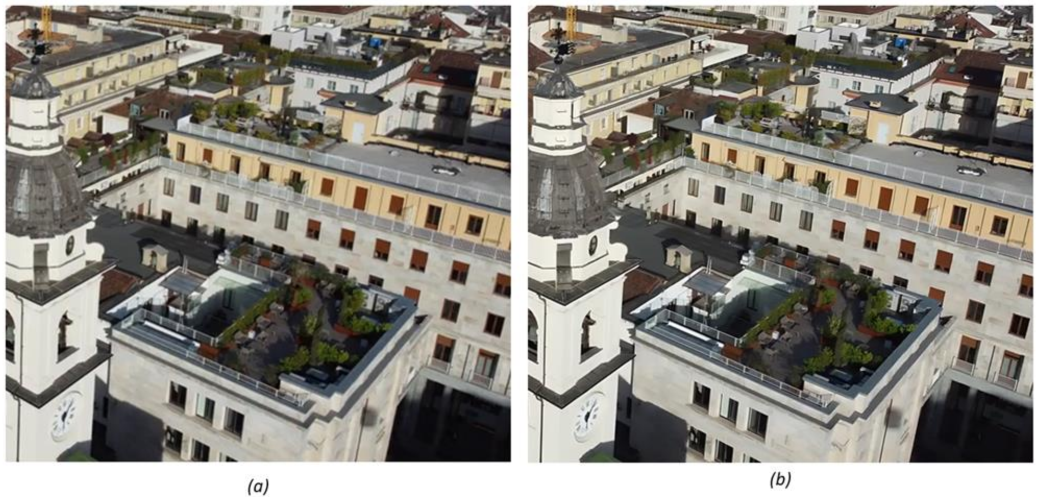


Figure 77: Comparison between bicubic (a) and our finetuning (b) on 4x upscale (zoom in for better view).

As stated before, the upscale multiplier is fundamental. The ability of classic upscalers decreases with the increase in value. A 2x upscale is easier to execute than a 4x because the stretching process is lighter, and therefore the number of artifacts that could be introduced is lower than in the 4x upscale case. The average scores represented in Table 18 support this thesis and the compared frames appear similar even if the details are better represented by our finetuned model (see the white railings on the roofs, in Figure 78). The two contenders perform mostly the same in terms of SSIM and MS-SSIM, while PSNR and VMAF 4k shows contrasting views highlighting once again how metrics can sometimes behave differently. The finest details could be lost by classical non-learnable filter during the upscaling process and in general the sharpness and contrast of images may be negatively affected, as can be noticed in Figure 78. At the same time, some metrics are not able to detect these slight differences because their values are the results of other evaluations or mathematical formulas that are not directly correlated to the characteristics that human eyes find when looking at the images. The PNSR in this third comparison is 2.0 dB higher for the bicubic case. If the priority were simply numerical accuracy, we could have used the MSE loss function to refine the model getting as close as possible to that score. Generally, when sharpness or the details' preservation is the priority, the L1 loss is the right choice. In this experiment we decided to put more emphasis on VMAF and on image quality rather than the simple numbers. Figure 78 clearly shows our approach: the perceived quality is higher for RBVSR than bicubic, and details like the white railing, remain almost as sharp as the ground truth. This consolidates once again the power of the proposed model, even if in this third case the visual difference is not so high as in the previous comparisons.

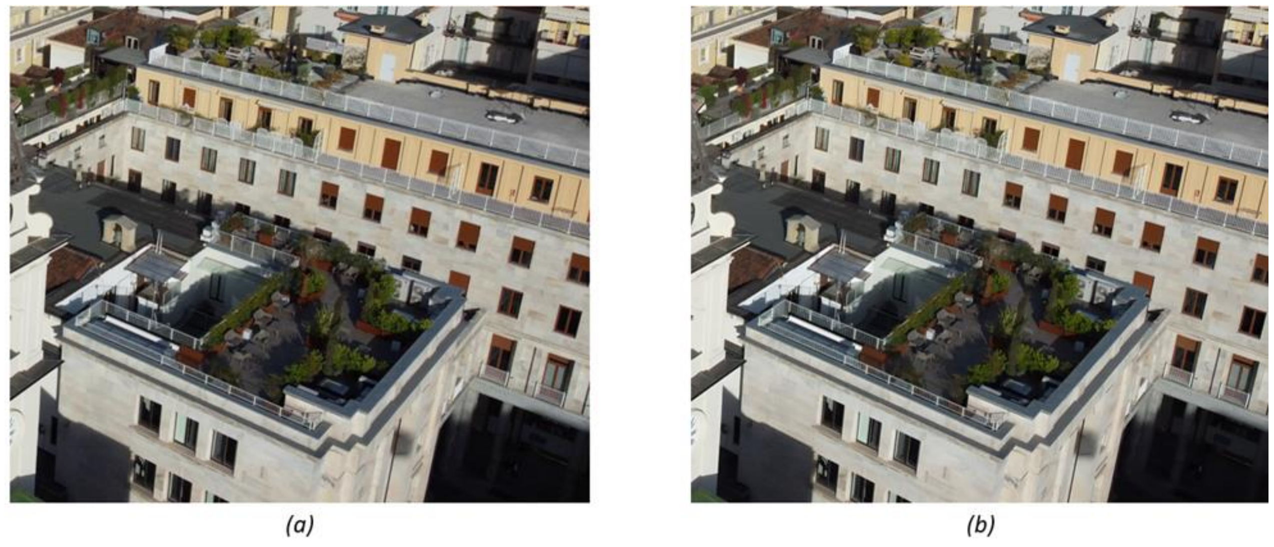


Figure 78: Comparison between bicubic (a) and our finetuning (b) on 2x upscale case (zoom in for better view).

7.1.4 Deployment and integration in the XReco platform

This component is built using docker-compose and provides the following services:

- **api_app:** This container runs the FastAPI backend, which acts as a middleware between the user's requests and the video upscaling process.
- **celery_worker:** This is the schedule for the jobs called by the API. This is the main backend component that runs the video upscale processes as asynchronous tasks.
- **redis:** Message broker and database for the queue and jobs. It manages the communication between the API and the Celery task executor.

The process workflow works as follows:

1. Upload a video to process to the server by calling the /upload API endpoint.
2. Request to process an uploaded video by calling the /process API endpoint. The user can specify the desired upscale factor (x2 or x4), and output (a new video or still keyframes).
3. Poll the API to check the status of a processing task, by calling the /tasks/{task_id} API endpoint.
4. Download the upscaled video/images, by calling the /download/{id} API endpoint.
5. Download a preview image of the upscaled video/images by calling the /preview/{id} API endpoint.

7.2 Blind Face Restoration

In the creation of high-fidelity 3D avatars, the quality of the facial texture map is paramount. These textures are often derived from 2D images which may suffer from low resolution, compression artifacts, or other forms of degradation. To address this challenge, a key component of our work involves the use of Single Image Super-Resolution (SR) techniques. SR is a class of image processing methods designed to generate a high-resolution

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

(HR) image from a single low-resolution (LR) counterpart. The primary objective is to restore high-frequency details that were lost during the image acquisition or compression process, leading to a significant enhancement in perceptual quality.

For the specific context of this project, which targets the generation of realistic 3D facial avatars, a specialized super-resolution method was investigated and implemented. The focus was on the effective upscaling and enhancement of facial textures to ensure that the final 3D models exhibit lifelike and detailed appearances.

This component relates to the requirement NF88.2 ("It MUST be possible to enhance the 2D Multimedia Shared Material using AI-based techniques") of the XReco system.

7.2.1 Investigated Method: Generative Facial Prior GAN (GFP-GAN)

The method selected for facial image super-resolution is the Generative Facial Prior GAN (GFP-GAN)[130]. This technique was designed for blind facial image restoration, where the degradation in the source image is unknown. GFP-GAN excels at achieving an optimal balance between the realism of the generated face and fidelity to the original subject's identity.

This balance is crucial for our application, as the goal is not only to create a visually appealing texture but also to ensure that the identity of the individual is accurately preserved in the 3D avatar. GFP-GAN's capability to deliver high-quality results stems from its use of a rich and diverse knowledge base of facial characteristics, which is embedded within a pre-trained generative network that acts as a facial prior.

7.2.2 Applications and Project Relevance

Within the scope of this project, GFP-GAN is directly applied to the task of 3D avatar creation. By processing low-resolution facial photographs, e.g., taken from RAI archive, the technique generates high-quality, detailed textures that can be mapped onto 3D head models. This results in avatars with a significantly higher degree of realism and visual fidelity. A practical application of this methodology was demonstrated within the RAI News Media Demonstrator, which featured the 3D reconstruction of Guglielmo Marconi as an Unreal Metahuman. To achieve a high-quality result, historical photographs from different sources, such as the RAI archives, were processed and upscaled with the described super-resolution approach, yielding a significantly improved facial texture for the final avatar, as shown in Figure 79. The successful implementation of GFP-GAN represents a significant step in our pipeline for generating next-generation, realistic 3D digital humans.

[130] https://github.com/TencentARC/GFPGAN

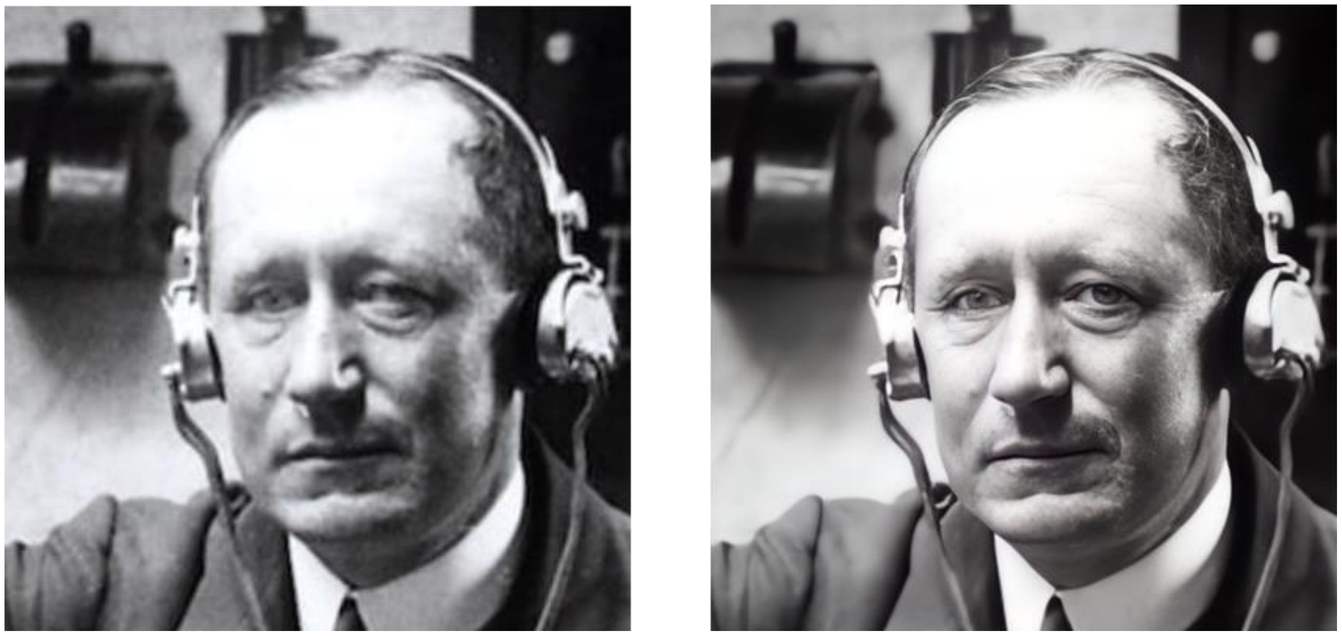


Figure 79: Comparison of the original face (left) and the restored face (right). The original face was extracted from: https://www.raicultura.it/cropgd/900x520/dl/img/2022/11/08/1667911382298_Marconi.jpg

7.2.3 Experimental results

To demonstrate the capability of GFP-GAN to preserve the identity of individuals, we applied the GFP-GAN model to the face images of celebrities in the *Labeled Faces in the Wild* (LFW) dataset.[131] LFW is one of the most used resources for face verification and recognition tasks. For each face (original, restored) we extracted the ArcFace embeddings[132] and computed the Cosine similarity between them. Figure 80 shows the resulting distribution. Most of the scores are in the range of 0.85 to 0.95, meaning that there is a very strong correspondence between each of the pairs of faces. For none of them the score is below 0.5, the threshold for which two faces are normally considered to represent the same individual.[133]

[131] https://scikit-learn.org/0.19/datasets/labeled_faces.html (last accessed June 06th, 2025)
[132] J. Deng, J. Guo, N. Xue and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 4685-4694, doi: 10.1109/CVPR.2019.00482.
[133] https://arxiv.org/abs/1801.07698 (last accessed June 06th, 2025)

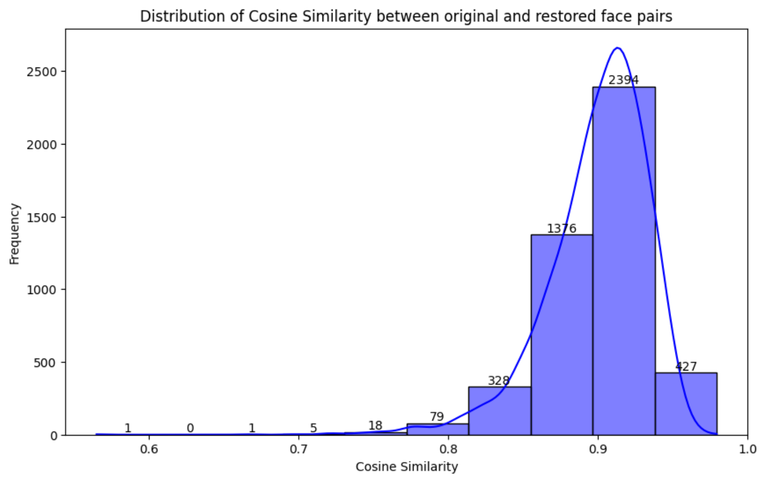


Figure 80: Distribution of the Cosine similarity between pairs of faces (original, restored) for the individuals from the LFW dataset.

7.2.4 Deployment and integration in the XReco platform

This component is built using docker-compose and provides the following services:

- api_app: this container runs the FastAPI backend, which acts as a middleware between the user's requests and the face restoration process.
- celery_worker: it is the schedule for the jobs called by the API. This is the main backend component that runs the face restoration processes as asynchronous tasks.
- redis: message broker and database for the queue and jobs. It manages the communication between the API and the Celery task executor.

The process workflow works as follows:

1. Upload a set of images to process to the server, by calling the /upload API endpoint.
2. Request to process an uploaded set of images, by calling the /restore API endpoint. The user may specify the desired upscale factor (x2, x4 or x8).
3. Poll the API to check the status of a processing task, by calling the /tasks/{task_id} API endpoint.
4. Download the restored images, by calling the /download/{id} API endpoint.
5. Download a preview image of the restored images by calling the /preview/{id} API endpoint.

7.3 Human-centred point cloud super-resolution

In D4.1 (Section 5.3.11), we have addressed the problem of single-view 3D super-resolution (3D SR), specifically in the context of enhancing sparse or low-resolution 3D geometry (such as depth maps or partial point clouds) captured from a single viewpoint. We have already introduced the concept of 3D SR as an extension of 2D super-resolution, emphasizing the analogous challenges in 3D data acquisition and representation. In particular, we focused on point cloud upsampling as a concrete instance of 3D SR and described our early approach, which involved converting single-view point clouds into Projected Normalized Coordinate Code (PNCC)[134] representations and applying 2D super-resolution techniques such as SwinIR.

In the current phase, we generalize and consolidate that approach into a more flexible framework for single-view 3D SR. Rather than relying on a specific 2DSR model or representation, we propose a method that converts the input 3D geometry into structured 2D image-like representations suitable for processing with efficient and well-established 2D architectures. This avoids the need for high-resolution RGB inputs or computationally intensive 3D operations.

Within this framework, we continue to use PNCC as a structured 2D encoding of 3D coordinates, enabling convolutional models to effectively learn geometric upsampling. We implemented and evaluated two model variants under this framework: one based on Swin Transformers[135] (SwinT-PNCC) to prioritize reconstruction quality, and another using Vision Mamba[136] (VM-PNCC) to emphasize computational efficiency. Both models operate exclusively on PNCC inputs and are capable of real-time inference.

We validated our framework on standard depth SR benchmarks, achieving strong performance across different upsampling scales. Importantly, our models achieve competitive results without the need for RGB guidance and while maintaining efficiency suitable for real-time applications.

7.3.1 Methodology

The proposed framework operates within a 3D-to-2D representation domain, converting input data into PNCC before applying 2DSR models. An overview of our pipeline is presented in Figure 81. The representation method must satisfy several key requirements:

- **Geometric independence**: The representation must encode purely geometric information, functioning independently of auxiliary inputs such as RGB data to ensure broad applicability across diverse scenarios.
- **Reversibility**: The method should maintain reversibility at the single-view level, meaning that given a 3D structure and its projection, the representation preserves all information within the inherent limitations of the projection itself. This property enables lossless recovery of the original projected form (depth map or point cloud) and facilitates conversion to any desired single-view 3D format.
- **Spatial compatibility**: The representation must retain a matrix-structured, image-like format that preserves spatial relationships, ensuring compatibility with standard 2DSR architectures without requiring specialised modifications.
- **Model flexibility**: The super-resolution component can utilise any method that operates exclusively on low-resolution input representations, without dependence on external guidance or auxiliary features. This

[134] Zhu, X., Liu, X., Lei, Z., & Li, S. Z. (2017). Face alignment in full pose range: A 3d total solution. IEEE transactions on pattern analysis and machine intelligence, 41(1), 78-92.
[135] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 10012-10022).
[136] Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., ... & Liu, Y. (2024). Vmamba: Visual state space model. Advances in neural information processing systems, 37, 103031-103063.

design choice maximises the framework's adaptability across different super-resolution approaches while maintaining computational efficiency.

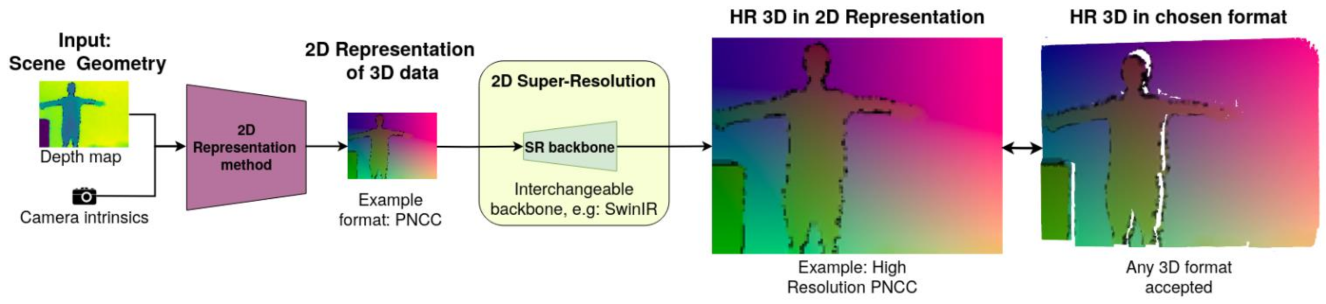


Figure 81: Overview of the pipeline with an example PNCC representation.

Choosing the right 3D representation is crucial for the framework. While depth maps are valid due to their image-like format, they lack spatial richness. Point clouds, on the other hand, require projection and are incompatible with 2DSR models. PNCC encodes normalized 3D coordinates as RGB values at each pixel. This preserves full geometric information per view and remains compatible with standard 2D architectures. PNCC is also reversible; depth maps can be recovered analytically by projection techniques (as PNCC pixels refer to points in the XYZ space), and point clouds are extracted by listing valid PNCC pixels as points. The hypothesis is that it will be a powerful format within the proposed framework. PNCC projection is presented in Figure 82. Each 3D point projected from the scene is assigned its (X, Y, Z) position value as RGB value of the image.

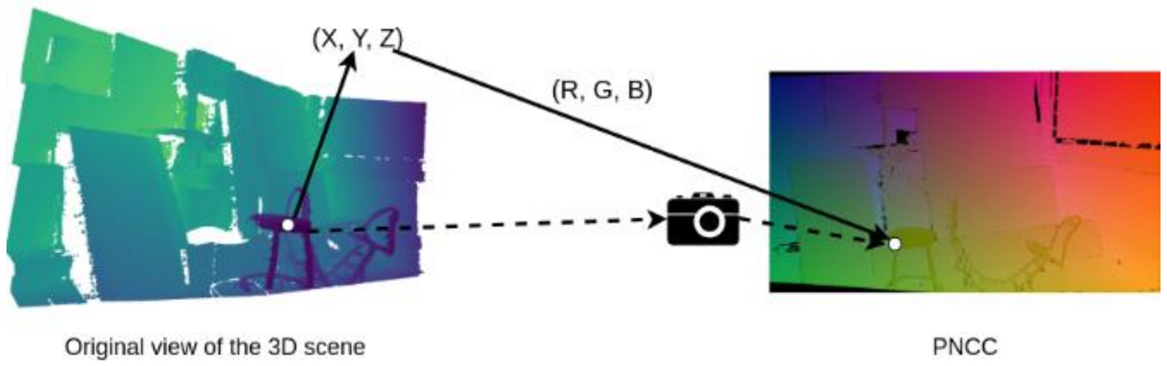


Figure 82: Illustration PNCC projection procedure.

Given depth and camera intrinsics, PNCC is computed as:

$$PNCC(u, v) = \left(\frac{(u - cx) \cdot d(u, v)}{f_x \cdot s}, \frac{(v - cy) \cdot d(u, v)}{f_y \cdot s}, \frac{d(u, v)}{s} \right)$$

where (f_x, f_x, cx, cy) are intrinsic parameters, $d(u, v)$ is depth, and s is a scale factor. The resulting coordinates are shifted and globally normalized to preserve the aspect ratio and fit within the RGB range. Pixels without valid depth are masked during training. These are excluded from loss computation and optionally filled using nearest-neighbour interpolation for stability. They are also excluded from any evaluation.

The projection and backprojection processes are resolution-dependent, meaning the camera intrinsics must be adjusted according to the resolution at which these operations occur. If the original intrinsic parameters correspond to the native (high) resolution, they need to be scaled when applied to lower-resolution data. For instance, if a low-resolution (LR) depth map is obtained via bicubic downsampling from a raw high-resolution image, the intrinsics should be proportionally downscaled when converting the LR depth to LR PNCC. Then, after super-resolution is applied, the original (unscaled) intrinsics are used for reconstruction.

7.3.2 2D Super-Resolution

Model architectures. The SwinIR architecture is adapted to serve as the high-accuracy baseline. Since the PNCC representation maintains an image-like structure, the model can operate without architectural changes to the core pipeline. This adapted version is referred to as SwinT-PNCC.

For time-sensitive scenarios, DVMSR[137], a recent Vision Mamba-based super-resolution model designed for efficiency, is adapted. Unlike the transformer-based SwinIR[138], the Mamba architecture offers significantly lower computational cost while maintaining competitive performance. In this setup, the teacher version of DVMSR is used without applying distillation. This variant is referred to as VM-PNCC.

In both cases, the architectures share a similar construction, comprising a shallow feature extractor (a CNN in both cases), a deep feature extractor composed of a Swin Transformer block and a Vision Mamba block, respectively, and an upsampler that combines these feature sets to generate an output. Two distinct output layers in the upsampler are explored for two different approaches: the first is a 3-channel output for predicting directly the PNCC, while the second is a single-channel output focused solely on the Z component (depth), after which the entire PNCC is calculated using projection techniques. An ablation study contrasts both methodologies within the VM-PNCC architecture.

Training procedure. The super-resolution models are trained directly from scratch in PNCC rather than transferring patterns learned from unrelated domains such as RGB. Training is supervised by only the valid high-resolution pixels, and the pixel-wise Charbonnier loss[139] is employed. While other loss functions such as L1 and L2 were considered, the Charbonnier loss yielded more stable and robust results in this setting. Further details on training configuration are provided separately.

[137] Lei, X., Zhang, W., & Cao, W. (2024). Dvmsr: Distillated vision mamba for efficient super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6536-6546).
[138] Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., & Timofte, R. (2021). Swinir: Image restoration using swin transformer. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1833-1844).
[139] Barron, J. T. (2017). A general and adaptive robust loss function. 2019 IEEE. In CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 10).

7.3.3 Results

We evaluate our 2D-to-3D-SR framework using two representative datasets: NYUv2[140] and RGB-D-D[141]. NYUv2 is used to assess the accuracy and efficiency of our implementations under controlled conditions. RGB-D-D, being a human-centred, real-world dataset, is employed to test generalization and applicability in realistic scenarios, particularly those involving human subjects.

For NYUv2, we use a custom variant with proper intrinsics and scene-based splits to prevent leakage. Low-resolution inputs are generated by bicubic downsampling, and all models are trained on this set. The results (Table 19, Table 20) show that both SwinT-PNCC and VM-PNCC achieve competitive accuracy and significantly lower inference times compared to the guided baseline SGNet.

Table 19: Results of Depth SR in NYUv2 dataset at several upscaling factors. Best results are bold and second best underlined (excluding bicubic baseline). All methods are trained on NYUv2.

APPROACH	RMSE (cm)			TIME (sec)			# PARAMS		
	x4	x8	x16	x4	x8	x16	x4	x8	x16
Bicubic	25.22	34.80	47.19	0.002	0.002	0.002	0	0	0
SGNet	11.66	21.26	**35.85**	0.596	0.438	0.436	36.4M	39.9M	86.6M
SwinT-PNCC	**9.99**	**19.15**	<u>39.04</u>	<u>0.164</u>	<u>0.121</u>	<u>0.055</u>	<u>11.7M</u>	<u>11.8M</u>	<u>11.8M</u>
VM-PNCC	<u>11.19</u>	<u>20.15</u>	39.48	**0.045**	**0.027**	**0.024**	**7.2M**	**7.3M**	**7.3M**

In RGB-D-D, the focus shifts to real-world performance. Despite the domain gap introduced by sensor variations and scene content, our models outperform SGNet and bicubic baselines in accuracy, runtime, and model size.

Table 20: Results of Depth SR on RGB-D-D at 4x upscaling. Best results are bold (excluding bicubic baseline). Methods are trained on NYUv2 to test model generalisation to other domains.

APPROACH	RMSE (cm)	TIME (s)	# PARAMS
Bicubic	0.222	0.002	0
SGNet	0.232	0.381	36.4M
SwinT-PNCC	**0.212**	**0.109**	**11.7M**

To further evaluate perceptual realism, we visualize point clouds reconstructed from predicted depth maps on RGB-D-D. The visualisations clearly show the benefits of our unguided method in human-centric scenes. SwinT-PNCC produces the most coherent surfaces, avoiding the noise artifacts present in SGNet and the deformation observed with bicubic upscaling. Figure 83 showcases the visualisation of 2 different views of the prediction of each approach (Bicubic upscaling, SGNet, and our SwinT-PNCC model).

[140] Silberman, Nathan, et al. "Indoor segmentation and support inference from rgbd images." Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12. Springer Berlin Heidelberg, 2012.
[141] He, Lingzhi, et al. "Towards fast and accurate real-world depth super-resolution: Benchmark dataset and baseline." Proceedings of the ieee/cvf conference on computer vision and pattern recognition. 2021.

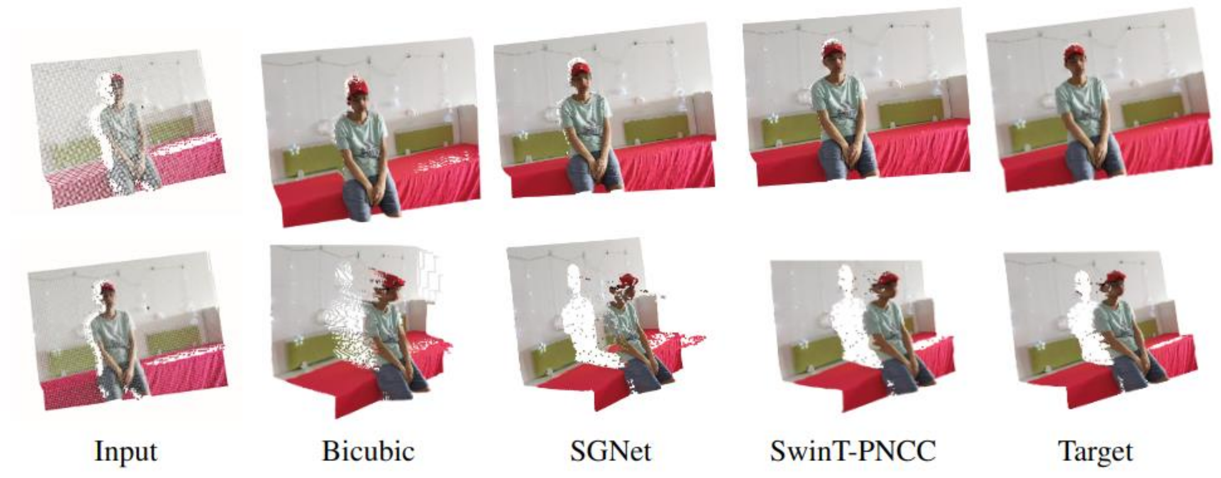


Figure 83: Qualitative results of x4 upscaling Depth SR in RGB-D-D, while trained on NYUv2.

The presented Depth-SR method has been deployed as a service via an API for upscaling 3D data with depth and camera parameters input. Future extensions of this work include expanding the current single-view 3D SR framework to multi-view settings, enabling its application as a point cloud upsampling solution that aggregates multiple surface observations. This would allow the framework to operate beyond single-view constraints while preserving its lightweight, image-based nature. We also plan to explore additional 3D data enhancement tasks within the same end-to-end framework, applying similar strategies to address challenges such as denoising, completion, or compression in structured 3D representations. Finally, we aim to improve the internal SR models by developing custom architectural modifications tailored to the properties of PNCC inputs and 3D geometry, potentially boosting both performance and robustness across diverse scenarios.

7.4 Human-centred point cloud data enhancement

A common challenge when synthesising virtual views from real 3D captures is the occurrence of self-occlusions, i.e. regions of a surface that are never visible from any input viewpoint and therefore appear as missing or undefined in the generated views. These occluded areas result in holes or gaps in the depth or geometry data, leading to incomplete or degraded virtual reconstructions. This phenomenon is illustrated in Figure 84.

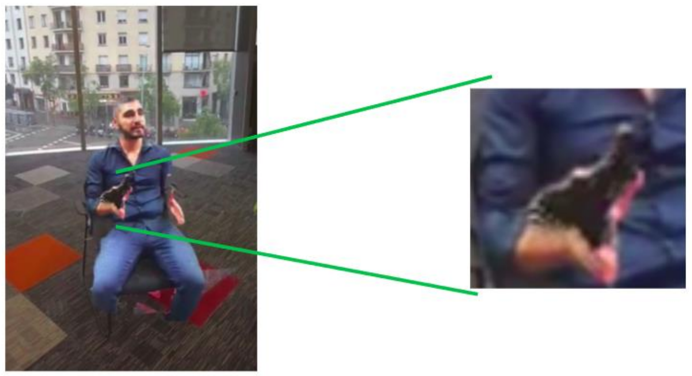


Figure 84: Self-occlusion problem illustration.

Self-occlusion presents a depth completion challenge requiring inference of plausible geometry for missing regions. However, like depth super-resolution, most existing methods depend heavily on high-quality RGB images to extract visual cues for depth enhancement, limiting applicability when RGB input is unavailable. We address this through a comprehensive 3D data enhancement perspective, treating geometric imperfections – sparsity, resolution degradation, and self-occlusions – as components of a unified restoration task. This enables end-to-end models capable of simultaneous multi-form enhancement. To our knowledge, no existing approaches target comprehensive, unguided 3D data enhancement without RGB input. We developed 3DPVM, a unified, unguided 3D data enhancement model performing both super-resolution and self-occlusion-aware depth completion end-to-end. The model operates directly on projected data without RGB guidance, optimised for practical runtime performance while recovering fine details and filling occluded regions in single-view representations.

Our approach contrasts with traditional depth completion methods, which predict dense depth maps from sparse inputs (LIDAR, stereo sensors). Guided methods leverage RGB structural and semantic cues through spatial affinity propagation (DySPN[142], NLSPN[143]), U-Net architectures[144], or dynamic fusion mechanisms[145]. Non-guided methods operate without RGB supervision, relying solely on sparse depth cues through classical approaches (sparse-to-dense) or diffusion processes[146]. Advanced techniques include RGB-D inpainting[147, 148] and multi-view completion[149, 150].

Critically, no existing approaches perform depth completion and super-resolution simultaneously in a unified framework. Current solutions employ sequential pipelines – either completion followed by super-resolution, or vice versa. These methods suffer from error propagation and suboptimal integration, as each stage operates independently without awareness of the full geometric degradation context. The prevalence of RGB-dependent approaches in both domains further validates our unified, RGB-free solution for applications with limited input information or real-time constraints.

7.4.1 Methodology

Building on the foundation of our prior 3D super-resolution work (Section 7.3), we develop 3DPVM, an unguided 3D data enhancement model capable of performing both super-resolution and depth completion simultaneously. The architecture is inspired by DVMSR[137], which leverages Vision Mamba for efficient upscaling.

[142] Lin, Yuankai, et al. "Dyspn: Learning dynamic affinity for image-guided depth completion." IEEE Transactions on Circuits and Systems for Video Technology 34.6 (2023): 4596-4609.
[143] Park, Jinsun, et al. "Non-local spatial propagation network for depth completion." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16. Springer International Publishing, 2020.
[144] Zhang, Youmin, et al. "Completionformer: Depth completion with convolutions and vision transformers." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.
[145] Wang, Yufei, et al. "Improving depth completion via depth feature upsampling." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.
[146] Lo, Kyle Shih-Huang, Jörg Peters, and Eric Spellman. "RoofDiffusion: Constructing Roofs from Severely Corrupted Point Data via Diffusion." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024.
[147] Lei, Jiabao, Jiapeng Tang, and Kui Jia. "Generative scene synthesis via incremental view inpainting using rgbd diffusion models." CoRR (2022).
[148] Dash, Ankan, Guiling Wang, and Tao Han. "Attentive Partial Convolution for RGBD Image Inpainting." Companion Proceedings of the ACM Web Conference 2024. 2024.
[149] Chen, Hao-Xiang, et al. "Circle: Convolutional implicit reconstruction and completion for large-scale indoor scene." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022.
[150] Chen, Honghua, Chen Change Loy, and Xingang Pan. "Mvip-nerf: Multi-view 3d inpainting on nerf scenes via diffusion prior." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

We introduce several modifications to adapt the framework for dual-purpose enhancement and robust operation without RGB input or prior knowledge of occluded regions. Unlike previous 3DSR implementations using nearest-neighbour interpolation for invalid pixels, 3DPVM incorporates partial convolutions[151] to distinguish and process valid versus missing regions. Our architecture comprises three stages: a shallow feature extractor, a deep feature extractor with Vision Mamba blocks, and an upsampler. Since the upsampler requires fully valid inputs, we ensure validity in deep features despite initial missing data.

The Partial Mamba module adapts Vision Mamba for partial inputs by using partial convolutions for embedding computation, assigning mean values to invalid pixels, and filtering fully invalid patches. Mean embeddings substitute patches without valid data, producing fully valid outputs. Residual connections retain the initial valid mask throughout the deep extractor to prevent unintentional updates.

The shallow feature extractor uses a single partial convolutional layer, potentially producing invalid areas. A filling module iteratively extends valid regains by updating newly recoverable invalid areas based on proximity to valid data. Without access to target valid masks, we cannot predefine background regions – particularly relevant for human models or structured foregrounds. Therefore, we add a mask prediction branch operating parallel to the enhancement pathway, mirroring the main architecture with its own filling module and upsampler to predict probability maps. Thresholding determines the final foreground mask. The architecture of our method is presented in Figure 85.

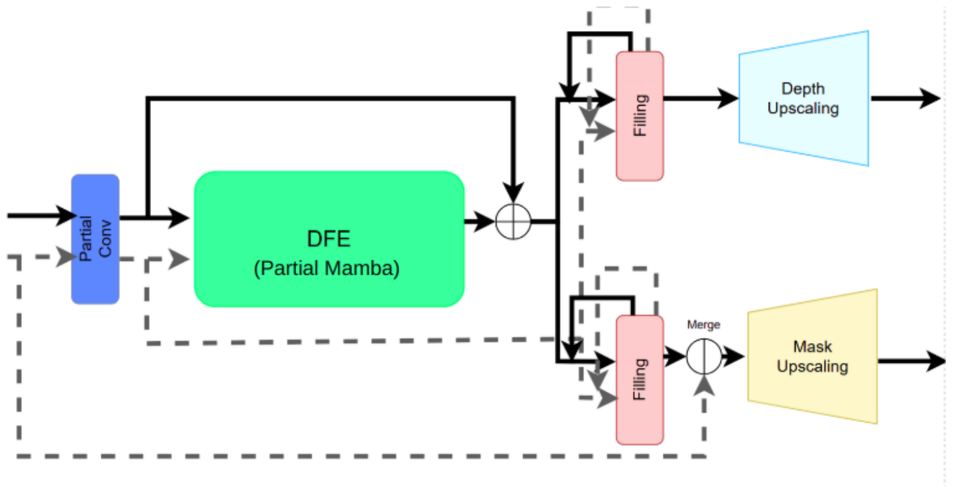


Figure 85: 3DPVM architecture.

The training objective combines multiple loss terms to support both enhancement accuracy and robust mask prediction: (1) A pixel-wise Charbonnier loss for the enhancement branch, encouraging accurate recovery of depth values. (2) A frequency-domain patch loss to suppress high-frequency noise and improve geometric regularity. (3) A binary cross-entropy (BCE) loss for the predicted mask to encourage accurate foreground/background separation. (4) A mask contour regularization term, based on patch-wise contour length, to ensure smooth and coherent mask boundaries. Concretely our loss term is:

[151] Liu, G., Reda, F. A., Shih, K. J., Wang, T. C., Tao, A., & Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. In Proceedings of the European conference on computer vision (ECCV) (pp. 85-100).

$$\mathcal{L}(\hat{y}, y) = \mathcal{L}_{depth}(\hat{y}_{depth}, y_{depth}) + \mathcal{L}_{mask}(\hat{y}_{mask}, y_{mask})$$

where

$$\begin{aligned}\mathcal{L}_{depth}(\hat{y}_{depth}, y_{depth}) &\\ &= \mathcal{L}_{charbonnier}(\hat{y}_{depth}, y_{depth}) \\ &+ \mathcal{L}_{freq}(\hat{y}_{depth}, y_{depth})\,\mathcal{L}_{charbonnier}(\hat{y}_{depth}, y_{depth}) \\ &= \mathcal{L}_{charbonnier}(\hat{y}_{depth}, y_{depth}) = \overline{\sqrt{(\hat{y}_{depth} - y_{depth})^2 + \varepsilon^2}}\end{aligned}$$

and the individual loss terms are:

$$\mathcal{L}_{freq}(\hat{y}_{depth}, y_{depth}) = \overline{DFT_{patch}(\hat{y}_{depth}) - DFT_{patch}(y_{depth})}$$

$$\mathcal{L}_{mask}(\hat{y}_{mask}, y_{mask}) = \mathcal{L}_{BCE}(\hat{y}_{mask}, y_{mask}) + \mathcal{L}_{contour}(\hat{y}_{mask}, y_{mask})$$

$$\mathcal{L}_{BCE}(\hat{y}_{mask}, y_{mask}) = \overline{(-(y_{mask} \cdot \log(\hat{y}_{mask})) + (1 - y_{mask}) \cdot \log(1 - \hat{y}_{mask}))}$$

$$\mathcal{L}_{contour}(\hat{y}_{mask}, y_{mask}) = \frac{1}{N_{patches}} \cdot \sum_{patch}^{N_{patches}} \left(\sum_{i,\, j\, \epsilon\, patch} \nabla\sigma(\hat{y} - \tau) - \sum_{i,\, j\, \epsilon\, patch} \nabla\sigma(y - \tau) \right)$$

This combined objective allows 3DPVM to learn structured, high-quality outputs that address both missing geometry due to occlusions and low-resolution detail in a unified, unguided pipeline. Following the same procedure as in DVMSR, the model is trained by small crops to allow using larger batches (due to GPU memory limitations).

Reprojection Data Augmentation. To enable learning of depth completion under self-occlusion scenarios, we implement a reprojection-based data augmentation strategy that introduces occlusions in a controlled, geometry-aware fashion. The goal is to simulate realistic missing regions that arise during virtual view synthesis, allowing the model to recover such areas using only depth information, without reliance on RGB inputs. This pipeline assumes access to a 3D reconstruction dataset containing multiple depth views per scene and corresponding camera parameters. For each training sample, a target view is selected as the ground truth, and a neighbouring source view is chosen to generate the occluded input. The selection process is based on a proximity distribution that ranks candidate views using camera position distance and overlap in visible regions, ensuring both spatial closeness and scene relevance. The depth map from the selected neighbouring view is then reprojected into the target view's coordinate frame. This

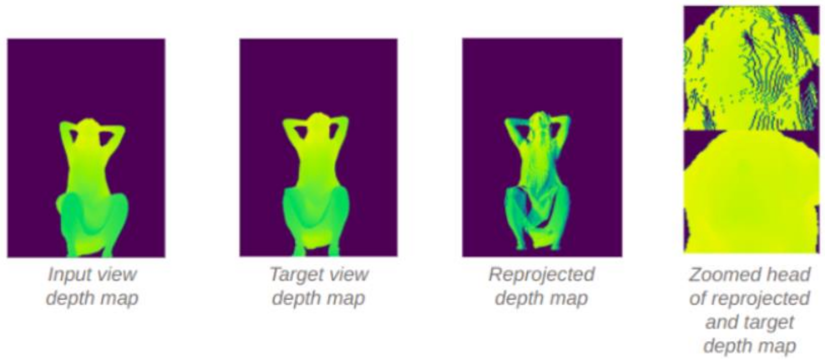


Figure 86: Example of the proposed data augmentation.

process naturally removes points from the target that are not visible from the source view, thereby creating a version of the input with realistic self-occlusion patterns. This reprojected depth map becomes the model input, and the original target view serves as the supervision signal. The resulting data augmentation on a sample depth map is presented in Figure 86.

To support training for both depth completion and super-resolution, we further downsample the reprojected depth to generate a low-quality (LQ) version of the occluded input. The resulting training pair consists of this augmented LQ map and the original high-quality (HQ) target, providing supervision for both enhancement tasks in a unified setup.

7.4.2 Experiments

We evaluated our proposed approach using the ActorsHQ dataset for both training and test (different sets, isolated actors and sequences were used for testing). We chain our data augmentation technique to the bicubic downscale degradation used previously in our 3DSR work to obtain the low-quality version of the samples, while using the original one as high-quality pair (input and target respectively). We compared our method with a bicubic upscaling and Nearest Neighbour interpolation (baseline) and a concatenation of Completionformer without RGB and a DVMSR model (as the state-of-the-art approach). Both models are trained from scratch to ensure they learn in the ActorsHQ domain for a fair comparison with our approach, including the proposed data augmentation pipeline for Completionformer. Completionformer is already forced to be RGB-blind during training, to ensure that it focuses solely on geometrical cues. DVMSR is trained with the high-resolution-low resolution ActorsHQ train, without self-occlusions.

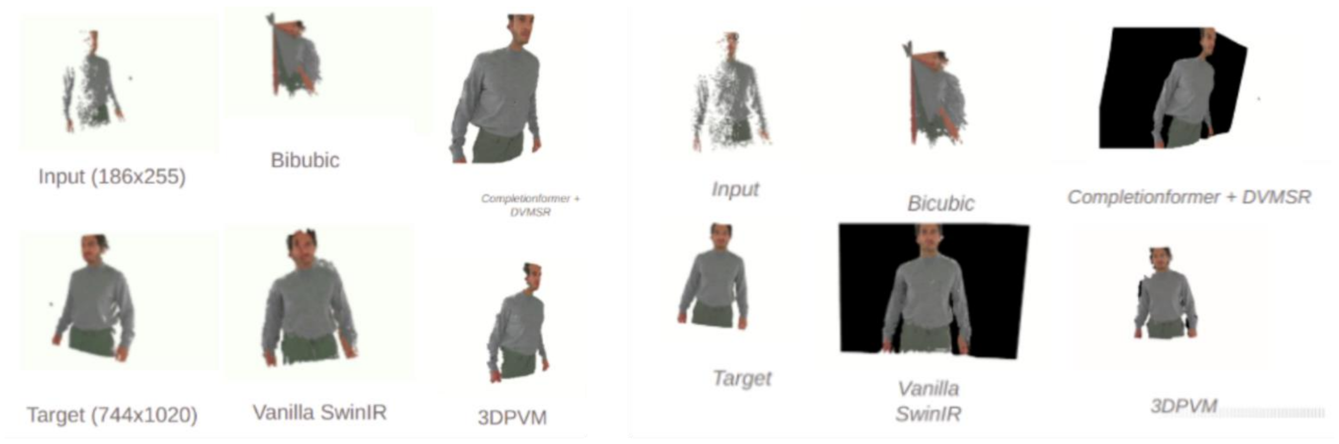
We make two types of evaluation: first, we use the target foreground mask (ignoring our mask prediction branch) to demonstrate that we are already superior in the data enhancement part. Second, we also demonstrate the results by using our predicted mask, while for the other approaches, which don't predict this mask, the whole image is evaluated.

Results (Table 21) using the target mask showcase the superior 3D data enhancement of our proposed 3DPVM approach in terms of accuracy and model weight. In time, we expected to outperform the state-of-the-art approaches by more, but we consider that this result is very variable because of the iterative nature of our filling module, which depends on the amount of missing information to recover:

Table 21: Results of depth enhancement on ActorsHQ. Best results are bold (excluding bicubic baseline). All methods are trained on ActorsHQ training-set.

APPROACH	RMSE (cm)	TIME (sec)	# PARAMS
Bicubic	11.07	0.021	0
Completionformer + DVMSR	16.40	0.178	85.2M
Vanilla SwinIR	9.90	0.346	11.7M
3DPVM	**8.94**	**0.172**	**7.6M**

We also include qualitative results to demonstrate the better performance of our approach. In Figure 87 views of the different resulting point clouds are shown. Using the target mask (a), 3DPVM already predicts a more plausible 3D shape. These differences are still more notable when using the predicted mask (b), as our model is the only one that does this task at the same time. In the qualitative results the reason of applying this mask prediction is clear, as otherwise when not disposing of the target mask the shown effect would be seen.



(a) with target mask *(b) without target mask*

Figure 87: Qualitative results of 3D data enhancement (SR+gap filling) with applying the target mask (a), and without (b).

Although our results look promising (and already unique in the literature) we expect them to improve in the near future. First, a better handling of the complete image context may be crucial for the quality of the results, especially for the mask prediction. We also expect to better exploit the power of the Partial Mamba by using alternative scans and partial modes (i.e. other kinds of padding than mean for the invalid data). The model can also easily achieve better inference times by replacing the iterative layers by something more deterministic, while probably also enhancing the output quality. Furthermore, we expect this to combine with other kinds of degradation such as sensor noise or data compression artifacts, to demonstrate that the model may be able to overcome these problems when trained with proper data. Finally, this makes even more sense to use in a multi-view setting, as the completion ability of 3DPVM may allow for filling occluded regions when using virtual views, generating complete Point Clouds.

7.5 3D Generation

3D Generation has been introduced to XReco as a very valuable service within the ecosystem. Apart from being able to create content when the data are available, there is a high demand in creating content from prompting, or from a very small amount of data. To this end, XReco explored and implemented text-to-3D technology powered by LLMs. The goal is to convert textual descriptions into detailed 3D models, which can be used in various applications such as virtual reality, gaming, and digital content creation. Four models were tested:

- Shap-E[152], a state-of-the-art model designed for generating 3D shapes from textual descriptions. It uses a combination of neural networks and probabilistic models to produce high-quality 3D geometries. Known for its high accuracy and ability to generate complex structures, Shap-E also supports fine-tuning to adapt to specific domains or improve performance on specialized tasks.

[152] Jun, H., & Nichol, A. (2023). Shap-e: Generating conditional 3d implicit functions. arXiv preprint arXiv:2305.02463.

- LLama Mesh[153] focuses on creating mesh representations of 3D objects from text, leveraging the power of LLMs to understand and interpret textual input, then generating detailed 3D meshes. LLama Mesh is praised for its precision and the high level of detail in its outputs, making it suitable for applications that require detailed and accurate 3D models.
- ThreeStudio[154] is a versatile 3D model generation tool that combines text-to-3D capabilities with a user-friendly interface. It uses LLMs to interpret textual descriptions and generate corresponding 3D models. Designed to be accessible to users with varying levels of expertise, ThreeStudio provides tools for both automatic generation and manual refinement of models
- Huyuan3D-1[155] utilizes advanced machine learning techniques to generate 3D models from text, focusing on generating realistic and high-fidelity 3D objects. Ideal for use in professional settings where quality is paramount, Huyuan3D-1 is known for its robustness and ability to handle a wide range of textual inputs.

Each model was evaluated based on its performance, accuracy, and ease of use.

Text-to-3D technology has a wide range of applications across different industries. In virtual reality and augmented reality (VR/AR), it can create immersive environments and objects, enhancing user experience by providing rich, interactive 3D content generated from simple textual inputs. In gaming, game developers can use this technology to generate game assets, characters, and environments quickly and efficiently, reducing development time and costs and allowing for rapid prototyping and iteration.

For digital content creation, content creators can generate 3D models for animations, videos, and other digital media, simplifying the content creation process and making it accessible to individuals without extensive 3D modelling expertise. Educational institutions and training programs can use text-to-3D models to create interactive learning materials and simulations, enhancing engagement and understanding through visual and interactive content. In e-commerce, online retailers can create 3D representations of products from descriptions, improving customer experience by providing detailed and interactive product views, potentially increasing sales.

Architects and designers can quickly generate 3D models of buildings, interiors, and objects from textual descriptions, facilitating rapid visualization and iteration of design concepts. Overall, text-to-3D technology offers significant benefits, including enhanced user experience, reduced development time and costs, simplified content creation, improved customer engagement, and accelerated design processes.

7.5.1 Detailed Comparison

In terms of accuracy and quality (Figure 88), Shap-E excels with high accuracy and the ability to generate complex and detailed structures. LLama Mesh offers very precise outputs with high detail, excelling in creating detailed mesh representations. ThreeStudio provides good accuracy with a balance between automatic generation and user control for refinement, while Huyuan3D-1 delivers excellent quality and realism, making it ideal for professional applications requiring high fidelity.

[153] Wang, Z., Lorraine, J., Wang, Y., Su, H., Zhu, J., Fidler, S., & Zeng, X. (2024). Llama-mesh: Unifying 3d mesh generation with language models. arXiv preprint arXiv:2411.09595.
[154] https://github.com/threestudio-project/threestudio/tree/main
[155] https://3d-models.hunyuan.tencent.com/

Performance-wise, Shap-E is efficient in generating models but may require fine-tuning for optimal performance. LLama Mesh offers high performance with quick generation times, suitable for real-time applications. ThreeStudio provides balanced performance, offering both automatic and manual options for users. Huyuan3D-1 exhibits robust performance, capable of handling complex and diverse inputs efficiently.

Regarding ease of use, Shap-E requires some expertise for fine-tuning and optimal use, whereas LLama Mesh is user-friendly with straightforward generation processes. ThreeStudio is highly accessible with tools for both beginners and advanced users, while Huyuan3D-1 is designed for professional use and may have a steeper learning curve.

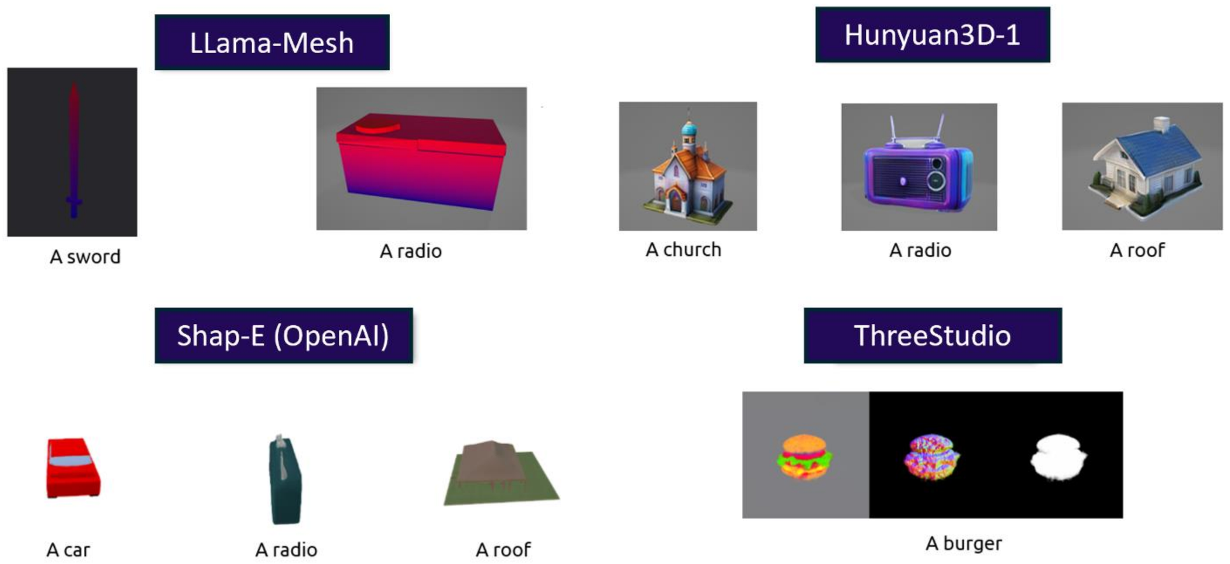


Figure 88: Comparison of the different text-to-3D models using the same prompts.

Shap-E is best for research and specialized applications requiring high accuracy. LLama Mesh is ideal for applications needing detailed meshes, such as gaming and simulations. ThreeStudio is suitable for a wide range of users, from hobbyists to professionals. Huyuan3D-1 is perfect for professional and high-fidelity applications, such as virtual reality and digital content creation.

Each of the four models tested in the XReco project has its own strengths and is suited to different applications. Shap-E and Huyuan3D-1 excel in generating high-quality, detailed models, while LLama Mesh offers precision and quick generation times. ThreeStudio provides a versatile and user-friendly approach, making it accessible to a broader range of users. The choice of model will depend on the specific requirements of the application and the level of detail and accuracy needed.

By implementing text-to-3D technology, the XReco project aims to revolutionize the way 3D content is created, making it faster, more accessible, and more efficient across various industries. As evidenced by the qualitative results in Figure 88, the Hunyuan3D-1 algorithm is integrated as text-to-3D service in XReco (Section 9.3).

8 Network acceleration infrastructure

This section presents the network acceleration technologies implemented to support XReco's computationally intensive media processing workflows. The infrastructure developments assess the critical performance requirements of GPU-driven NeRF rendering, 3D reconstruction, and large-scale media file processing through advanced memory management and unified computing frameworks.

8.1 Direct memory management

As described in D4.1, XReco's architecture is structured around a three-tier development model that encompasses the user experience, backend data management, and the deployment of systems on infrastructure. A core focus lies in infrastructure development, where network acceleration becomes essential due to the heavy demands of image- and video-based data flows. As XReco leverages cloud computing and GPU-driven NeRF and 3D reconstruction routines, the seamless transfer and processing of large, high-resolution media files require low-latency communication and efficient bandwidth usage. Kubernetes (K8s)[156] is employed to orchestrate AI workloads across compute clusters, particularly for large-scale models that exceed the capabilities of a single node. To mitigate latency in shared memory operations between nodes, remote direct memory access (RDMA) is utilized, enabling faster interconnects. Furthermore, XReco integrates the Unified Computing Framework (UCF) to streamline the development of AI-driven pipelines through tools like the graph composer, microservice builder, and application builder. This modular and scalable approach allows the decoupling of service, application, and infrastructure layers. A headless service deployment method ensures efficient RDMA connections within K8s environments, facilitating robust memory sharing. A prototype cluster equipped with GPUs and Network Interface Cards (NICs) is currently being deployed at CERTH, laying the foundation for scalable and efficient infrastructure tailored to XReco's media-intensive applications.

The architecture of RDMA as key enabling technology for UCF and the utilization of K8s was introduced in D4.1. In this document, we will first describe the results obtained on the implementation of RDMA, and then the development of a UCS/UCF demonstrator (a collaboration between NVIDIA and CERTH). Figure 89 shows the generalisation of the direct memory access for distributed processing, as presented in D4.1. The final goal is that the different processing elements can seamlessly share data over memory, skipping CPU processing time of the network elements. On top of this enabling technology, compartmentalised deployment of workloads as well as AI-pipelines can be built at low latency.

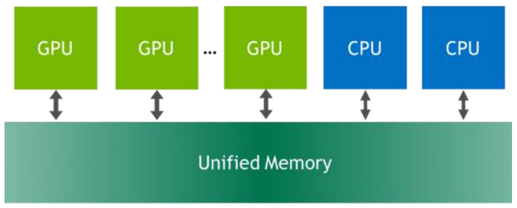


Figure 89: Generalisation of the direct memory access approach for distributed processing.

Figure 90 shows the high-level system utilised to benchmark the implementation within XReco of the shared memory approach; the setup leverages GPU Direct RDMA to enable high-speed, low-latency communication between NVIDIA GPUs and NICs, bypassing traditional data movement bottlenecks. Unlike conventional methods that require data to be copied from GPU memory to system memory and mediated by the CPU, GPU Direct RDMA

[156] https://kubernetes.io/

allows direct peer-to-peer data transfer, eliminating intermediate memory copies and reducing CPU overhead. This setup relies on specific CUDA[157] APIs that expose GPU memory for the creation of memory keys (MKEYs), which are essential for enabling RDMA operations. It also utilizes inter-driver callbacks between NIC and GPU drivers through the dmabuf framework[158] to ensure seamless memory sharing and data movement. However, GPU Direct RDMA is not compatible with systems that use Heterogeneous Memory Management (HMM) or Unified Virtual Memory (UVM), as these rely on kernel-managed memory rather than CUDA-managed memory. This architecture is particularly suited for scenarios requiring fast data exchange, such as AI model training or inference distributed across GPU clusters, and in particular the rendering of NeRF calls as in XReco.

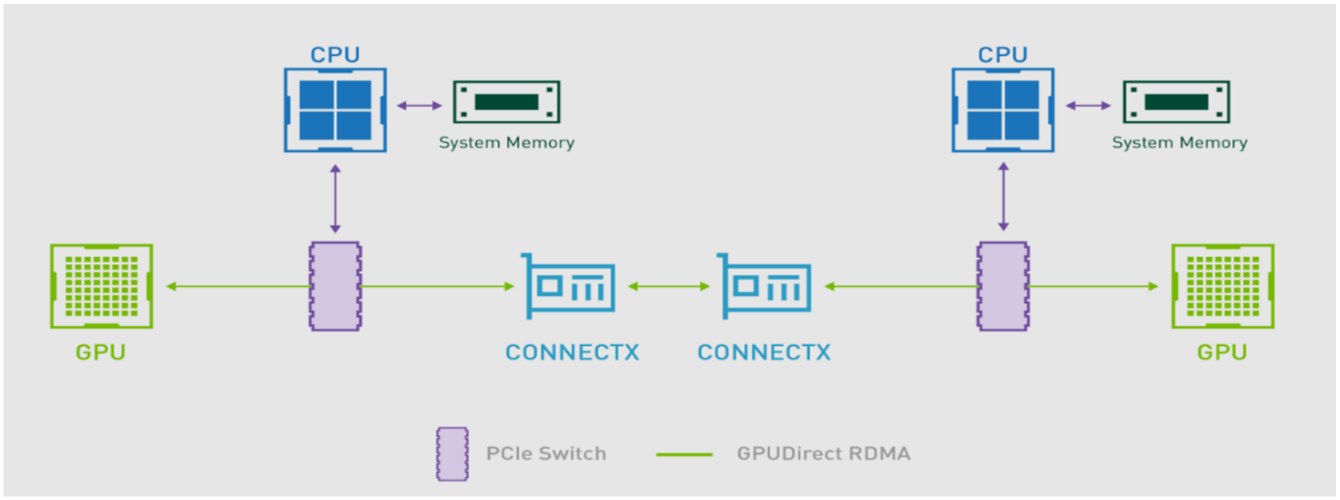
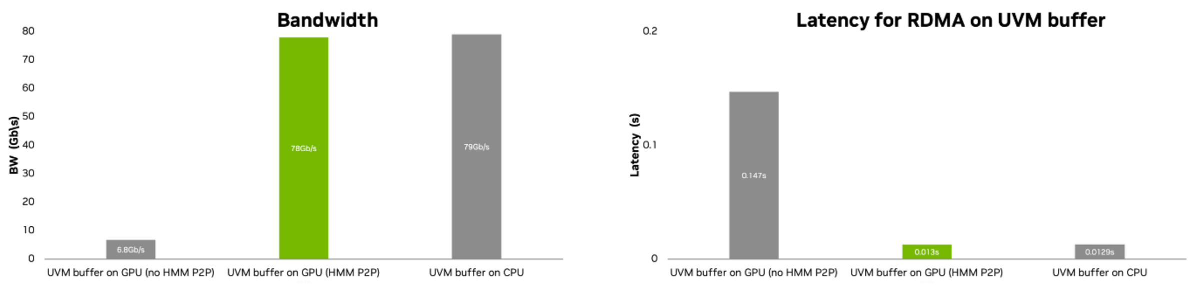


Figure 90: Implemented system for results generation.

Figure 91 shows the improvement in peer-to-peer performance when using direct memory access. The results show how bandwidth is only utilized at a fraction when utilizing the full stack, i.e., the data needs to follow the entire memory path. However, when direct memory is utilized, we reach a utilisation of around 80%, which is very positive considering there is always a network overhead. The network overhead includes the headers of the different networking stacks and the encryption overhead. As a result, around 80% bandwidth for a 100Gbit/s connection is achieved. Figure 91 (b) shows the latency reduction, amounting to circa 10% only of the latency induced when no memory sharing is employed. Operating at decimal fraction of the latency improves the user experience on rendering NERF models.

[157] https://developer.nvidia.com/cuda-toolkit
[158] https://docs.kernel.org/driver-api/dma-buf.html



(a) *(b)*

Figure 91: (a) Preliminary results of available bandwidth with memory sharing for GPUs and CPUs. (b) Latency reduction on shared memory buffers.

A 10× reduction in memory I/O latency can significantly improve the performance of NeRF rendering, especially in real-time or interactive applications. NeRF relies heavily on memory access, as rendering requires sampling multiple points along each viewing ray, querying a neural network for colour and density values at those points, and accumulating these results to produce the final image. Modern NeRF implementations, such as Instant-NGP[159], make use of data structures like multi-resolution hash grids, which require frequent and often random memory accesses. Lower latency directly accelerates these lookups, reducing delays in preparing input features for neural network inference.

Furthermore, the small neural networks used in NeRFs still require repeated access to weights and intermediate activations, which can become bottlenecks if memory latency is high. A 10× reduction in latency minimizes such stalls, leading to more efficient execution of the model. Since each ray can involve hundreds of sampling points, and each point triggers multiple memory operations, this latency improvement translates into a significant increase in overall ray throughput. As a result, NeRF rendering becomes faster and more scalable, with observed rendering speedups in the range of 2× to 5× depending on the hardware and model architecture.

In latency-sensitive use cases, such as real-time rendering for augmented or virtual reality, this improvement is particularly impactful. It enables higher frame rates and reduces visual lag, enhancing the user experience. Therefore, advances in memory technologies that reduce access latency, such as those leveraging 3D-stacked memory, near-memory processing, or novel caching techniques, are well-aligned with the computational needs of NeRF and similar volumetric rendering models. These results underscore the importance of low-latency memory systems in supporting next-generation neural rendering workloads.

8.2 Unified Cloud Services Tools (UCS/UCF)

Unified Cloud Services Tools (UCS Tools, UCS), also known as the Unified Compute Framework (UCF), is a toolset to develop cloud-native applications, and to streamline the development of AI-driven pipelines. It hosts optimized microservices that can be independently deployed, managed, and scaled within an application. It can be run in a distributed model, on multiple devices, across the cloud to edge and embedded platforms. Finally, it

[159] Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. ACM transactions on graphics (TOG), 41(4), 1-15.

allows domain experts to create their own microservices using UCS-compatible NVIDIA SDKs. Under the hood, a Microservice Specification is created to define infrastructure requirements, API endpoints, labels and other configurations. The Specification is built into a UCS Microservice, based on top of a container image. Furthermore, a single or multiple Microservices can be combined in an UCS Application.

This architecture decouples the different layers (service, application, and fabric), and enables individual configuration management while maintaining interoperability, hence allowing for scalability with a simple load and run execution in different environments. Additionally, utilising UCF pipelines continues to provide support for RDMA memory sharing applications.

Microservice Registry is the collection of UCS Microservices used during the development and operation. The total collection can come from both local repositories and/or cloud-hosted solutions like NGC (NVIDIA GPU Cloud). To work with existing applications, or to create new ones, the framework is both available through GUI interface, called UCS Studio, or command line interface (CLI) tools for Microservice and Application building.

8.2.1 Jetson Orin setup

A proof-of-concept setup was created to test the performance of UCF pipelines, consisting of two components. The first component, the development system, is hosting UCS Tools, and is responsible for creating and configuring the UCF pipelines. The Microservice and Application building CLI tools were installed, requiring Ubuntu 22.04 + x86_64 platform, on a Windows-based workstation PC via WSL2. The second component, responsible for deploying the UCS Microservices/Applications and hosting the Kubernetes cluster, a Jetson Orin AGX Developer Kit 64GB was used. Figure 92 shows the installation and configuration steps to complete the deployment system.

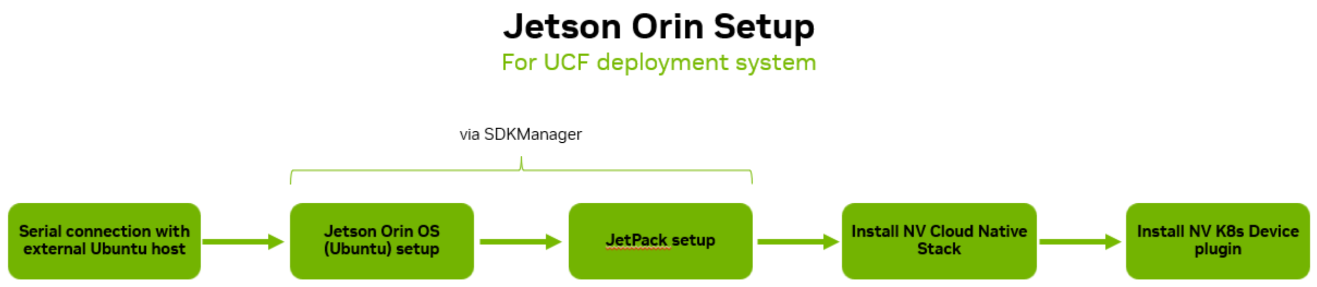

Jetson Orin Setup
For UCF deployment system
via SDKManager
Serial connection with external Ubuntu host
Jetson Orin OS (Ubuntu) setup
JetPack setup
Install NV Cloud Native Stack
Install NV K8s Device plugin

Figure 92: Roadmap of setting cluster on Jetson Orin.

Using the Orin out of the box requires reflashing of the OS and installing essential libraries. After powering up and booting in force recovery mode, the Orin and the installation host can be connected via USB, communicating on serial connection. The installation has to be an Ubuntu running machine (Windows with WSL2 is not suitable).

To complete the installation, NVIDIA's SDKManager is recommended. It requires an NVIDIA Developer account, which can be created for free. During the installation process, the right device type is to be selected to complete installing Ubuntu 22.04. The SDKManager also completes the installation of the newest available JetPack SDK version (6.2), and package consisting of kernel modules, drivers, necessary firmwares. The Jetson Orin has 64 GB built-in storage, which can be extended with an SD cards and/or NVMe type SSD. Due to the installed software and used container image sizes, in this system a 250 GB NVMe SSD extension was added, and the Jetson Orin OS flashed on it.

After completing the setup, and connecting to the network via wireless connection, the next installation was NVIDIA's Cloud Native Stack (v13.2), a software collection to run cloud native workloads on NVIDIA GPUs. It contains many libraries necessary to run cloud-native, GPU accelerated applications, such as Containerd, NVIDIA Container Toolkit, Kubernetes (including Microk8s), Helm, NVIDIA GPU Operator, etc. Additionally, NVIDIA's k8s-device-plugin is necessary to be installed to configure the Kubernetes cluster, enabling it to recognise the Jetson Orin's GPU as allocatable resource, and to be requested and used by the Kubernetes Nodes/Pods.

8.3 Test setup

To test and compare the performance of the UCF pipeline compared to a traditional Kubernetes deployment, a benchmark was created using PyTorch, a widely used python library in machine learning, while resource usage metrics were collected with built in Kubernetes and OS (L4T Tegra) tools.

Testing was based on PyTorch container image available on NGC (NVIDIA GPU Cloud), using version 25.04-py3-igpu. In the UCF case, UCS Microservice was configured based on the image, packaged into a UCS Application, and deployed via helm install on the Jetson Orin's cluster. In the control case, a Kubernetes manifest file was created for single run Pod job.

PyTorch's functions were benchmarked. The test calculates the batched dot product of different sized matrices, ranging from [1,1], [1, 64] to [10000,10000]. Two methods are used to achieve the batched dot products, one by a combination of multiplying and summing (torch.mul/torch.sum), and another by built-in PyTorch function (torch.bmm). These calculations were run on 1, 4, 16, and 32 process threads. Figure 93 shows the calculation times of these matrices in microseconds:

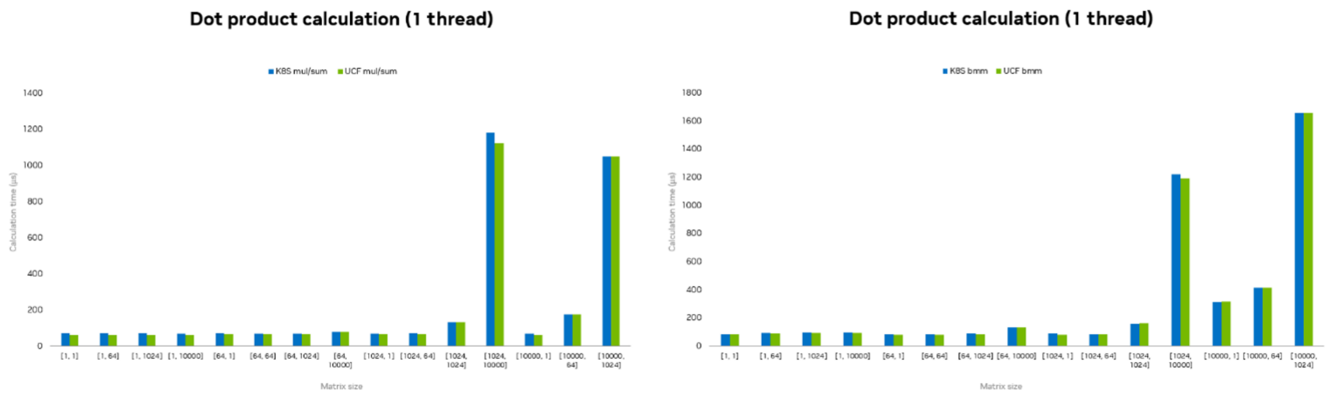


Figure 93: Matrix dot product calculations on Kubernetes and UCF.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

The calculations reveal that from matrix sizes of [1,1] to [10000, 1024], execution time is within a few microseconds in the Kubernetes and UCF cases, both scaling evenly with the size. At a significant jump, at matrix size [10000,10000], is where an approx. 2% increase in execution time can be noted, both in mul/sum and bmm cases, as can be seen in Figure 94. In every case, the number of threads does not impact performance, regardless of matrix size.

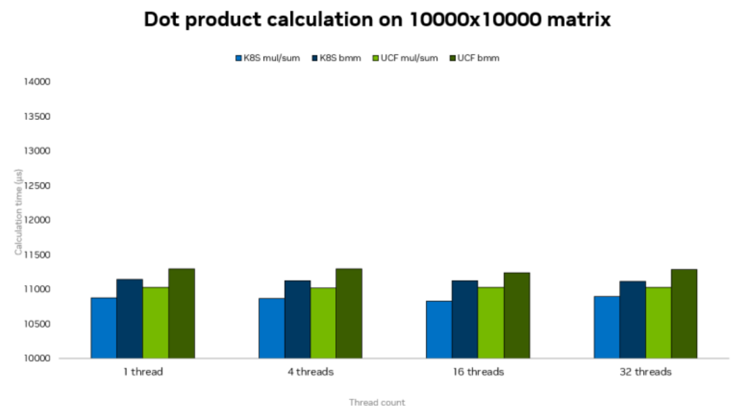
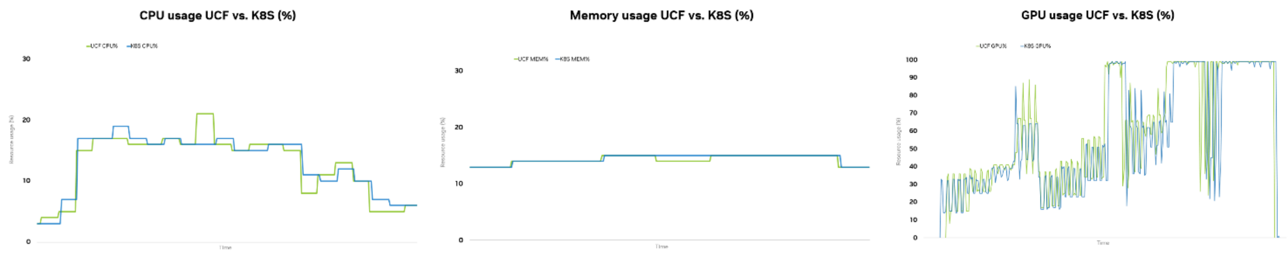


Figure 94: 10000 x 10000 size matrix dot product calculations on Kubernetes and UCF.

During the benchmarking test calculations, the demand for a variety of resources was monitored, such as CPU, memory and GPU utilization. The Kubernetes Node executing the UCS Application and the Kubernetes Pod jobs was measured via "kubectl top node" command, showing CPU and memory usage (MEM) in ratio to total available of each resource. For GPU, L4T Tegra OS included tool “tegrastats” was utilized, which queries the Jetson Orin device’s GPU usage.

Figure 95 shows the CPU, memory and GPU utilization, respectively, of the UCF and Kubernetes test cases:



(a) *(b)* *(c)*

Figure 95: (a) CPU usage, (b) Memory usage, and (c) GPU usage on Kubernets and UCF.

The results show very similar resource usage, in either CPU, memory, or GPU. This indicates that utilizing the UCF pipeline does not require extra resources compared to traditional Kubernetes deployments, meanwhile retaining features like RDMA support, individual configuration management for scaling, and NGC integration.

9 Authoring tool development and service communication

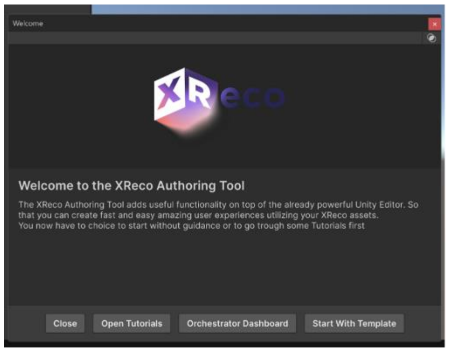
The various assets that can be found or generated on XReco can be used in a wide variety of applications. XReco provides three authoring tool experiences for compositing XR applications each requiring different sets of digital skills, considering professional workflows as well as more intermediate ones.

9.1 Unity-based authoring

In D4.1 (Section 7.1), a very high-level overview of the state of the Unity-based authoring tool was presented. Since then, many additional features have been implemented and applied on Unity to achieve flexibility and customisation tailored to the identified user-base targeting ease-of-use.[160] More specifically, we provide an easy link to the Orchestrator Dashboard (Section 9.3), so that the user has fast access to the assets. These assets can then easily (via Drag'n'Drop) be added to the scene. We also made a couple of templates available, so that the user does not need to start from scratch but can directly start with a solid base for the most common use cases.

9.1.1 Usage Overview

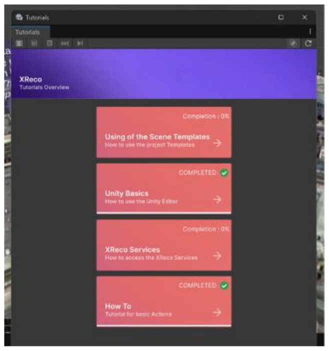
The user first needs to install Unity and then he can open the Authoring Tool (AT) project itself. In contrast to the plain Unity Editor, the user does not start from scratch, they are welcomed with a dialog (Figure 96-a). From this dialogue they can directly choose to first start with some interactive tutorials (Figure 96 – b, c). Additionally, they can choose to start from a template scene (Figure 97). The template scenes contain not only some out-of-the-box functionality, but are also documented and color-coded so that the user can use these as a starting point to rapidly create applications for his own content



Welcome to the XReco Authoring Tool

The XReco Authoring Tool adds useful functionality on top of the already powerful Unity Editor. So that you can create fast and easy amazing user experiences utilizing your XReco assets.
You now have to choice to start without guidance or to go trough some Tutorials first

(a)



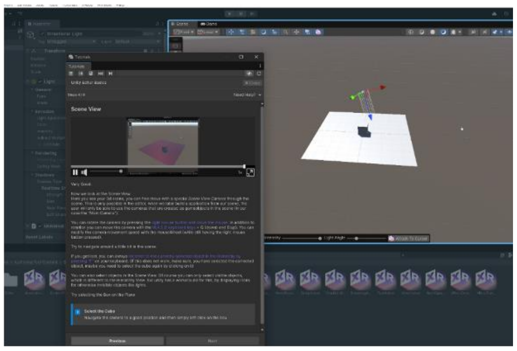
Completion : 0%
Using of the Scene Templates
How to use the project Templates

COMPLETED ✓
Unity Basics
How to use the Unity Editor

Completion : 0%
XReco Services
How to access the XReco Services

COMPLETED ✓
How To
Tutorial for basic Actions

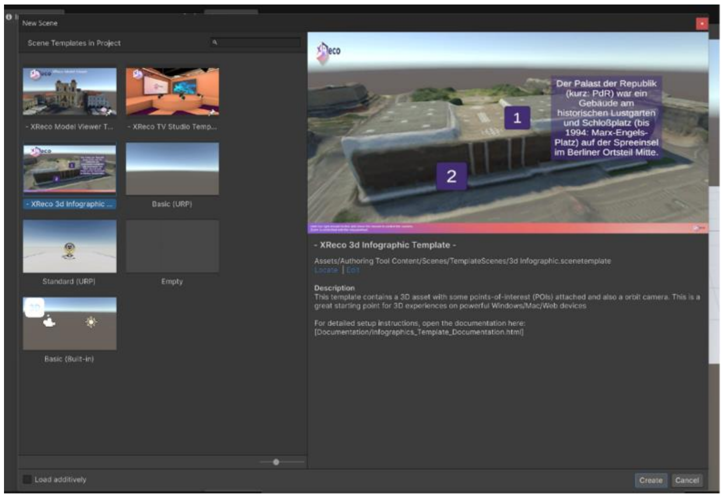
(b)



(c)

Figure 96: Unity based authoring project start. (a) Introduction screen when opening Unity. (b) Available interactive tutorials. (c) exemplary tutorial.

[160] For an animated overview of all features, visit our tutorial https://www.youtube.com/watch?v=9m2LAkUIEEk

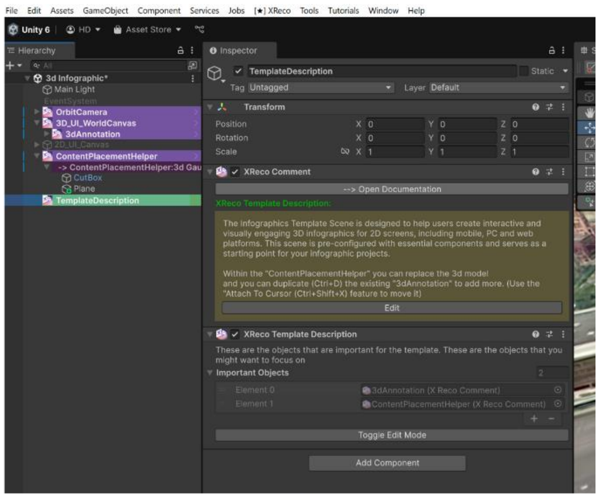

New Scene
Scene Templates in Project
- XReco Model Viewer T...
- XReco TV Studio Temp...
- XReco 3d Infographic ...
Basic (URP)
Standard (URP)
Empty
Basic (Built-in)
Der Palast der Republik (kurz: PdR) war ein Gebäude am historischen Lustgarten und Schloßplatz (bis 1994: Marx-Engels-Platz) auf der Spreeinsel im Berliner Ortsteil Mitte.
1
2
Load additively
Create
Cancel

- XReco 3d Infographic Template -

Assets/Authoring Tool Content/Scenes/TemplateScenes/3d Infographic.scenetemplate
Locate | Edit

Description
This template contains a 3D asset with some points-of-interest (POIs) attached and also a orbit camera. This is a great starting point for 3D experiences on powerful Windows/Mac/Web devices

For detailed setup instructions, open the documentation here:
[Documentation/Infographics_Template_Documentation.html]



Tag Untagged Layer Default

```
Position   X 0   Y 0   Z 0
Rotation   X 0   Y 0   Z 0
Scale      X 1   Y 1   Z 1
```

XReco Template Description:

The Infographics Template Scene is designed to help users create interactive and visually engaging 3D infographics for 2D screens, including mobile, PC and web platforms. This scene is pre-configured with essential components and serves as a starting point for your infographic projects.

Within the "ContentPlacementHelper" you can replace the 3d model
and you can duplicate (Ctrl+D) the existing "3dAnnotation" to add more. (Use the "Attach To Cursor (Ctrl+Shift+X) feature to move it)

These are the objects that are important for the template. These are the objects that you might want to focus on

Important Objects 2

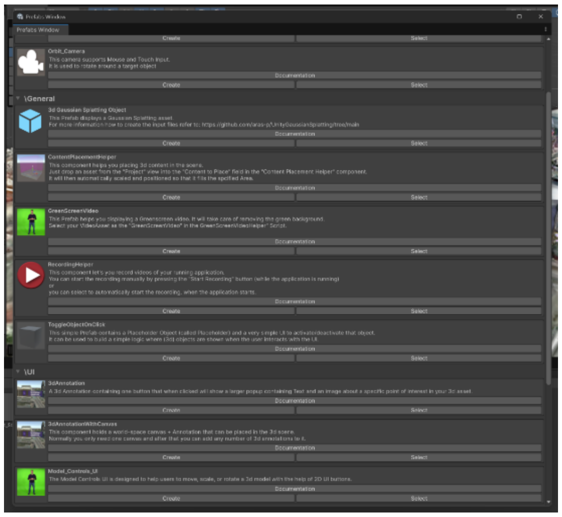
Element 0	3dAnnotation (X Reco Comment)
Element 1	ContentPlacementHelper (X Reco Comment)

(a) (b)

Figure 97: (a) Exemplary template scene. (b) Unity-based template functionality.

The AT also adds a range of reusable Prefab Objects to the Unity editor, which fit into many of the XReco use cases, easy to use and fully documented. These prefabs can be used in any scene.

The toolbar (Figure 99) also adds more functionality and connections to other packages, such as opening documentation pages for the integrated services developed within XReco, or backing up the current project.



Orbit_Camera
This camera supports Mouse and Touch Input.
It is used to rotate around a target object

\General

3d Gaussian Splatting Object
This Prefab displays a Gaussian Splatting asset.
For more information how to create the input files refer to: https://github.com/aras-p/UnityGaussianSplatting/tree/main

ContentPlacementHelper
This component helps you placing 3d content in the scene.
Just drop an asset from the "Project" view into the "Content to Place" field in the "Content Placement Helper" component.
It will then automatically scaled and positioned so that it fills the spcified Area.

GreenScreenVideo
This Prefab helps you displaying a Greenscreen video. It will take care of removing the green background.
Select your VideoAsset as the "GreenScreenVideo" in the GreenScreenVideoHelper" Script.

RecordingHelper
This component let's you record videos of your running application.
You can start the recording manually by pressing the "Start Recording" button (while the application is running)
or
you can select to automatically start the recording, when the application starts.

ToggleObjectOnClick
This simple Prefab contains a Placeholder Object (called Placeholder) and a very simple UI to activate/deactivate that object.
It can be used to build a simple logic where (3d) objects are shown when the user interacts with the UI.

\UI

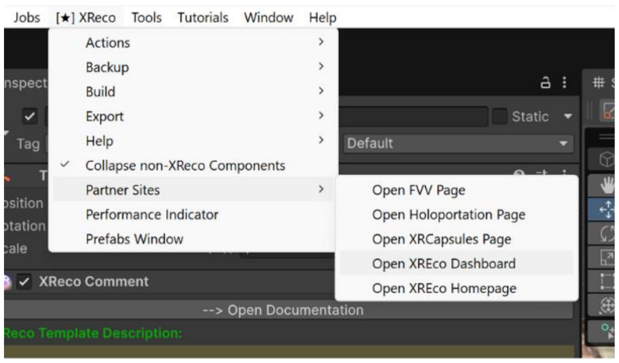
3dAnnotation
A 3d Annotation containing one button that when clicked will show a larger popup containing Text and an image about a specific point of interest in your 3d asset.

3dAnnotationWithCanvas
This component holds a world-space canvas + Annotation that can be placed in the 3d scene.
Normally you only need one canvas and after that you can add any number of 3d annotations to it.

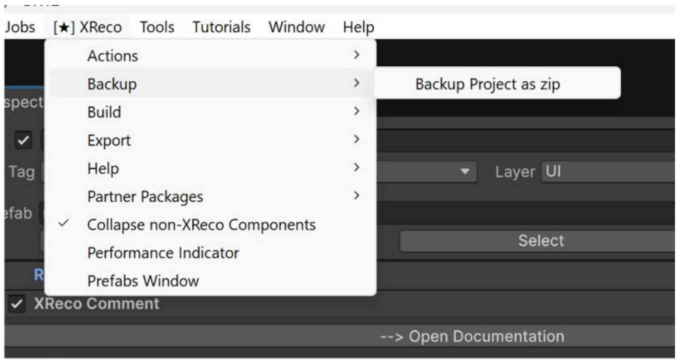
Model_Controls_UI
The Model Controls UI is designed to help users to move, scale, or rotate a 3d model with the help of 2D UI buttons.

Figure 98: Prefabs available in the Unity-based AT.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.



(a)



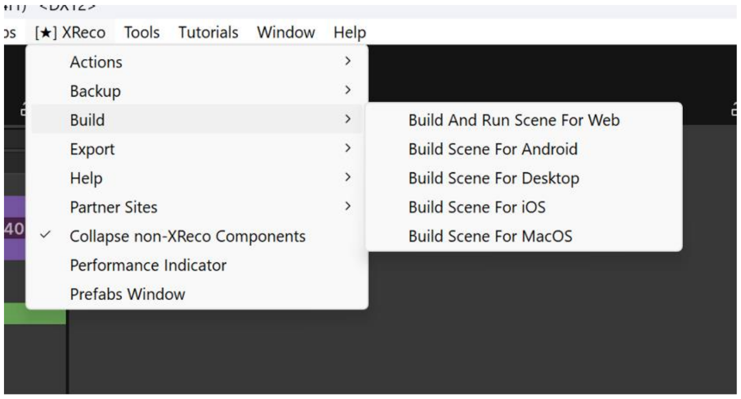
(b)

Figure 99: Unity-based AT toolbar options.

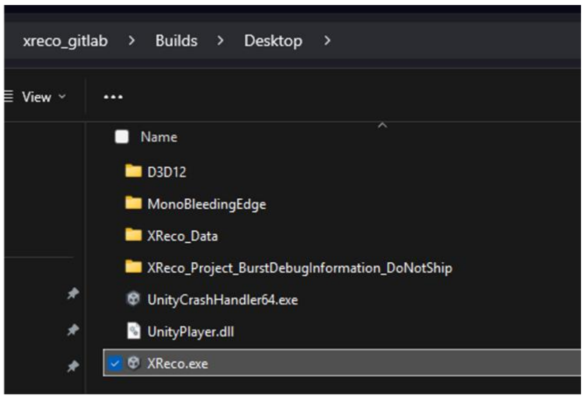
The AT also adds some useful 3rd Party packages, like GLTF importer or 3D Gaussian Splatting Support. The 3DGS package was also modified to support WebGPU builds.

9.1.2 Outcome

When the user is done with creating the application they can create a standalone app for different platforms, like mobile devices (iOS and Android), an HTML5 web application or a desktop experience (Figure 100-a). The output can then be shared or uploaded, so that other people can run the app without the need to install the Authoring Tool themselves (Figure 100-b).



(a)



(b)

Figure 100: Unity-based AT outputs. (a) Building for different platforms. (b) Output executable (for Windows in this example).

9.2 XRCapsules

XRCapsules is a web-based open source-application for creating XR experiences. Its goal is to serve as an easy way to generate Demonstrations, place assets in space, and program the execution of asset animations and access to triggers in a very simple way, to finally package it and build experiences using native Unity applications.

9.2.1 Motivation

Maintaining a complete XR development department that includes programmers, 3D artists, and ongoing application support can be an expensive endeavour. The cost of skilled professionals to develop and maintain an application can quickly add up, particularly when considering the need for regular updates and support. Additionally, software costs and hardware required to support the development process can also be significant.

Outsourcing development work may seem like a cost-effective solution, but it can come with its own set of challenges. Communication issues and cultural differences may arise, leading to delays and misunderstandings. Additionally, it may not provide the level of control and oversight that a complete development department can offer.

9.2.2 Templates and no-code

Using templates and no-code solutions can provide significant advantages over traditional development methods. With templates, developers can save time and effort by starting with pre-built structures and designs rather than building everything from scratch. This allows for a more streamlined development process, as well as faster turnaround times for new projects.

No-code solutions can also be incredibly beneficial, as they allow non-technical users to create and implement their own solutions without having to rely on specialized developers. This can be especially useful for smaller businesses or organizations that may not have the resources to maintain their own development teams. With no-code solutions, users can simply drag and drop elements to create their own custom solutions, without having to worry about coding or technical details.

Overall, using templates and no-code solutions can provide a more efficient and cost-effective approach to development, while also empowering non-technical users to create their own solutions (Figure 101).

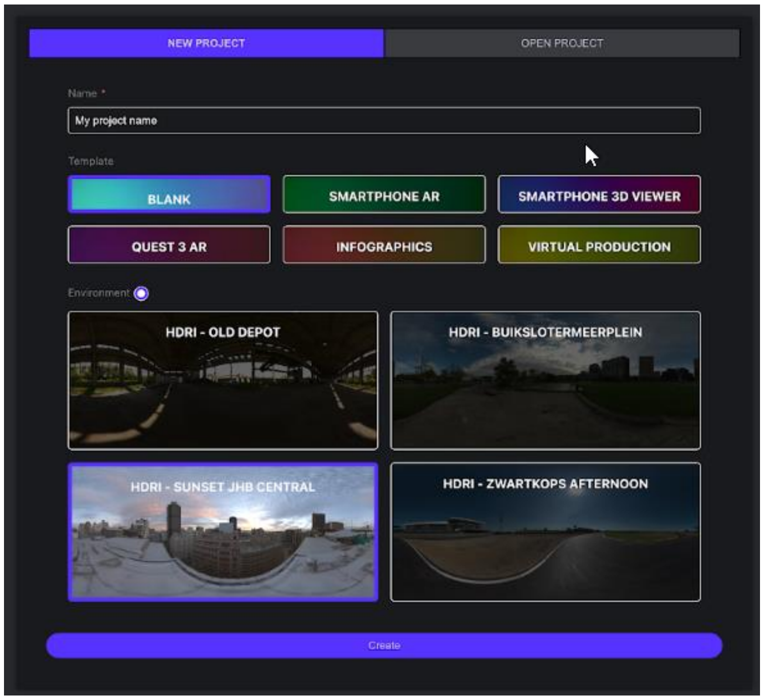


Figure 101: Overview of template creation in XRCapsules. The user can start from a blank project, templates for smartphone AR, Virtual production etc. In addition, the user can select a starting environment map illumination with pre-defined HDR images.

9.2.3 Flow and components

9.2.3.1 Assets

Assets are the building blocks of XRCapsules. They are the media elements that can be linked to templates to facilitate the creation of solutions, specifically the main demos like Home TV, GPS Based Walking, and GPS Based Car. There are various types of assets that can be used in the XRCapsules platform, which include:

- **3D Model.**
- **3D Character:** A three-dimensional character that can be animated and rendered in real-time.
- **360° Environment:** A panoramic image that covers the entire 360° field of view.
- **HDR files**: which are often used in lighting 3D scenes. These files contain a wider range of colour and brightness information than standard image files, which allows for more realistic and dynamic lighting in a 3D environment.

In addition to these, there are also video assets which can be linked to XRCapsules:

- **Text:** A textual element that can be used to display information.
- **Image:** A static image that can be used to provide visual information.
- **Video:** A video that can be played within the experience.
- **360° Video:** A video that covers the entire 360° field of view.
- **Stream:** A live video that is streamed from a server
- **FVV (Free Viewpoint Video):** A video that allows the viewer to move the camera position freely.
- **NeRF (Neural Radiance Fields):** A technique to represent 3D scenes as continuous functions, allowing for novel views to be rendered with high quality.
- **Point cloud:** A 3D point cloud that can be used to represent real-world objects.

9.2.3.2 Triggers

Triggers are used to interact or activate with assets in XRCapsules. There are two types of triggers:

- **User trigger:** This type of trigger is any way of interacting with an asset by the user. This includes clicking with a mouse or on a touch device, pressing a previously programmed key, or using sensors built into the device, such as blinking, gyroscope, etc.
- **Time trigger**: This type of trigger generates interaction from a specific time, which can be determined by the start of execution (absolute time) or from a user trigger (relative). The spawn of an asset in the scene is a time trigger.

Using triggers, assets can be animated and made interactive within XRCapsules, allowing for the creation of immersive and engaging experiences without effort.

9.2.3.3 Interface

We define this tool as a Platform-as-a-Service (PaaS), a web solution that connects with different elements and facilitates the creation or editing of XReco elements in an easy way.[161]

We start with a **webgl-based interface**, which we developed using three.js, always seeking universality of execution on different platforms (Figure 102).

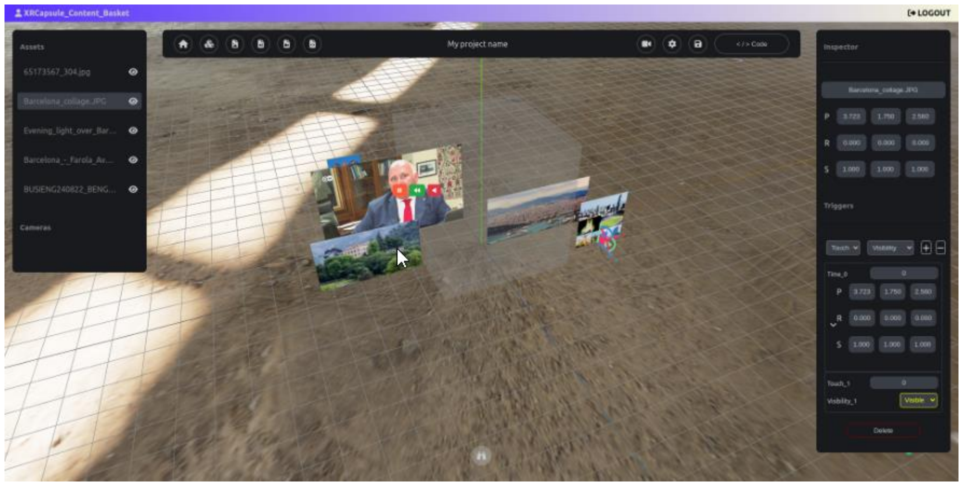


Figure 102: An example scene, showcasing the XRCapsules interface on a web viewer.

In any case, our goal is not to fully visualise all assets, but rather their physical representation through a bounding box or associated proxy and perform simulation and play in the associated XRCapsules Player application.

9.2.4 JSON interchange format

The exponential rise of 3D content creation across industries - ranging from architecture to virtual production - has exposed the inefficiencies of traditional monolithic file formats like USD and FBX, which bundle geometry, metadata, animations, and textures into a single structure. These formats, while powerful, introduce challenges in terms of version control, scalability, and real-time rendering, particularly in web, mobile, and cloud environments. JSON, when used as a reference format, solves these problems by externalizing heavy assets (GLB/GLTF) and retaining only the logical scene description (transformations, hierarchies, triggers, metadata) in a lightweight, readable structure. This separation of structure from content significantly reduces file size, allows selective loading of assets, and enhances modularity - making it ideal for dynamic rendering workflows, distributed production pipelines, and device-agnostic deployment.

Within the XRCapsules tool developed by Mediapro/Visyon as part of the XReco project, JSON enables non-technical users to compose rich XR experiences through a no-code interface, outputting JSON descriptors that orchestrate 3D scenes by referencing external assets. This descriptive format is cloud-native and optimized for both authoring and playback. It supports versioning, delta updates, and distributed rendering architectures by ensuring only the JSON layer (typically under 1MB) needs to be transmitted across networked environments, while heavy assets (10–500MB) remain cached or streamed as needed. Compatibility with WebGL/WebGPU and Unity ensures broad ecosystem integration, while we have implemented extensibility mechanisms (like rendering hints -"prefer_gaussian_v2") to future-proof scenes against evolving rendering technologies.

Use cases already include BIM/IFC translation for AEC, level design in gaming, object libraries in museums, and cloud-rendered XR demos in automotive or broadcast. For example, a museum viewer may browse thousands

[161] For an animated overview of all features, visit our tutorial https://www.youtube.com/watch?v=qTzAUb008VM

of digital artifacts with near-zero latency, loading only selected assets and their contextual metadata dynamically. Meanwhile, virtual production environments benefit from JSON's orchestration layer, which allows sub-millisecond updates across multi-node rendering clusters. This architecture positions JSON not just as a practical solution for asset interchange, but as a fundamental enabler of the next generation of spatial computing - where content is streamed, remixed, and rendered contextually in real time across diverse platforms and devices.

In Annex I – Section 11.3 provides a detailed overview of the JSON schema used for XRCapsules.

9.3 Orchestrating between user interfaces and services

In D4.1 the Orchestrator dashboard was presented showcasing the services integrated until December 2023. In this document we describe the latest developments of XReco's UI designed to support the full pipeline of XR content creation and reuse. Further developments of the Orchestrator dashboard are presented along with a new main component, the XReco Marketplace. This section focuses on the Orchestrator Dashboard, which serves as the central interface for interacting with the platform's core services.

The Orchestrator enables users to upload, search, and organize assets such as images, videos, and 3D models. It facilitates reconstruction and optimization workflows and supports publishing to the Marketplace. It also provides direct access to tools like XRCapsules (Section 9.2), ZAUBAR CMS (Section 9.3.1), and the Unity-based Authoring Tool (Section 9.1) for XR content creation.[162]

Key functionalities of the Orchestrator Dashboard include:

- **Browsing and searching content**: Search and browse external sources (e.g., Sketchfab, Wikimedia) or internal assets in the Neural Media Repository (NMR). Uploads are enriched with AI-based tagging for improved discoverability.
- **Managing content baskets**: Organise assets into user-defined baskets to manage projects and workflow inputs. Baskets serve as the context for launching processing jobs.
- **Launching reconstruction and asset optimization workflows**: Configure and launch reconstruction and optimization services directly from the dashboard.
- **Publishing assets**: Publish assets to the Marketplace with a built-in licensing wizard.
- **Accessing additional tools**: Seamlessly transition from asset management to XR content creation using XReco's connected authoring environments.

The high-level architecture of the Orchestrator Dashboard is illustrated in Figure 103.

[162] For an animated overview of all features, visit our tutorial https://www.youtube.com/watch?v=IU-drRzLZj4

The Orchestrator integrates a wide range of services and APIs, acting as the central hub for asset ingestion, processing, and publication within the XReco ecosystem. It connects with the following vertical services:

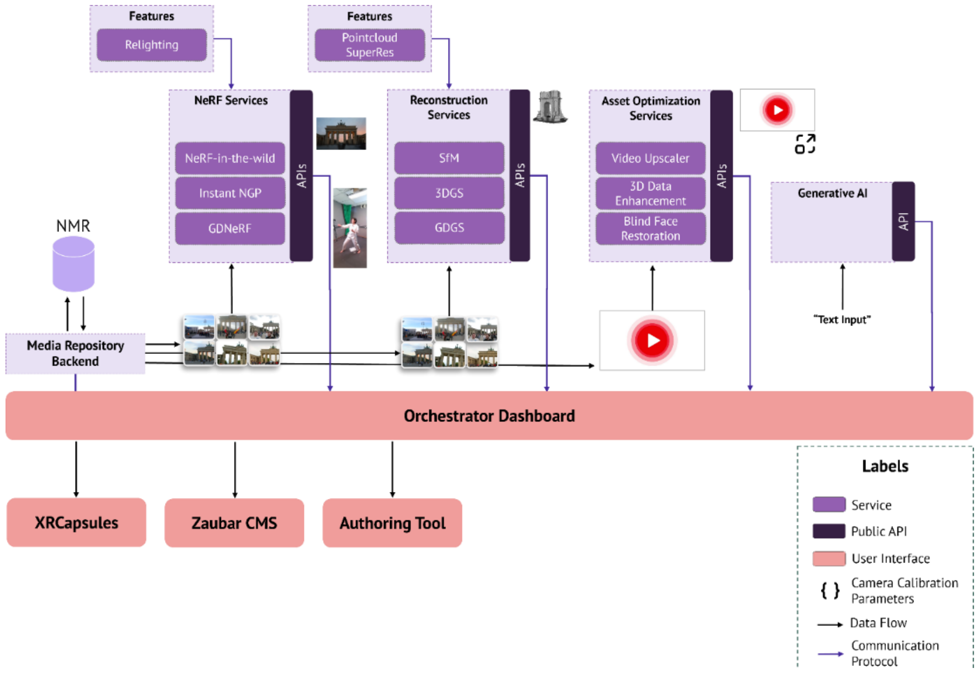


Figure 103: High-level architecture of the Orchestrator.

- NeRF Services (NeRF-in-the-wild, Instant NGP, GDNeRF) for 3D generation from 2D images.
- Reconstruction Services (SfM, GDGS, 3DGS) for 3D scene reconstruction.
- Asset Optimisation Services (Video Upscaler, 3D Data Enhancement, Blind Face Restoration) for improving the quality of images, video and 3D models.
- Generative AI Service for text-to-3D model generation.

The Orchestrator also integrates with key horizontal components and APIs of the XReco platform:

- The Neural Media Repository (NMR) for internal content storage, semantic enrichment, and advanced search. The NMR is enriched by AI-powered descriptor extraction services that generate metadata such as tags (from news tagging and object detection), visual similarity features, cross-modal descriptors (enabling image-to-model search), and semantic video summaries. These metadata enhance the discoverability and contextualization of assets within the Orchestrator interface.
- The Metasearch Backend for federated search across external repositories such as Sketchfab, Wikimedia, RAI, DW, and Europeana. It also powers the content baskets feature, allowing users to collect, organise, and manage assets from both external and internal sources within the Orchestrator interface.
- The Monetisation Management API for publishing assets to the Marketplace.
- Keycloak for secure authentication and centralised user management across the platform.

Together, these integrations enable a seamless workflow from asset acquisition and processing to content publishing and XR experience creation. The Minimum Viable Product (MVP) of the XReco Platform, including the fully functional Orchestrator Dashboard, has been implemented. The following sections provide a detailed walkthrough of the user experience.

9.3.1 XReco Platform Overview

When users access the XReco platform, they are first presented with the Login Page (Figure 104), where they enter their email and password to sign in.

Upon login, users are directed to the Orchestrator Dashboard. Before exploring its features, it is useful to understand the main navigation bar (Figure 105) available on all platform pages:

Figure 104: XReco platform: Login page.

1. **Asset Upload:** A dedicated page for uploading new assets to the platform.
2. **Orchestrator:** Main dashboard page of the XReco platform.
3. **Marketplace:** A separate area for discovering XR content (covered in D3.2).
4. **User Options:** Account settings mainly related to the Marketplace, such as profile management and purchased assets (covered in D3.2).
5. **Service Jobs:** Area for monitoring and managing ongoing/completed jobs.

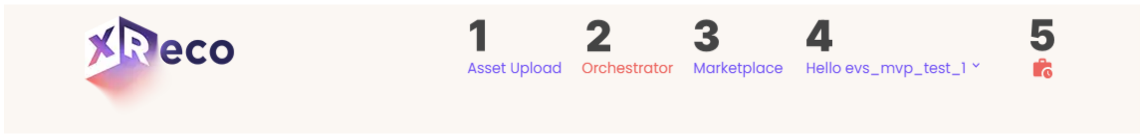


Figure 105: XReco Platform: Navigation bar.

The Orchestrator page is organized into several key sections: Repositories Selection, Content Basket Management, XReco Services, XReco Authoring Tools, and the Asset Browser. These sections are highlighted in Figure 106.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

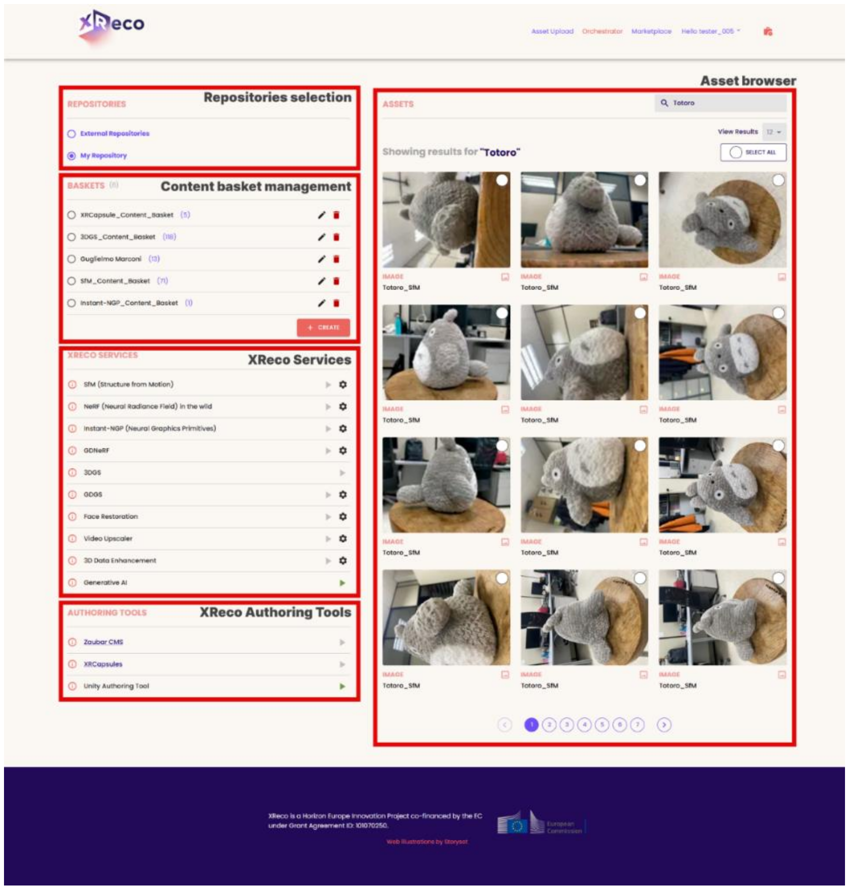


Figure 106: Orchestrator: Main page overview.

9.3.2 Repositories Selection

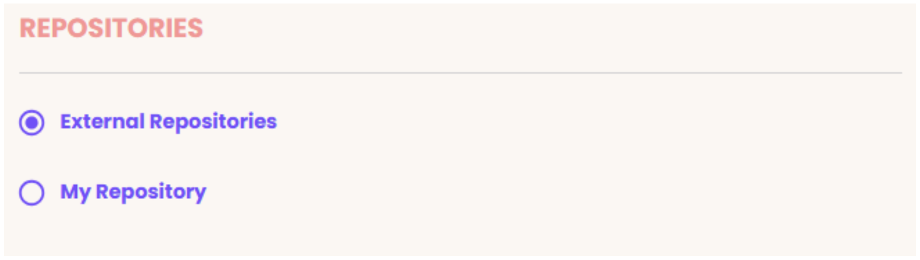


Figure 107: Orchestrator: Repositories selection

The Repositories Selection panel (shown in Figure 107) allows users to define the source from which they wish to browse or search for multimedia assets. Two options are available:

- **External Repositories**: This option provides access to external multimedia platforms, made available through integration with the Metasearch Backend.
- **My Repository**: This corresponds to the Neural Media Repository (NMR), which contains an organisation's internally managed multimedia content and is connected via the NMR Backend.

9.3.3 Content Basket Management

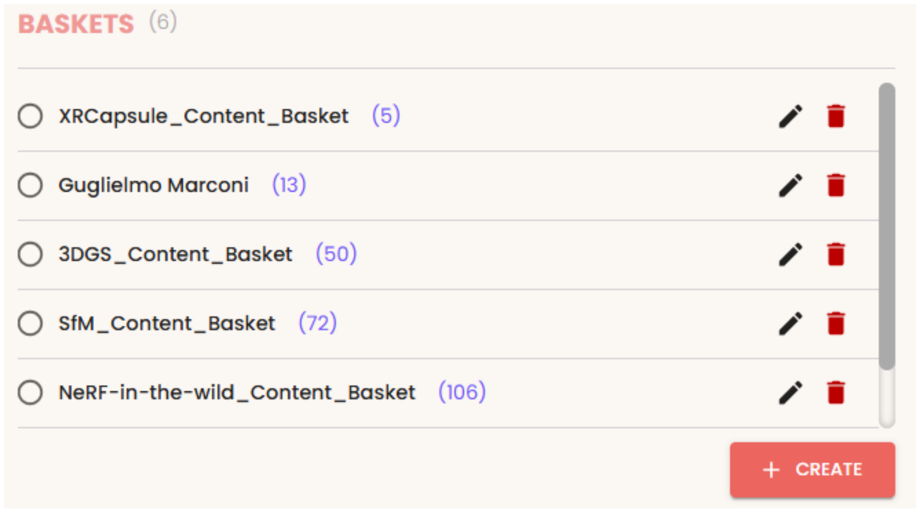


Figure 108: Orchestrator: Content basket management.

The Content Basket Management panel (shown in Figure 108) enables users to organise assets into curated collections known as content baskets. These baskets serve as user-defined workspaces that support efficient grouping, processing, and project organization.

Users can create a new basket by clicking the “+ CREATE” button, which opens a prompt to name the basket. Existing baskets can be renamed or deleted as needed. One basket can be designated as the active workspace at any given time.

Assets displayed in the Asset Browser - whether originating from search results or other baskets - can be added to a basket through a simple selection workflow. By clicking the white selection circle at the corner of an asset tile (1), a new “+ ADD” button (2) will become visible. Clicking this button opens a dropdown menu (3) to choose the target basket. This interaction is illustrated in Figure 109.

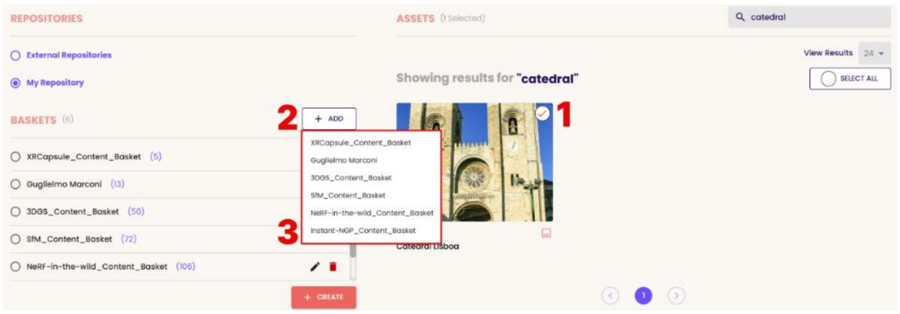


Figure 109: Orchestrator: Add asset to basket.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

9.3.4 Asset Browser

The Asset Browser provides a dynamic and user-friendly interface for searching, previewing, and managing multimedia assets within the XReco platform. The content displayed in the Asset Browser is determined by the active selection in the Repositories panel or by the currently selected content basket.

Only one content source - either a repository (External or NMR) or a content basket - can be active at any given time. When a repository is selected, the Asset Browser activates a search interface via a search bar located in the top-right corner of the section. This search bar is disabled when a content basket is selected, in which case the Asset Browser displays the assets already added to that basket.

Several features in the Asset Browser are always available, regardless of the selected source:

- **Bulk asset selection**: A “Select All” button allows users to select all assets displayed on the current page. Individual asset selection is also possible by clicking the circle icon in the top-right corner of each asset tile, as mentioned previously.
- **Pagination and display options:** Users can navigate through result pages and adjust the number of assets shown per page, improving the usability of the interface when working with large datasets.

The following subsections provide a more detailed overview of how the search operates when working with external repositories and with the Neural Media Repository.

9.3.4.1 Search External Assets (Metasearch)

When the “External Repositories” option is selected in the Repositories panel, users can perform searches across integrated external platforms such as Sketchfab, Wikimedia Commons, RAI, Deutsche Welle, and Europeana. This functionality is powered by the Metasearch API and supports a wide variety of asset types, including images, videos, 3D models, audio files, and textual resources.

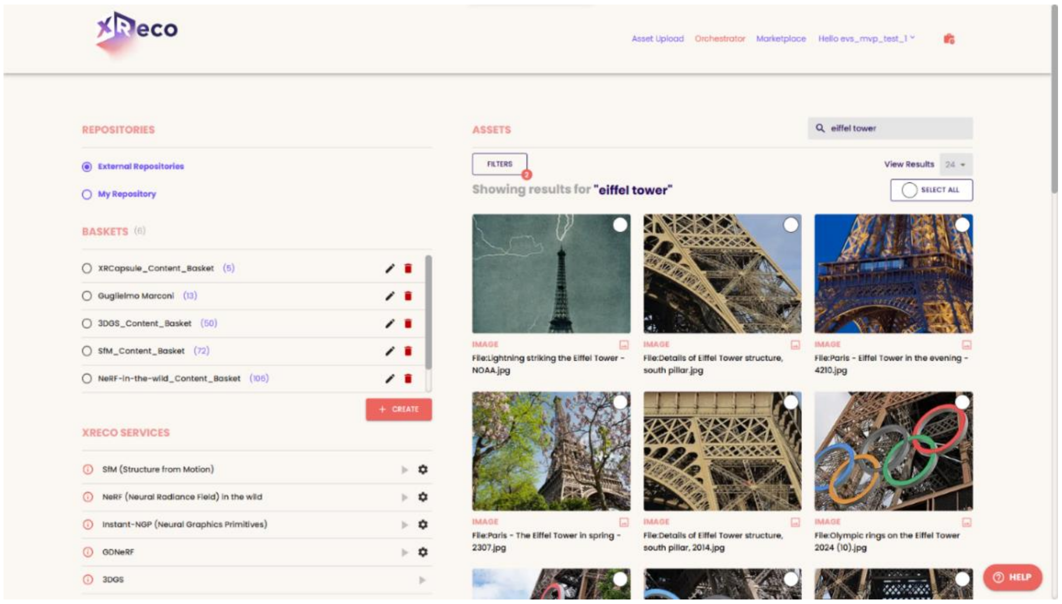

XReco
Asset Upload
Orchestrator
Marketplace
Hello evs_mvp_test_1
REPOSITORIES
External Repositories
My Repository
BASKETS (6)
XRCapsule_Content_Basket (5)
Guglielmo Marconi (13)
3DGS_Content_Basket (50)
SfM_Content_Basket (72)
NeRF-in-the-wild_Content_Basket (106)
+ CREATE
XRECO SERVICES
SfM (Structure from Motion)
NeRF (Neural Radiance Field) in the wild
Instant-NGP (Neural Graphics Primitives)
GDNeRF
3DGS
ASSETS
eiffel tower
FILTERS
View Results 24
Showing results for "eiffel tower"
SELECT ALL
IMAGE
File:Lightning striking the Eiffel Tower - NOAA.jpg
IMAGE
File:Details of Eiffel Tower structure, south pillar.jpg
IMAGE
File:Paris - Eiffel Tower in the evening - 4210.jpg
IMAGE
File:Paris - The Eiffel Tower in spring - 2307.jpg
IMAGE
File:Details of Eiffel Tower structure, south pillar, 2014.jpg
IMAGE
File:Olympic rings on the Eiffel Tower 2024 (10).jpg
HELP

Figure 110: Orchestrator: Search from external repositories.

As shown in Figure 110, users initiate a search by entering a query—e.g., “Eiffel Tower”—in the search bar. The resulting assets are displayed in the Asset Browser, organised by relevance.

To refine search results, users can click the “Filters” button, which opens a panel with several filtering options (illustrated in Figure 111). Available filters include:

- **Category:** Filter results by content type (e.g., image, video, 3D model).
- **License Type:** Limit results to assets with specific usage rights (e.g., CC0, CC BY).
- **Source Repository:** Restrict results to a particular external provider (e.g., Sketchfab).

After selecting the desired filters, users must click the “Apply filters” button to execute the filtered search.

Selecting an asset from the search results opens a detailed preview pop-up (examples shown in Figure 112). This interface not only displays the asset itself but also presents the associated metadata. The layout of this preview adapts to the asset type. For example, a 3D model will include an interactive viewer, while audio and video assets feature playback controls.

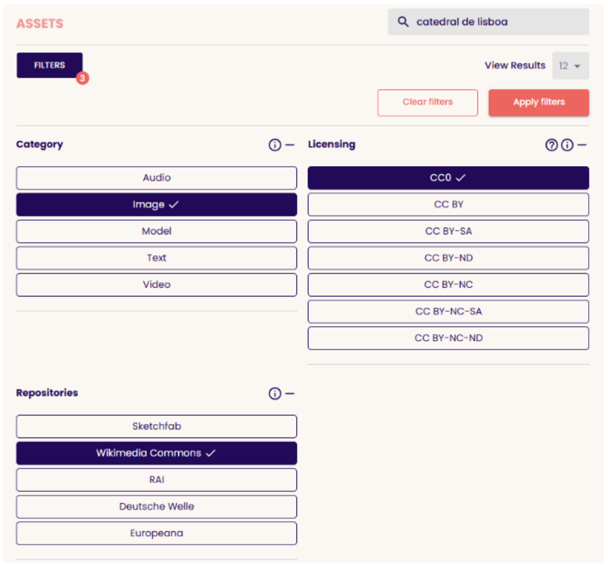
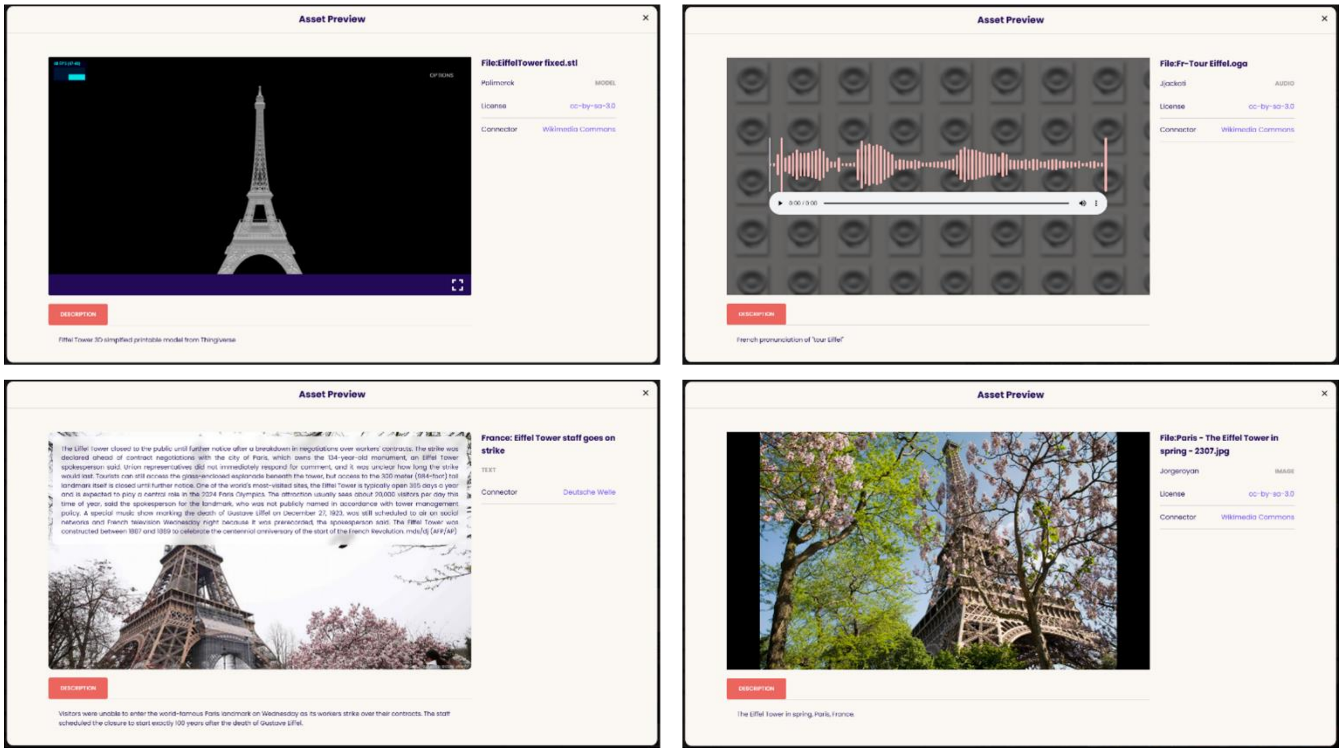


Figure 111: Orchestrator: filters (external assets).



File:EiffelTower fixed.stl

Polimorok	MODEL
License	cc-by-sa-3.0
Connector	Wikimedia Commons

Eiffel Tower 3D simplified printable model from Thingiverse

File:Fr-Tour Eiffel.oga

Jjackoti	AUDIO
License	cc-by-sa-3.0
Connector	Wikimedia Commons

French pronunciation of "tour Eiffel"

The Eiffel Tower closed to the public until further notice after a breakdown in negotiations over workers' contracts. The strike was declared ahead of contract negotiations with the city of Paris, which owns the 134-year-old monument, an Eiffel Tower spokesperson said. Union representatives did not immediately respond for comment, and it was unclear how long the strike would last. Tourists can still access the glass-enclosed esplanade beneath the tower, but access to the 300 meter (984-foot) tall landmark itself is closed until further notice. One of the world's most-visited sites, the Eiffel Tower is typically open 365 days a year and is expected to play a central role in the 2024 Paris Olympics. The attraction usually sees about 20,000 visitors per day this time of year, said the spokesperson for the landmark, who was not publicly named in accordance with tower management policy. A special music show marking the death of Gustave Eiffel on December 27, 1923, was still scheduled to air on social networks and French television Wednesday night because it was prerecorded, the spokesperson said. The Eiffel Tower was constructed between 1887 and 1889 to celebrate the centennial anniversary of the start of the French Revolution. mds/dj (AFP/AP)

France: Eiffel Tower staff goes on strike

TEXT	
Connector	Deutsche Welle

Visitors were unable to enter the world-famous Paris landmark on Wednesday as its workers strike over their contracts. The staff scheduled the closure to start exactly 100 years after the death of Gustave Eiffel.

File:Paris - The Eiffel Tower in spring - 2307.jpg

Jorgeroyan	IMAGE
License	cc-by-sa-3.0
Connector	Wikimedia Commons

The Eiffel Tower in spring, Paris, France.

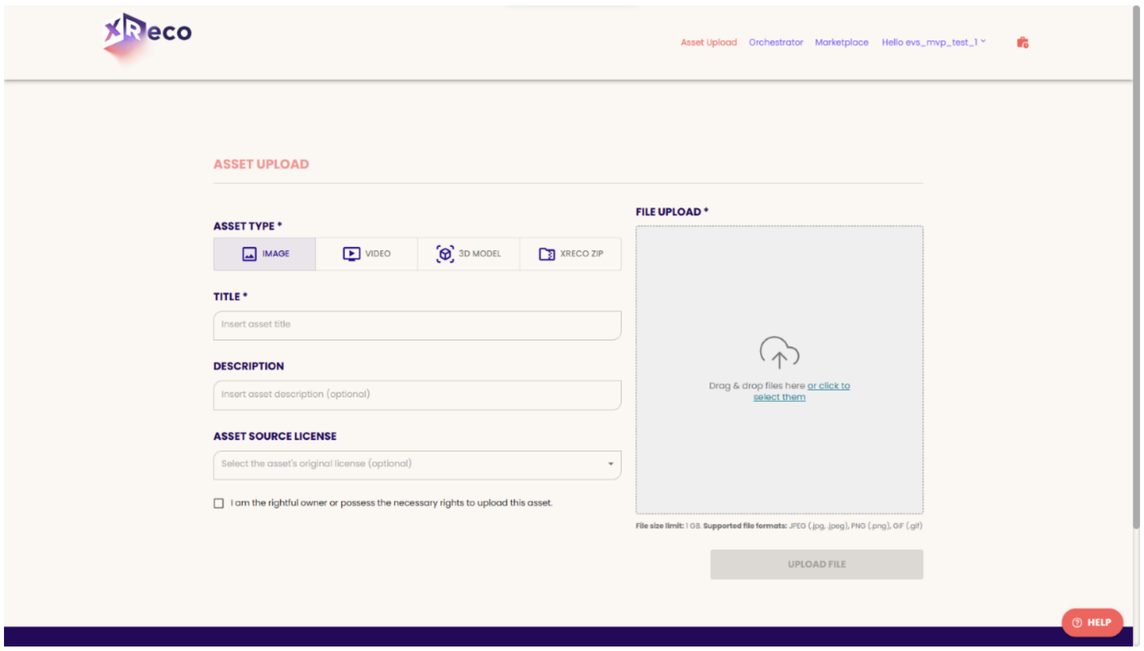
Figure 112: Orchestrator: Asset preview (external assets). (from top left to bottom right: 3D model, audio, text, image).

Metadata shown in the preview includes essential information such as the asset's title, description, type, source repository, licensing terms, and user details. This contextual information allows users to assess whether an asset is suitable for reuse or integration into their content workflows.

When external assets are added to a content basket, they are ingested into the Neural Media Repository (NMR). Once ingested, these assets gain access to advanced tools such as AI tagging and visual similarity search. These functionalities are described in the following section.

9.3.4.2 Search Organisation's Assets (NMR)

Assets can be ingested into the Neural Media Repository (NMR) through two methods: either by selecting external assets and adding them to a content basket, which triggers automatic ingestion, or by manually uploading files via the Asset Upload page shown in Figure 113.



ASSET UPLOAD

ASSET TYPE *
IMAGE | VIDEO | 3D MODEL | XRECO ZIP

TITLE *
Insert asset title

DESCRIPTION
Insert asset description (optional)

ASSET SOURCE LICENSE
Select the asset's original license (optional)

☐ I am the rightful owner or possess the necessary rights to upload this asset.

FILE UPLOAD *
Drag & drop files here or click to select them

File size limit: 1 GB. **Supported file formats:** JPEG (.jpg, .jpeg), PNG (.png), GIF (.gif)

UPLOAD FILE

Figure 113: Asset upload page.

The Asset Upload page, accessible from the top navigation bar, provides a simple interface for uploading local files to the platform. Users can either drag and drop files or browse their device to upload images, videos, 3D models, or ZIP archives. ZIP files are suitable for complex or unsupported formats and must follow the NMR's required structure - details of which are available via an on-screen pop-up. A title is required for each upload, while a description and license information are optional. Once submitted, assets are automatically processed and become searchable within the NMR.

After uploading, users can return to the Orchestrator. By selecting "My Repository" in the Repositories panel, they can search their organisation's assets using the NMR API. As shown in Figure 114, users can enter a query—e.g., *"Einstein Tower Dataset"*—into the search bar to locate relevant files, which are then displayed in the Asset Browser.

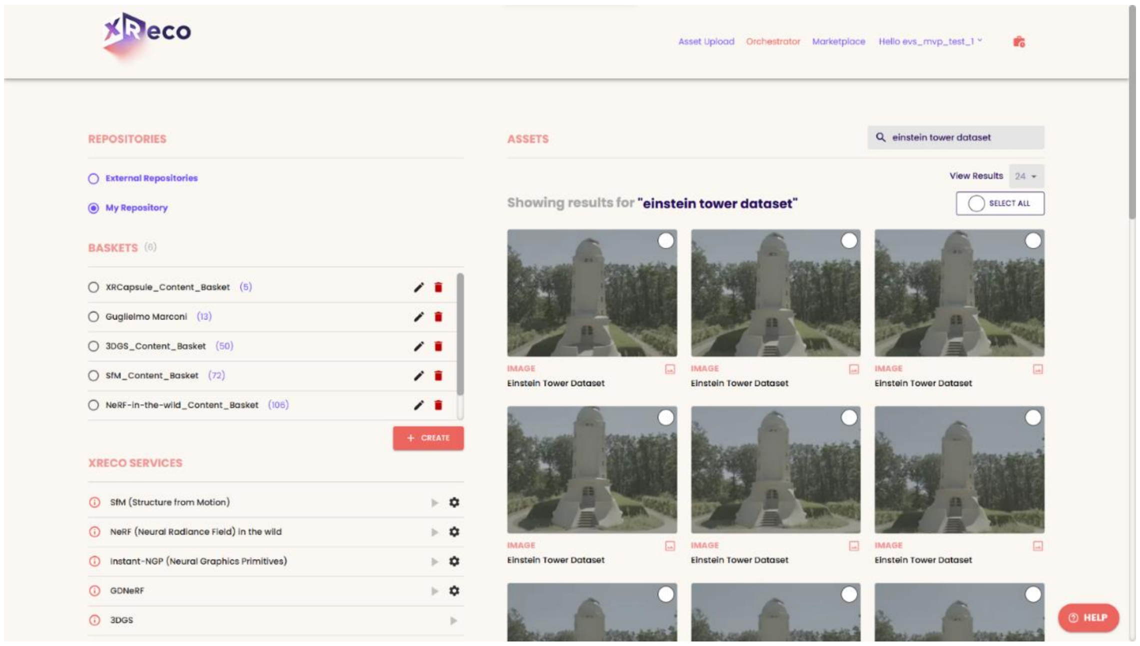
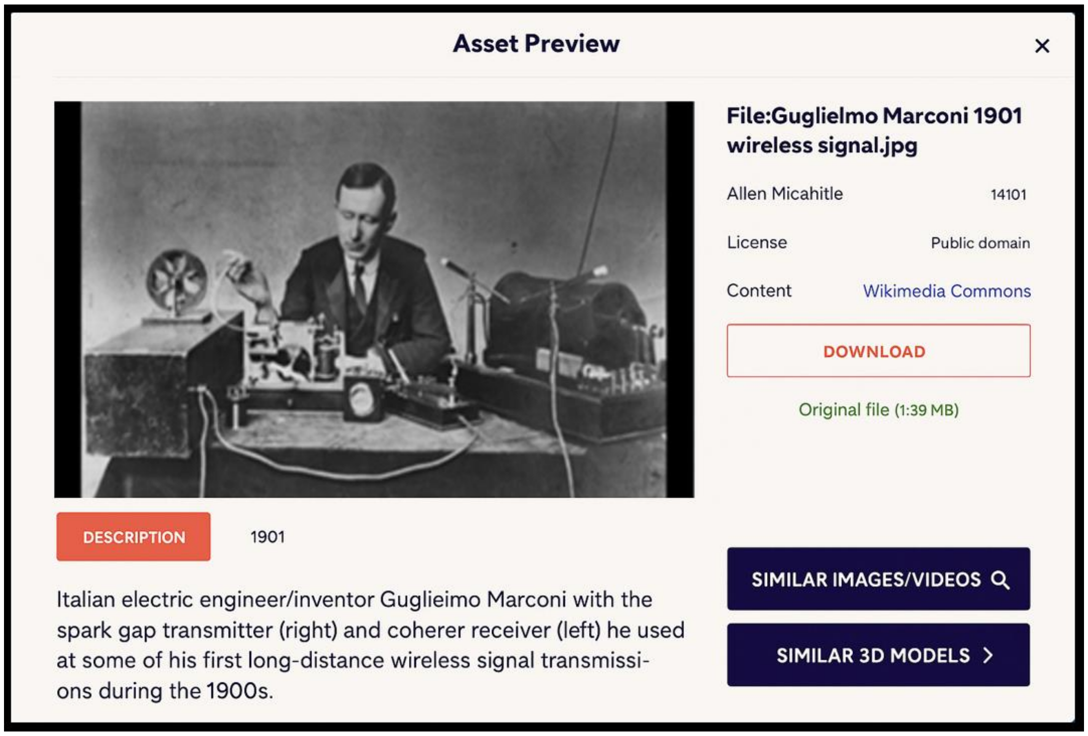


Figure 114: Orchestrator: Search NMR assets.

Selecting an asset opens a detailed preview pop-up. While similar to the asset preview examples shown before, it includes additional NMR-specific capabilities such as:

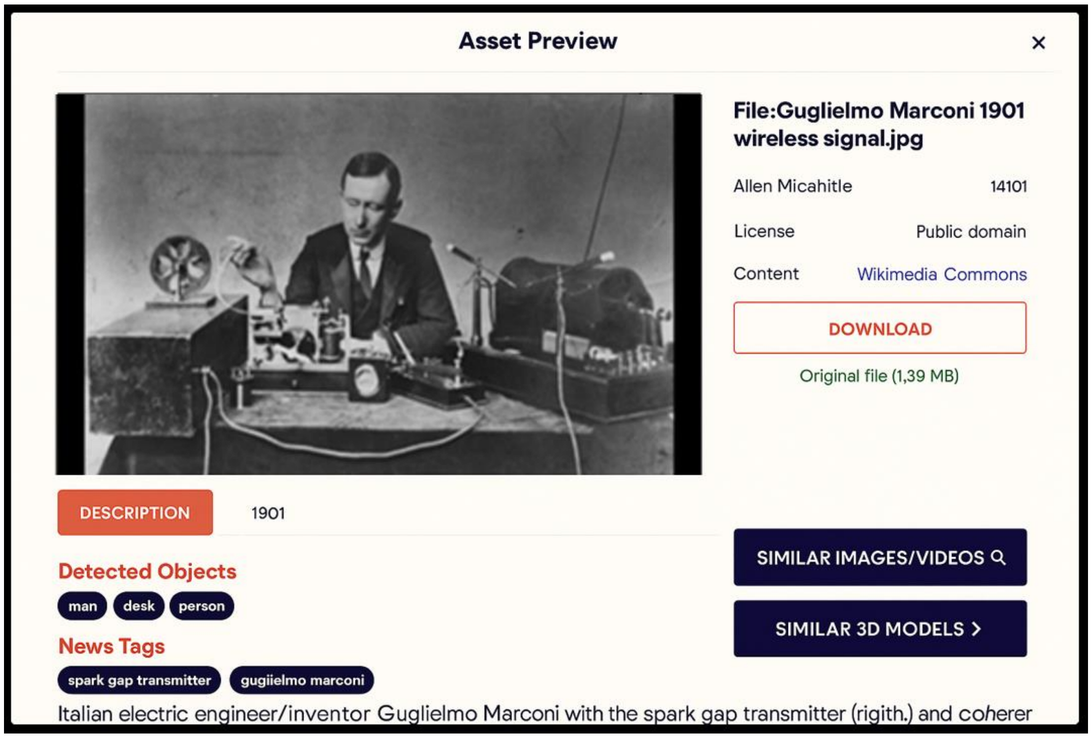
- AI-generated tags for detected objects and extracted keywords, which are displayed as clickable filters to explore related content.
- Visual similarity search, enabling discovery of visually similar images, video segments, and 3D models. This includes cross-modal image-to-model retrieval for more effective asset discovery.
- An option to publish uploaded assets to the XReco Marketplace, described further in D3.2.

Figure 115 and Figure 116 illustrate an example of an image ingested into the NMR. Below the image, two tabs are available: Description and Tags. Figure 115 shows the Description tab with asset details, while Figure 116 displays the Tags tab featuring AI-generated tags. In this example, “Detected Objects” such as “tie” and “person” are identified on the image, alongside “News Tags” extracted from the description, such as “Guglielmo Marconi”. Clicking any tag allows users to find other assets containing the same tag (e.g., selecting “person” retrieves assets where persons were detected).



Italian electric engineer/inventor Guglieimo Marconi with the spark gap transmitter (right) and coherer receiver (left) he used at some of his first long-distance wireless signal transmissi-ons during the 1900s.

Figure 115: Orchestrator: Description tab in asset preview.



Italian electric engineer/inventor Guglielmo Marconi with the spark gap transmitter (rigith.) and coherer

Figure 116: Orchestrator: Tags tab in asset preview.

Additionally, both figures show the “Similar Images/Videos” and the “Similar 3D models” buttons at the bottom right. The first initiates a search for visually similar images and videos, while the second finds 3D models visually related to the selected asset. In Figure 117, we can see an example of a similarity search.

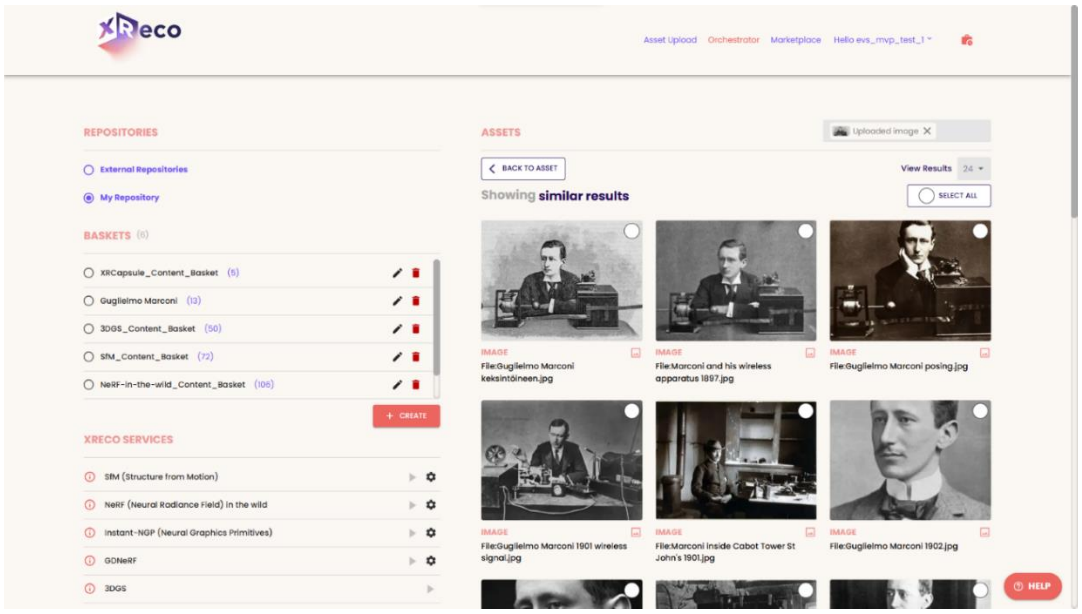
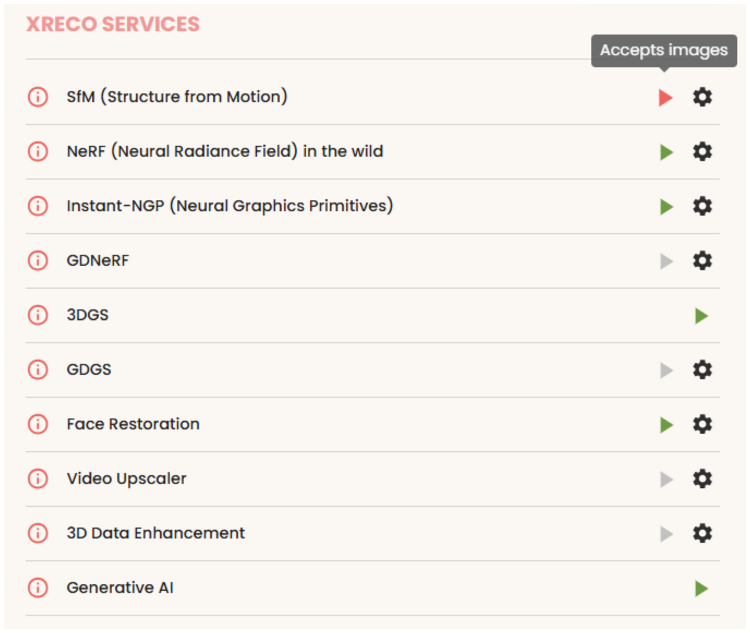

XReco
Asset Upload
Orchestrator
Marketplace
Hello evs_mvp_test_1
REPOSITORIES
External Repositories
My Repository
BASKETS (6)
XRCapsule_Content_Basket (5)
Guglielmo Marconi (12)
3DGS_Content_Basket (50)
SfM_Content_Basket (72)
NeRF-in-the-wild_Content_Basket (106)
+ CREATE
XRECO SERVICES
SfM (Structure from Motion)
NeRF (Neural Radiance Field) in the wild
Instant-NGP (Neural Graphics Primitives)
GDNeRF
3DGS
ASSETS
Uploaded image
BACK TO ASSET
View Results 24
Showing similar results
SELECT ALL
IMAGE
File:Guglielmo Marconi keksintöineen.jpg
IMAGE
File:Marconi and his wireless apparatus 1897.jpg
IMAGE
File:Guglielmo Marconi posing.jpg
IMAGE
File:Guglielmo Marconi 1901 wireless signal.jpg
IMAGE
File:Marconi inside Cabot Tower St John's 1901.jpg
IMAGE
File:Guglielmo Marconi 1902.jpg
HELP

Figure 117: Orchestrator: Results of a similarity search.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

9.3.5 XReco Services



XRECO SERVICES

Accepts images

- SfM (Structure from Motion)
- NeRF (Neural Radiance Field) in the wild
- Instant-NGP (Neural Graphics Primitives)
- GDNeRF
- 3DGS
- GDGS
- Face Restoration
- Video Upscaler
- 3D Data Enhancement
- Generative AI

Figure 118: Orchestrator: XReco services.

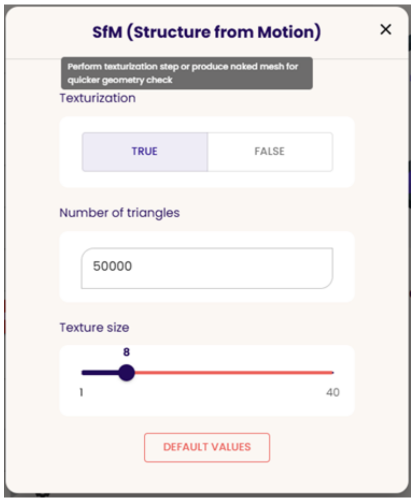
The XReco Services section (Figure 118) provides users with access to the available XReco services within the Orchestrator Dashboard. Each service is represented by a tile that includes:

- An "Info" icon to view a brief description of the service.
- A "Settings" icon to configure service parameters.
- A "Play" icon to launch the service.

Hovering over the "Play" icon reveals the required input types for that service (see Figure 118, top). For example, the Structure from Motion (SfM) service accepts only image inputs. As such, the currently selected content basket must contain at least one image to run the service. When the input criteria are met, the "Play" icon turns green; otherwise, it remains greyed out. If users attempt to launch a service without meeting the input requirements, an error message will appear.

Advanced users can customize processing parameters via the service configuration pop-up (Figure 119), or they can proceed with default settings.

Once a service is launched, users can track its progress in the Service Jobs tab (Figure 120). This interface displays both ongoing and completed jobs, along with relevant details such as the service name and associated content basket.



SfM (Structure from Motion)

Perform texturization step or produce naked mesh for quicker geometry check

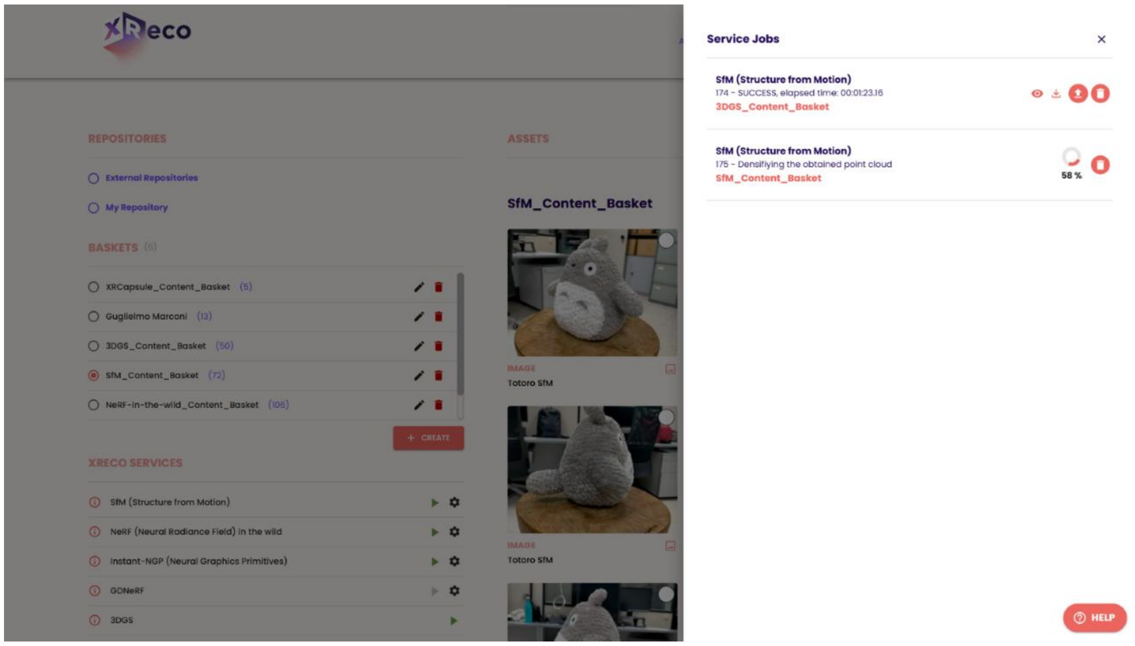
Texturization: [✓ TRUE] FALSE

Number of triangles: 50000

Texture size: 8 (1 – 40)

Figure 119: Orchestrator: Service configuration pop-up.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.



Service Jobs

SfM (Structure from Motion)
174 - SUCCESS, elapsed time: 00:01:23.16
3DGS_Content_Basket

SfM (Structure from Motion)
175 - Densifying the obtained point cloud
SfM_Content_Basket
58 %

Figure 120: Orchestrator: Service jobs tab.

For ongoing jobs, users can view the progress percentage, status messages, or cancel the job. For completed jobs, the following options are available:

- Preview the result (typically a 3D model).
- Download the output to their local device.
- Upload the result to the NMR for further use or sharing.
- Delete the result if no longer needed.

Services integrated on the Orchestrator include:

- **SfM (Structure from Motion)** – Developed by UPM. Reconstructs textured 3D meshes from multiple images by estimating camera poses and triangulating a point cloud. It accepts one or more images as input and outputs a 3D model in GLTF or OBJ format.
- **NeRF (Neural Radiance Field) in the wild** – Developed by CERTH. Learns a Neural Radiance Field from unstructured images taken under varying lighting or occlusions. It accepts a set of images as input and outputs trained NeRF model weights.
- **Instant-NGP (Neural Graphics Primitives)** – Developed by CERTH. Rapidly trains and renders NeRFs using NVIDIA's Instant-NGP for efficient 3D reconstruction. It accepts one or more images or a video as input and outputs a PLY model.
- **GDNeRF** – Developed by i2CAT. Performs real-time NeRF view synthesis from sparse inputs using a volumetric renderer. It accepts a ZIP archive of images as input and outputs RGB and depth videos, along with trained model weights.
- **3DGS** – Developed by i2CAT. Uses 3D Gaussian Splatting to generate dense neural renderings from input media. It accepts images, a video, or a ZIP archive as input and outputs a PLY model.

- **GDGS** – Developed by i2CAT. Synthesises novel views from sparse images in real-time using 3D Gaussian Splatting. It accepts a ZIP archive of images as input and outputs RGB and depth videos, along with trained model weights.
- **Face Restoration** – Developed by RAI. Enhances the resolution and texture quality of facial images for improved reconstruction. It accepts a set of images as input and outputs the enhanced images.
- **Video Upscaler** – Developed by RAI. Upscales videos by 2x or 4x and returns high-resolution video or frame sets. It accepts a video as input and outputs an upscaled video or upscaled image frames.
- **3D Data Enhancement** – Developed by i2CAT. Improves the resolution and geometry of existing 3D models. It accepts a 3D model as input and outputs an enhanced 3D model.
- **Generative AI** – Developed by Capgemini. Generates realistic 3D models from natural language prompts using Hunyuan3D-1. It accepts a text prompt as input and outputs a 3D model in ZIP format.

All service results can be downloaded or uploaded to the Neural Media Repository (NMR) for future reuse and sharing within the platform.

9.3.6 XReco Authoring Tools

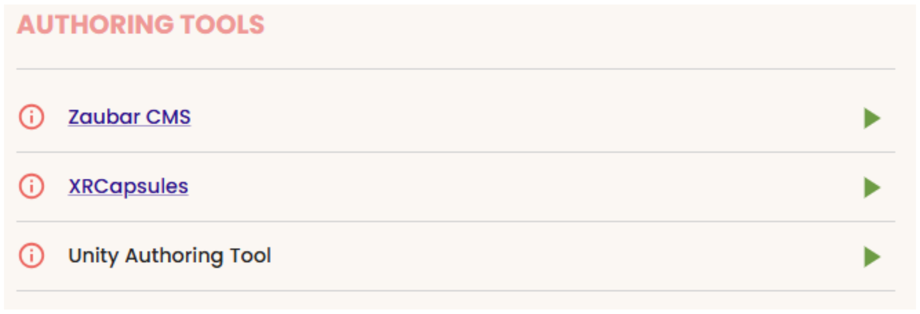


Figure 121: Orchestrator: Authoring tools section.

The Orchestrator Dashboard includes direct integration with the XReco platform's authoring tools, such as ZAUBAR CMS, XRCapsules, and the Unity Authoring Tool. These tools can be accessed through the Authoring Tools panel (Figure 121) within the dashboard.

For XRCapsules and ZAUBAR CMS, which are web-based applications, the Orchestrator provides streamlined interaction:

- Clicking the “Play” button for XRCapsules opens the web app with the current content basket automatically loaded - no login required.
- For ZAUBAR CMS, the selected basket's assets are uploaded directly to the tool. Users can then access the CMS using their XReco credentials.

In the case of the Unity Authoring Tool, which is a desktop application, clicking the “Play” button opens a pop-up with installation instructions.

These integrations support a smooth handoff between asset preparation in the Orchestrator and immersive content creation in the selected authoring environment.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

9.4 CMS-based authoring

The ZAUBAR CMS-based authoring tool has undergone significant evolution between D4.1 and D4.2, reflecting the maturation from conceptual framework to production-ready implementation. While the previous version focused on high-level capabilities and applications, the current iteration provides technical specifications and workflow documentation.[163]

D4.1 emphasised the innovative integration of AR, AI, and CMS capabilities with features like AI mural maker and generative pipeline concepts. The current update transforms these concepts into technical implementations, detailing the microservices architecture, Unity3D integration, and specific authoring workflows. The Romania 1989 revolution example has evolved from a conceptual application to a technical case study demonstrating precise GPS coordination, scene management, and object transformation capabilities.

ZAUBAR's CMS-based authoring tool is a browser-accessible, Unity-integrated authoring platform purpose-built for location-based AR experiences. Technically, it functions as part of a microservices architecture designed for extended reality (XR) content creation, providing a seamless authoring pipeline from multimedia ingestion to immersive scene delivery. Below is a more detailed technical description contextualised with an outdoor historical AR tour on Victory Square (Piața Victoriei) in Timisoara, Romania.

9.4.1 Architecture and platform integration

ZAUBAR's CMS authoring tool represents a sophisticated browser-accessible platform specifically designed for location-based AR experiences within the XReco ecosystem. The system operates as a Unity-integrated authoring environment that seamlessly connects with the broader XReco infrastructure through a microservices architecture. It serves as a front-end client for authoring XR experiences, interacting with:

- **Neural Media Repository (NMR)** for content indexing and semantic tagging.
- **3D asset reconstruction services**, e.g. via NeRF or Gaussian Splatting.
- **Multimedia retrieval systems** using CLIP-encoded multimodal embeddings.
- **REST APIs and Dockerised microservices**, ensuring portable, scalable deployment.

9.4.2 Location-aware AR content creation

The core strength of the ZAUBAR platform lies in its location-awareness capabilities. Content creators can precisely anchor AR experiences to specific geographic coordinates using GPS-based positioning combined with Visual Positioning System (VPS) technologies. Each media asset can be configured with multiple trigger types:

- Spatial triggers (GPS-, VPS-based).
- Scene transitions.
- Audio and text overlays.
- 3D reconstructions and animations from scene photogrammetry or neural rendering pipelines.

For instance, an AR tour in Victory Square, Timisoara, can include:

[163] For an animated overview of all features, visit our tutorial https://www.youtube.com/watch?v=ez6uQHKVxxs

- A neural rendering reenactment of the 1989 revolution using NeRF.
- Live multimedia overlays, such as historical images, video testimonials, or AI-narrated monologues rendered in Mixed Reality.
- Various interactive elements that can be triggered and experienced based on distance.

9.4.3 Content ingestion & management

The CMS authoring system provides tools for multimodal content ingestion and management. The platform supports diverse media formats including text, audio, video, and 3D assets, organising them into user-defined content baskets that facilitate project management and workflow organization.

A distinctive feature of the platform is its mobile-first workflow integration. Content creators can utilise a companion smartphone and multimedia content on-site before synchronising the materials back into the CMS for further refinement and integration.

9.4.4 Scene creation and spatial management

Scene creation within the ZAUBAR platform follows a structured approach that balances creative flexibility with technical precision. Each scene represents a discrete spatial experience tied to specific GPS coordinates or visual triggers, enabling creators to design location-specific content that activates when users reach designated areas.

The scene initialisation process allows creators to assign human-readable names, detailed descriptions, and unique identifiers through the CMS interface or Unity-based editor. Metadata fields support target audience specification and language localisation, ensuring content accessibility across diverse user groups. The geospatial anchoring system utilises GPS coordinates visualised through an integrated map interface, supporting both point-based placement and area-based geofencing for more flexible activation zones.

9.4.5 Connecting scenes in a tour

Connected experiences can be created through the tool's tour orchestration capabilities. Scenes can be linked into coherent narrative sequences or thematic tours using tour management tools. Each tour consists of an ordered list of scenes that can pass state data between locations, enabling persistent user progress tracking, content unlocking mechanisms, and choice-dependent narrative branching.

Navigation tools within the CMS include Tour Mode simulation capabilities, allowing creators to preview the complete user journey before deployment. End users experience seamless navigation through integrated mini-maps and route guidance that clearly indicates completed scenes and upcoming destinations. This approach enables complex multi-location experiences such as historical tours that progress from Victory Square through Timisoara's Cathedral to museum locations, each building upon previous narrative elements.

9.4.6 Object placement and transformation

Scene composition involves object management capabilities supporting diverse media types. The platform accommodates 3D models in standard formats, video panels, images and spatial audio. A complete list of the supported formats per media type is presented in Table 22.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Table 22: Supported formats and features within ZAUBAR's CMS-based authoring tool.

Object type	Supported formats/features
3D Models	glTF, OBJ, FBX (via Unity3D)
Video Panels	MP4 overlays, animated textures
Images	PNG/JPEG billboards, UI cards
Audio	Spatial sound, ambient layers
Text	Labels, annotations, subtitles
Holograms	Volumetric Video, NeRF/FVV

Object transformation tools enable precise positioning, rotation, and scaling through Unity3D's scene graph or intuitive CMS interface sliders. Dynamic transformation capabilities support real-time adjustments based on user interaction or temporal triggers. The platform includes intelligent snap-to-surface and ground-plane detection systems that ensure realistic object placement within the AR environment.

9.4.7 Material and visual adjustments

Each object supports material customisation and shader manipulation for visual fidelity and thematic style. Material editing options include RGBA value specification, gradient application, and texture mapping capabilities for diffuse, normal, and specular map integration. Transparency and opacity controls enable fade effects and layered visual compositions, while emissive properties support highlighting and atmospheric lighting effects.

The shader support system accommodates Unity-compatible rendering pipelines including Standard PBR, Unlit, and AR-optimised mobile shared. Custom shader importation and preset selection through the CMS interface enable advanced visual effects and thematic consistency across complex AR experiences.

9.4.8 CMS and Unity integration workflow

All scene elements are accessible from the CMS asset browser, where users can:

- Drag assets into the scene,
- Preview in real-time,
- See semantic-tags, related content, and reuse frequency.

Unity3D's play mode or WebGL viewer provide real-time preview. The experience can be emulated via GPS spoofing or virtual simulation for testing purposes.

9.4.9 Implementation example: Victory Square Historical Experience

The practical application of these capabilities can be demonstrated through a historical AR experience centred on Victory Square in Timisoara. The implementation process begins with scene creation using GPS coordinates (45.7537° N, 21.2257° E) and descriptive metadata identifying the location's significance during the 1989 Romanian Revolution.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

Content placement involves positioning historically accurate 3D models, e.g. period-appropriate barricades, using ground anchor technology for realistic integration with the physical environment. Audio elements including protest recordings are configured with precise spatial ranges, creating immersive soundscapes that activate 10-meter proximity zones. Video panels displaying archival footage provide historical context while maintaining visual integration with contemporary environment.

Scene transitions enable seamless progression to related locations, such as the Cathedral, creating a cohesive narrative experience that guides users through historically significant sites. Material adjustments to 3D objects, including weathering effects through roughness and transparency modifications, enhance visual authenticity and emotional impact. The complete experience integrates into the broader "Voices of Freedom – Timisoara AR Walk" tour, demonstrating the platform's capability to create meaningful, educational, and emotionally resonant AR experiences that connect users with historical events and locations.

10 Transformation services related KPIs

This section provides an overview of how the developed XReco transformation services address both the functional and non-functional requirements set forth in the project and demonstrates progress towards the project's strategic objectives. The focus is on mapping the capabilities of the XReco platform – including content search, reconstruction, enhancement, and authoring tools – to the specific requirements and KPIs defined for enabling effortless XR and immersive media creation.

The context of this section is rooted in "Objective 3: To develop and integrate novel technologies for enabling effortless XR and immersive media creation". This objective is supported by a set of KPIs that measure the maturity, user acceptance, and usability of the delivered solutions. While KPIs that have to do with user acceptance are part of WP5, the relevant KPIs for WP4 are the following:

- **KPI3.1**: At least **three (3)** XR services are above TRL6.
  The KPI is considered to be achieved, since many of the services have been evaluated in real working conditions for the realisation of XReco's demonstrators. More specifically, SfM, FVV, holoportation, as well as NeRF, and face reconstruction technologies, have been applied in professional contexts for demonstrating the broadcasting and the tourism/automotive scenarios (D5.2).

To comprehensively evaluate both functional and non-functional requirements, we present Table 23, which provides a curated list of key XReco requirements. Each requirement is accompanied by a clear indication of its fulfilment status, as well as a reference to the relevant section or evidence within this deliverable for validation. This structured approach ensures transparent and traceable assessment of the platform's capabilities. Additionally, we provide a mapping with the technical validation experiments defined in D2.2. More specifically, the experiments that are relevant to the technologies presented in this deliverable are 3D Reconstruction EXP-4.1, Neural Rendering EXP-4.4, Freeviewpoint Video (FVV) EXP-4.2.

Table 23: Evaluation of functional and non-functional requirements, and mapping to experiments defined in D2.2. "-" indicates that the requirement does not map to an experiment.

Req. Code	Requirement	Achieved	Evaluation	EXP
FR.118.1	The Volumetric Video Services MUST provide Free Viewpoint video services live and streaming purposes	YES	By-design (Section 6.3)	EXP4.2
NF.119.2	The Volumetric Video Services SHOULD provide Free Viewpoint video services with a motion-to-photon latency <= 300 ms	YES	Section 6.3.5	EXP4.2
NF.120.2	The Volumetric Video Services SHOULD provide Free Viewpoint video services with a capturing frame rate >= 25fps	YES	Section 6.3.5	EXP4.2
NF.121.2	The Volumetric Video Services MUST provide Free Viewpoint video services with a motion-to-photon latency <= 250 ms	YES	Section 6.3.5	EXP4.2
NF.122.2	The Volumetric Video Services MUST provide Free Viewpoint video services with a capturing frame rate >= 30fps	YES	Section 6.3.5	EXP4.2
FR.131.1	The FVV View Renderer SHOULD be able to work on a cloud environment as a virtualized service	YES	Section 6.3.5	EXP4.2
FR.132.1	The FVV Live system SHOULD be able to support several simultaneous View Renderers.	YES	Section 6.3.3.1	EXP4.2
NF.160.1	The Volumetric Video Services MUST provide Free Viewpoint video services with a capturing resolution >= 1080p	YES	Section 6.3.5	EXP4.2
NF.161.1	The Volumetric Video Services MUST provide Free Viewpoint video services with a minimum camera range >= 90 degrees	YES	Section 6.3.5	EXP4.2
NF.162.1	The Volumetric Video Services MUST provide Free Viewpoint video services with a maximum Transmission BW per stereo camera <= 100 Mbps	YES	Section 6.3.5	EXP4.2
NF.163.1	The Volumetric Video Services MUST provide Free Viewpoint video services with a rendering frame rate >= 30fps	YES	Section 6.3.5	EXP4.2
NF.164.1	The Volumetric Video Services MUST provide Free Viewpoint video services with a rendering resolution >= 1080p	YES	Section 6.3.5	EXP4.2
NF.165.1	The Volumetric Video Services MUST provide Free Viewpoint video services with Virtual Camera control with 6 degrees of freedom	YES	Section 6.3.5	EXP4.2
NF.148.1	The XReco Neural Rendering Services MUST be able to render objects from multi-view RGB images of the	YES	Table 3	EXP4.4

Req. Code	Requirement	Achieved	Evaluation	EXP
	same object with a minimum rendering quality PSNR >= 24dB			
NF.149.1	The XReco Neural Rendering Services MUST be able to render objects from multi-view RGB images of the same object with a minimum rendering quality SSM >= 0.75	YES	Table 3	EXP4.4
NF.150.1	The XReco Neural Rendering Services MUST be able to render objects from multi-view RGB images of the same object with a minimum rendering quality LPIPS <= 0.25	YES	Table 3	EXP4.4
NF.156.1	The XReco services MUST provide 3D reconstruction utilizing neural implicit functions with a minimum rendering quality with a PSNR >= 24dB	YES	Table 7	EXP4.1
NF.157.1	The XReco services MUST provide 3D reconstruction utilising neural implicit functions with a minimum Rendering Quality with a SSIM >= 0.75	YES	Table 7	EXP4.1
NF.158.1	The XReco services MUST provide 3D reconstruction utilising neural implicit functions with a minimum Rendering Quality with a LPIPS <= 0.25	YES	Table 7	EXP4.1
NF.159.1	The XReco services MUST provide 3D reconstruction utilising neural implicit functions with a minimum Geometry Quality with a Chamfer Distance <= 4.5	YES	Table 7	EXP4.1
NF.154.2	The XReco services MUST provide 3D building reconstruction with a minimum Rendering Quality (through objective measurements) with a PSNR >= 24	YES	Table 3, Table 7	EXP4.1
NF.235.2	The XReco services MUST provide 3D building reconstruction with a minimum Rendering Quality (through objective measurements) with a SSIM >= 0.75	YES	Table 3, Table 7	EXP4.1
NF.236.2	The XReco services MUST provide 3D building reconstruction with a minimum Rendering Quality (through objective measurements) with a L^2 distance between 3D objects <= 0.01, resizing them so their bounding box diagonal is 1.	YES	Table 3, Table 7	EXP4.1
NF.237.2	The XReco services MUST provide 3D building reconstruction with a minimum Rendering Quality (through subjective measurements) with a MOS >= 3	YES	Figures 28, 29, 30, 31, 32	EXP4.1

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

11 Conclusion

The XReco platform's mature service ecosystem addresses distinct use cases across the content creation spectrum. Content search and filtering services excel in multilingual news environments through NewsTagger's fine-tune approach and location-based discovery via the Mixed Reality interface, enabling intuitive field reporting scenarios where traditional search interfaces can prove impractical.

3D reconstruction service selection depends primarily on object characteristics and deployment constraints. SfM proves optimal for static architectural subjects and cultural heritage documentation requiring high geometric accuracy, while NeRF-based implementations address complex lighting scenarios where photogrammetry usually struggles. 3DGS provides the ideal solution for real-time rendering applications where immediate visual feedback takes precedence over absolute precision.

Human-centred reconstruction services target distinct application domains based on capture complexity requirements. GDNeRF technology is efficient in sparce camera configurations, while FVV enables immersive communication and telepresence applications requiring 6DoF navigation around human subjects.

The authoring tool selection follows a tiered approach addressing varying technical expertise levels. Unity-based tools offer maximum flexibility for more experienced developers requiring custom functionality, while XR-Capsules occupies the strategic middle ground between complexity and accessibility through template-based authoring. CMS-based authoring addresses location-specific content creation, proving most effective for tourism applications and geospatial storytelling scenarios.

Public platform integration leverages the containerised microservices architecture to enable scalable deployment across diverse infrastructure environments. Core reconstruction services are available through public cloud deployment with appropriate resource management, while authoring tools maintain hybrid models balancing accessibility with performance requirements for intensive development workflows.

Ease of use considerations address varying technical expertise through progressive complexity models, with entry-level users benefiting from template-driven interfaces and automated processing workflows while advanced users retain detailed parameter control. The 10x reduction in memory I/O latency particularly can benefit resource intensive applications, while the modular architecture enables gradual platform adoption without requiring comprehensive workflow transformation.

The successful containerisation and validation through real-world deployment scenarios demonstrates readiness for broader adoption while providing a foundation for continuous improvement based on production use-cases.

The advancements documented in this deliverable position XReco as a leading solution for organisations seeking to modernise and streamline their XR content creation workflows. The modular, microservices-based architecture enables seamless integration with existing media production pipelines, reducing adoption barriers for broadcasters, cultural institutions, and creative agencies. The platform's ability to automate content discovery, enhance legacy assets, and deliver immersive, location-based experiences address critical industry needs for efficiency, scalability, and innovation.

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

With the growing demand for immersive media in sectors such as broadcasting, tourism, education, or heritage preservation, XReco offers a compelling value proposition. The platform's support for real-time authoring, and its compatibility with industry-standard tools (such as Unity) further enhances its appeal to both technical and non-technical users.

Looking ahead, XReco is well positioned to drive the next wave of innovation in XR content creation and distribution. The project's commitment to openness and interoperability (open-source technologies with permissive licenses, REST APIs for service communication) enables ongoing ecosystem expansion, including third-party service integration and community-driven enhancements. Future developments may focus on:

- Expanding use cases: Broadening the platform's applicability to novel domains such as live events or interactive storytelling.
- Leveraging immersive technologies: Incorporating advances in generative AI, multimodal retrieval, and real-time rendering to maintain technological leadership.
- Strengthening community engagement: Fostering partnerships with industry stakeholders, academic institutions, and open-source communities to accelerate innovation and adoption.

By continuing to refine and expand its capabilities, XReco is poised to become a cornerstone of the immersive media ecosystem, supporting the creation of rich, engaging, and accessible XR experiences for diverse audiences worldwide.

Annex I: Extended information for components context

11.1 Details on sample structure for RSS items in News Content Tagging

Prompts are structured using the Llama-3.1-8B-Instruct template applied to the following structured messages (for each RSS item):

```
[
        {
                "role": "system",
                "content": "You are a multilanguage AI assistant aimed to suggest tags in news
                articles."
        },
        {
                "role": "user",
                "content": "Analyze the following news article and assign a list of representative
                tags to the content, returning always at least 3.

                Title: <title of the article>

                Subtitle: <subtitle of the article (if present)>

                Text: <text of the article>

                The response has to be given in the following format "[<tag1>, <tag2>,.. ]", where
                the <tag>s are the tags identified and written in the same language of the article.
                Do not add any further text."
        },
        {
                "role": "assistant",
                "content": <list of the tags assigned by journalists as [<tag1>,<tag2>,...]>
        }
]
```

The LoRA adapters obtained with the fine-tuning process described above are available on Hugging Face[164].

11.2 GDNeRF API Implementation

The GDNeRF API provides an interface for training and running inference on GDNeRF-based models. It is built using **FastAPI**, with asynchronous job handling through **Celery** and **Redis**. It is readily available at http://195.251.117.31:8000/docs.

[164] https://huggingface.co/raicrits/Llama_NewsTagger_XRECO (last accessed May 29th, 2025)

Key Endpoints

POST /upload_zip/{dataset_name}
Uploads a dataset to the system. The dataset must be structured with the necessary manifest.json, calibration file, RGB images, and depth maps.

POST /train
Initiates fine-tuning on an uploaded dataset. Requires an experiment name, dataset reference, quality mode, and downsize factor.

GET /check_status/{task_id}
Returns the status of an ongoing task (e.g., upload, training) using the task ID.

GET /download_experiment/{experiment_name}
Downloads the rendered RGB and depth videos as well as the trained model weights for a given experiment.

POST /infer_views_from_zip
Performs inference using a zipped dataset with videos and camera parameters to synthesize novel views from few inputs.

GET /download_preview
Fetches the last generated video result from a completed inference task.

Typical Workflow

1. **Upload a dataset** using /upload_zip/{dataset_name}. The dataset should be zipped and follow the GDNeRF structure.
2. **Train the model** with /train, providing experiment settings.
3. **Check progress** via /check_status/{task_id} to monitor uploads or training.
4. **Download results** once training completes using /download_experiment/{experiment_name}.
5. **Infer new views** from uploaded video-based data using /infer_views_from_zip.
6. **Retrieve latest output** with /download_preview.

11.3 XRcapsule JSON Specification

In this part of Annex I we provide the structure, required fields, data types, and expected behaviour for XRcapsule JSON files. XRcapsule is a modular JSON format designed to represent interactive XR scenes that can be authored with simple templates and executed on mobile and desktop devices using the XRcapsule player.

11.3.1 Root structure

```
{
  "$schema": "string",
  "Metadata": MetadataObject,
```

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

```
 "Capsules": [ CapsuleObject, ... ]
}
```

11.3.2 MetadataObject

```
{
 "FileVersion": "string",
 "XRCapsuleEditorVersion": "string"
}
```

Field	Type	Description
FileVersion	string	Version of the XRcapsule file format.
XRCapsuleEditorVersion	string	Version of the editor that created the file.

11.3.3 CapsuleObject

```
{
 "Name": "string",
 "TargetDevices": "string",
 "Workspace": WorkspaceObject,
 "Assets": [ AssetObject, ... ],
 "Scene": SceneObject
}
```

Field	Type	Description
Name	string	Human-readable name of the capsule.
TargetDevices	string	Target platform (e.g., “Smartphone AR”).
Workspace	WorkspaceObject	Defines unit scale and environment.
Assets	list of AssetObject	Media elements used in the scene.
Scene	SceneObject	Scene graph including objects, cameras, and triggers.

11.3.4 WorkspaceObject

```
{
 "Volume": [ number, number, number ],
 "Unit": "string",
 "Environment": {
  "Name": "string",
  "Type": "string",
  "SourceURL": "string | null"
```

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

```
  }
}
```

Field	Type	Description
Volume	[float]	Dimensions of the workspace.
Unit	string	Typically “meters”.
Environment	object	HDR environment config.

11.3.5 AssetObject

```
{
  "AssetUUID": "string",
  "Name": "string",
  "Type": "string",
  "SourceURL": "string",
  "Metadata": {
    "XReco": {
      "Notes": "string",
      "BoundingBox": [float, float, float],
      "Polygons": int,
      "Resolution": [int, int] | null
    }
  }
}
```

11.3.6 Asset Types

- "3D.Model"
- "Image"
- "Video"
- "360 Degree Video"
- "Stream"
- "FVV"
- "NeRF"
- "Cloudpoint"

11.3.7 SceneObject

```
{
  "Cameras": [ CameraObject, ... ],
  "Objects": [ SceneObjectItem, ... ],
  "Triggers": [ TriggerObject, ... ]
}
```

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.
The content of this document is © the author(s). For further information, visit xreco.eu.

11.3.8 CameraObject

```
{
 "Name": "string",
 "SceneUUID": "string",
 "Enabled": true,
 "Transform": TransformObject,
 "FollowTarget": {
  "Enabled": true,
  "TargetPosition": [string, string, string]
 }
}
```

11.3.9 SceneObjectItem

```
{
 "Name": "string",
 "AssetUUID": "string",
 "SceneUUID": "string",
 "Visible": true,
 "Enabled": true,
 "Transform": TransformObject
}
```

11.3.10 TransformObject

```
{
 "Position": [string, string, string],
 "Rotation": [string, string, string],
 "Scale": [string, string, string]
}
```

All values are stringified floats for cross-platform compatibility.

11.3.11 TriggerObject

```
{
 "Type": "string",
 "TriggerParameters": [ "string", ... ],
 "SceneTargetUUID": "string",
 "OnActivate": "string"
}
```

11.3.12 Trigger Types

- "Touch - Visibility"
- "Touch - Transform"
- "Time - Visibility"
- "Time - Transform"

11.3.13 Best Practices

- Use consistent units and volume scale across all capsules.
- Include at least one visible object and one environment HDR to avoid empty scenes.
- UUIDs must be unique within each capsule.
- Maintain readable and descriptive Name fields for clarity.

11.3.14 Example Minimal XRcapsule

```
{
 "$schema": "https://xrcapsule.visyon.tech/schema#",
 "Metadata": {
  "FileVersion": "0.0.2",
  "XRCapsuleEditorVersion": "1.0.0"
 },
 "Capsules": [
  {
   "Name": "Example Capsule",
   "TargetDevices": "Smartphone AR",
   "Workspace": {
    "Volume": [5, 5, 2.5],
    "Unit": "meters",
    "Environment": {
     "Name": "HDRI_Env",
     "Type": "HDRi",
     "SourceURL": "https://example.com/env.hdr"
    }
   },
   "Assets": [],
   "Scene": {
    "Cameras": [],
    "Objects": [],
    "Triggers": []
   }
  }
 ]
}
```

XReco is an HorizonEurope Innovation Project co-financed by the EC under Grant Agreement ID: 101070250.

Annex II: RGB-D Cameras information

11.4 Capture Devices

As the goal of the capture setup is to be consumer-grade, the cost and accessibility of the capture devices must be carefully considered, in addition to meeting the essential technical requirements—most notably, the ability to estimate depth in real time. The devices that meet these criteria generally fall into three main categories, each defined by its underlying depth estimation technology. A summary of these technologies is provided in Table XX.

Stereo cameras operate based on the same principles as human binocular vision. They use two sensors placed at a fixed distance from each other (known as the *baseline*) to estimate depth by analyzing disparities between corresponding points in the two images. These cameras tend to be more affordable, as they rely on widely available imaging sensors (color, monochrome, or infrared). However, the computational cost of post-processing the image data is relatively high. The accuracy of stereo cameras is strongly influenced by both the baseline and the distance to the object. Additionally, they are generally poor at detecting flat surfaces. On the positive side, stereo cameras can function in outdoor environments—as long as they do not rely on infrared light, which can lead to overexposure under sunlight. While they are effective for basic distance measurement, the overall reconstruction quality is insufficient for our specific use case.

Time-of-Flight (ToF) cameras represent an active sensing technology. They operate by emitting a pulse of light and measuring the time it takes for the light to reflect off surfaces and return to the sensor, thereby estimating depth. A significant limitation of this technology is its reduced accuracy in outdoor settings due to ambient light interference. Additionally, certain materials—such as deep black or highly reflective surfaces—may fail to reflect the infrared (IR) light properly, resulting in missing depth data. Nonetheless, ToF cameras generally offer superior reconstruction quality compared to stereo cameras. Although the widely used Kinect 4 Azure camera has been discontinued, the Orbbec Femto Bolt camera—based on the same technology—has emerged as its de facto replacement in the sector.

Structured light cameras are another form of active depth-sensing technology. These systems consist of two primary components: a projector that emits a known (often invisible IR) light pattern onto the scene, and a sensor that captures the resulting image. Depth is estimated by analysing distortions in the projected pattern. Like ToF systems, structured light cameras are vulnerable to ambient light interference and thus perform poorly in outdoor environments.

Table 24: Summary of depth estimation algorithms along with exemplary camera technologies.

Depth Cameras	Stereo Vision	ToF	Structured Light
Advantages	low hardware requirements	long detection distance	convenience for miniaturization
	low cost	large tolerance to ambient range	low resource consumption
	high robustness to light disturbance	high frame rate	high resolution

Depth Cameras	Stereo Vision	ToF	Structured Light
Disadvantages	large calculation complexity	high equipment requirements	small tolerance to ambient light
	strong object texture dependence*	high resource consumption	short detection range
	limited measurement range	low edge accuracy	high noise
Representative	Zed2	Kinect4Azure	Kinect v1
	Oak-D	Orbbec Femto Bolt	

The replacement device needs to offer similar quality and price range. After looking at different options (as shown in Table 25) in the market, the best option we found was the Orbbec Femto Bolt cameras which use the same technology as the K4A devices.

Table 25: Table with the main characteristics of different RGB-D cameras studied as replacement for the Kinect 4 Azure sensor.

Camera	Type	Range (m)	Error (mm)	Colour Res	Depth Res	FPS (RGBP / D)	Color FoV (Hº x Vº)	Depth FoV (Hº x Vº)	Shutter Type (RGBP / D)	IMU
Kinect 4 Azure	ToF	0.5 - 3.86	<11 mm + 0.1%	3839 x 2160	640x 576	30 / 30	90° x 59°	75º x 65º	Rolling/Global	yes
Orbbec FemtoBolt	ToF	0.3- 5.4	<11 mm + 0.1%	3840 x 2160	640 x 576	30 / 30	80º x 51º	75º x 65º	Rolling/Global	yes
RealSense D435	AIRS	0.3 - 3	< 2% at 2m	1920 x 1080	1280 x 720	30 / 90	69° × 42°	87° × 58°	Rolling/Global	no
RealSense D435i	AIRS	0.3 - 3	< 2% at 2m	1921 x 1080	1281 x 720	30 / 90	69° × 42°	87° × 58°	Rolling/Global	yes
OAK-D Pro	AIRS	0.7 - 12	~2% at <4m	4056 x 3040	1280 x 800	60 / 120	66° x 54° / 78°	80° x 55° / 89.5°	Rolling/Global	no
Zed2	Neural Stereo	0.3 - 20	--	2208 x 1242	2208 x 1242	100	110° x 70° / 120°	110° x 70° / 120°	E.S. Rolling Shutter	yes

The main issues when doing this migration was getting familiar with their SDK and inner workings. As a bonus, we were able to make use of their hardware to synchronize the cameras and improve the visual quality of the reconstruction. During this change we also implemented the use of higher resolutions getting a final reconstruction with more points and better quality of the results.