D4.1

XR and Media Transformation Services, API and Authoring Tools v1

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	36 months

Deliverable title	XR and Media Transformation Services, API and Authoring Tools v1
Deliverable number	D4.1
Deliverable version	V1.0
Previous version(s)	n/a
Contractual Date of delivery	31.12.2023
Actual Date of delivery	23.12.2023
Nature of deliverable	Report
Dissemination level	Public
Partner Responsible	CERTH
Author(s)	Antonis Karakottas (CERTH), Vangelis Chatzis (CERTH), Werner Bailer (JRS), Francisco Morán (UPM), Daniel Berjón (UPM), Teresa Hernando (UPM), Javier Usón (UPM), Julián Cabrera (UPM), Cláudia Marinho (MOG), J.J. Vegas Olmos (MLNX), Jesús Luque Muriel (Visyon), Marieta Caballero (Visyon), Roberto Iacoviello (RAI), Alberto Ciprian (RAI), Anne-Sophie Panzer (ZAUBAR)
Reviewer(s)	Werner Bailer (JRS), Francisco Morán (UPM), Nico Patz (DW)
EC Project Officer	Andreea Popescu

Abstract	This report provides the first version of XReco's transformation backend technologies that will drive XReco's services as well as authoring tool and APIs development. It presents WP4 developments until M16 from an XR-ready content creation perspective.
Keywords	Neural rendering, Neural Radiance Fields, NeRF, Structure from Motion, SfM, Volumetric Capturing, Neural reconstruction, 3D reconstruction, 4D reconstruction, Unity3D

XRECO is a Horizon Europe 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 mediaverse-project.eu.

Revision History

Version	Date	Modified By	Comments
V0.1	25/09/2023	Antonis Karakottas (CERTH)	First table of contents
V0.2	20/10/2023	Antonis Karakottas (CERTH)	Updated table of contents
V0.3	12/12/2023	CERTH, JRS	First Draft
V0.4	15/12/2023	CERTH, i2CAT, NVIDIA, RAI, UNIBAS, UNITY3D, UPM, Visyon, ZAUBAR	Second Draft
V0.5	17/12/2023	CERTH	Final Second Draft
V0.6	18/12/2023	JRS	Final Second Draft Review
V1.0	23/12/2023	DW	Final Document

XRECO is a Horizon Europe 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 mediaverse-project.eu.

Glossary

ABBREVIATION	MEANING
2DSR	2D data enhancement, image super-resolution
AR	Augmented Reality
ASR	Automatic Speech Recognition
CLIP	Contrastive Language-Image Pre-training
CMS	Content Management System
FP	Feature Point
FVV	Free Viewpoint Video
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HMD	Head-Mounted Display
HR	High resolution
K8s	Kubernetes, an open-source system
LLM	Large language model
LR	Low resolution
ML	Machine Learning
MLP	Multi-layer-perceptions
MR	Mixed Reality
NeRF	Neural Radiance Fields
NMR	Neural Media Repository
NN	Neural Network
RGB-D	RGB (Colour) + Depth
SfM	Structure from Motion
TF-IDF	Term-frequency/inverse document frequency
VR	Virtual Reality
WP	Work Package
XR	eXtended Reality

Table of Contents

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

Index of Figures

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

Index of Tables

1 Executive Summary

This deliverable represents a significant milestone withing the XReco project, aiming to develop and deliver a set of advanced technologies, suitable for creating XR experiences. The primary objectives include harnessing the latest advancements in content filtering, neural rendering, asset aggregation, and volumetric content creation. These technologies will form individual services, which will be integrated into the XReco platform following a micro-service architecture, emphasizing modularity. This deliverable describes the current progress in XR experience authoring, which is realised via Unity3D as well as other tools which are developed entirely within XReco or introduced by XReco partners and extended within the scope of the project. The presented technologies, although applicable to various XR contexts, consider the use cases of the project that involve immersive journalism and tourism (e.g., creating 3D scenes of monuments for XR journalism contexts or XR location-based tourism scenarios).

In a nutshell, this deliverable provides a description regarding the activities within WP4 until M16, encompassing five tasks:

- **T4.1: Content sourcing and filtering**: Technologies for content search, incoming content monitoring and filtering according to a particular topic or production, and XR interfaces for content retrieval.
- **T4.2: Neural rendering services**: Algorithms for reconstructing 3D scenes from 2D image data utilizing Neural Radiance Field (NeRF) approaches.
- **T4.3: 3D asset aggregation and optimisation services**: Technologies for 3D reconstruction from 2D image data employing computer vision as well as machine learning pipelines, along with technologies for content optimisation (colour image and 3D image super-resolution)
- **T4.4: XR volumetric and free-viewpoint video services**: Services for producing human-centred volumetric and free-viewpoint video, utilising RGB-D data.
- **T4.5: APIs and authoring tool development**: The development of authoring tools as well as 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.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

2 Introduction

The focus of this deliverable is WP4, which is responsible for developing and delivering a set of vertical technologies that will enable XR content creation for XR experience realisation according to the use cases and the requirements specified in WP5 and WP2 respectively. The technologies developed under this WP are intended to leverage the latest advancements in context-based content filtering, neural rendering, asset optimisation, volumetric video, and free viewpoint video. This document has a two-fold character. First it presents the backend technologies and tests conducted for content search and XR content creation and optimization. Secondly, it describes the developments concerning the authoring of XR experiences through the usage of authoring tools, considering a variety of digital competences, from professional to less technical users, realised through different authoring applications to be evaluated in different scenarios.

WP4 is divided into five tasks, each with its own set of objectives and responsibilities. Considering the diverse technical tasks in WP4, this deliverable is organised as follows:

In Section 3, backend and frontend technologies for content retrieval are presented. More specifically, components for retrieving and filtering content according to a specific topic are described. These components build on top of the Neural Media Repository (NMR) described in D3.1. Additionally, Mixed Reality (MR) frontend components are presented, allowing content search through XR devices (such as headset displays and smartphones) via facilitating image-based object recognition modules.

Section 4 provides a comprehensive overview of NeRF technologies that have been successfully implemented and tested for their utilisation within XReco. It also presents early qualitative and quantitative results on standard benchmarks. The section is structured to cover a broad range of use cases, including both static and dynamic human-centred scenes. The NeRF technologies discussed span a wide array of applications. These include NeRF algorithms for learning radiance fields of generic object scenes, as well as in-the-wild based scene fitting. Furthermore, the section also delves into dynamic RGB-D based human-centred reconstruction, which utilises NeRF algorithms. Moreover, Section 4 also reflects the current state of research in this field.

As NeRF-based scenes are a relatively novel technology, they are not directly supported by current 3D and rendering engines. Section 5 provides a set of backend technologies for scene reconstruction that output standard triangle-based meshes. Additionally, technologies for optimising (i.e., upsampling) 2D and 3D content are presented. These can be utilised in many contexts, either for straightforward optimisation, or as intermediate technologies for enhancing the inputs of other components.

Section 6 delves into human-centred 3D reconstruction, presenting different technologies able to reconstruct dynamic 3D humans in real-time, utilising RGB-D sensors. More specifically, UPM's Free-Viewpoint Video (FVV) Live module is presented, along with its extensions developed within XReco. The FVV Live system can interpolate between multiple RGB-D streams in real-time and provide the ability to stream 3D human-centred content in studio-based scenarios. Additionally, extensions to i2CAT's holoportation system concerning the compression and streaming of 3D content are presented in this section.

Finally, Section 7 considers the authoring of XR experiences by first describing extensions in the Unity3D editor developed for the XReco platform. Then Visyon's XR-Capsules are presented, which is a minimal authoring solution providing the ability to composite 3D scenes, activate actions according to user-defined triggers, and

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

export Unity3D-based scenes. Furthermore, XReco's Orchestrator module is described, that will enable user-friendly usage of XReco's backend technologies trough web-based interfaces and API-based communication. Finally, ZAUBAR's CMS-based authoring tool and its extensions within XReco are presented.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. 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 Neural Media Repository (NMR) described in D3.1. As these components are part of the authoring workflow, they are addressed in WP4.

These components are backend components that filter incoming content to select items relevant for stories being worked on, and interfaces for media search, including novel search paradigms such as search in mixed reality.

3.2 Content sourcing and filtering

3.2.1 Concept

The content sourcing and filtering component aims to automatically select content items from those ingested into the Neural Media Repository, based on the relevance of those items for stories being worked on. The stories being worked on are represented by the content collected so far 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.

While the topic is related to recommendation, it is purely content focused. A user profile is entirely irrelevant, as journalists may work on a number of topically diverse topics at the same point in time. The problem can be broken down into two key steps: (i) deciding whether the item is topically related to the story being worked on, a problem known as topic detection or threading, and (ii) determining whether the item contains new information or perspectives related to the content already collected. This aspect may be on an information content level, i.e., containing new facts, but may also include supporting information for already known facts, or different visual representations of the same content. Apart from relevance and novelty detection, we propose to consider also approaches for multi-document question answering, which has been a very active research topic recently. The question of what is novel or diverse in a content item with regard to a story represented by already collected items can be posed as a problem of whether questions derived from the new item can be answered from this content set.

Most of the work related to these research topics focuses on text. We see two main ways of how multi-modality can be achieved in these approaches. The first is to stick with the text-based approach, but use transcription and captioning (image to text, video to text) models to derive textual information from audiovisual content. The second is to use joint visual-text embeddings, based on vision-language foundation models, which enables to use the same approaches in the common feature space. It is not yet clear which approach is the best and whether the same approach is best suited for both steps in the workflow.

A recent survey paper on extraction of news narratives[1] lists 14 criteria to be considered: relevance, surface similarity, topic distribution, entities, coherence, coverage, dispersion, diversity and redundancy, output structure, article structure, content references, temporal references, burstiness and frequency and temporal distance. Not all of those are applicable in a journalistic context, where the story is emerging, incoming items are often raw content, and the content set is not available in retrospect (which would e.g., be needed to assess coverage). Thus, we address the criteria of surface similarity in the initial grouping step, while entities, diversity and redundancy are addressed in the novelty detection step.

In the following subsections, we summarise the state of the art related to this problem and describe the status of the topic threading and novelty detection approaches implemented so far (focusing on text) before we outline future plans.

3.2.2 Related work

There is a large body of work on (multimodal) topic detection and clustering, often termed topic detection and tracking (TDT). Many (also quite recent) works rely on rather traditional text analysis approaches such as term frequency/inverse document frequency (TF-IDF) (e.g., Story Forest[2]), Latent Dirichlet Allocation (LDA) (e.g., [3] and [4]) or KL-Divergence of selected topic sentences[5]. Other approaches use dynamic topic modelling and Hidden Markov Models to infer event stages from the stream of news documents[6]. Other works treat the similarity measurement of a new item with regard to the previous ones as an information retrieval and ranking problem[7]. However, this approach assumes that it mines the question from those posted on the web to already published news articles.

Named entities are without question an important feature of news items. Thus, some approaches focus on detecting named entities (using e.g., models such as RoBERTa[8]) and using them for chaining news stories[9]. Named

[1] Keith Norambuena, Brian Felipe, Tanushree Mitra, and Chris North. "A survey on event-based news narrative extraction." *ACM Computing Surveys* 55.14s (2023): 1-39.
[2] Liu, Bang, et al. "Story forest: Extracting events and telling stories from breaking news." ACM Transactions on Knowledge Discovery from Data (TKDD) 14.3 (2020): 1-28.
[3] Xu, Guixian, et al. "Research on topic detection and tracking for online news texts." IEEE access 7 (2019): 58407-58418.
[4] Chengyu Wang, Xiaofeng He, and Aoying Zhou. 2018. Event phase oriented news summarization. World Wide Web 21, 4 (2018), 1069–1092.
[5] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11). ACM, New York, NY, 745–754.
[6] Mele, Ida, Seyed Ali Bahrainian, and Fabio Crestani. "Event mining and timeliness analysis from heterogeneous news streams." Information Processing & Management 56.3 (2019): 969-993.
[7] Nicholls, Tom, and Jonathan Bright. "Understanding news story chains using information retrieval and network clustering techniques." Communication methods and measures 13.1 (2019): 43-59.
[8] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692* (2019).
[9] Gedikli, Fatih, A. Stockem Novo, and Dietmar Jannach. "Semi-automated identification of news story chains: A new dataset and entity-based labeling method." Proceedings of the 9th International Workshop on News Recommendation and Analytics (INRA 2021) co-located with 15th ACM Conference on Recommender Systems (RecSys 2021). 2021.

entity-based approaches have also been proposed for determining novelty in these story chains[10]. A more modern variant of this approach used BERT to detect relations between entity pairs, and then performs novelty classification on semantic triples[11]. However, focusing on named entities may be too limiting in some cases.

More modern approaches use sentence embeddings as a basis for determining story clusters. USTORY[12] is based on RoBERTa, while another approach[13] uses Sentence-BERT[14]. Another group of neural network-based approaches uses graphs as representations and graph NNs for processing (e.g., [15] and [16]). Another language model approach that is interesting in our context is the summarization of news topics based on question answering[17]. In NN-based novelty detection, one type of approaches uses sentence embeddings, and uses attention to determine document-level novelty of sentences[18]. In addition to using a NN for embedding of news, a news recommendation approach[19] uses a recurrent NN to train an evolving model of the state of the story.

Multi-document question answering, in particular in open domains, has received a lot of attention in recent works. For the detection of cross-references in a set of documents, Caciularu et al.[20] use questions generated from key sentences in one of the documents to obtain answers from the other documents. In order to improve the accuracy of multi-document question answering, Lu et al.[21] propose to generate a knowledge graph from the set of documents to be queried and uses the graph to support the language model in answering questions about

[10] Panagiotou, Nikolaos, et al. "A general framework for first story detection utilizing entities and their relations." *IEEE Transactions on Knowledge and Data Engineering* 33.11 (2020): 3482-3493.

[11] Ma, Nianzu, et al. "Semantic Novelty Detection and Characterization in Factual Text Involving Named Entities." arXiv preprint arXiv:2210.17440 (2022).

[12] Yoon, Susik, et al. "Unsupervised Story Discovery from Continuous News Streams via Scalable Thematic Embedding." arXiv preprint arXiv:2304.04099 (2023).

[13] Polimeno, Alessandra, et al. "Improving and Evaluating the Detection of Fragmentation in News Recommendations with the Clustering of News Story Chains." arXiv preprint arXiv:2309.06192 (2023).

[14] Reimers, Nils, and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[15] Yang, Boming, et al. "Going Beyond Local: Global Graph-Enhanced Personalized News Recommendations." Proceedings of the 17th ACM Conference on Recommender Systems. 2023.

[16] Hu, Linmei, et al. "Graph neural news recommendation with long-term and short-term interest modeling." Information Processing & Management 57.2 (2020): 102142.

[17] Wang, Xuezhi, and Cong Yu. "Summarizing news articles using question-and-answer pairs via learning." The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I 18. Springer International Publishing, 2019.

[18] 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.

[19] Zhu, Qiannan, et al. "Dan: Deep attention neural network for news recommendation." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. No. 01. 2019.

[20] 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).

[21] Lu, Xiaolu, et al. "Answering complex questions by joining multi-document evidence with quasi knowledge graphs." Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2019.

the documents. Another retrieval-guided method[22] proposes end-to-end training using pseudo-labels, trained with the expectation-maximisation framework.

3.2.3 Topic threading for text

We aim to leverage the power of large language models (LLMs) for threading topics in news items. We build on the recently proposed USTORY framework. One key motivation for that work is the observation that using all text of a news item for article embedding gives the same weight to discriminative content of the text and to other information, resulting in suboptimal clustering performance. The framework reaches state-of-the-art performance on common benchmark datasets.

The framework uses a pretrained LLM (RoBERTa) in this case to perform an embedding of all sentences of an article. The clustering of articles is performed using a theme identification (using a traditional approach with TF-IDF). For new articles, sentences can then be weighted with regard to a theme and produce a candidate embedding for an article in the context of a theme. In order to account for the development in stories, the contribution of keywords to themes is weighted in a time-decaying manner. The assignment of new articles is based on a confidence metric derived from the relative similarities to the top ranked and alternative themes. Articles resulting in low confidence are not assigned but used to seed new clusters.

We have modified the USTORY framework to make it compatible with a newer deep learning framework version. We have also adjusted the data preprocessing and sentence embedding to require less RAM than the original implementation. The results reported in the USTORY paper on the Wikipedia Current Events Portal (WCEP) 2018 dataset[23] could be reproduced.

We aim to perform topic threading for multimodal data. This requires modifying the embedding as well as the keyword-based theme representation. For the embedding, we have replaced the RoBERTa sentence embedding with a CLIP[24] embedding, in particular with the CLIP ViT-L-14 available as a pretrained model for the Sentence Transformers library[25]. For text embedding, this model has the limitation that the maximum length of the token sequence supported is shorter than that of RoBERTa, requiring long sequences to be truncated. For video embedding, we can work with different modalities: First, text obtained via automatic speech recognition (ASR) is processed like a text article. Second, we consider key frames extracted from the video as ”sentences” and represent the visual representation of the video as a sequence of image embeddings. We plan to compare an approach of visual activity based key frame extraction (i.e., not synchronised with the text) as well as key frames triggered by the sentences in the text transcript (this would yield synchronised visual and image features). For the WCEP 2018 text dataset we can show that the performance with the CLIP ViT-L-14 is almost exactly the same as that with using RoBERTa (despite the fact that only a short token sequence is supported by CLIP).

[22] Singh, Devendra, et al. "End-to-end training of multi-document reader and retriever for open-domain question answering." Advances in Neural Information Processing Systems 34 (2021): 25968-25981.
[23] Ghalandari, Demian Gholipour, et al. "A Large-Scale Multi-Document Summarization Dataset from the Wikipedia Current Events Portal." *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.
[24] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PMLR, 2021.
[25] https://www.sbert.net/

For the keywords, we currently assume that we stay with a text-based representation. The keywords obtained from text can then be used to obtain a ranked list for each image embedding. We are aware that this approach requires some share of textual content to be available and does not allow to discover new key concepts only appearing in the visual domain.

3.2.4 Novelty detection for text

We consider the approach of considering the novelty and relevance of a news item with regard to the story so far as a question answering problem as a very timely one, with the potential to leverage the progress in LLMs and question answering. A recent work, named Peek Across[26], uses the idea of question extraction from one document in a collection to discover cross-links in other documents. The paper builds on a recently proposed method for extracting question-answer pairs from text named QAsem[27], and uses another method for better contextualising the question[28]. Using these methods, the approach in the paper consists of pretraining a multi-document model named QAMDen on the NewSHead dataset[29], which is then fine-tuned on the particular document set for question answering.

We have set up Peek Across to work on a story thread in order to perform question answering. This currently requires running the fine-tuning whenever the thread is modified, which is an issue to be improved in the future. We are currently working on exploring how questions extracted from a new item using QAsem and the answers obtained from posing them to the story can be best used to derive a novelty and relevance score.

3.2.5 Future plans

We are currently in the process of completing the implementation of the story threading for visual content and selecting and preparing an appropriate test dataset. Adjustments to the chosen approach concerning the choice of embedding and topic representation may be needed.

The novelty detection approach is still to be completed and evaluated. One limitation we can already foresee is that past known facts are reported as novel. For example, in a series of news items on the current war in Israel, a newspaper may publish a background piece mentioning a meeting of Yitzhak Rabin and Yasser Arafat in 1993. As this fact has not occurred in the previous articles, it is likely to be flagged as novel. This issue could be addressed by also including results in response to the questions retrieved from open knowledge sources like Wikipedia.

[26] 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).
[27] 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.
[28] 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.
[29] Gu, Xiaotao, et al. "Generating representative headlines for news stories." *Proceedings of The Web Conference 2020*. 2020.

We currently think to address the novelty aspect in the text domain, using the embeddings for visual content obtained in the threading step. This may neglect purely visual novelty (e.g., a video showing evidence of facts only verbally reported before), and thus further extensions to the approach may be needed in future.

3.3 Mixed Reality User Interface

The realm of efficient content retrieval has surged, thanks to the exponential rise in multimedia data. UNIBAS's *(MR)*^{2} content search tool, is a new concept denoting Mixed Reality Multimedia Retrieval. *(MR)*^{2}, capitalizes on the transformative capacities of Mixed Reality, featuring a live query function that empowers users to initiate queries intuitively through interaction with real-world objects. Within the new framework, we seamlessly integrate cutting-edge technologies such as object detection (YOLOv8[30]), semantic similarity search (CLIP), and data management (Cottontail DB[31]) within vitrivr. Through autonomous generation of queries based on object recognition in the user's field of view, *(MR)*^{2} creates an immersive retrieval of comparable multimedia content from a connected database. This research attempts to redefine the user experience with multimedia databases, harmoniously uniting the physical and digital domains. The success of our iOS prototype application signals promising results, setting the stage for immersive and context-aware multimedia retrieval in the years of MR.

3.3.1 Overview

As technology evolves rapidly, it unveils novel and captivating avenues for interacting with digital data, leading to an overwhelming influx of multimedia content. However, traditional retrieval techniques are needed to help manage this vast data volume. This section delves into the convergence of Artificial Intelligence (AI), Mixed Reality (MR), and multimedia retrieval, culminating in the creation of *(MR)*^{2}—a transformative concept seamlessly uniting the physical and digital realms.

The motivation behind our research arises from the growing demand for seamless user interactions with multimedia content. Conventional retrieval systems, reliant on text-based queries, often fail to deliver users the desired immersive experience. In MR environments, our goal is to facilitate effortless engagement with multimedia content by harnessing the capabilities of AI-powered object detection.

To exemplify the potential of *(MR)*^{2}, we present a use case featuring a user in a city centre adorned with an MR headset. Besides menu navigation, our system empowers users to engage directly with historical buildings and recognizing them through object detection. Concentrating on a historical artefact, *(MR)*^{2} can dynamically provide additional information about it and suggest similar artworks, transforming art exploration into an immersive journey.

Our investigation strives to redefine multimedia retrieval in MR environments through a robust framework integrating AI-driven object detection, XR technologies, and multimedia retrieval. This section introduces *(MR)*^{2} and illustrates the revolutionary impact of AI-powered live queries on user interactions within MR environments.

[30] https://github.com/ultralytics/ultralytics

[31] Gasser, R., Rossetto, L., Heller, S., & Schuldt, H. (2020, October). Cottontail DB: an open source database system for multimedia retrieval and analysis. In *Proceedings of the 28th ACM International Conference on Multimedia* (pp. 4465-4468)

3.3.2 Foundation

This section delves into multimedia retrieval, object detection, and visual-text co-embedding. Multimedia retrieval focuses on efficient content search across diverse datasets using AI-generated ranked lists. Object detection in mixed reality relies on advanced AI techniques like YOLOv8 for real-time identification. Ultralytics' YOLOv8 stands out in applications like autonomous driving. As exemplified by CLIP, visual-text co-embedding enhances multimedia retrieval robustness through AI-driven integration of visual and textual features. CLIP's transformative impact extends to tasks like zero-shot image classification.

3.3.2.1 Multimedia Retrieval

Multimedia retrieval refers to searching and retrieving content from diverse data sets, including images, videos, audio, and text, based on user queries. This process is critical in content-based image retrieval and video recommendation applications. The main challenge lies in developing efficient retrieval systems that can handle multimedia data in different modalities and formats[32]. AI plays a central role in developing effective retrieval engines that can produce a ranked list of documents based on the relevance of the user's query.

3.3.2.2 Object Detection

Detecting and interacting with physical objects in real-time in mixed reality (MR) environments demands a specialized approach to object detection. Advanced AI techniques like YOLOv8[33] and Faster R-CNN[34], particularly in deep learning, have revolutionized object detection in images and video streams. These techniques form the foundation for object detection in MR scenarios, allowing swift and accurate identification of objects in the user's surroundings. Ultralytics[35] has introduced YOLOv8, which strategically divides input images into grid cells for predicting objects. This version employs a deep neural network with convolutional layers and feature fusion techniques to enhance its ability to detect objects of varying sizes and contexts. YOLOv8's efficient architecture and feature fusion make it a go-to choice in applications like autonomous driving, surveillance, and object recognition due to its speed and reliability.

3.3.2.3 Visual-Text Co-Embedding

Visual-text co-embedding is a powerful technique that fuses visual and textual features to enhance multimedia retrieval systems. The groundbreaking Contrastive Language-Image Pre-training (CLIP[36],[37]) architecture represents a significant advance in this field. CLIP uses AI to create a unified embedding space that compares text and images directly. This integration allows textual metadata to seamlessly blend with visual content, resulting in more robust and context-aware retrieval systems. The AI-driven CLIP architecture features a jointly

[32] S. Rüger, “Multimedia Information Retrieval,” Synthesis Lectures on Information Concepts, Retrieval, and Services, vol. 1, no. 1, pp. 1–171, Jan. 2009, doi: 10.2200/s00244ed1v01y200912icr010.
[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016, doi: 10.1109/cvpr.2016.91.
[34] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/tpami.2016.2577031.
[35] https://docs.ultralytics.com
[36] Mokady, R., Hertz, A., & Bermano, A. H. (2021). Clipcap: Clip prefix for image captioning. arXiv preprint arXiv:2111.09734.
[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever: Learning Transferable Visual Models From Natural Language Supervision. CoRR abs/2103.00020 (2021)

trained vision and text encoder, allowing images and their textual descriptions to be encoded in the same space. Using a contrastive loss function, CLIP effectively brings similar image-text pairs closer while pushing dissimilar pairs apart in the shared space. CLIP's versatility extends to tasks such as zero-shot image classification, illustrating the transformative impact of AI on multimedia retrieval.

3.3.3 Concept

(MR)² stands at the forefront of revolutionising multimedia retrieval within mixed reality environments, setting the stage for the harmonisation of the natural and virtual world. This groundbreaking system harnesses the power of advanced machine learning technologies to enable real-time interaction and a user-centric design, reshaping the way users engage with their surroundings.

3.3.3.1 Key Principles

- **Immersion**: *(MR)²* is crafted to immerse users in a mixed reality environment, seamlessly blending physical and digital realms. The overarching goal is to transport users into an augmented space, allowing them to interact with their surroundings while effortlessly accessing and engaging with digital content. The immersive experience aims to captivate users, making them feel fully present within the mixed-reality environment.
- **Real-time Interaction**: Central to *(MR)²* is a robust emphasis on real-time interaction. This principle ensures that users can perform actions and receive responses without perceptible delays. Whether capturing images, selecting objects or retrieving multimedia content, *(MR)²* prioritises immediate and fluid interactions. This commitment enhances the overall user experience, making it dynamic and engaging.
- **User-Centric Design**: *(MR)²* adopts a user-centric design approach, placing the user's needs and preferences at the forefront. The system is meticulously crafted to be intuitive, user-friendly, and adaptable to individual requirements. This user-centric design spans the entire user journey within the mixed reality environment, aiming to cater to a diverse user base and ensure the concept is accessible and enjoyable for all.
- **Integration of Cutting-Edge ML Models**: To achieve accurate object detection and relevance in content retrieval, *(MR)²* integrates cutting-edge machine learning models. These models represent the pinnacle of ML and computer vision research standards, underscoring *(MR)²*'s commitment to leveraging technological advancements to provide users with precise and meaningful results.

3.3.3.2 Architecture

The architecture of (MR)², illustrated in Figure 1, comprises two integral components:

- **Frontend:**
  The frontend is designed to capture user interactions and perform computations on the device. Offering three query modalities—object detection, area selection, and text queries—the frontend ensures that query results are not only accurate but also presented in an engaging manner. This approach facilitates user comprehension and utilization of the information retrieved.
- **Backend:**
  The backend handles data from queries, processes inputs using a sophisticated machine learning model, and conducts similarity searches using feature vectors. This cohesive model aligns seamlessly with

(MR)²'s principles, facilitating the merger of real and digital worlds, ensuring real-time functionality, and prioritizing a user-centric design. Moreover, the flexibility in ML model selection positions *(MR)²* to adapt to future breakthroughs in the field.

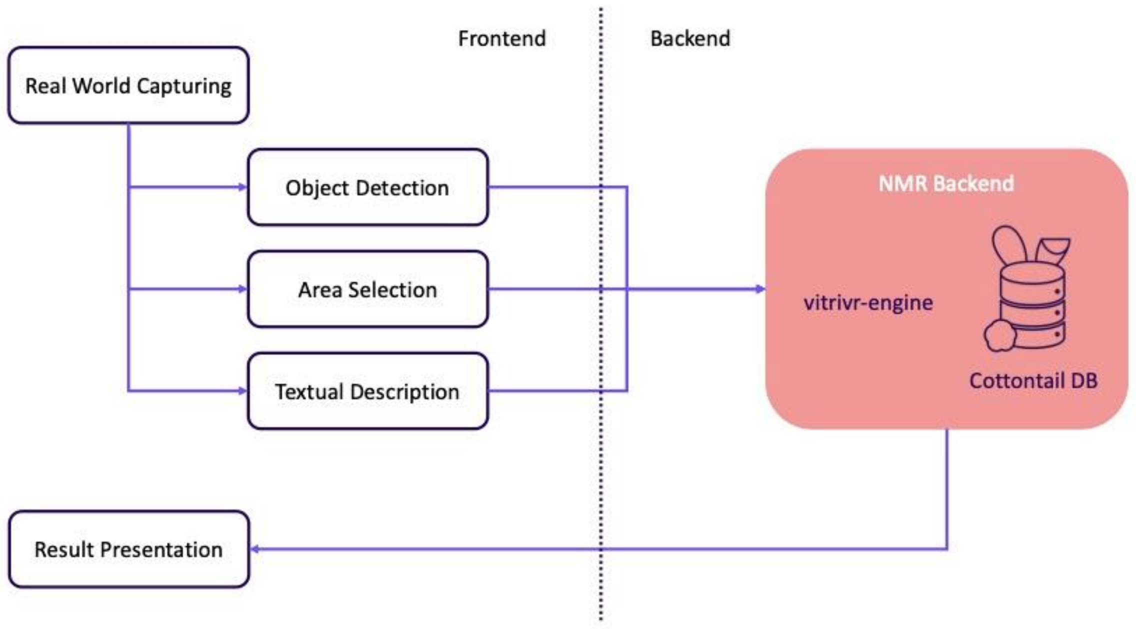


Figure 1: Conceptual Architecture of (MR)²

3.3.4 Implementation

This chapter comprehensively explores the prototype implementation of *(MR)²*, providing insights into its intricate architecture and critical components. As an iOS application tailored for iPhones and iPads, (MR)² establishes seamless communication with the NMR backend, encompassing the vitrivr engine and Cottontail DB. This collaborative integration forms a robust foundation for the system's operations.

- **iOS Application:** At the core of *(MR)²*'s functionality is the iOS application, meticulously crafted in Swift. This component catalyses mixed-reality interactions and robust object detection. Users experience a responsive interface that effortlessly adapts to real-world surroundings, thanks to the integration of the camera feed using AVFoundation. This integration allows for a smooth transition between front and back cameras, enhancing the interactive experience.
- **NMR Backend (vitrivr-engine and Cottontail DB):** The NMR backend, housing vitrivr-engine and Cottontail DB, is the retrieval centre for *(MR)²*. The backend manages CLIP feature extraction through a RESTful API and executes similarity searches in Cottontail DB. This real-time process ensures the prompt retrieval of the top 100 similar objects, showcasing the system's efficiency in delivering meaningful results to users.

(MR)²'s integration of the camera feed using AVFoundation goes beyond mere visual capture. It captures the essence of real-world surroundings, offering users an interactive and immersive experience. The app's commitment to a responsive interface allows users to effortlessly switch between front and back cameras, contributing to the fluidity of the overall interaction.

The powerful combination of Apple's Vision and CoreML frameworks, featuring the YOLOv8 model, empowers *(MR)²*'s object detection capabilities. This dynamic approach ensures the identification of objects in the live camera feed, providing users with real-time bounding boxes for a comprehensive understanding of their surroundings, seen in Figure 2 (a).

The backend performs CLIP feature extraction on captured images, followed by real-time similarity searches in Cottontail DB. This meticulous process ensures the prompt retrieval of the top 100 similar objects, solidifying *(MR)²*'s commitment to delivering efficient and meaningful user results.

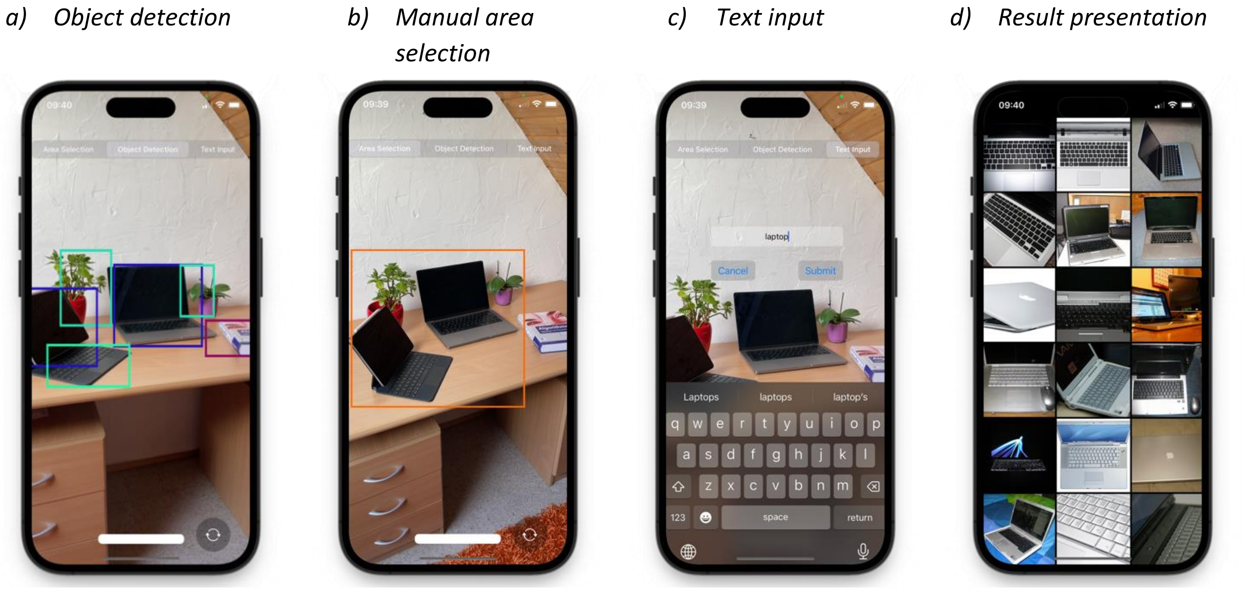


Figure 2: Different views while using (MR)²

The app does not just retrieve results; it presents them in real-time within a dedicated ViewController. The scrollable grid, strategically designed to display the most similar images with the highest similarity in the top-left corner, offers users a visually intuitive way to navigate through results, shown in Figure 2 (d). Users can enlarge individual objects for closer inspection, enhancing the overall user experience.

Beyond live queries, *(MR)²* caters to diverse user needs with additional query options. Users can initiate region-based queries through a touch input, positioning a rectangle on the screen within the camera feed (Figure 2 (b)). For text queries, presented in Figure 2 (c), users enter descriptions, and CLIP enables cross-modal similarity comparisons with image content. This flexibility ensures that *(MR)²* is responsive and adaptable to varying user preferences and query types.

3.3.5 Evaluation

This section offers a thorough evaluation of *(MR)²*, covering both analytical and user-centric aspects. We analyse functional performance, including object detection inference time, query response time, and real-world usability. The section concludes with a focused discussion on overall performance and future advancements.

3.3.5.1 Performance and User Evaluation

To evaluate *(MR)²*'s functional prowess, we undertook a comprehensive two-phase assessment with an analytical evaluation followed by a user-centric exploration.

The analytical evaluation meticulously examined pure performance metrics, measuring the inference time for object detection and query response time. With a median inference time of 24.8ms, *(MR)²* ensures real-time applicability. The average query time at 4191ms showcases remarkable efficiency, particularly given the extensive ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset[38] stored in Cottontail DB.

Simultaneously, the user evaluation engaged 14 participants in real-world scenarios, spotlighting practical usability. Participants with moderate to high technology affinity (average ATI score of 4.43[39]) found *(MR)²* to be user-friendly and efficient. Reinforcing this positive perception, the System Usability Scale (SUS) score of 87 reflects the overall favourable views on usability[40].

Open feedback from users echoed the encouraging SUS results, highlighting *(MR)²*'s intuitiveness and practicality. While minor concerns surfaced, such as overlapping bounding boxes when multiple objects are detected, participants expressed a strong inclination to continue using *(MR)²*, underscoring its user-friendliness and potential for widespread adoption.

3.3.5.2 Discussion

While *(MR)²* has garnered positive feedback, we understand the perpetual need for improvement. Our commitment to enhancing user experience involves continuous exploration, especially in broadening the range of supported objects. By incorporating user feedback, *(MR)²* remains on a trajectory of evolution and improvement. Currently, we are actively pursuing three exciting pathways that promise a more immersive future in multimedia retrieval:

1. Advancements are underway in query modes, immersive result presentation, device integration, and cutting-edge machine learning techniques, such as OCR, ASR, and tag integration.
2. An exploration into complexity and context awareness is unfolding, focusing on temporal queries to enrich the search landscape. Pioneering immersive result presentation in MR contexts, seamlessly overlaying search results, is a key area of interest.

[38] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, Apr. 2015, doi: 10.1007/s11263-015-0816-y.
[39] T. Franke, C. Attig, and D. Wessel, "A Personal Resource for Technology Interaction: Development and Validation of the Affinity for Technology Interaction (ATI) Scale," International Journal of Human–Computer Interaction, vol. 35, no. 6, pp. 456–467, Mar. 2018, doi: 10.1080/10447318.2018.1456150.
[40] "SUS: A 'Quick and Dirty' Usability Scale," Usability Evaluation In Industry, pp. 207–212, Jun. 1996, doi: 10.1201/9781498710411-35.

3. Beyond enhancing iOS accessibility, we are venturing into compatibility with MR glasses like Meta Quest Pro or future Apple Vision Pro devices, opening new possibilities for more immersive applications. Crucial to shaping the future of MR multimedia retrieval are advancements in object detection models and self-training approaches.

3.3.6 Related Work

In the dynamic landscape of XR, a tapestry of multimedia retrieval systems has laid the groundwork for innovations preceding *(MR)²*, offering diverse perspectives on integrating XR with multimedia and enriching user interactions in immersive environments.

Prioritizing text input in VR, vitrivr-VR[41] [42], intricately connected to [43], spearheads innovation in VR interfaces. This system diverges from *(MR)²*'s live queries, carving its path in virtual reality exploration. Meanwhile, [44] ventures into projecting multimedia objects for visual analytics, utilizing the multi-dimensional multimedia model (M[45] within the VR realm. Although providing advanced visual analysis support, ViRMA lacks *(MR)²*'s object detection and an automated query approach.

Shifting the focus to complete cultural heritage exploration, *GoFind!*[46] seamlessly blends content-based multimedia retrieval with Augmented Reality (AR). In contrast to *(MR)²*, *GoFind!* places a spotlight on historical exploration, embracing varied query modalities to enhance user engagement in augmented environments.

3.3.7 Conclusions

(MR)², a new Mixed Reality (MR) Multimedia Retrieval concept that uses MR technology's power to transform how users interact with multimedia content. At the core of *(MR)²* lies an innovative approach to query formulation and a live query option that seamlessly connects the digital and physical worlds. Our prototype on iOS devices demonstrated how *(MR)²* can enhance the user's multimedia retrieval experience.

[41] F. Spiess, R. Gasser, S. Heller, L. Rossetto, L. Sauter, and H. Schuldt, "Competitive Interactive Video Retrieval in Virtual Reality with vitrivr-VR," in Proceedings of the 27th International Conference on MultiMedia Modeling (MMM 2021) – Part II, ser. Lecture Notes in Computer Science, vol. 12573. Prague, Czech Republic: Springer, Jun. 2021, pp. 441–447. [Online]. Available: https://doi.org/10.1007/978-3-030-67835-7_42

[42] F. Spiess, P. Weber, and H. Schuldt, "Direct Interaction Word-Gesture Text Input in Virtual Reality," in IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR 2022), Virtual Conference. IEEE, Dec. 2022, pp. 140–143. [Online]. Available: https://doi.org/10.1109/AIVR56993.2022.00028

[43] L. Rossetto, I. Giangreco, C. Tanase, and H. Schuldt, "vitrivr: A flexible retrieval stack supporting multiple query modes for searching in multimedia collections," in Proceedings of the 24th ACM international conference on Multimedia. ACM, Oct. 2016.

[44] A. Duane and B. Þ. Jónsson, "ViRMA: Virtual Reality Multimedia Analytics," in ICMR '22: International Conference on Multimedia Retrieval. Newark, NJ, USA: ACM, Jun. 2022, pp. 211–214. [Online]. Available: https://doi.org/10.1145/3512527.3531352

[45] S. Gíslason, B. Þ. Jónsson, and L. Amsaleg, "Integration of Exploration and Search: A Case Study of the M3 Model," in Proceedings of the 25th International Conference on MultiMedia Modeling (MMM 2019) – Part I, ser. Lecture Notes in Computer Science, vol. 11295. Thessaloniki, Greece: Springer, Jan. 2019, pp. 156–168. [Online]. Available: https://doi.org/10.1007/978-3-030-05710-7_13

[46]

(MR)² utilizes the YOLOv8 model for object detection. While we acknowledge the opportunities for expanding the supported object types, *(MR)²* exemplifies the potential for object recognition in MR environments. Additionally, we leverage the CLIP machine learning model for similarity searches to enhance retrieval accuracy.

The positive user evaluations have underscored *(MR)²*'s potential for engaging multimedia retrieval experiences. The live query option and seamless MR integration received particular acclaim, validating our user-centred design.

As we look to the future, *(MR)²* lays the foundation for further advancements. Very important, of course, is to use the frontend to be able to select specific multimedia data on-site in the XReco use cases. Therefore, the retrieval should be incorporated there.

4 Neural Rendering Services

4.1 Overview

This section describes the work done in T4.2 Neural Rendering Services. T4.2 is a pivotal element of the XReco project, focusing on the implementation and release of XReco's neural rendering services, as well as their integration with industry-standard 3D engine workflows. This task is set to play a vital role in the delivery of advanced XR applications.

The primary objective of T4.2 is to integrate existing solutions and advance current research efforts in the field of Neural Radiance Fields (NeRF). The goal is to provide a range of options for creating three-dimensional (3D) representations from image collections within XReco's repository. These 3D life-like representations are fundamental for the immersive XR experiences that XReco envisions.

The following subsections investigate the development activities of different neural rendering algorithms considering three kinds of scenarios: i) General object NeRF-based reconstruction; ii) In-the-wild scenarios in which NeRF algorithms are trained on unstructured datasets; and iii) Human-centred NeRF-based reconstruction from sparse viewpoints utilising RGB-D data.

4.2 Instant Neural Graphics Primitives

There have been many advancements in computer graphics as a research field, especially since the advent of the original NeRF algorithm[47]. Many research works target NeRF scalability, quality, as well as training and rendering speed. A notable work, Instant Neural Graphics Primitives by Muller et al. [48] (INGP) targets fast NeRF convergence via encoding the 3D space with multi-resolution voxel grids and enabling fast voxel grid queries by utilising hash tables. This led to very fast NeRF training times (minutes instead of hours), providing a very pertinent candidate for employing within XReco, as it enables fast prototyping and user-feedback.

[47] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1), 99-106.

[48] Müller, Thomas, et al. "Instant neural graphics primitives with a multiresolution hash encoding." *ACM Transactions on Graphics (ToG)* 41.4 (2022): 1-15.

An INGP implementation was first provided to XReco's end-users as a standalone MS Windows application which helped the users understand the power as well as the drawbacks of the algorithm in a hands-on manner. The circulated application features image collection or video uploading, with the ability of estimating camera poses, and then initiates NeRF training on the specified data collection. The User Interface of the mentioned Windows application can be described in the two following figures. Figure 3 shows the application as it starts, where the users can upload images or video of the scene and start estimating camera pose and position. Another available option is to upload an already estimated scene (e.g., using an SfM algorithm) to start the train immediately. Figure 4 shows what is the result when the scene is training and a panel where the user can view the progress of the training, optimise the camera, take a snapshot, and export mesh, volume, or slices. From the GUI there is also an option to render a video from the trained scene. This GUI-enabled application was circulated to end-users within the consortium to offer the ability to have a hand-on experience on future tool implementations.

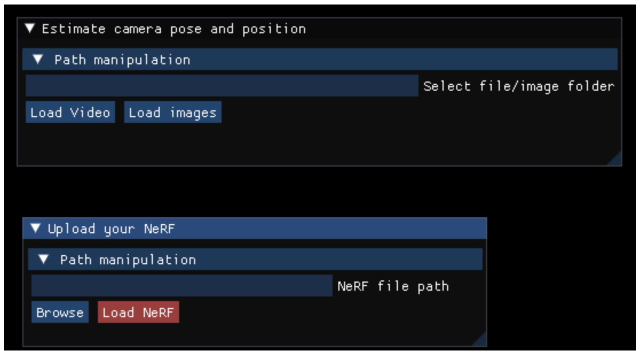
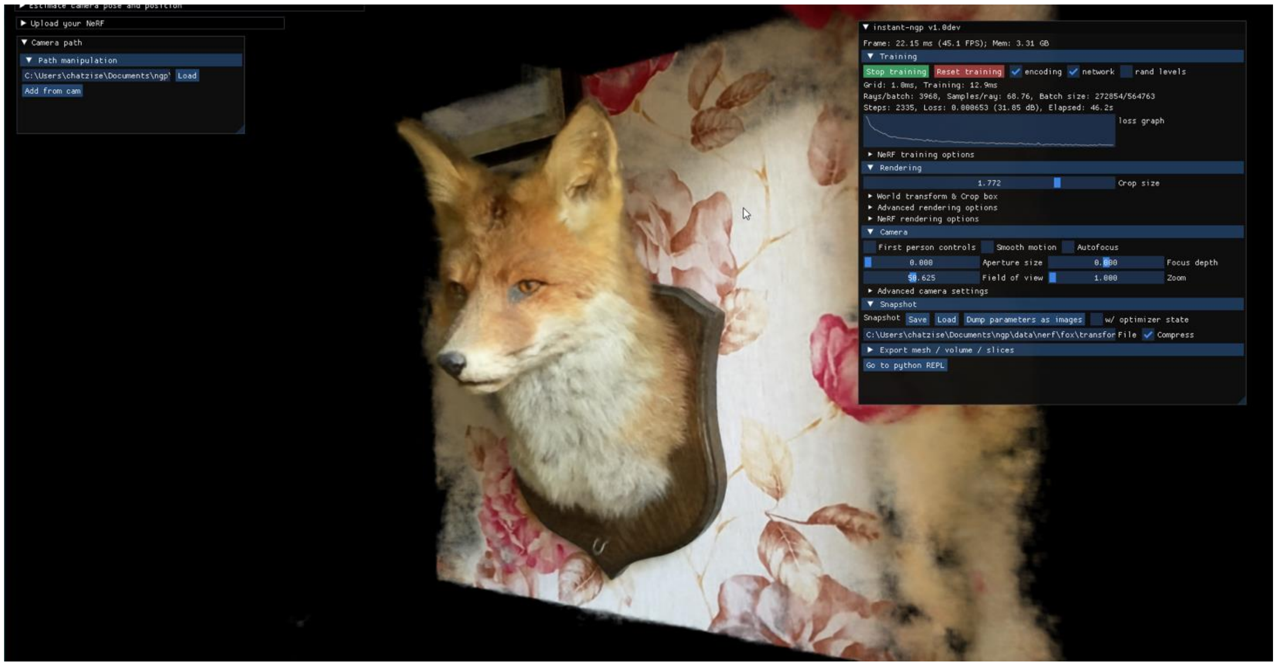


Figure 3: INGP User Interface: Video and Image collection uploading.


```
▶ Estimate camera pose and position
▶ Upload your NeRF
▼ Camera path
▼ Path manipulation
C:\Users\chatzise\Documents\ngp\  Load
Add from cam

▼ instant-ngp v1.0dev
Frame: 22.15 ms (45.1 FPS); Mem: 3.31 GB
▼ Training
Stop training  Reset training  ✓ encoding  ✓ network  rand levels
Grid: 1.0ms, Training: 12.9ms
Rays/batch: 3968, Samples/ray: 68.76, Batch size: 272854/564763
Steps: 2335, Loss: 0.000653 (31.85 dB), Elapsed: 46.2s
                                                    loss graph
▶ NeRF training options
▼ Rendering
                1.772                               Crop size
▶ World transform & Crop box
▶ Advanced rendering options
▶ NeRF rendering options
▼ Camera
First person controls  Smooth motion  Autofocus
        0.000      Aperture size          0.000        Focus depth
        50.625     Field of view          1.000        Zoom
▶ Advanced camera settings
▼ Snapshot
Snapshot  Save  Load  Dump parameters as images  w/ optimizer state
C:\Users\chatzise\Documents\ngp\data\nerf\fox\transfor File  ✓ Compress
▶ Export mesh / volume / slices
Go to python REPL
```

Figure 4: INGP User Interface: Live NeRF training interface.

4.3 NeRF in-the-wild

The reasoning behind providing a fast NeRF algorithm to XReco's end users is to make them aware of the technology, its advantages and disadvantages, as well as receiving early feedback. Despite the fact that INGP offers fast training (and thus enables fast prototyping), its results in terms of quality are subpar when compared to more recent research efforts and algorithms. Additionally, INGP is a generic NeRF algorithm which does not concretely focus on similar aspects as XReco in terms of content aggregation. One of the main considerations in XReco, is the ability to search for content inside a large pool of assets, which centralises content from different organisation repositories. In order to enable NeRF-based reconstruction from different sources (i.e., collections of images of the same scene, but potentially shot at different timings, meaning that there could be different illumination settings in each image in the collection), NeRF-in-the-wild (NeRFw)[49] was reimplemented to be adapted for use within XReco. NeRFw is an algorithm that extends the capabilities of NeRF, providing the ability to handle unstructured collections of photographs taken in the real-world in an unconstrained manner, with the goal of creating radiance fields in in-the-wild settings (a subset of such training images is presented in Figure 5), by providing two key extensions over the original NeRF algorithm.

First, is its ability to handle variable illumination. This means that the system can be trained on images captured at different times of day under different lighting conditions. This is achieved by incorporating a learnable appearance embedding and training a neural network to output a different RGB colour conditioned on that embedding.

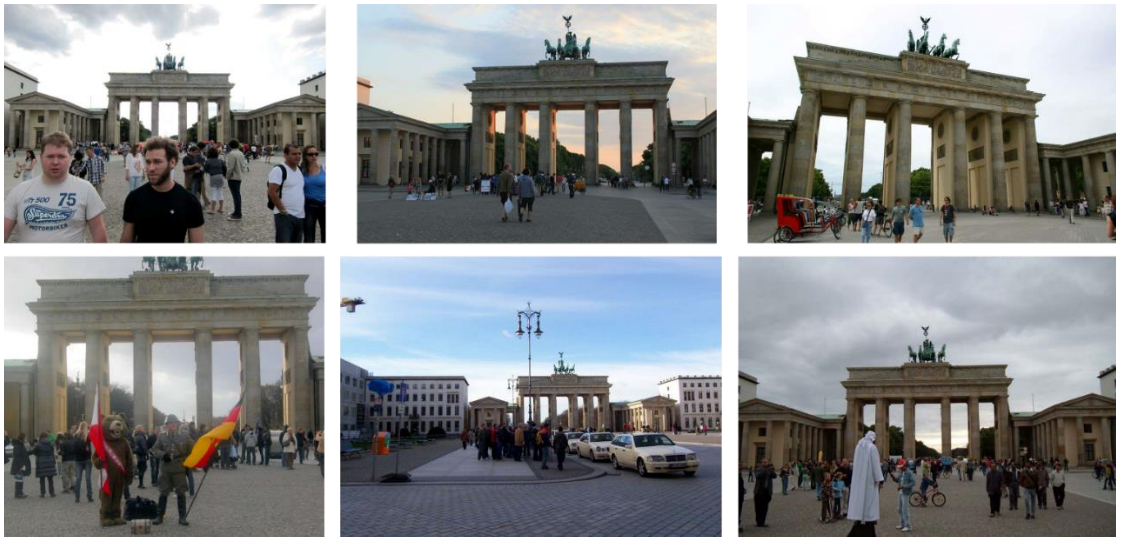


Figure 5: An exemplary subset of in-the-wild training data used to train the NeRFw algorithm[50].

Another significant feature is its ability to remove transient occlusions. Similarly, another neural network is included in the training pipeline, which learns a transient embedding outputting the probability of a pixel being

[49] Martin-Brualla, Ricardo, et al. "Nerf in the wild: Neural radiance fields for unconstrained photo collections." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[50] Bi, X., Chen, Y., Liu, X., Zhang, D., Yan, R., Chai, Z., ... & Liu, X. (2021). Method Towards CVPR 2021 Image Matching Challenge. arXiv preprint arXiv:2108.04453.

transient (occluded) or not. This probability is used as a probability mask during training for blending the final colour during rendering and is able to disregard occlusions in a scene (Figure 7).

The original NeRFw algorithm, does not introduce other novelties except from the ones mentioned. Therefore, training convergence is rather slow, and even slower than the original NeRF due to the incorporation of two additional multi-layer-perceptions (MLP) in the training pipeline. To that end, the algorithm was modified by replacing the original positional encoding[47] with the multi-resolution hash grid encoding presented in INGP[49]. This resulted in achieving 10x faster convergence times (though training convergence does not yet achieve desirables times and stays in the order of hours). Even though the training duration is large, the qualitative results are encouraging, as presented in Figure 6.

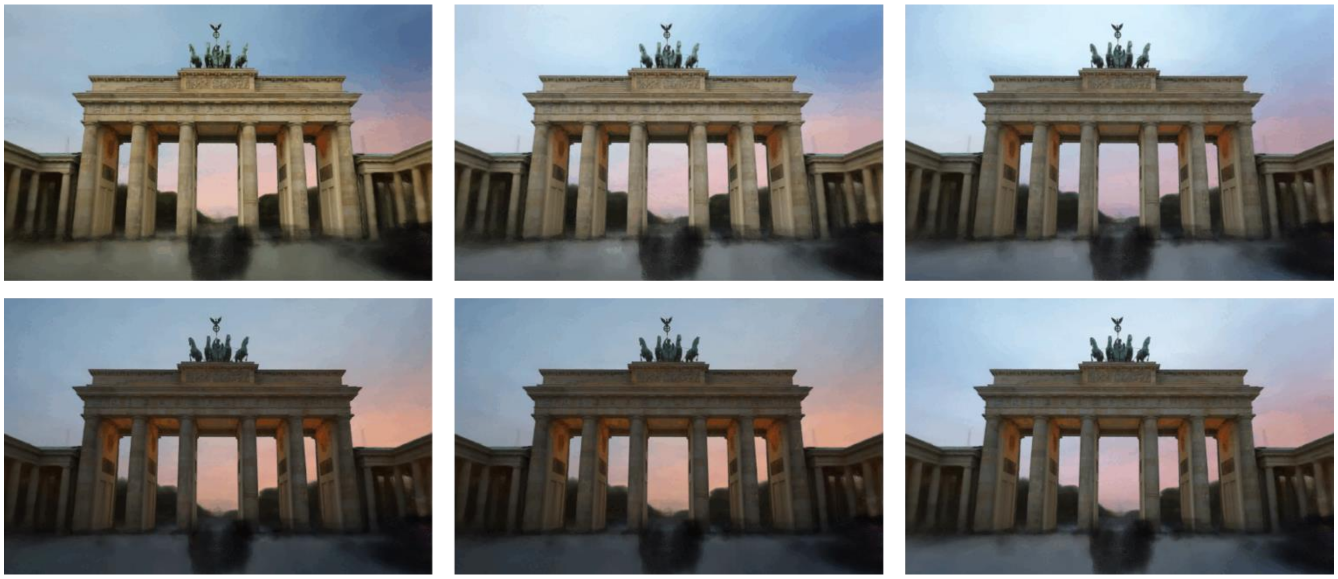


Figure 6: NeRFw appearance embedding space interpolations on the same scene.

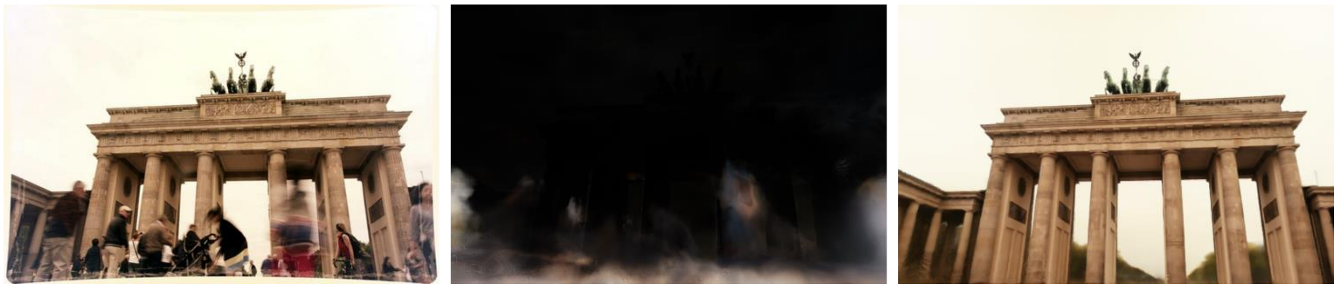


Figure 7: NeRFw transient embedding effect.

Figure 7 shows from left to right: An original image from the training dataset, the estimated transient embedding for that image, the same viewpoint rendered taking into account the transient embedding.

For future work, our focus lies on conducting experiments aimed at speeding up training convergence. Also, the intention is to encapsulate these developments within a Docker container, ensuring encapsulation and portability. Furthermore, leveraging the FastAPI framework will facilitate the integration of these capabilities into the XRECO project, thereby enhancing its functionality and usability. Additionally, there is a planned initiative to facilitate the exportation of the model for integration within the Unity3D platform.

4.4 Human-centred NeRF

NeRF approaches have been proven to excel at highly realistic novel view synthesis, not only in the task of novel-view synthesis for general static scenes, but in dynamic human-centred scenarios too. However, they lack in many parts when considering user friendliness in such human-centred scenarios. First, their requirement for large amounts of training viewpoints (common scene datasets used for NeRF training contain dozens to hundreds of viewpoints) is considered an obstacle, as this becomes an important barrier: having that many cameras surrounding a human subject potentially degrades user experience and makes setting up a new capturing rig difficult (requiring a lot of human effort) and expensive due to the number of cameras required. To that end, methods that focus on reducing the number of viewpoints needed were investigated. Additionally, another strategy that was employed to address the issue of insufficient coverage was RGB-D data utilisation during training. Moreover, standard NeRF training pipelines typically use SfM algorithms for viewpoint estimation. These algorithms require a large amount of overlap among the different views. This cannot be easily achieved in a human-centred capturing scenario, in which camera positions are optimised for covering the user. To that end, viewpoint calibration is required. Finally, NeRF algorithms are designed to basically overfit a single scene, and thus, they are not able to directly infer the properties of novel scenes. Therefore, the line of generalizable NeRF algorithms was additionally explored.

4.4.1 Related work

Many research works focus on reducing the number of needed input views by supplementing the training process with extra information and improve on the sampling strategy or add new components to the training loss. DietNerf[51] is based on the principle that independently of the viewpoint, the reconstructed scene is always the same. Therefore, a semantic loss term is additionally employed, based on features extracted from a vision transformer trained on hundreds of millions of images annotated with natural language. DS-NeRF[52] adds a depth loss in the training process. In most cases an SfM algorithm is used to extract the unknown camera poses, usually COLMAP[53]. Part of the algorithm's output is a sparse point cloud of the scene that allows to extract the depth value of some of the pixels in the input images. Comparing the expected depth with the depth rendered by the NeRF algorithm is added as an extra supervision term. NerfingMVS[54] uses the same depth values to guide a monocular depth estimation network to extract a dense depth map that is then used as depth priors to train a

[51] Jain, A., Tancik, M., & Abbeel, P. (2021). Putting nerf on a diet: Semantically consistent few-shot view synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5885-5894).
[52] Deng, K., Liu, A., Zhu, J. Y., & Ramanan, D. (2022). Depth-supervised nerf: Fewer views and faster training for free. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12882-12891).
[53] Schönberger, J. L. (2018). Robust methods for accurate and efficient 3D modeling from unstructured imagery (Doctoral dissertation, ETH Zurich).
[54] Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., & Zhou, J. (2021). Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5610-5619).

NeRF. RGBDNeRF[55] uses traditional volume reconstruction techniques to obtain new views of the scene and pretrain a NSVF on those and then finetunes the network on the original images using in both stages a patch.

4.4.1.1 Generalizable NeRF algorithms

A branch of generalizable NeRF algorithms (i.e., algorithms that are able to generalise across different scenes) emerged with MVSNeRF[56]. These approaches allow the synthesis of new views from a few input views from the scene without any training. These models can be finetuned in order to obtain better results on the specific scene. MVSNeRF warps 2D image features onto a plane sweep volume, which is used to interpolate features on samples along the rays of the novel view. ENeRF[57] proposes to estimate depth bounds from a cost volume obtained from the plane sweep volume, which is also used for feature interpolation. Additionally, they employ a coarse and fine volume in order to be able to perform real-time view synthesis. Figure 8 illustrates the approach that ENeRF follows.

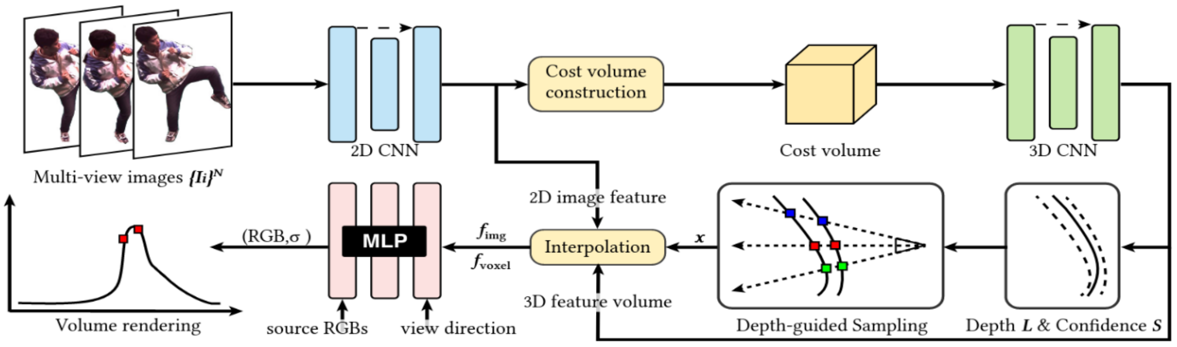


Figure 8: ENeRF's generalisable approach.

More recently, S-VolSDF[58] proposed to combine feature probability volumes in order to predict the signed distance to the surface from each view and combine them in order to render the surface with VolSDF[59]. This approach is proven to outperform existing methods for mesh estimation given a set of sparse views. Nevertheless, this approach is non-generalizable and is not suited for real-time view synthesis. Neo360[60] is a recent algorithm that generalizes to novel scenes and has been proven to be effective in a sparse camera setting.

[55] Dey, A., & Comport, A. I. (2022). RGB-D Neural Radiance Fields: Local Sampling for Faster Training. arXiv preprint arXiv:2203.15587.

[56] Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., & Su, H. (2021). Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 14124-14133).

[57] Lin, H., Peng, S., Xu, Z., Yan, Y., Shuai, Q., Bao, H., & Zhou, X. (2022, November). Efficient neural radiance fields for interactive free-viewpoint video. In SIGGRAPH Asia 2022 Conference Papers (pp. 1-9).

[58] Wu, H., Graikos, A., & Samaras, D. (2023). S-VolSDF: Sparse Multi-View Stereo Regularization of Neural Implicit Surfaces. arXiv preprint arXiv:2303.17712.

[59] Yariv, L., Gu, J., Kasten, Y., & Lipman, Y. (2021). Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems, 34, 4805-4815.

[60] Irshad, M. Z., Zakharov, S., Liu, K., Guizilini, V., Kollar, T., Gaidon, A., ... & Ambrus, R. (2023). Neo 360: Neural fields for sparse view synthesis of outdoor scenes. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 9187-9198).

This approach constructs an image-conditioned triplane representation which allows the model to infer scene features. They combine this local representation with global features to obtain the colour and density of the input points. Additionally, they perform instance segmentation of the objects in each scene, which is determined by an additional output of the NeRF's MLP.

Based on the reviewed literature, the approach that is closer to our goal is ENeRF, as it is applicable in real-time settings. Nevertheless, as will be shown in the results section, it is not well-suited for a sparse set of views. As Neo360 was published one year after the start of the project, we do not consider it as a baseline. Nonetheless, it is not applicable to a real-time setting. Therefore, we consider ENeRF as our baseline model. We don't consider approaches that involve human priors as we want to explore the potential of model-free approaches.

4.4.2 Databases

We benchmark the methods considered on standard databases which consist of general scenes and human-centred sequences containing depth maps associated with each view. We have also focused on datasets in which the camera calibration parameters are given as their approximation and improvement is not part of our research:

- DTU[61]: The scenes include a wide range of objects captured under controlled settings with a robotic arm. It contains approximately 4000 images from 80 scenes. As this dataset is captured under a dense camera setting, sparse views are sampled from the complete set.
- CWI[62]: Human-centred dataset captured by a multi-view setup of 7 cameras surrounding the subject (Figure 9). They were recorded with Kinect4Azure cameras[63]. It contains a total of 20 sequences with different subjects performing certain actions.
- ActorsHQ[64]: Actors-HQ is a high-fidelity dataset of clothed humans in motion. The dataset features multi-view recordings of 160 synchronised cameras that simultaneously capture individual video streams of 12MP each. As such, the dataset is tailored for the tasks of photo-realistic novel view and novel pose synthesis of humans.

[61] Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., & Aanæs, H. (2014). Large scale multi-view stereopsis evaluation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 406-413).

[62] Reimat, I., Alexiou, E., Jansen, J., Viola, I., Subramanyam, S., & Cesar, P. (2021, June). CWIPC-SXR: Point Cloud dynamic human dataset for Social XR. In Proceedings of the 12th ACM Multimedia Systems Conference (pp. 300-306).

[63] https://learn.microsoft.com/en-us/azure/kinect-dk/

[64] Işık, M., Rünz, M., Georgopoulos, M., Khakhulin, T., Starck, J., Agapito, L., & Nießner, M. (2023). Humanrf: High-fidelity neural radiance fields for humans in motion. arXiv preprint arXiv:2305.06356.

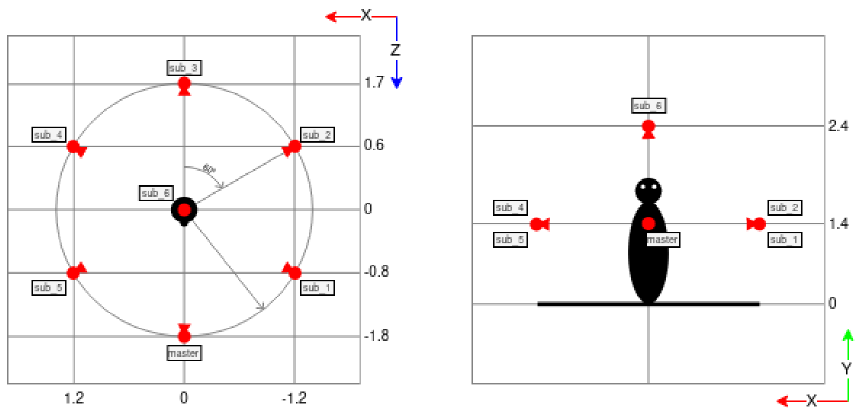


Figure 9: Camera distribution on the CWI dataset. From left to right: Top view, and frontal view.

For the generalizable depth based NeRF approach, the general scenes are used during the training process of the NeRF. Human-centred sequences are considered for both training and evaluation settings in order to evaluate the zero-shot inference capabilities, as well as the inference quality on finetuned scenarios.

4.4.3 Depth-assisted Data Augmentation for NeRF initialization

We take the data augmentation idea of RGBDNeRF[55] and develop a two-step training pipeline that can be used in combination with other methods and is easy to be combined with other extensions and improvements.

The proposed pipeline consists of a first phase in which a point cloud representation of the scene is generated, and a set of novel views are rendered as a direct projection of that point cloud. The rendered images are then used in an initial training step of the NeRF network. At a second stage, the NeRF model is trained using the original images. The hypothesis is that at the first stage the network is fed with enough views to learn the radiance field of the scene and in the second phase we train it to synthesise more realistic colours and textures, given that training with the generated views achieves a quality similar to what would be achieved using only classical reconstruction methods with noticeable artefacts typical to NeRF networks (such as floating points).

The flexibility of the pipeline permits effortless integration with other NeRF algorithms and allows us to leverage the fact that no depth information is required, as the training process itself is able to generate point clouds of the training scenes. However, we can also experiment with adding additional training features, such as depth images instead of point clouds.

Figure 10 shows the fill NeRF pipeline: The original images, depth maps and camera parameters are used to generate novel views (images, depth maps, camera parameters and masks), which are then used to train a NeRF network. The original images are also used to extract masks. The trained network's weights are used to initialise a second NeRF model that is trained with the original images, camera parameters and masks.

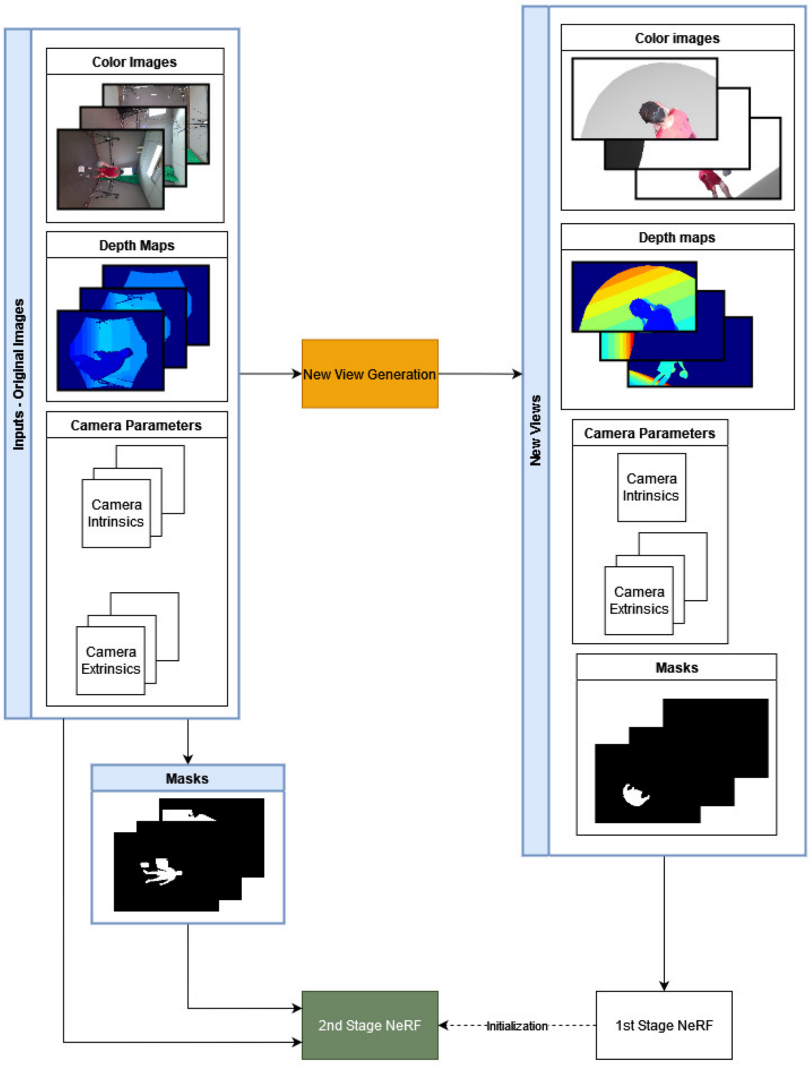


Figure 10: The full NeRF training pipeline

4.4.4 Novel View Generation

The process of generating novel views is based on classical volume reconstruction techniques. The colour images, depth maps and camera-intrinsic parameters are used to generate individual point clouds that are then fused together to form a single one representing the entire scene. The floor and walls of the room are removed by using a cylindrical filter around the person in the centre of the scene. Then the virtual scene can be modified by adding synthetic objects, such as a floor plane or walls. From this scene we can also extract the depth maps and binary masks (images that indicate if a pixel is occupied by the subject to be reconstructed or not) of the subject as well as the camera parameters. Figure 11 visualises the novel view generation pipeline: Colour and depth images, as well as camera intrinsics parameters are used to generate a single point cloud for each viewpoint. These are then fused into a single one using the extrinsics parameters of each viewpoint (i.e., camera positions and orientation). The fused point cloud is filtered using a cylindrical spatial filter. The fused point cloud can be rendered from novel viewpoints to generate new training samples.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

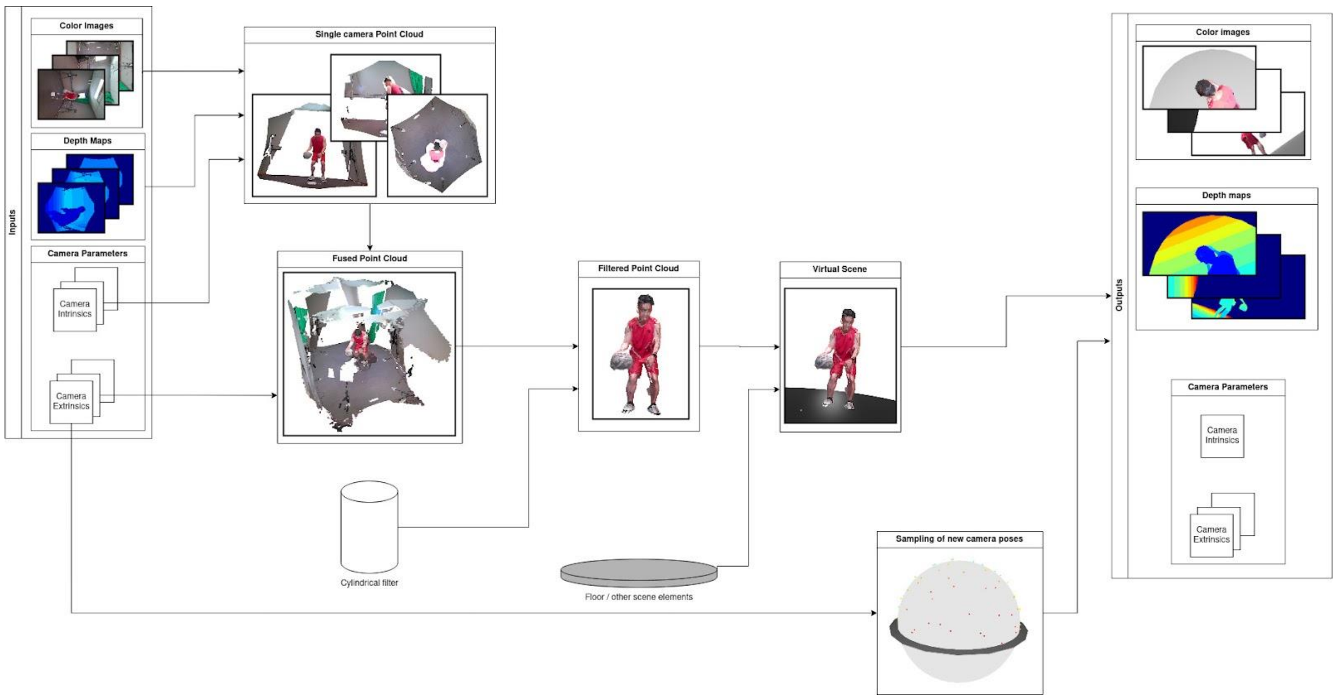


Figure 11: Novel view generation pipeline

4.4.5 Experiments and results

The first batch of experiments utilise the CWI dataset. The same subset of views has been chosen as test viewpoints and kept from all stages of training for all experiments. The rest of available viewpoints have been added one at a time, in each testing configuration with different numbers of views, with the goal of testing the effect of providing a higher number of samples model. INGP was chosen as a baseline method, considering its speed and good results.

Figure 12 presents the Peak Signal to Noise Ratio (PSNR) and the Learned Perceptual Image Patch Similarity (LPIPS) results averaging over the scenes tested and incrementing the number of views. Binary masks were used during training for removing the background as it is not considered for reconstruction. Training on ground truth images only is presented in blue. Pink lines correspond to first training with synthetic images in the first stage. Orange lines include fine-tuning the model with the original images, and the green lines correspond to adding a small amount of training steps using depth supervision. In Peak Signal to Noise ratio (PSNR, top) higher means better. In Learned Perceptual Image Patch Similarity (LPIPS, bottom) lower means better.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

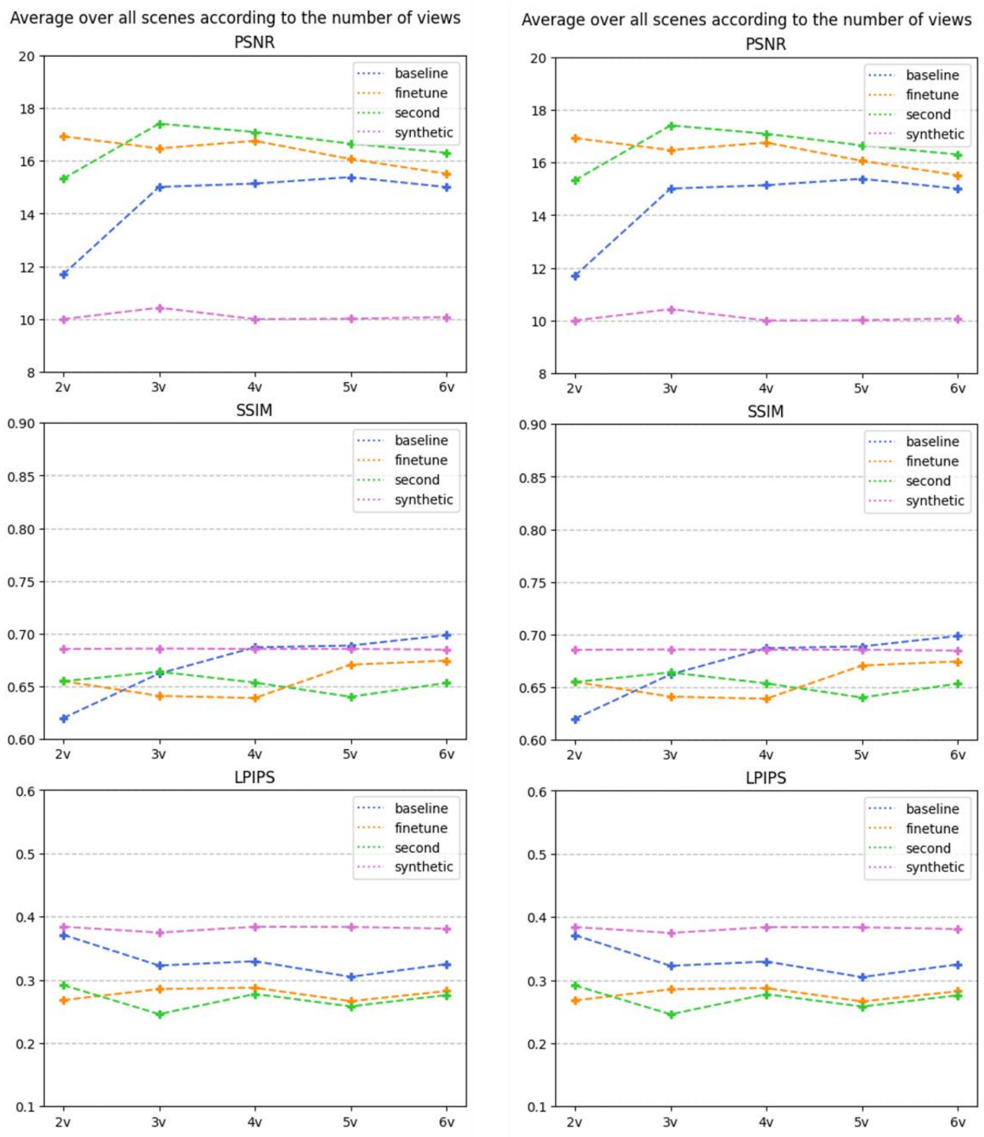


Figure 12: Plots presenting the average metrics for all scenes for the test image according to the number of views.

As we can see, the quality of the results improves only until 3 views, in terms of PSNR. However, it does not seem to improve when more views are added. This is due to the fact that the cameras in the CWI dataset are placed to cover the central subject as completely as possible with the smallest number of cameras. This means that when one is taken out, that view will be the furthest away from all the other cameras and consequently where the network and training will invest less of their resources. Moreover, since it only has 7 views, we only keep one for testing and the metrics consider a single test sample. To that end, the next phase of experiments considers datasets with more viewpoints available, as they facilitate better testing and experimentation settings.

As can be observed the visual synthetic results give the worse results, part of it is that some of the white background bleaches through and a white fog covers most of the person, but when doing cross sections of the resulting NeRF, the geometry of the person can be seen there, but still is quite bad quality and no better than using classical volume reconstruction techniques. Figure 13 shows rendered images from NeRF trained on synthetic images. Left: a rendered image from the viewpoint of one of the training cameras, where the subject is mostly obscured by white fog. Right: a rendered image of the same model, but cutting the space of rendered points. The main subject has a halo of empty space between them along with white fog.

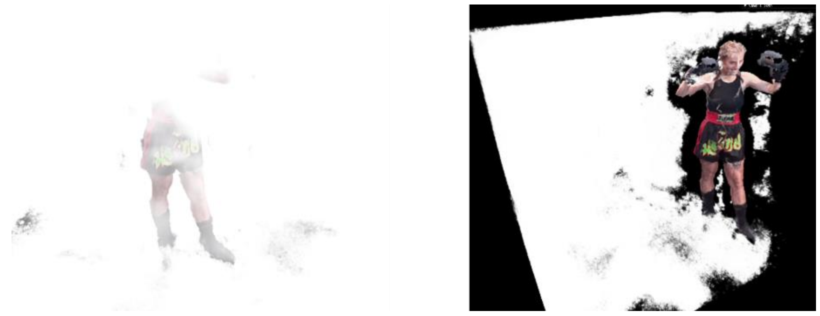


Figure 13: Rendered images from NeRF trained on synthetic images.

In Figure 14 we can see the results in different stages of the training: (a) ground truth image, (b) rendered image using only 6 input images from the same viewpoint, (c) rendered image with the full pipeline, (d) presents the same setting in (b), but with the extra depth supervision incorporated during training. In all cases 6 real view images were used as input and training was executed with binary masks except for the last screenshot in the step with depth because masks make the results worse. The screenshots are from the viewpoint of the training camera.

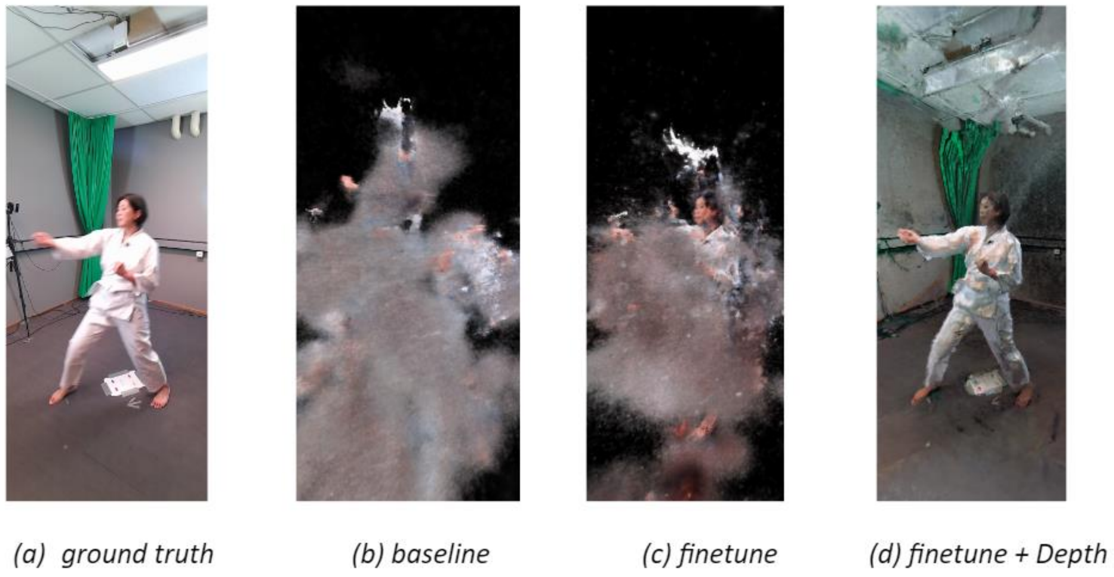


Figure 14: Different stages of training

The second part of the experiment considers adding depth supervision in all training steps, as well as utilising the ground truth images only. Quantitative results are presented in Figure 15. The blue lines represent training with the ground truth images only. The whole pipeline is represented by orange. The green crosses represent the full pipeline without depth supervision in general, but finetuned at a later stage as described above. The x-axis represents the weight of the depth supervision loss. The results with depth supervision 0.0 serve as reference and come from the previous experiments. All the different settings presented were trained with 6 viewpoints for supervision.

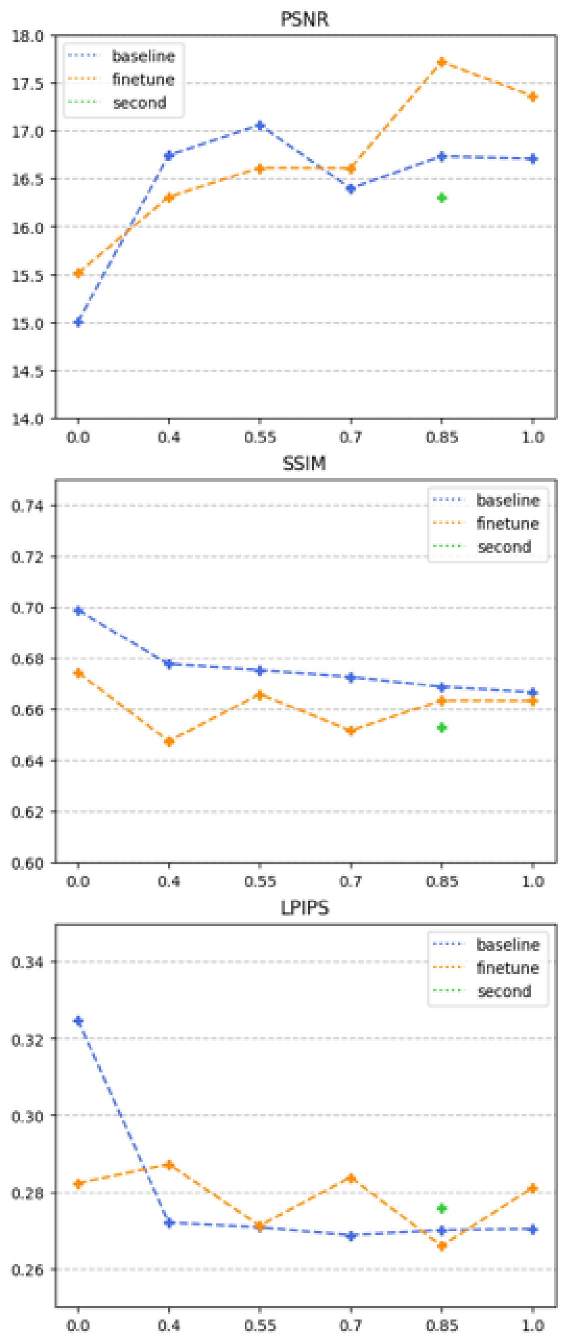


Figure 15: Average PSNR and LPIPs scores for all scenes while utilising depth supervision.

In this case (Figure 15) we can see that using the whole pipeline improves the results only within a range of weight values for the depth loss in terms of both PSNR and LPIPS. However, analysing the visual results (Figure 16), it is possible to appreciate that the quality of the full pipeline is better and similar to the results obtained applying the full pipeline adding a last finetuning step with depth. Figure 16 shows (a) ground truth image, (b) rendered image with real images only and depth supervision, (c) rendered image with depth supervision and fine-tuning, (d) rendered image without depth supervision in the main training pipeline with depth supervision added at the last training steps. All steps using depth supervision were trained with a weight of 0.85 for the depth supervision loss term

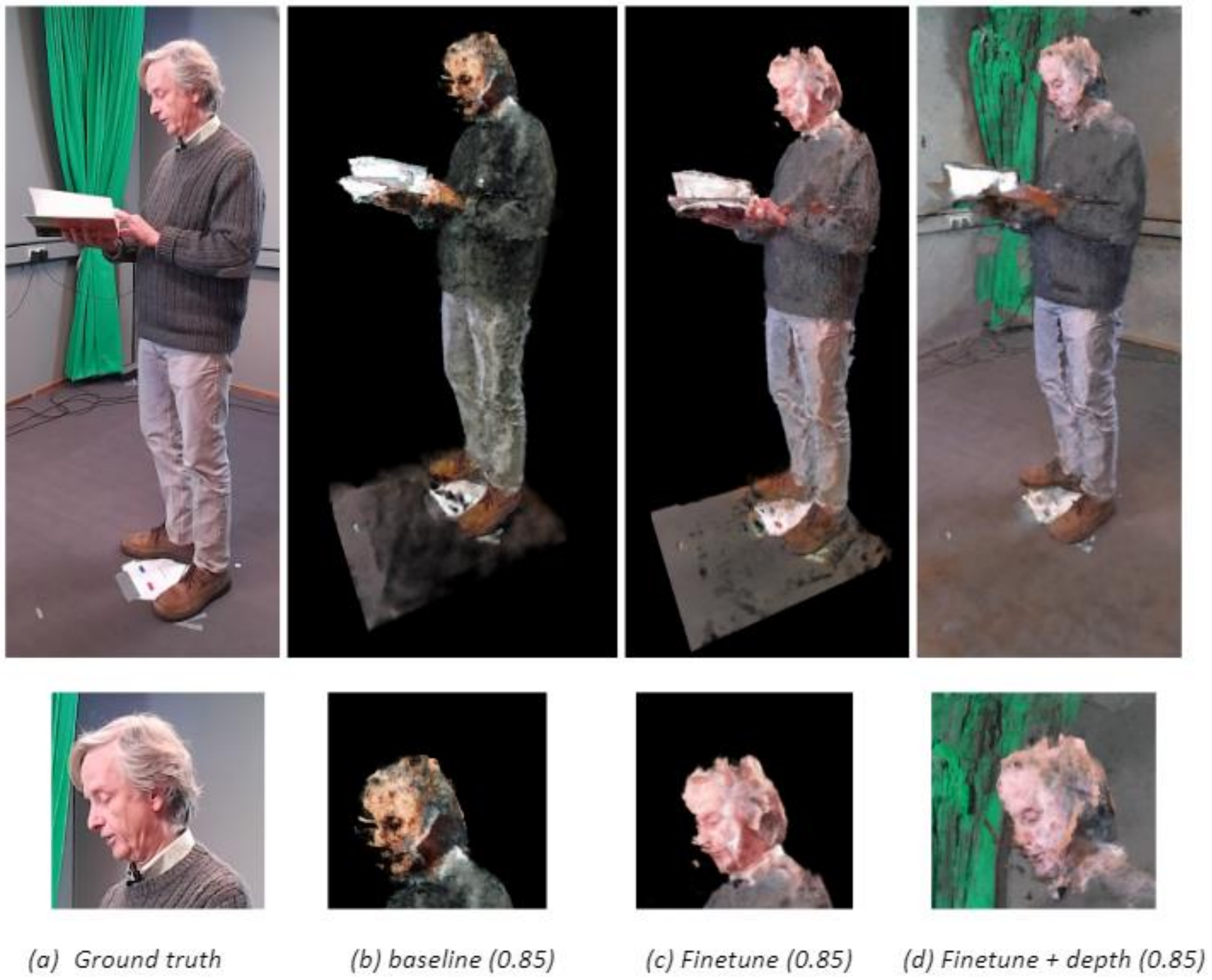


Figure 16: Visual results.

4.4.6 Generalizable depth-based NeRF

Another issue with NeRF algorithms is the considerable time required to train a model on a specific scene. Generalizable NeRF algorithms are proposed for alleviating long NeRF training times. In the following, we aim to integrate Generalizable NeRF algorithms with depth-based information. This not only enables us to work with a minimal number of images but also reduces the time required to fit a model on a specific scene. As discussed in Section 4.4.1.1, the approach closest to our objective is ENeRF.

ENeRF's architecture is presented in Figure 17. Given multi-view images of a static scene or a dynamic scene at one frame, a cost volume is constructed by a 2D Convolutional Neural Network (CNN). This is processed by a 3D CNN that outputs a 3D feature volume, as well as the coarse 3D geometry of the scene (represented by depth and confidence maps). The estimated geometry guides the ray marching sampling process, to sample points near the surface, which significantly accelerates the volume rendering process. Additionally, the 3D feature volume provides rich geometry-aware information, for generalizing radiance fields across different scenes.

During test time, the features for each sampled point are accumulated in order to estimate how the pixels of the input viewpoints will be blended. To that end, ENeRF is an IBR (Image-based rendering) method as the target image is a combination of pixels of the input views. We believe that this is the reason why ENeRF does not produce good quality results when it is trained on a sparse set of viewpoints. Not only ENeRF is limited in this matter, but also presents non-smooth transitions and many artefacts in scenes that do not have a dense camera setting.

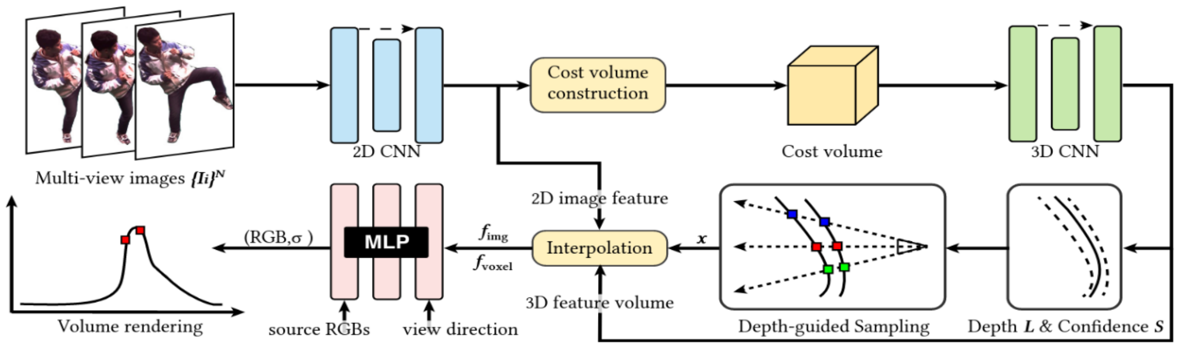


Figure 17: ENeRF architecture, for real time novel view synthesis.

Based on ENeRF limitations, we developed a novel method which we call Generalizable Depth-Based NeRF (GDNeRF). GDNeRF currently consists of the following improved components:

- Depth-based feature volume: Features are projected from the source views into grid cells based on a gaussian distribution which is centred at each viewpoints depth image (provided RGB-D cameras). The standard deviation is greater on depth discontinuities where the estimation might contain more errors.
- Multilevel feature volume: In order to model scene features at different scales we compute feature volumes at different scales so that they model pre-filtered features, similar to ZipNeRF.
- Depth-based supervision: Depth images can be used not only to compute a probabilistic feature volume but also to supervise the depth probability distribution for each ray. Additionally, we also supervise the estimated depth image, based on the density MLP of the NeRF model.
- ZipNeRF's antialiasing: ZipNeRF weights feature volumes at different scales based on the distance from the ray origin to the point sampled on the ray. Similarly, using the multilevel feature volumes, we weight each feature based on the grid cell space at each level and the approximate radius of the conical frustum at that point.

Figure 18 illustrates the overall process of our method. The first step is to render a coarse approximation of the scene, which will also provide an estimation for the corresponding depth images at a low-resolution level. This estimate will save a great deal of computational time in the training process, as the spatial dimensionality increases significantly and the computation without this estimation would take too much time for a real-time application.

First, we extract feature maps at different scales, which will be used depending on the step of the rendering process, i.e. if it is the coarse or the fine rendering. Then, feature maps are projected based on the depth estimated by the Kinect camera in a probabilistic manner. The feature volume is processed with a 3D CNN for which multilevel feature volumes are extracted. From these, a depth estimation for each ray in the target image is estimated. In each ray, samples are drawn according to this depth distribution. Then, ZipNeRF's feature fusion is applied to obtain prefiltered features. Finally, volumetric rendering is applied to obtain the final image. This process is repeated at a larger spatial resolution to obtain a fine render. The depth estimation from the coarse step is reused to define the bounds of the feature grid, hence aiding in the fine levels.

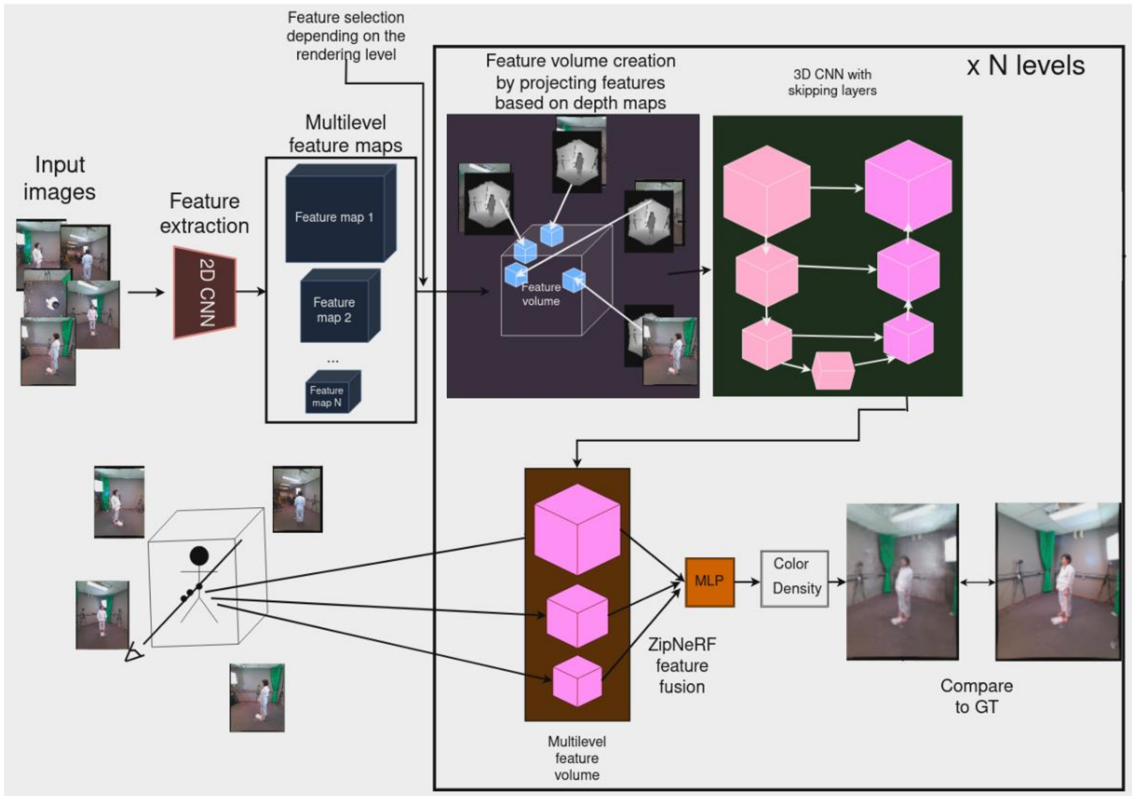


Figure 18: Overview of the GDNeRF approach.

4.4.7 Depth-based feature volume

The depth-based feature volume F is computed by defining a 3D grid in the target camera space C_{tar}. The feature F_{ijk} with respect to a source camera C_{src1} is computed by projecting a point in the grid x_{ijk} into the source image of reference. This point can be obtained by applying the corresponding camera matrix transformations. The feature of the source view at that point will be weighted depending on how close the depth at that point is to the known camera depth. Specifically, given the point x_{ij}, we project it to world coordinates X_{ijk}. Then, the

distance d_{ijk} from X_{ijk} to the camera origin osrc1is measured. The closer that d_{ijk} is to the ground truth distance of the projected point X_{ijk} in the source camera plane, the more the feature will be represented in x_{ijk}.
Specifically, the feature at F_{ijk} for the source camera C_{src1} will be:

$$F_{ijk} = N(d_{ijk};d_{ijksrc}, \sigma_{ijksrc})\, f_{ijksrc}$$

where d_{ijksrc} is the ground truth depth of the source camera C_{src1}, σ_{ijksrc} is the estimated standard deviation at that point on Csrc1 and fijksrc the feature of the corresponding point on C_{src1}.
This process is repeated for each of the M source cameras C_{src1}, ..., C_{srcM}. In order to obtain a combined feature volume from all views, we average all the feature volumes to obtain a single one with all the fused features. This makes intuitive sense, as the more one feature is observed from different points of view, the more consistent it should be. Features that are obtained from a single camera but are visible from few views also are well represented, as the depth will coincide with the source camera and a great weight will be assigned.
The depth-based feature volume is combined with the plane sweep volume from ENeRF in order to have information of features that are not visible in any view or for those for which we do not have a depth estimation.

4.4.8 ZipNeRF antialiasing

Similar to ZipNeRF, we weight the features of each level of the multiresolution hash encoding based on the distance from the ray origin to the sample on the ray. Specifically, given the mean cell space s_i of the grid level i, we follow the standard deviation p of the approximate gaussian that represents the visible frustum at that point p given by ZipNeRF. In ZipNeRF, p is approximated as:

$$p = d_p r / 2$$

where d_p is the distance from the camera origin to p and r controls the aperture of the conical frustum. We compute the corresponding weight of the feature at the grid level i based on their proximity. This weight is computed as:

$w_i = 1 /(p-s_i)2$

Weights are softmaxed in order to interpolate the prefiltered feature according to its scale.

4.4.9 Depth supervision losses

Apart from the losses that ENeRF employs, we additionally propose new supervision and regularization losses to improve the results of the new views. ENeRF uses the following losses:

- Photometric loss: L2 difference between the synthesized image and the ground truth image.
- Perceptual loss: Difference of feature maps extracted from the synthesized and ground truth image. Feature maps are extracted from a pretrained VGG network.

Additionally, in order to regularize the network training with the depth information that we have available, we incorporate the following terms:

- L1 depth supervision: In regions where we have a depth ground truth, we compare both the first depth estimation from the feature volume and the expected depth based on NeRF's particle density.
- DSNeRF depth distribution supervision: By using our approximation of the standard deviation of each pixel in the depth map, we regularize the distribution of weights so that the density concentrates around

the depth ground truth with the given standard deviation. Similar to DSNeRF, we minimize the KL-divergence between these two density distributions.
- Depth smoothness: Observing that in some cases NeRF tends to have some depth discontinuities, we also enforce that the depth in a local region is smooth by measuring the standard deviation in local patches.

4.4.10 Experimental results

For our experiments, we utilised a dedicated training server equipped with an NVIDIA RTX A5000. Throughout this period, we conducted a variety of experiments aimed at addressing various issues and debugging different components of our method. We started by trying to train INGP in a sparse camera setting of a custom recorded video. Figure 19 shows that it generates a cloud of inconsistent particles due to insufficient information of the scene.

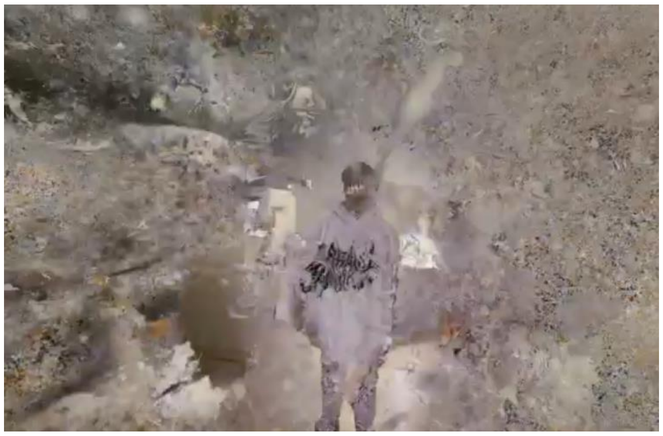


Figure 19: Qualitative results of INeRF trained on a sparse set of views (using NeRFStudio's implementation[65]).

When exploring the limitations of ENeRF, we generated a 360° video from 5 input views surrounding the subject of interest. Figure 20 depicts some of these frames. We can observe that as ENeRF is an IBR method, it is not well-suited for sparse scenarios. When synthesizing views that are close to one of the input views, ENeRF gives reasonable results, but when interpolating an intermediate point between two cameras it generates many artefacts as can be seen in the image on the right.

[65] Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., ... & Kanazawa, A. (2023, July). Nerfstudio: A modular framework for neural radiance field development. In ACM SIGGRAPH 2023 Conference Proceedings (pp. 1-12).

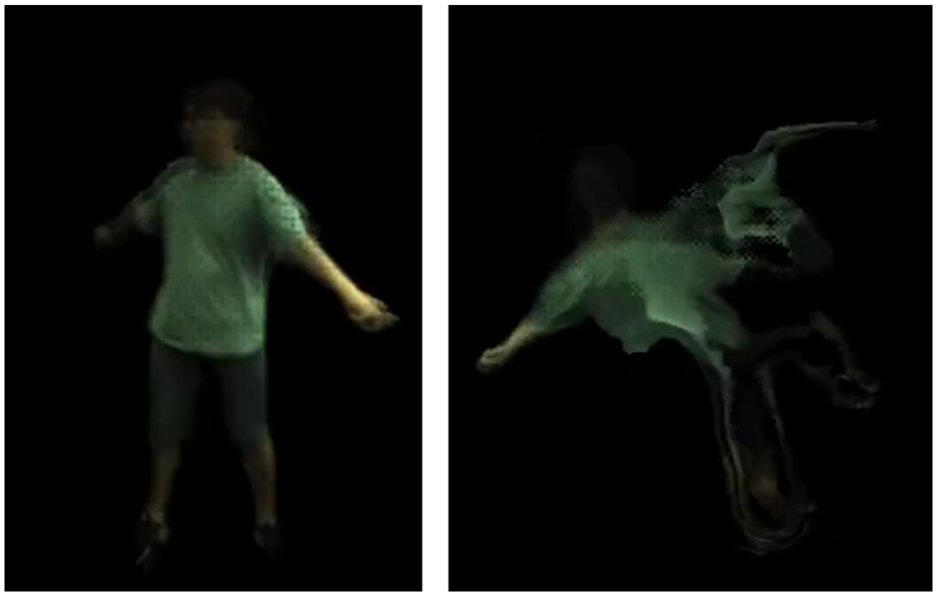


Figure 20: ENeRF experiments on ZJU-Mocap. Left: ENeRF results when the view is close to one of the five input cameras. Right: ENeRF results when the view is in the middle of two sparse input cameras.

In order to analyze the components of our approach, we decided to use the sequence *s20_karateka* from the CWI dataset. We use the first 300 frames for training our GDNeRF network for 100 epochs which takes approximately 1 day. After the training process, we perform a qualitative and quantitative analysis with the remaining frames of the video sequence (unseen during training). Figure 21 shows that our method can infer good results from 3 sparse views, providing more consistent geometry and texture. Note that both the RGB image and depth estimation with GDNeRF provides better results. After our initial trainings, we saw that some parts of the background were much more inconsistent compared to ENeRF, which made us perform an in-depth analysis of the components of our method. We performed several tests in order to determine the root cause of this issue, like using the ground truth depth map for the ray interval estimation. Nonetheless, it resulted in similar results as observed in Figure 22.

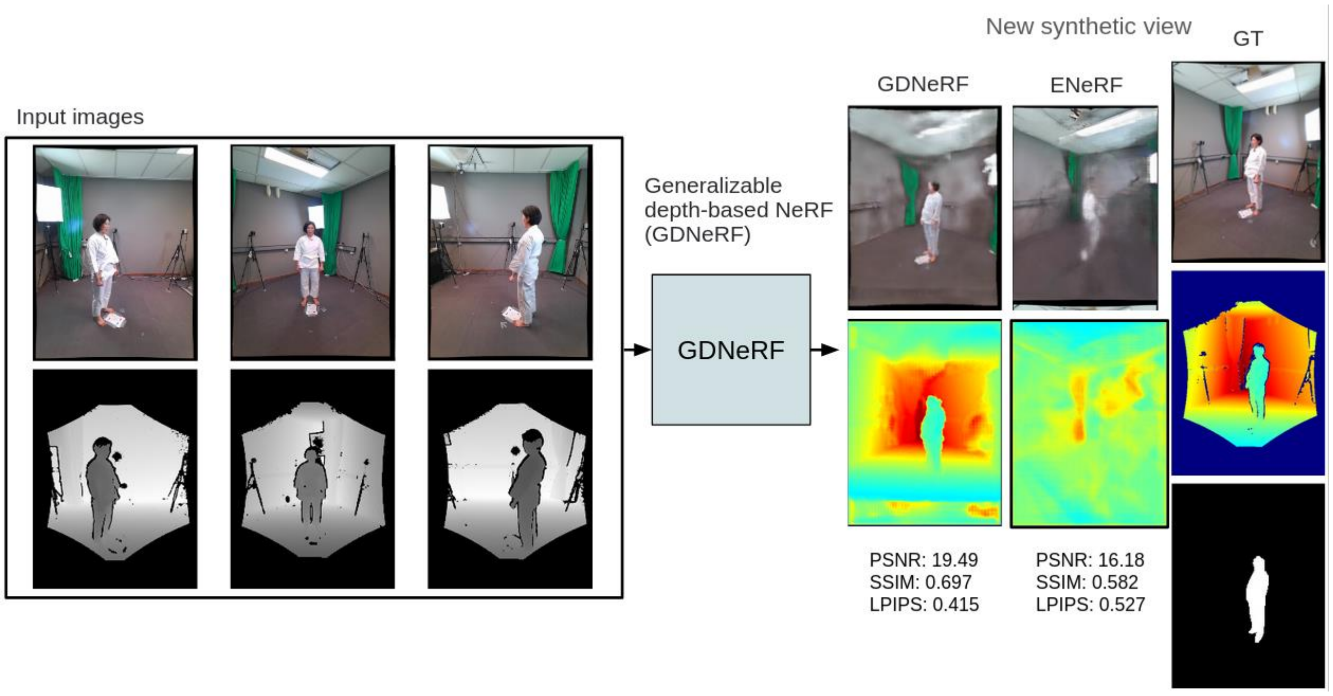


Figure 21: GDNeRF comparison with ENeRF in the sequence s20_karateja.

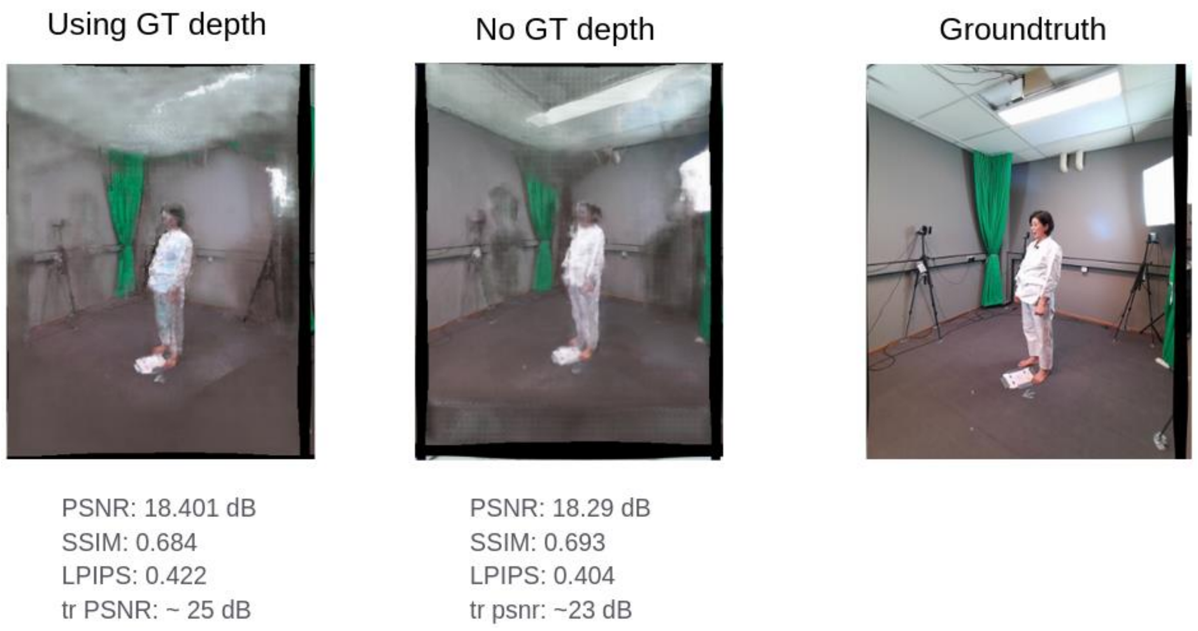


Figure 22: GDNeRF experiment comparing with the ground truth depth estimation in the target camera.

After evaluating the different components of the model, we realise there was an inconsistency in the scale of the feature maps that were projected from the coarse to the fine level, which resulted in the training not being able to correctly learn to represent the scene and generalize. We observed this after isolating the coarse network and observing that it was able to converge adequately with good results. Once the setting of the scale of each of the coarse and fine components was corrected, we obtained results that resemble the ground truth with much more fidelity. An example of the final results can be seen at Figure 23.

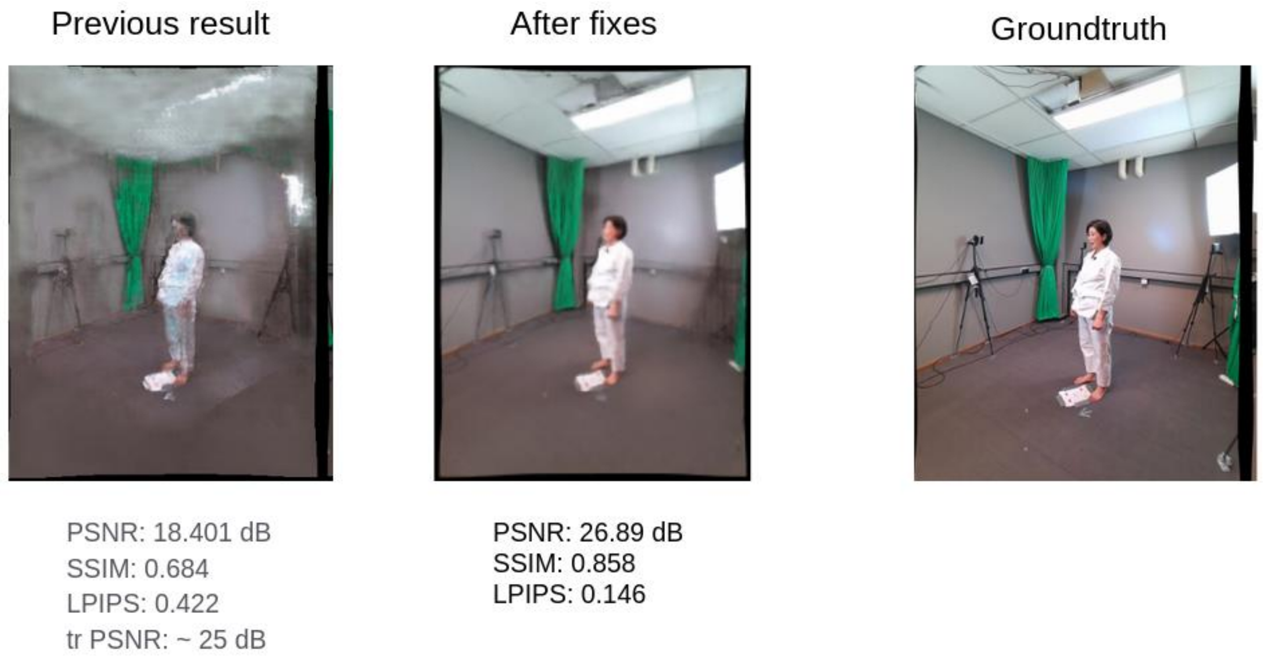


Figure 23: GDNeRF results after fixing the problem with scaling issues between the coarse and fine levels.

At this point we obtain a good reconstruction of the background and person from three sparse views. However, there are still blurring and minor artefacts in the occluded regions due to the sparsity of the input cameras. In the future, we plan to address these artefacts and increase the high frequency content by adding a generative component into the GDNeRF model.

4.5 NeRF from drone footage

Considering RAI's role as a user and content providing partner, the following describes the test results conducted employing INGP for the reconstruction of the Basilica di Superga captured from aerial footage. A RAI drone was used to capture detailed images of the building from different angles (Figure 24), providing a rich source of data for the algorithm. The uniqueness of the test lies in the fact that the video was not specifically created for the experiment; rather, RAI's archival footage was used to verify the goodness of the final result.

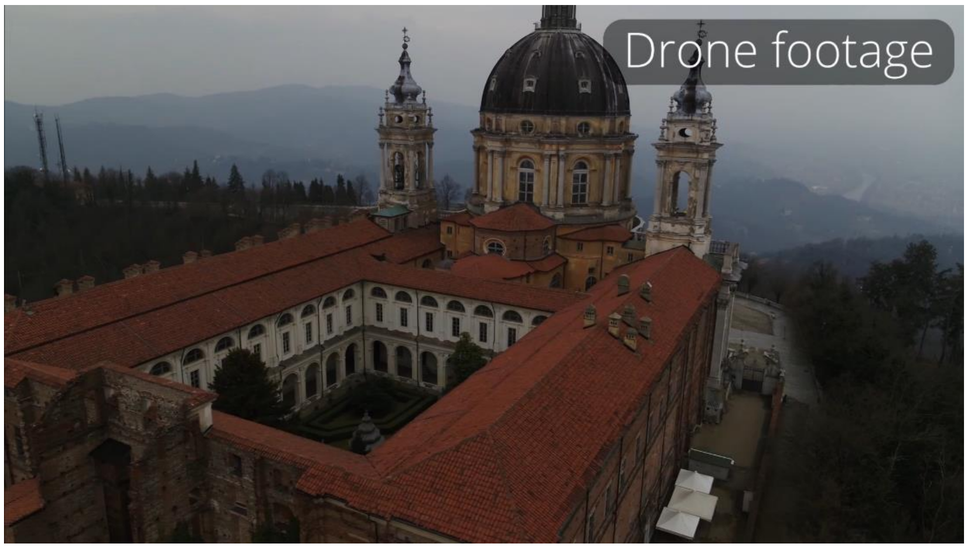
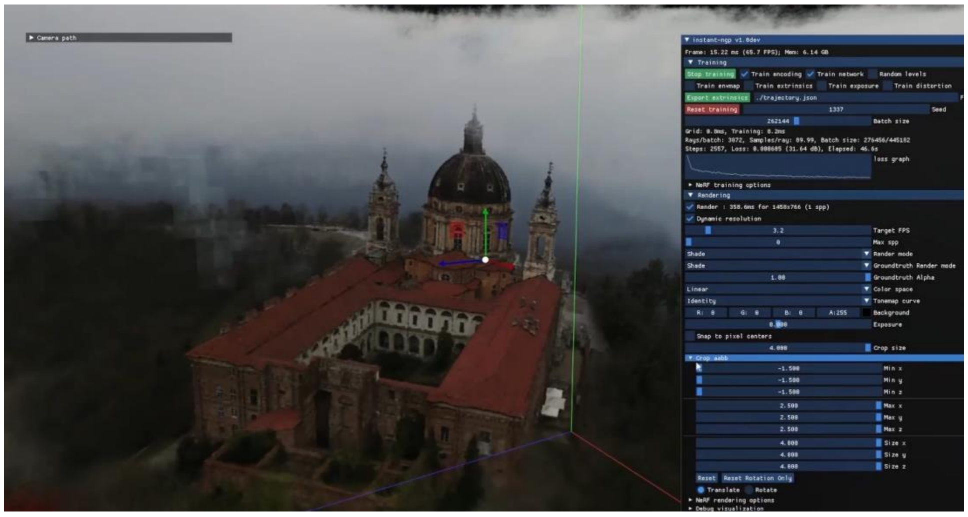


Figure 24: Drone footage of the Basilica of Superga

The system works on a set of analyzed images and to optimize the model reconstruction. Tests were performed by customizing the image processing parameters. Subsequently, the network was trained by working on the parameters to find the right balance between quality and rendering speed. As a first step, a crop was made directly on the area of interest to reduce the computational load by focusing on the subject. After obtaining an acceptable initial result, the frames per second (fps) parameter was adjusted to achieve a sharp and detailed reconstruction (Figure 25).

Once a detailed model was obtained, the network training was stopped, and a snapshot was saved for future customization.



```
▼ instant-ngp v1.0dev
Frame: 15.22 ms (65.7 FPS); Mem: 6.14 GB
▼ Training
Stop training  ✓ Train encoding  ✓ Train network  Random levels
Train envmap  Train extrinsics  Train exposure  Train distortion
Export extrinsics  ./trajectory.json
Reset training  1337  Seed
262144  Batch size
Grid: 0.0ms, Training: 8.2ms
Rays/batch: 3072, Samples/ray: 89.99, Batch size: 276456/445182
Steps: 2557, Loss: 0.000685 (31.64 dB), Elapsed: 46.6s
loss graph
► NeRF training options
▼ Rendering
✓ Render : 358.6ms for 1458x766 (1 spp)
✓ Dynamic resolution
3.2  Target FPS
0  Max spp
Shade  Render mode
Shade  Groundtruth Render mode
1.00  Groundtruth Alpha
Linear  Color space
Identity  Tonemap curve
R: 0  G: 0  B: 0  A:255  Background
0.000  Exposure
Snap to pixel centers
4.000  Crop size
▼ Crop aabb
-1.500  Min x
-1.500  Min y
-1.500  Min z
2.500  Max x
2.500  Max y
2.500  Max z
4.000  Size x
4.000  Size y
4.000  Size z
Reset  Reset Rotation Only
● Translate  Rotate
► NeRF rendering options
► Debug visualization
```

Figure 25: NeRF training phase

The results obtained were very interesting (Figure 26). The Basilica di Superga was reconstructed with great precision, allowing for a final render usable in television contexts. The 3D model requires minimal processing as the obtained mesh is not geometrically perfect. However, by working on resolution and density parameters, usable models can be achieved, even in the context of Virtual Production.

Successfully conducted experiments on historical buildings like the Basilica di Superga highlight the potential of this technology for future applications. Detailed modeling could be used for educational purposes, cultural heritage preservation, and the creation of realistic digital scenarios.

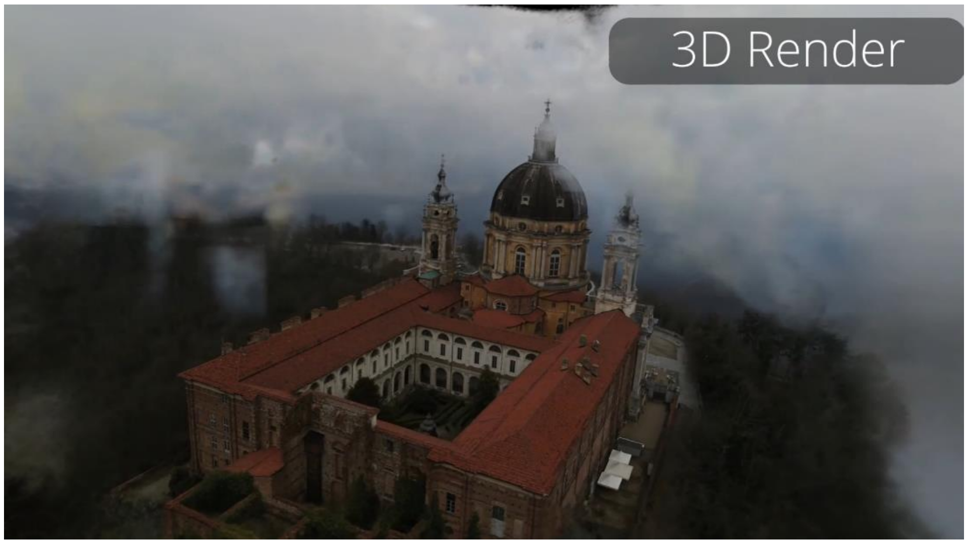


Figure 26: NeRF 3D Reconstruction.

4.6 Network acceleration infrastructure

XReco's architecture considers cloud computing solutions as underlying fabrics, where NeRF calls are made and executed on GPUs. Video or image-based flows are then transferred over the network, towards storage, consumption, or training. Hence, there is clearly three tiers of XReco developments, which are addressed concurrently:

- The user perspective – how to use XReco's solutions.
- The backend perspective – how data is generated, used, and managed.
- The infrastructure perspective – how XReco's systems are deployed on the infrastructure.

The following figure provides a graphical overview of these three areas.

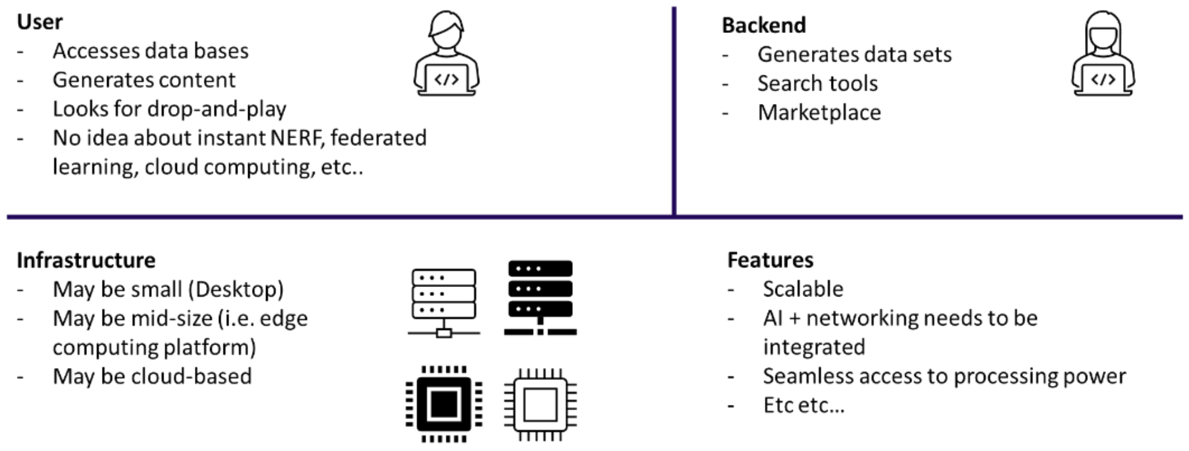

User
- Accesses data bases
- Generates content
- Looks for drop-and-play
- No idea about instant NERF, federated learning, cloud computing, etc..
Backend
- Generates data sets
- Search tools
- Marketplace
Infrastructure
- May be small (Desktop)
- May be mid-size (i.e. edge computing platform)
- May be cloud-based
Features
- Scalable
- AI + networking needs to be integrated
- Seamless access to processing power
- Etc etc...

Figure 27: XReco areas of development

This subsection develops the area of infrastructure development in XReco, which inherently relates to network acceleration. As processing power is readily available and can be scaled up, the challenge for users and backend is normally on how to experience low latency response on the processes. Network acceleration is crucial when dealing with image or video-based streams as in XReco for several reasons, primarily to ensure a smooth and efficient data transfer process. Here are some key reasons why network acceleration is important in this context:

1. **Large Data Volume**: Image and video files are typically large, especially when dealing with high-resolution content. Accelerating the network helps in reducing the time it takes to transfer these large files, ensuring a faster and more responsive streaming experience.
2. **Real-time Requirements:** Video and image streams often require real-time delivery, especially in applications such as video conferencing, online gaming, or live streaming. Network acceleration helps minimize latency, ensuring that the content reaches the viewer with minimal delay.
3. **Bandwidth Optimization**: Efficient network acceleration techniques can help optimize bandwidth usage. By compressing and optimizing the data being transmitted, you can reduce the amount of bandwidth required for streaming, making it more feasible for users with limited bandwidth or in situations with network congestion.
4. **Quality of Service (QoS)**: Accelerating the network contributes to maintaining a consistent Quality of Service. This is critical for applications where the quality of the image or video stream directly impacts the user experience. QoS mechanisms ensure that the content is delivered with the necessary speed and reliability.
5. **Adaptive Streaming**: Many video streaming services use adaptive streaming protocols that adjust the quality of the stream based on the viewer's network conditions. Network acceleration supports seamless transitions between different quality levels, ensuring a continuous and uninterrupted streaming experience as the network conditions change.
6. **Content Delivery Networks (CDNs)**: CDNs leverage network acceleration to distribute content across multiple servers strategically located around the world. By caching content closer to the end-users, CDNs reduce latency and enhance the overall streaming experience.
7. **Global Accessibility**: In the case of video or image streaming services that have a global audience, network acceleration becomes essential for delivering content efficiently across diverse geographic

locations. This helps in overcoming the challenges posed by long-distance data transmission and varying network conditions.

The implementation of the NeRF routines (T4.2) is done employing kubernetes[66] (aka K8s), which is a powerful container orchestration platform, enabling to drop specific NeRF workloads to a computing cluster efficiently. This is necessary when distributing AI pipelines over multiple compute nodes, as quite often huge models do not fit the memory and compute resources of a single node. One of the challenges in deploying K8s workloads is how to minimize latency when distinct nodes are sharing memory information. This occurs very often when training or inferring models, as data is constantly IO from the system (read/write operations). The following figure shows how in a server-client relation operating on a pod (which can be any type of processing platform), network interface cards (NIC) interconnect the units ideally through remote direct memory access (RDMA), thereby enabling low-latency memory exchanges.

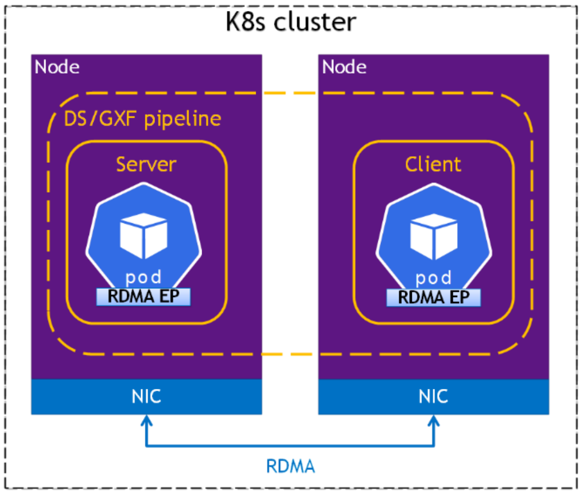


Figure 28: K8s cluster

In the context of XReco, we are considering employing the unified computing framework (UCF)[67] in order to build AI-driven video/image content pipelines. The following figure provides a high-level step-by-step diagram of how the UCF generate pipelines, containers registries, microservices registries, an application registry and finally drop on the processing infrastructure. The main components that drive this generation are:

- Graph composer: a GUI based application to create an AI graph app (drop-and-play approach for quick pipeline composition)
- Microservice builder: an aggregation engine for containers/graphs with endpoints (servers and clients) and configurable parameters
- Application builder: a tool to build UCF applications from UCF microservices.

[66] https://kubernetes.io/
[67] https://developer.nvidia.com/ucf

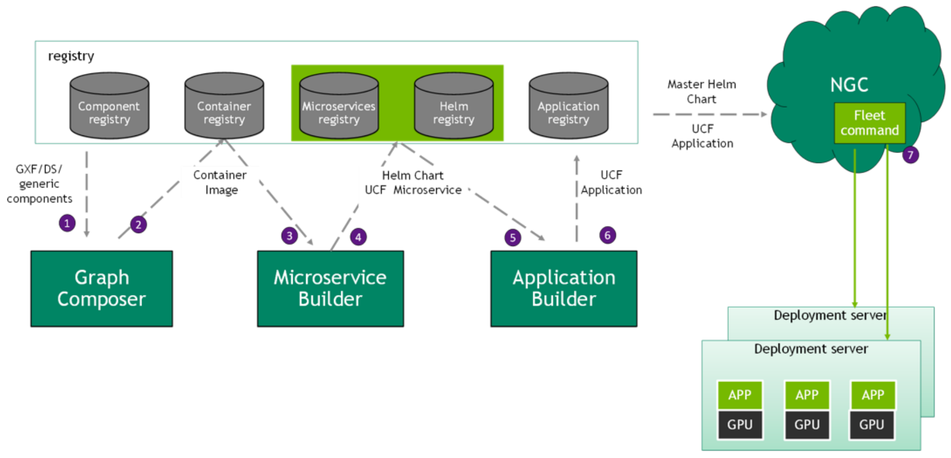


Figure 29: UCF pipeline

The following figure shows how the UCF application builder highlighted in the previous figure combines with the NeRF calls in XReco and the underlying K8s clusters for data processing. This architecture allows to decouple the different layers (service, application, and fabric), hence allowing for scalability.

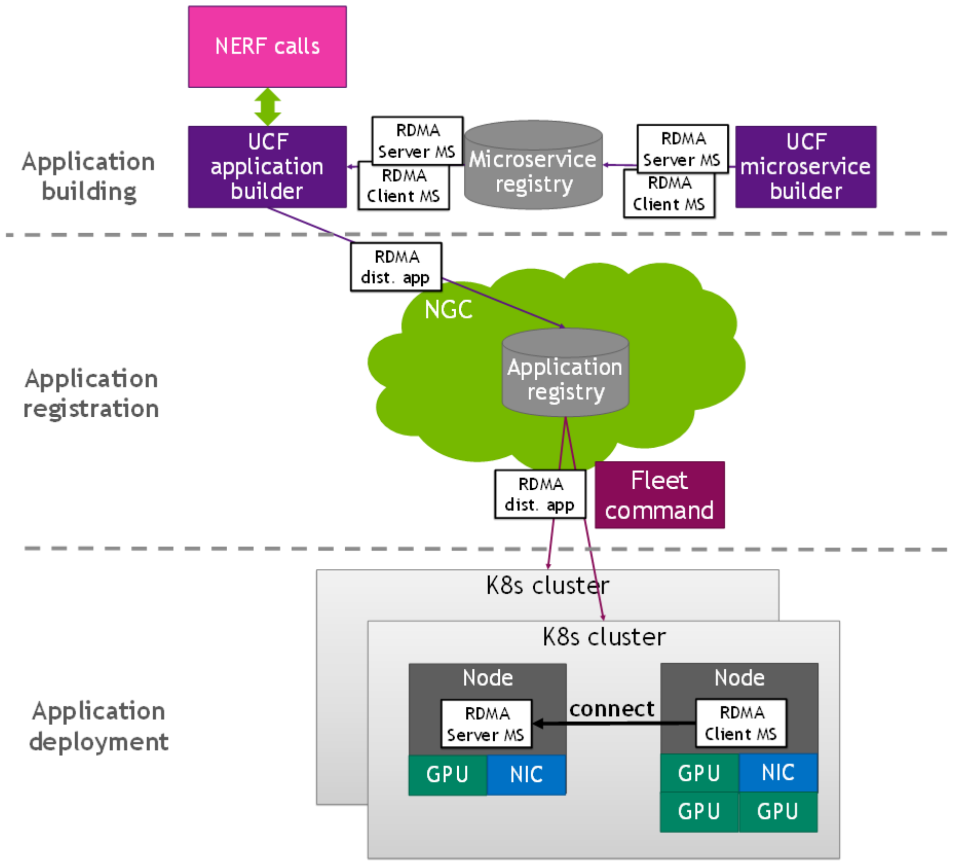


Figure 30: Overall XRECO architecture in relation to K8s

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

In XReco, we have developed an approach to enable the deployment of UCF pipelines on K8s, while maintaining RDMA application deployment for memory sharing. The following figure highlight this new methodology.

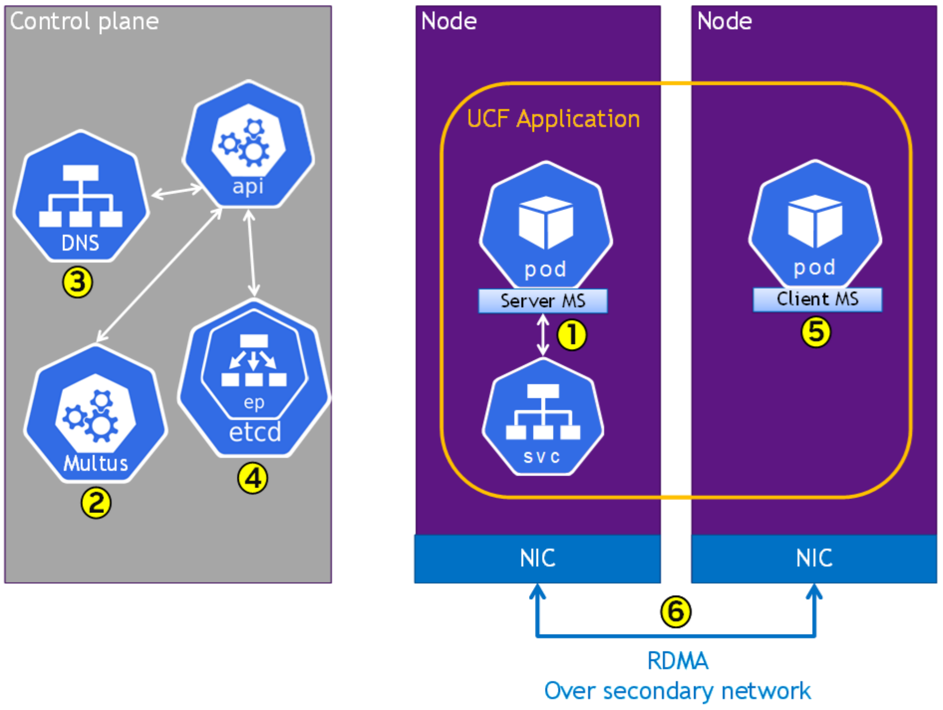


Figure 31: Headless service deployment

The methodology involves the following steps:

1. Expose the server as a headless server.
2. Deploy multi-user-service-controller to track the secondary network IP addresses and save it in the end-point-slice.
3. Server client's init container queries the DNS server.
4. DNS server returns all IP addresses in the end-point-slice.
5. Init container filters the non-RDMA IP addresses and picks one address from the remaining IPs.
6. Server client connects to the server with the chosen destination IP.

In the context of XReco, a small cluster is being built at CERTH with network interface cards and GPUs. The following figure shows the basic schematic of the setup, which allows for scalability if desired.

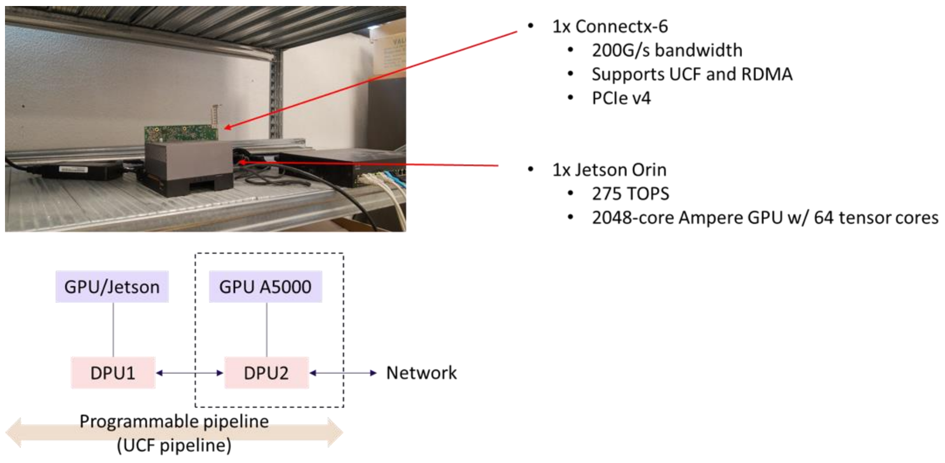


Figure 32: CERTH setup

5 3D asset aggregation and optimisation

5.1 Overview

The following section provides algorithms for Structure for Motion (SfM) and its extension in in-the-wild scenarios, enabling the aggregation of 2D images from different sets into standard 3D triangle-based models. Additionally, it provides an overview of super-resolution algorithms that target both 2D and 3D super-resolution. These can be used as stand-alone techniques, or to enhance the inputs of other algorithms (e.g., enhancing the resolution of training samples for NeRF or SfM algorithms).

5.2 Structure from Motion in-the-wild

5.2.1 Traditional 3D reconstruction pipeline

3D scene reconstruction methods take as input an unordered set of 2D images of the scene, captured from different viewpoints and yield as output some kind of 3D model representing it. Modern 3D scene reconstruction techniques based on NeRF do not produce an explicit 3D model, but traditional ones such as ours do: they yield a classic 3D triangle mesh representing the shape of the surface, together with a texture map capturing its appearance. To do so, they start by building a cloud of relevant 3D points of the scene, which then become the vertices of an initial triangle mesh, which is normally simplified before being textured. The first step of the traditional 3D reconstruction pipeline consists therefore in obtaining a 3D point cloud, and this is typically done by detecting as many relevant FPs (Feature Points) as possible in the 2D images, and then matching FPs from different images that correspond to the same 3D point. Solving this problem, known as SfM (Structure from Motion), enables to infer simultaneously the location of the 3D points and the pose (location and orientation) of the cameras used to capture the 2D images.

Several methods exist for detecting and describing FPs, of which the most widely accepted is SIFT (Scale-Invariant Feature Transform)[68], and there are several SfM techniques and public/free applications such as VisualSFM[69]. There are even complete MVG (Multiple View Geometry) libraries such as openMVG[70], which not only solve the FP detection and matching problem, but also implement the whole SfM pipeline summarized above. In the traditional 3D reconstruction method, we have implemented for XReco, we have carefully handpicked some openMVG tools, but complemented them with our own implementations.

5.2.2 Generation of denser 3D point clouds using A-KAZE

SIFT is quite robust against perspective distortions and illumination changes, but yields point clouds which are often too sparse for reliable surface recovery, so we also use an alternative FP extractor, A-KAZE[71], that outputs significantly more points in every image, therefore yielding more detailed point clouds, which can later be meshed with significantly better results. The performance of the A-KAZE FP extractor drops significantly in shaded areas, but this can be alleviated using local contrast enhancing algorithms such as CLAHE[72] or, even better, MSRCR[73] on the source images before FP extraction.

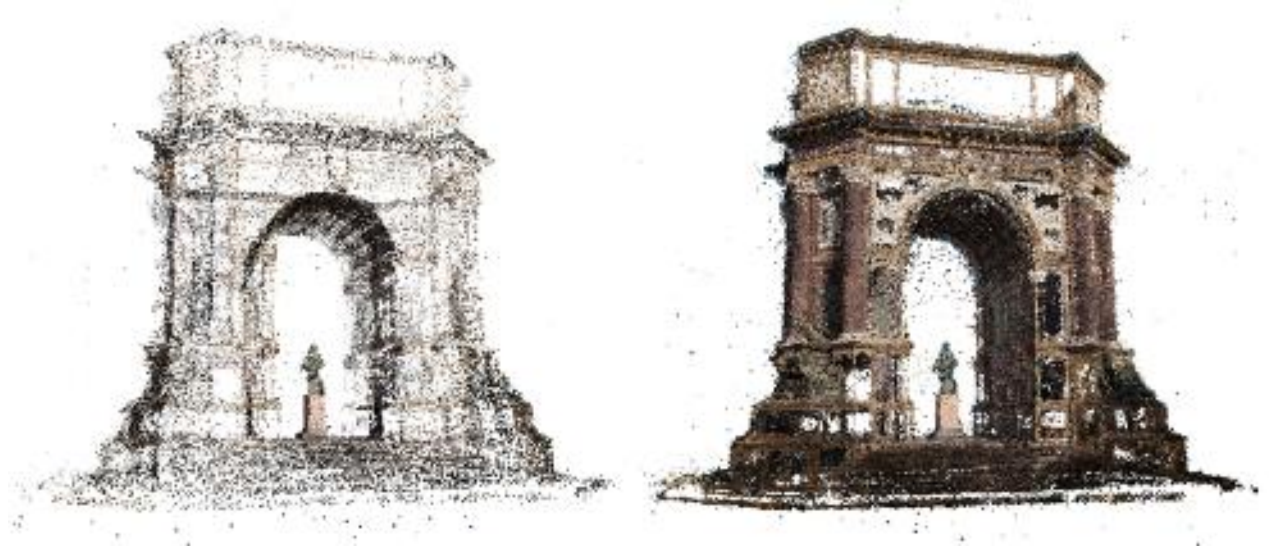


Figure 33: Comparison between SIFT (left) and A-KAZE (right)

[68] 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.

[69] http://ccwu.me/vsfm/

[70] https://openmvg.readthedocs.io/

[71] 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/).

[72] 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.

[73] 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.

Figure 33 compares a sparse point cloud of Turin's Arco Valentino obtained with SIFT FPs (left) vs. a much denser one obtained with A-KAZE FPs (right), and enhanced input images using the MSRCR algorithm to facilitate detection in shaded areas.

In addition to balancing the results between sunlit and shaded areas, the enhancement of source images produces the side effect of dramatically increasing the number of detected FPs. As a rough approximation, we have found photos of buildings to yield ~5k FPs using SIFT (the exact figures vary depending on the texture and shape of the subject) and 5-6 times as many, using A-KAZE on original images. On top of that, by enhancing the original images with MSRCR, we have managed to obtain an extra ~8x, and thus a total factor of 40-50x. It is worth noting that MSRCR-enhanced images do not seem to provide any significant increase of the number of FPs detected by SIFT.

This 40-50x increase is however a mixed blessing: while it is most desirable in terms of point cloud density, it has a very significant negative impact in computational demands, because exhaustive pairwise matching between two sets of FPs requires quadratic time on the number of points. In addition, the number of possible pairs of images in a set also grows quadratically on the number of images. Therefore, we have focussed on performance optimization to alleviate the problem and cut down on processing time, both trying to reduce the number of pairs of images to be matched and the per-pair computational cost.

5.2.2.1 Prioritisation of pairs of images

In the absence of any previous information, FPs should be extracted on every input image and matches sought between all possible image pairs. However, many pairs will have no overlap (for not depicting the same portion of the object) and therefore the computation time associated with them will be wasted, so the aim is to not even to consider them. This problem is also present when using SIFT, however it is much less severe because each image has significantly fewer points. The key observation is that both the intrinsic and extrinsic parameters of the cameras are well estimated using SIFT and A-KAZE, therefore we can obtain a sparse point cloud using SIFT and use it as a statistic sample of sorts to decide what image pairs are relevant and should be computed with A-KAZE to contribute to the detailed point cloud.

The results of the SfM module mentioned in Section 5.2.1 include the camera parameters, the locations of the 3D points and, even more crucially in this context, their projections onto the appropriate source images or, more precisely, the 2D FPs that originated from each 3D point. Thus, we can estimate the amount of overlap between two images by simply counting the number of points projected onto them. If two given images share no points, we can safely discard that image pair for lack of (relevant) overlap. For each image, we count the number of FPs it shares with each of the other images, and then sort them in decreasing order, so that we can compute matches against the images with the most significant overlap first, and discard pairs with negligible overlap.

Finally, pairwise matches can be computed in either depth-first order (i.e., for each of the images we compute matches with all its significant pairs) or breadth-first order (i.e., we compute the most significant pairs for every image first, then the second most significant, and so on). Best results are obtained if all significant pairs are computed; but if the user cannot afford to wait, breadth-first order gives the best results for an allotted time, and if the user is not satisfied with those results, she can resume the computation to refine the cloud.

5.2.2.2 Reduction of per-pair computational cost

The standard procedure to compute FP matches between two images is to exhaustively try to find the best match between all FPs from image A and all FPs from image B, which results in quadratic complexity, then estimate a geometric model to reject outliers (estimation of the fundamental matrix F[74] with RANSAC[75]). The bulk of this pairwise FP matching procedure is completely regular and thus amenable to efficient parallelisation, which we have implemented both on CPU and GPU. However, its inherent complexity is anyway excessive, therefore algorithmic optimisation is required to further reduce processing time.

As previously stated, while point clouds yielded by SIFT are too sparse for reliable meshing, camera pose estimation and intrinsic parameters are essentially well estimated, which means two things:

1. There is no need to estimate camera projection matrices again when using denser FPs. This simplifies the SfM phase of the 3D reconstruction, which must only optimize the location of 3D points from their projections, which is significantly faster than the joint estimation of camera parameters and point locations.
2. We can obtain the fundamental matrix F for any pair of cameras from their projection matrices, which enables us to perform smarter matching, as we will see next.

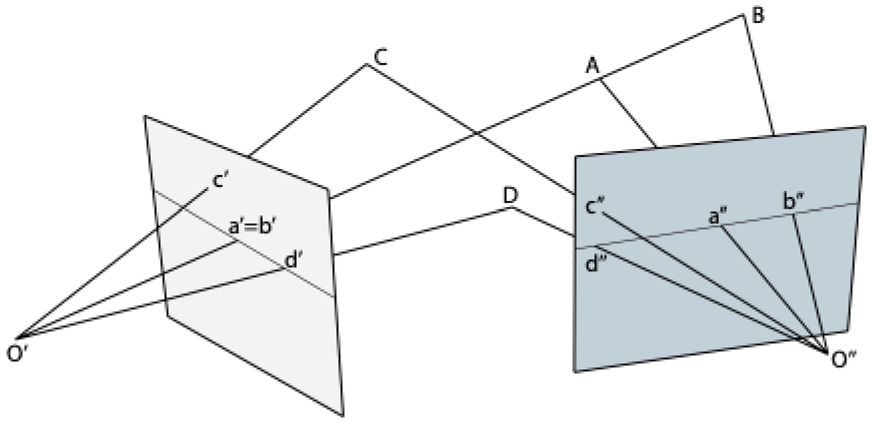


Figure 34: Epipolar geometry

Epipolar geometry helps rule out incompatible projections. The idea is that only projections that lie on the plane determined by the centres of both cameras and the projection whose match is to be determined are geometrically possible. Thus, a’ is geometrically compatible with a’’, b’’ and d’’, but never with c’’.

Since we now have the F matrix, we do not need to compute all potential pointwise matches, we can first check whether the locations of the candidate FPs are viable as illustrated in Figure 34, and only if they are we proceed to compute the distance between their descriptors. This approach results in more than a 10x speed increase in the CPU, even factoring in the locations of the points in the pointwise comparison.

[74] Q. T. Luong, O. D. Faugeras, “The fundamental matrix: Theory, algorithms, and stability analysis”. Springer IJCV, vol. 17, p. 43–75, Jan. 1996. DOI: 10.1007/BF00127818.
[75] M. A. Fischler, R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. Communications of the ACM, vol. 24, no. 6, p. 381–95, Jun. 1981. DOI: 10.1145/358669.358692.

Strictly speaking, this procedure still has quadratic time complexity, but we have exchanged for most FPs a data- and compute-intensive computation (comparing descriptors) for a lighter one (vector product to compute epipolar distance). On a CPU, this can be parallelised reasonably well by mapping one execution thread to each FP in image A and iterating through all the FPs in image B, so that there are no possible race conditions. Unfortunately, such a naive mapping does not work so well on a GPU, because execution threads are grouped into warps that must execute the same instructions synchronously and, for maximum efficiency, reuse or coalesce memory transactions. The solution is to make use of the fact that in a GPU we have some degree of control in thread scheduling. Thus, instead of using one flat list of FPs per image, we divide the images into tiles and make one list of FPs per tile, effectively grouping neighbouring FPs which will roughly satisfy geometric constraints on a per-tile (vs. per-point) basis. This results in two benefits: all FPs located in tiles that do not satisfy geometric constraints can be disregarded in a single operation (Figure 35), and all FPs from tiles that do satisfy them are relevant candidates to match all other FPs mapped to the same thread block, enabling shared reading and making much more efficient use of memory bandwidth.

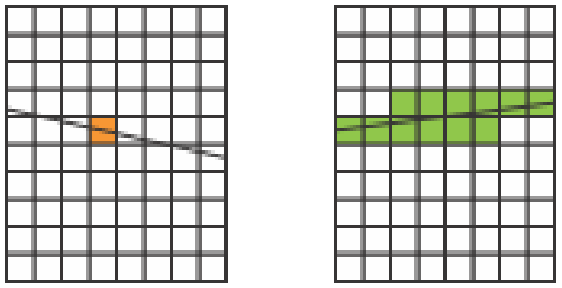


Figure 35: Grouping neighbouring FPs

All FPs in the orange tile (left) can only find their matches roughly around the corresponding epipolar line, i.e., within the green tiles (right), so all FPs outside the green tiles may be ignored.

5.2.3 3D mesh reconstruction

5.2.3.1 Poisson surface reconstruction and initial 3D mesh cleaning

Once we have the 3D point cloud and the corresponding normal vectors (to the inherent surface) correctly estimated, we can perform a Poisson surface reconstruction stage, which yields a 3D mesh. A clear advantage of Poisson meshing is that it closes any potential holes in the resulting surface, which could result from low density areas in the point cloud, due to glossy surfaces, bad illumination conditions, etc.

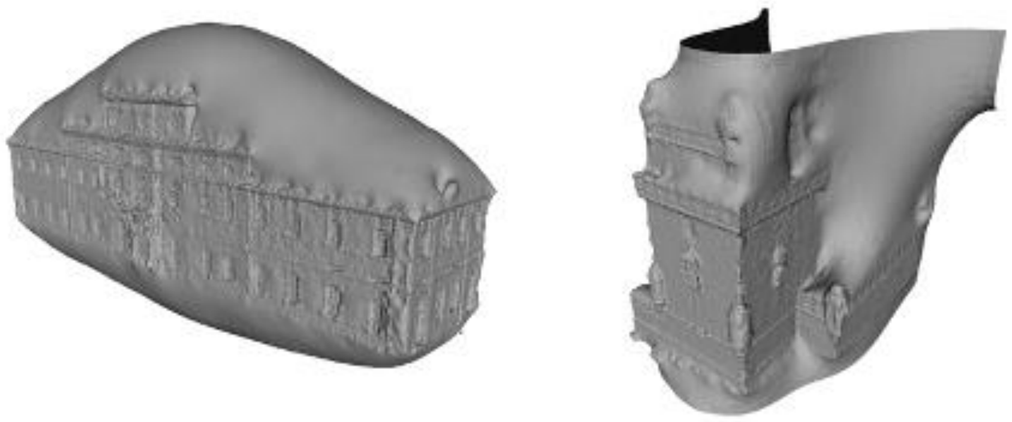


Figure 36: Two different results from the Poisson surface reconstruction stage run on point clouds of Turin's Villa la Tesoriera (left) and Lisbon's Torre de Belém (right).

On the other hand, areas that are not covered at all, such as roofs or hidden walls, will also be closed with a big "bubble", as illustrated by Figure 36-left. These other surfaces should be removed, as they should not be part of the final model. Besides, although 3D meshes yielded by Poisson surface reconstruction are normally watertight, sometimes they are not: see Figure 36-right.

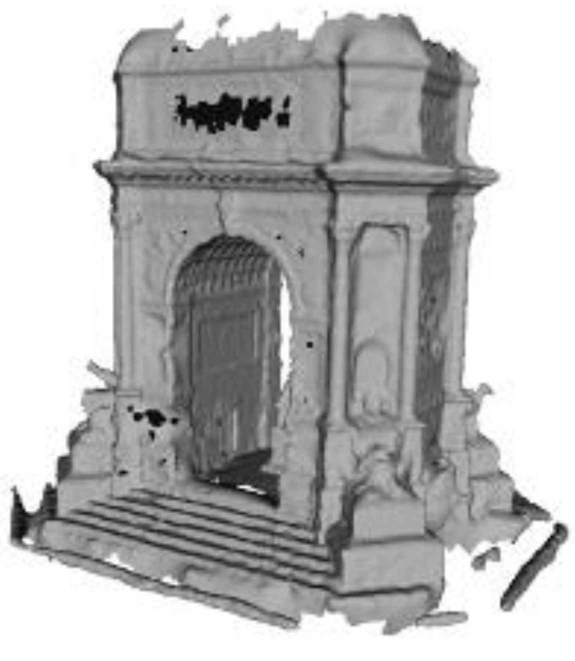


Figure 37: Result of a naïve mesh cleaning process run on a model of Arco Valentino.

To remove these unwanted mesh areas, state-of-the-art systems calculate the minimum distance from each vertex of the mesh to the point cloud and remove vertices further away than a certain threshold. This technique might work with very dense point clouds produced with laser scanners or similar devices, however, when dealing with point clouds originated through SfM, it can delete more triangles than desirable. This is due to the nature of SfM: for example, both areas of the 3D model without high frequency components in their texture (e.g., a plain-coloured wall) and shiny surfaces yield few FPs in the corresponding images, and therefore few 3D points, which in turn leads the Poisson surface reconstruction stage to carve a hole where there should not be one. This is the case in Figure 38, where we can see a big hole in the top area of the arch, due to the lack of detectable FPs in the marble plank.

Another problem we have had is that the reconstruction stage creates sometimes several isolated mesh components for each individual model. This is because there are areas in the point cloud which do not correspond to the particular building we are reconstructing, but to nearby buildings which are unavoidable when taking pictures of the central subject. As these components are not important, we keep the one with highest number of polygons and discard the rest.

To remove only unwanted triangles in the model, we analyze the boundary of the mesh, i.e., the set of edges belonging to only one triangle. If the model is closed, and represents a reconstructed building, we remove the triangles in the mesh boundary corresponding to the roof and ground, thus creating a one-triangle hole. Then we check the distance from each edge in the boundary to the processed point cloud and, if it is greater than a certain threshold, the corresponding triangle is deleted. This threshold is determined by the average distance among points in the cloud. To reduce the computation time, we can downsample the point cloud by placing a 3D voxel grid over the input point cloud and approximating by their centroid all points contained in each voxel. This approach is a bit slower than approximating them with the center of the voxel, but it represents the underlying surface more accurately. This process assumes there will only be unwanted surfaces in the roof and in the ground of the reconstructed models. As we iterate through the boundaries of the mesh, areas with a low count of 3D points, such as windows, are kept closed.

5.2.3.2 3D mesh smoothing and simplification

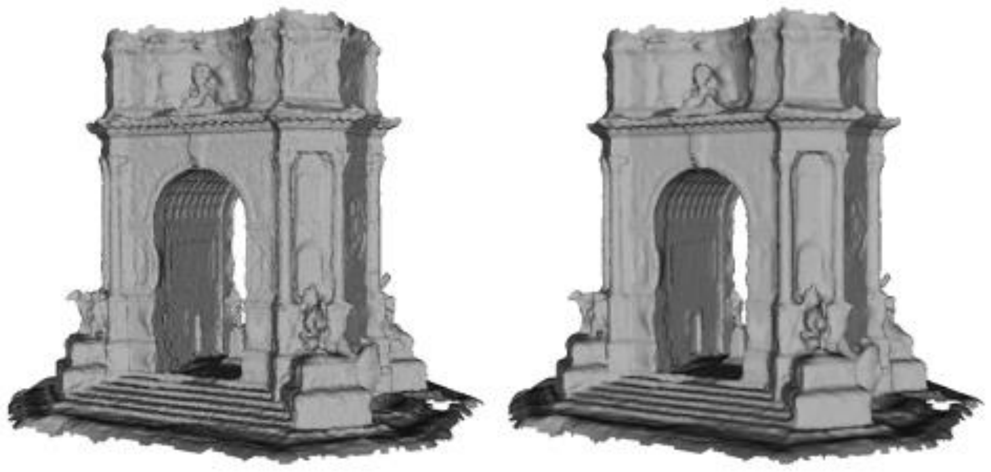


Figure 38: Cleaned 3D meshes of Arco Valentino before (left) and after (right) the Laplacian smoothing stage.

To ensure a good result, the Poisson surface reconstruction stage normally assumes a high octree depth. This means that the resulting mesh will resemble the point cloud accurately and have many very small triangles. On the other hand, as the point cloud will probably present some noise, the resulting mesh will also be noisy. To reduce this noise, a Laplacian smoothing stage is performed, as illustrated by Figure 38.

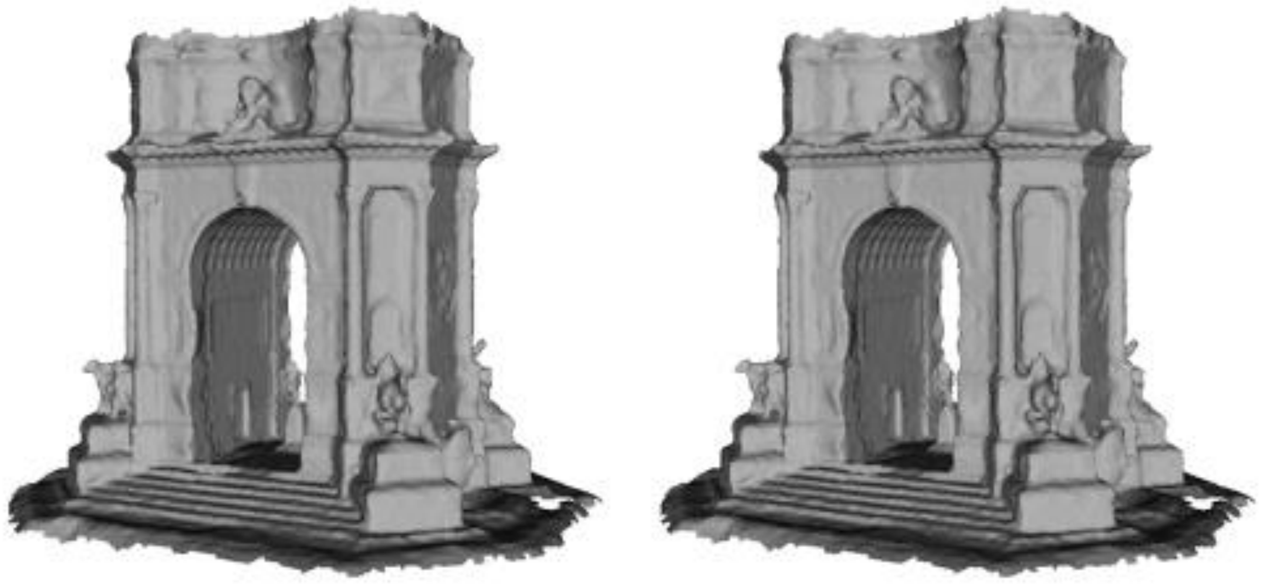


Figure 39: Two meshes representing Arco Valentino with ~200k (left) vs. ~20k (right) vertices.

Finally, we apply a decimation process based on QEM (Quadric Error Metrics)[76] to make sure that simpler (i.e., flatter) mesh areas are represented with fewer triangles while mesh corners and details are preserved. Figure 39 shows how the resulting mesh can have a much lower triangle count but still represent accurately the original shape.

5.2.4 3D mesh seamless, static multi-texturing

A) Colour blending

To obtain a realistic result, the 3D mesh undergoes a multi-texturing process[77] which yields a seamless texture atlas calculated by combining the colour information from the set of images used for the reconstruction. We suppress the colour seams due to image misalignments and irregular lighting conditions that multi-texturing approaches typically suffer from, while minimizing the blurring effect introduced by simpler colour blending techniques.

The key idea of this system is very different from other state-of-the-art techniques, which either search for discontinuities and iterate to reduce them or blend the colour information provided by the images matching areas with similar colour. In our case, each camera is ranked with respect to how well it "sees" each triangle (using the projected area of the triangle onto the image and the normal of the triangle with respect to the pointing direction of the virtual camera).

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

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

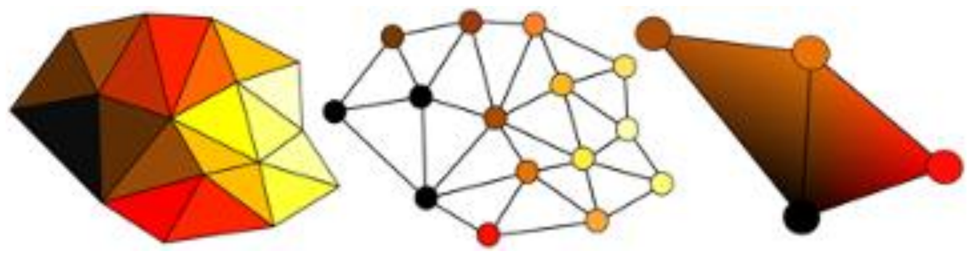


Figure 40: Two meshes representing Arco Valentino with ~200k (left) vs. ~20k (right) vertices

After assigning a rating per camera and triangle, we calculate a rating per camera and vertex by averaging the ratings of triangles surrounding each vertex (Figure 40). The final stage interpolates the ratings inside the triangle. This way, we create a seamless texture across the model, where each triangle has contribution from more than one image.

To save storage/transmission information, we create a texture atlas by unwrapping the 3D model into 2D charts. The colour contribution for each pixel of the atlas is calculated using the following expression:

$$Clr_i = \frac{\sum_j^N r_{ij} Clr_{ij}}{\sum_j^N r_{ij}}$$

where r_{ij} are the ratings of every camera that sees the considered triangle. These charts are efficiently packed into a texture atlas using a block packing algorithm.

5.2.5 Occlusion detection

Although the multi-texturing approach is able to detect occlusions produced by the geometry of the model, it cannot ensure photo consistency across the texture if there are partial occlusions of the model produced by foreign elements, either permanently (e.g., a lamppost placed in front of the landmark being reconstructed) or temporarily (e.g., a person walking by). To detect such occlusions, we must evaluate if the contribution from a particular camera to a certain pixel is too different from those of the other cameras.

However, banning the contribution of a particular camera is not as simple as it looks. As explained above, the key of this system is how the ratings are distributed smoothly across the 3D mesh. This means that if we discard the colour information of a camera in one particular pixel or group of pixels, we might be introducing a discontinuity in the ratings function and, therefore, a potential visible seam. To guarantee the continuity of the ratings function, we need to transfer the banning to the vertex rating, and its null value will be correctly interpolated. However, for low polygonal resolution meshes, this solution might be too drastic, so the final solution is subdividing the 3D mesh (using a trivial midpoint approach: we do not want to modify its geometry) to bring its polygonal resolution closer to the texture atlas resolution. Figure 41 shows a multi-textured 3D mesh resulting from this process.

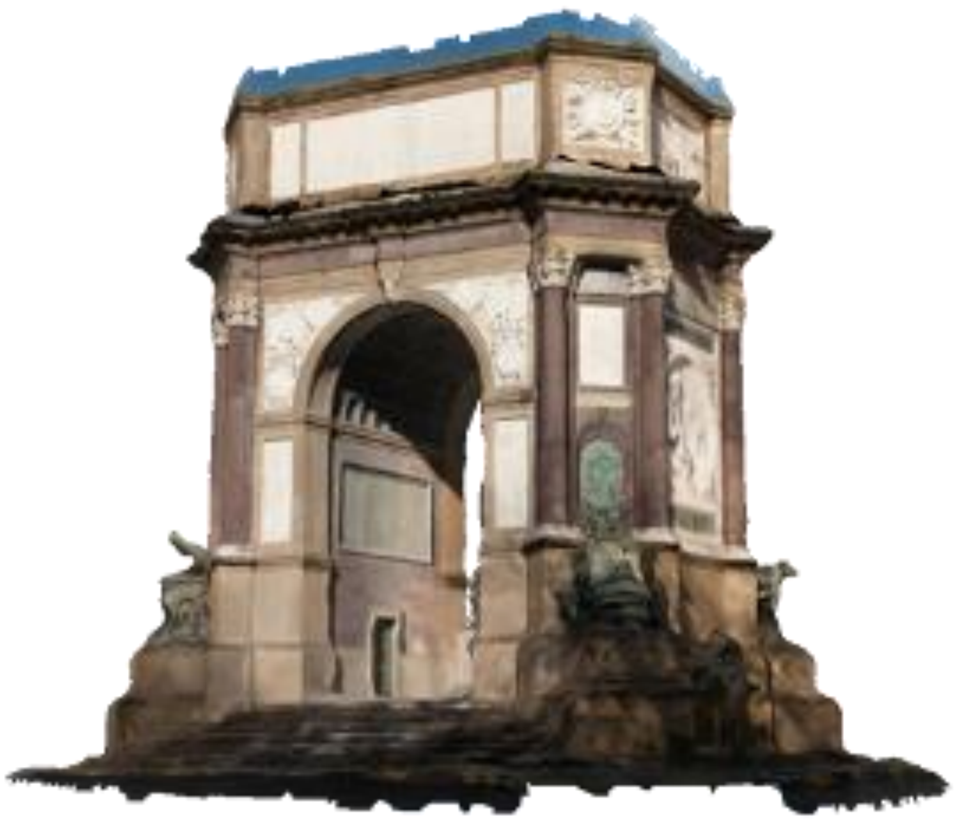


Figure 41: Result of applying the multi-texturing stage on the 3D mesh of Arco Valentino.

5.3 Point cloud super-resolution

Captured data is often limited by several aspects, such as the acquisition device specifications (e.g., resolution) and manufacturing defects (e.g., camera sensor noise) or the environmental conditions (e.g., low light conditions). Therefore, data enhancement techniques are used to improve the representation power of the intended domain, acting as a whole topic involving several problems (e.g., super-resolution, denoising, gap-filling or artefact recovery). The criterion of this improvement may be interpreted in multiple ways (e.g., fidelity to the domain or perceptual favourability). In this section the main objective is to improve the 3D data obtained by the different sensors.

5.3.1 2D data enhancement

Images are a representation of a captured slice of a certain domain through a 2D projection. In this first stage of our work, we have focused on one of the (most common) problems in 2D data enhancement, image super-resolution (2DSR). The problem of Image Super-Resolution aims to take an input image of a given low resolution (LR) and build its higher resolution (HR) version. The solution ideally will interpolate novel values while visually representing a likely capture of the domain (i.e., recovering the lost information such as the HR textures).

We focus on the task of supervised 2D SR, which is composed of pairs of LR (input) and HR (target) images. Our goal is to upscale the LR image as close as possible to its HR pair. When working with natural images, these pairs are generated in two different ways. The first is directly capturing the scene with different devices of different resolutions. This method presents two key challenges: i) it needs two devices to capture the same exact scene (which implies capturing it at the same time) and ii) the two devices must be placed at exact position in the scene. The second method, which is the most commonly used, consists of capturing the HR image and then down sampling it to retrieve its LR pair. This step often involves bicubic interpolation.

5.3.2 Datasets

Martin et al.[78] created in 2001 a content-variated dataset of 100 natural images (by sampling the BSD500 dataset) for unrelated purposes that has been a benchmark for 2DSR since then. Bevilacqua et al.[79] proposed a set of 5 images that are visually relevant to qualitatively evaluate 2DSR techniques. This was later expanded by a larger set of 14 images by Huang et al.[80] at the same time they presented another benchmarking dataset containing 100 urban scenes. All these mentioned datasets have been captured naturally for their HR images and artificial downgrade (bicubic downscale) has been applied for the LR pair. In the non-real domain, Aizawa et al. also created a dataset for Manga scan images for also unrelated purposes but relevantly used in the 2DSR context by applying bicubic downscale for obtaining the LR pairs (usual benchmark). Agustsson et al. proposed the DIV2K dataset, a collection of 1000 diverse 2K-resolution images with artificial degradation for their LR pairs. It is often used for training 2DSR approaches. Later, Lugmayr et al.[81] appended a set of Flickr images to the DIV2K dataset, creating the DF2K dataset.

5.3.3 Metrics

Choosing a good metric for 2DSR evaluation is a hard task. The most common metrics are PSNR and SSIM, while others like MSE or MAE are sometimes used. The issue with these metrics is that they are pixel-wise (except SSIM, but it is often correlated to PSNR) and this does not correctly represent the representation power of the enhanced image. For this reason, other metrics such as LPIPS represent perceptual similarity by using unrelated Neural Networks that seem to correlate to our perception. NIQE is the most relevant unsupervised metric (which doesn’t compare to the target image but instead evaluates the quality of the generated one). Finally, subjective metrics like MOS are also used.

5.3.4 Related work

2DSR is a problem that has gone through different stages. The arrival of CNN’s produced a CNN-based stage, and later the research direction was on edge-directed methods. Some years ago, using GANS for 2DSR was a trend.

[78] Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001, July). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001 (Vol. 2, pp. 416-423). IEEE.
[79] Bevilacqua, M., Roumy, A., Guillemot, C., & Alberi-Morel, M. L. (2012). Low-complexity single-image super-resolution based on nonnegative neighbor embedding.
[80] Huang, J. B., Singh, A., & Ahuja, N. (2015). Single image super-resolution from transformed self-exemplars. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5197-5206).
[81] Lim, B., Son, S., Kim, H., Nah, S., & Mu Lee, K. (2017). Enhanced deep residual networks for single image super-resolution. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 136-144).

Ledig et al. proposed SRGAN, which added an adversarial term to the traditional CNN training, which was later improved by Wang et al. with ESRGAN. Zhang et al. introduced RCAN, which applied (Channel-wise) Attention for the first time and became State of the Art in 2018. With the popularization of Transformers, a new stage began. SwinIR, proposed by Liang et al.[82], is still a very relevant approach, which works using Swin Transformers (which theoretically makes use of larger regions in the image) (Figure 42). However, Chen et al.[83] discovered that it still could use a larger region of it and proposed HAT, which used a hybrid attention transformer that worked at different levels (local level, window level, and among adjacent windows) and proved that it could use a larger portion of the image (Figure 43).

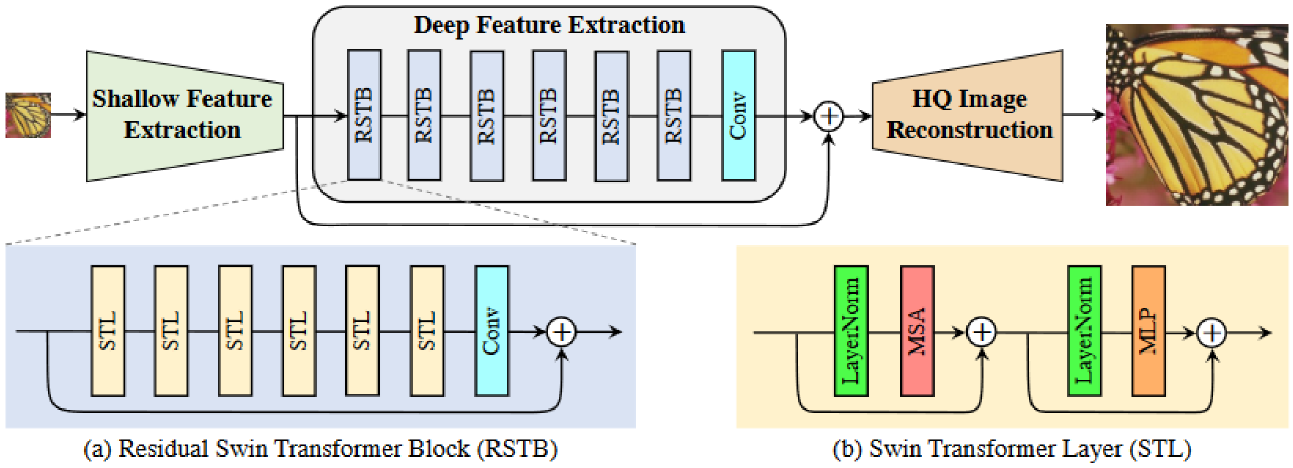

Shallow Feature Extraction
Deep Feature Extraction
RSTB
Conv
HQ Image Reconstruction
STL
LayerNorm
MSA
MLP
(a) Residual Swin Transformer Block (RSTB)
(b) Swin Transformer Layer (STL)

Figure 42: SwinIR architecture

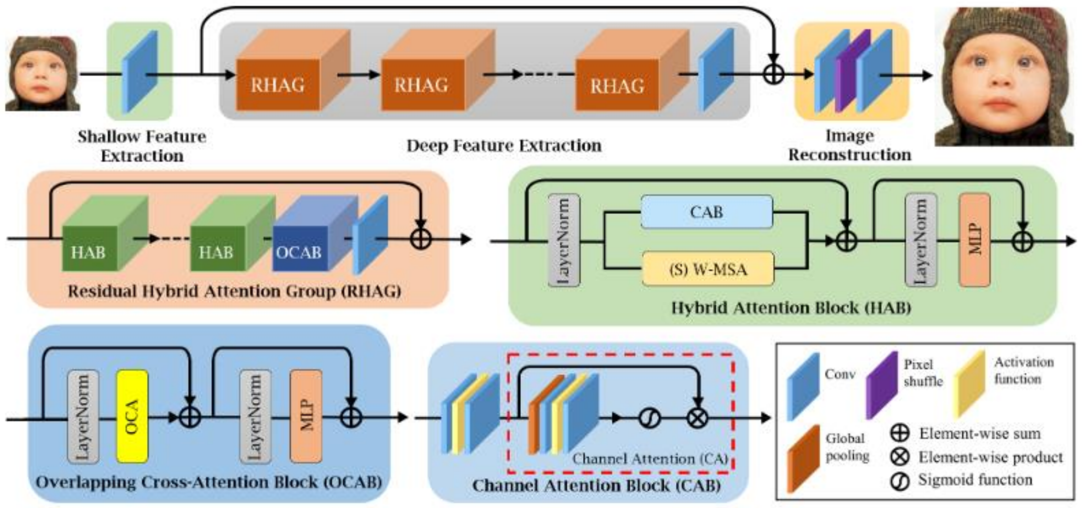


Figure 43: HAT architecture.

[82] 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).
[83] Chen, X., Wang, X., Zhang, W., Kong, X., Qiao, Y., Zhou, J., & Dong, C. (2023). HAT: Hybrid Attention Transformer for Image Restoration. arXiv preprint arXiv:2309.05239.

5.3.5 3D data enhancement

Current technology allows us to capture 3D Data as we do with 2D. As 2D Data uses cameras as acquisition devices (and implies a 2D projection), there are 3D sensors that allow us to capture the space content and doesn't require a 3D to 2D projection. We already know that 2D Data Enhancement implies all the 2D Data capturing (through images) issues and enhancement solutions, and 3D Data Enhancement does the same for 3D Data. The same kind of limitations that image acquisition may suffer can be present in capturing the 3D scene (e.g., device noise, 3D resolution). In the same way, 3D Data Enhancement involves a whole set of problems, including 3D Super-Resolution (3DSR).

3D Resolution defines the 3D space detail (number of slots in the space to occupy for volumes in the content) as 2D Resolution defines the 2D Space detail (number of slots to occupy for shapes in the content). So, SR in the 3D context implies taking a lower-resolution 3D capture (e.g., LR Point Cloud) and getting a higher-resolution representation of the same scene (e.g. HR Point Cloud). Since 3D Data can be represented in several ways that may refer to different things. In the case of Point Clouds, which means obtaining more points to better represent the volumes (with higher detail textures and shapes), and it is often referred to as Point-Cloud Upsampling. From now on, when we refer to 3DSR we will concretely refer to Point Cloud Upsampling.

In the context of 3DSR/Point Cloud Upsampling, the same concept as in 2D is applied: we need pairs of LR and HR Point Clouds (fewer and more points). As in 2D SR, the acquisition of the pair can be done naturally (with different devices) or artificially (natural acquisition of HR Point Cloud and artificial downgrade to LR Point Cloud). In the case of an artificial downgrade of the HR Point Cloud, if we apply a simple downsampling we can get fewer points but equally represented in resolution (same slots), so instead a quantification of the input Point Cloud is applied.

5.3.6 Datasets

For the task of 3DSR, there are two main benchmarking datasets, although many different experiments are conducted in different publications (most of them private). First, Lian et al.[84] presented SHREC-15, a collection of human sequences representing the position of their joints, artificially downgraded. Second, Li et al. presented a collection of 3D models in their PU-GAN publication. These 2 datasets are often used for benchmarking.

5.3.7 Metrics

In 3DSR (and in general in any Point Cloud recovery problem) most of the metrics are based on Point Cloud distances. The most common one is Chamfer distance. Earth Mover's Distance is also often used. Uniformity metrics such as Normalized Uniformity Coefficient (NUC) are also often used. Also, most of the times the problem is converted to a binary problem of points close to the target ones within a certain threshold, and a binary classification, such as F-score, is applied. Finally, a 3D version of MOS (evaluating Point Clouds) is also used.

[84] Lian, Z., Godil, A., Fabry, T., Furuya, T., Hermans, J., Ohbuchi, R., ... & Wuhrer, S. (2010, May). SHREC'10 Track: Non-rigid 3D Shape Retrieval. In 3DOR@ Eurographics (pp. 101-108).

5.3.8 Related work

In 3DSR, CNN's also started to play an important role, with the particularity that they needed to apply 3D Convolutions (which are relatively hard to compute). However, in the State of the Art these are only used at the input and 2D features flow through the Networks. Yu et al.[85] presented PU-Net, which was the first Deep Learning-based approach on this task. Jian et al.[86] presented later ARGCN which used Graph Convolutions and incorporated an adversarial term, and finally, Li et al.[87] presented PU-GAN, an adversarial version of PU-Net which also incorporates Self-Attention.

5.3.9 3D to 2D Projections

3D Data has a structure that requires a way of working over it, which cannot always be accomplished. Sometimes a projection to 2D Data must be applied. For example, a Point Cloud can be projected to a Depth Image using some camera with a given calibration. However, a Depth Image represents for each pixel in the resulting 2D map only the depth value in a single channel. Guo et al., in their paper of 3DDFA for face alignment proposed an approach to embed the true XYZ positions of the projected points called PNCC, which will be very relevant to our work (Figure 44). PNCC consists in colouring the points in RGB meaning the XYZ position and projecting them to the 2D plane to generate a 2D image embedding the 3D positions of all projected points. Except for the occluded points (from the projection), this operation is reversible to reconstruct the 3D Point Cloud.

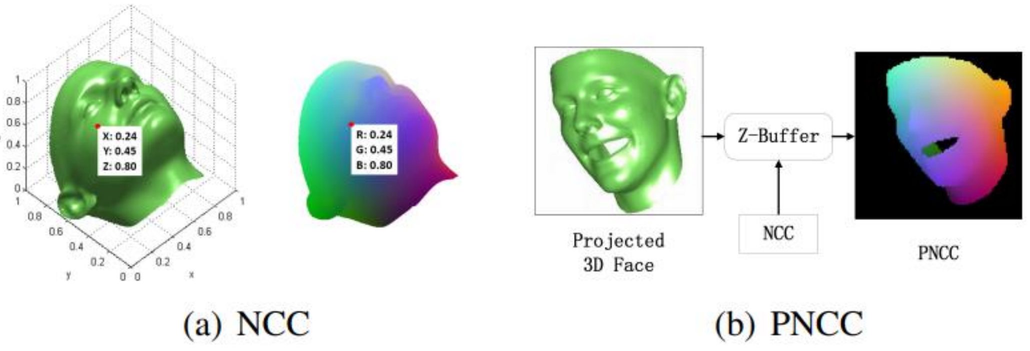


Figure 44: PNCC generation on 3DDFA.

5.3.10 2D Super resolution

We created a custom testing set for benchmarking different approaches. That set is composed of the previously mentioned Set5, Set14, BSD100, Urban100 and Manga109 as well as some custom samples including faces,

[85] Yu, L., Li, X., Fu, C. W., Cohen-Or, D., & Heng, P. A. (2018). Pu-net: Point cloud upsampling network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2790-2799).
[86] Jiang, J., Wang, A., & Aizawa, A. (2021, April). Attention-based relational graph convolutional network for target-oriented opinion words extraction. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (pp. 1986-1997).
[87] Li, R., Li, X., Fu, C. W., Cohen-Or, D., & Heng, P. A. (2019). Pu-gan: a point cloud upsampling adversarial network. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 7203-7212).

bodies, urban scenarios and extremely LQ images. Furthermore, we also created a Qualitative set for evaluating the desired approaches.
We tested the two main State of the Art 2D SR approaches, SwinIR and HAT with a SR scale factor of 4. We also included a Real-Time method, EDFN, presented by et al. in 2022. We evaluated PSNR and SSIM as well as the time taken by each method (and fps). The experimental results can be seen in Table 1.

Table 1: Results on the main 2D SR approaches for our 2D SR set.

	PSNR	SSIM	Time (ms)	Speed (fps)
SwinIR	23.50 dB	0.73	0.59	~2
HAT-S	25.17 dB	0.76	0.57	~2
EFDN (RT)	23.07 dB	0.72	0.014	~70

The qualitative results of the method were also reviewed, and some pertinent samples are presented in Figure 45.

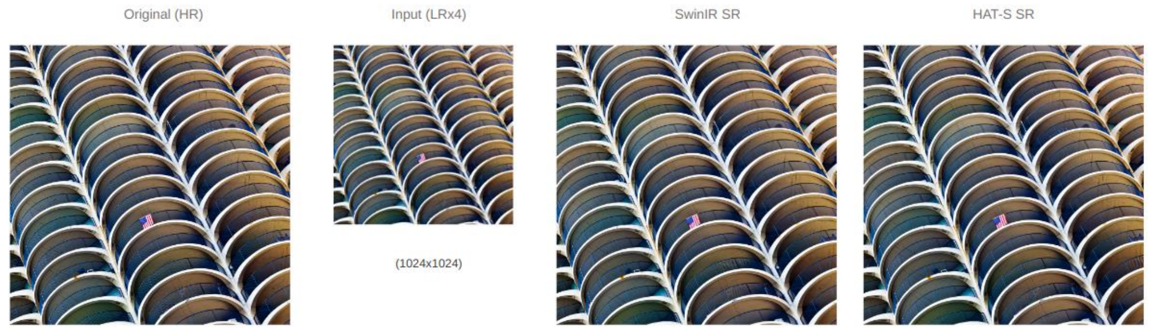


Figure 45: Qualitative comparison of different SR algorithms where high frequency problems are apparent.

In Table 1, it is evident that the numeric difference between SwinIR and HAT is not so relevant quantitatively and is probably a numerical. Furthermore, SwinIR has more available models (for different scales and cases), so we decided to use SwinIR in our comparisons from now on. Instead, the EDFN approach, while being much faster, is close quantitatively but clearly worse in qualitative performance, so we compared SwinIR and EDFN in our created qualitative set. We include some examples of this comparison (Figure 46 and Figure 47):

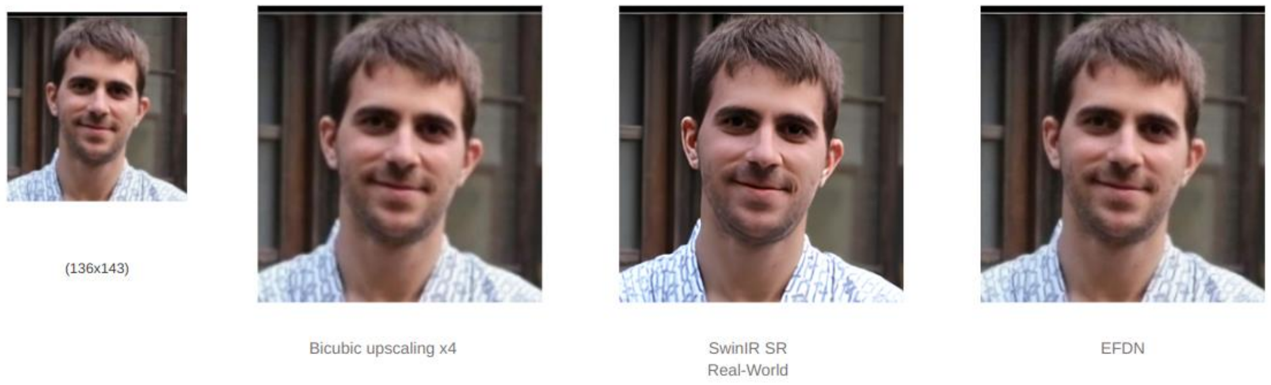


Figure 46: A qualitative comparison of different Super-Resolution (SR) algorithms on an image displaying a face.

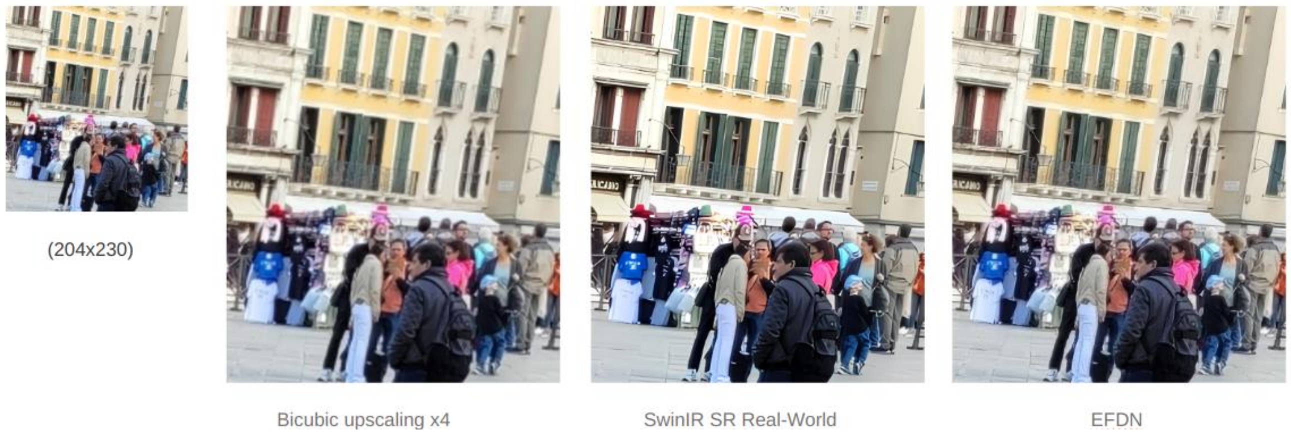


Figure 47: A qualitative comparison of various Super-Resolution (SR) algorithms in an image containing rich content in terms of frequency and colours.

The conclusion drawn is that unless real-time conditions are explicitly required, we prefer to conduct our experiments using SwinIR, as it appears to deliver superior performance.

5.3.11 3D Super Resolution

The main idea in our work is that we want to enhance the 3D Data but avoiding the cost of 3D processing. To do so, we propose to use PNCC as a 3D to 2D projection and apply a 2DSR approach (lighter than 3D SR) for further 3D Reconstruction. This way, we will achieve a higher resolution version of the input Point Cloud in a lighter way than regular 3DSR. To our knowledge, this is a novel idea that has not yet been tested. So, our pipeline will consist of generating a PNCC from an input Point Cloud, applying a 2DSR approach over this PNCC and reconstructing the resulting upsampled Point Cloud from this higher resolution PNCC.
Our first step consists in generating the PNCC from the input Point Cloud. We tested our pipeline with the samples in the CWI dataset, which contains the RGB+D data from a series of sequences. Since the calibration files are available, we can build the Point Cloud from the Depth maps, and therefore assign an RGB color (being the computed XYZ position) to each pixel in the Depth Map image (Figure 48). For benefitting from the whole scale of the value range, we put each channel between 0 and 1 in 32 bits.

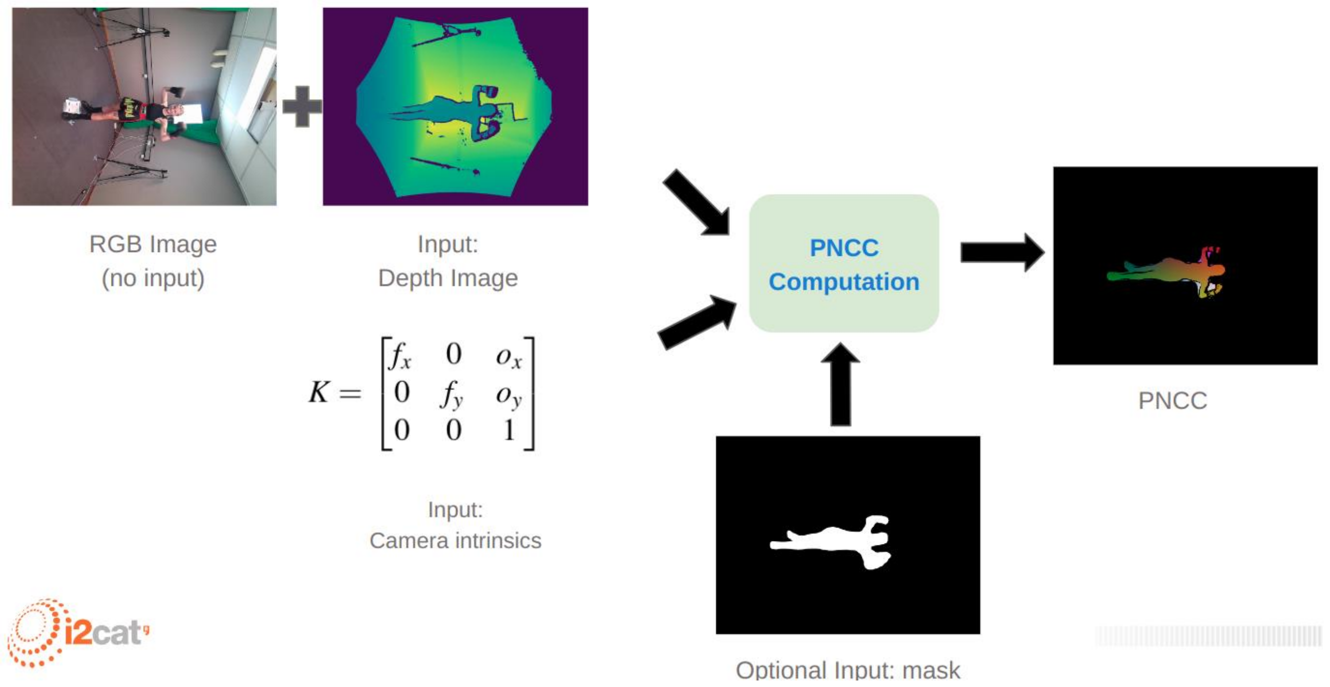


Figure 48: PNCC generation with our pipeline.

Figure 49 shows some (masked) examples of this 3D to 2D projected generation.

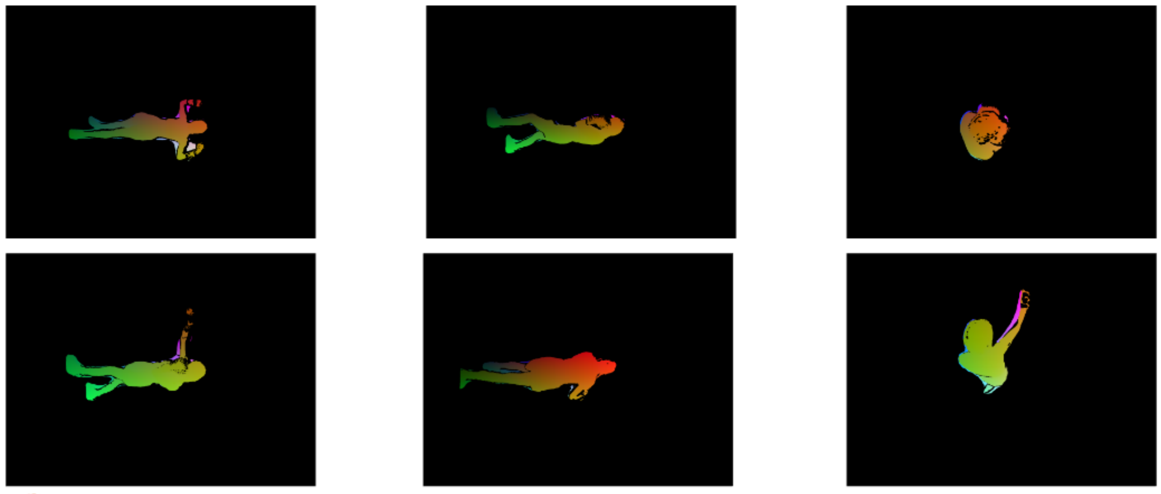


Figure 49: Examples of generated PNCC samples.

At this point, we can apply different 2D SR approaches on the PNCC images. We included in our experiments Bicubic upscaling (baseline) and the pretrained (in natural RGB image) version of SwinIR, both with a x4 upscaling factor. We got some result on the algorithms we used to perform this step, evaluating them with some mentioned 2DSR metrics in one of the CWI samples.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

The last step is to reconstruct the final Point Cloud from the Upscaled PNCC to evaluate the 3D reconstruction. As can be seen in Table 2, bicubic downscaling is still better than the SwinIR version which has been trained in natural RGB images (2D pre-trained model). That is because PNCC Data has a completely different structure and meaning than what it has seen before. Some experimental results on the mentioned methods using the CWI database are depicted in Figure 50.

Table 2: Results on the baseline approaches for PNCC SR on a PNCC sample using bicubic interpolation and 2D pre-trained SwinIR model.

	Chamfer distance
SwinIR x 4 (RGB trained)	8.647
Bicubic upsampling x 4	3.481

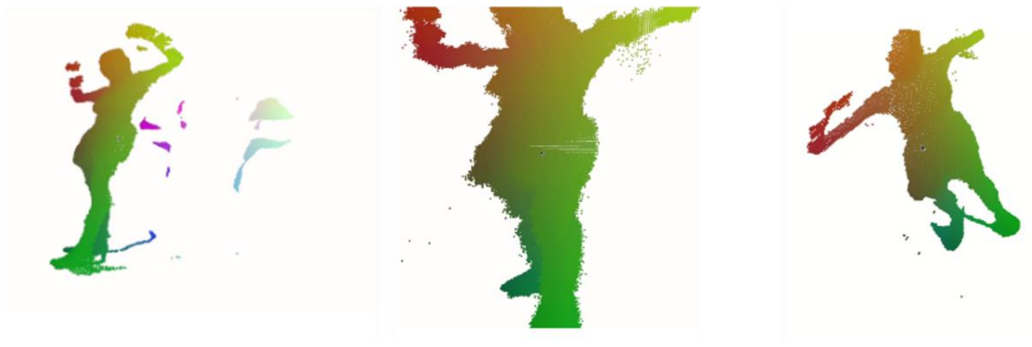


Figure 50: PNCC SR with different approaches, a pre-trained SwinIR model, and bicubic upsampling x 4.

5.3.12 3D projected model training

We aim to get a SwinIR model that learns to upscale the 3D Point Cloud through its PNCC, but as seen in the previous experiments, which requires training it in the PNCC Data. We prepared a training set on several sequences of the CWI Dataset, some for validation and some for evaluation and test. Table 3 shows some preliminary results on the test set.

Table 3: Results of the PNCC-trained SwinIR models vs baseline approaches on our PNCC test set.

	PSNR	LPIPS	NIQE	Chamfer
SwinIR (PNCC fine-tuned)	**29.99 dB**	**0.089**	10.11	**1.583**
SwinIR (RGB pre-trained)	27.56 dB	0.131	7.48	3.871
Bicubic	27.22 dB	0.220	**10.78**	3.303

As we can see in the table, our trained model obtains better results than the other baseline methods. Some qualitative examples can be seen in Figure 51.

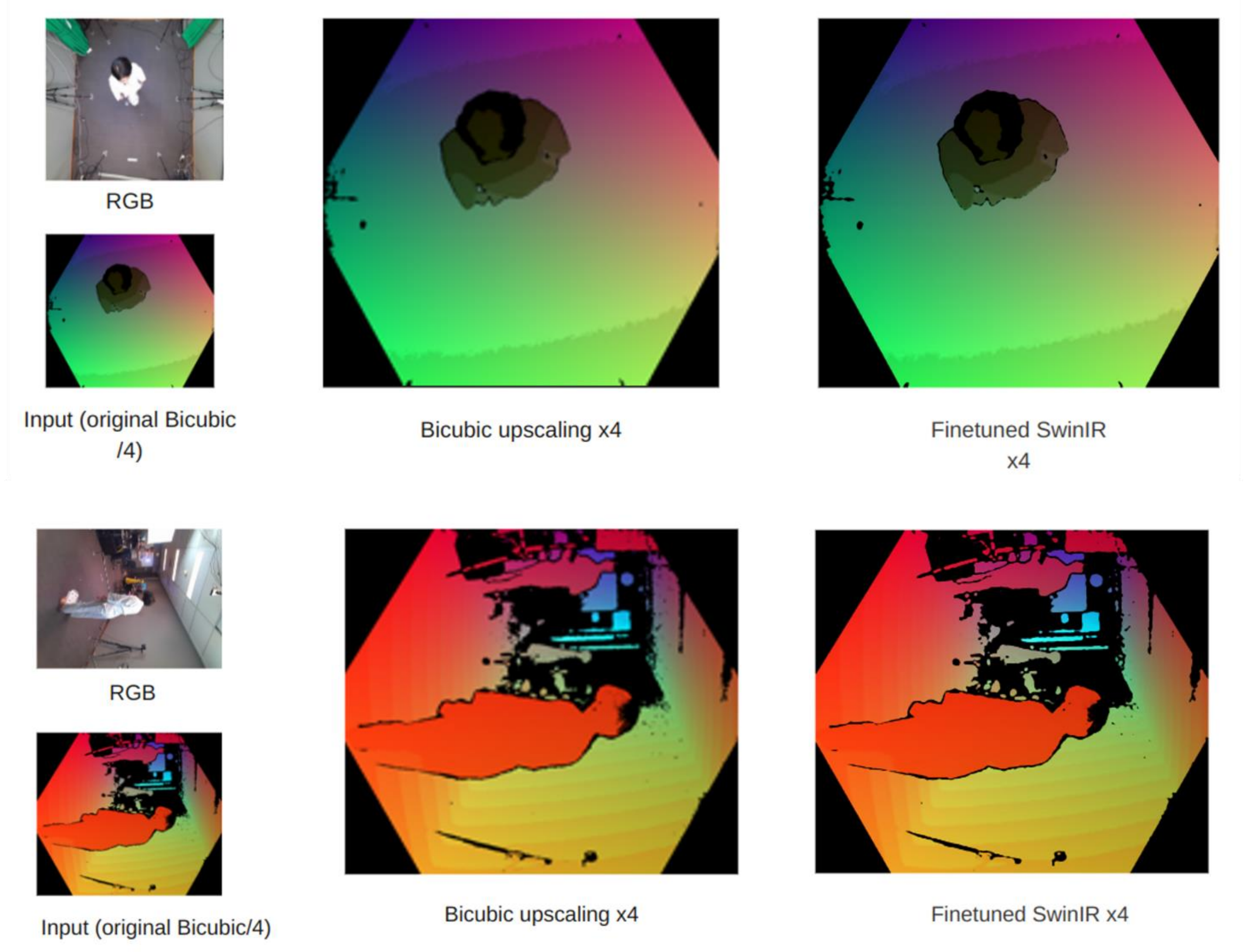


Figure 51: SwinIR trained on PNCC experimental results comparison.

In the future we plan to continue training the model from scratch and train it with more diverse databases in order to be more robust to other types of scenes. We are also planning to finish adapting the PNCC conversion pipeline to use the RGB projection maps provided for the compression algorithms and try to integrate the data enhancement inside the 3D reconstruction pipeline.

5.4 Colour image super-resolution

Single Image Super-Resolution (SR) is a technique that is commonly used to improve the resolution of low-quality images and generate high-quality counterparts. This process primarily focuses on restoring high-frequency information lost during image acquisition or compression, resulting in a more enhanced visual quality.

In the context of facial image super resolution (targeting facial 3D reconstruction) two super-resolution methods were investigated, one of which is focused on the upscale and enhancement of facial textures for 3D avatars, while the other has more general purposes. GFP-GAN[88] is a specially trained Generative Adversarial Network (GAN) for blind facial image restoration, with an emphasis on prioritizing facial details. It can achieve an optimal balance between realism and fidelity in the image restoration process. This is made possible through the utilisation of a broad and diverse knowledge base embedded in a pre-trained generative network, specifically designed for facial image restoration.

As shown in Figure 52, GFP-GAN consists of a degradation removal module and a pretrained face GAN architecture serving as a facial prior. These components are connected through latent code mapping and various levels of Channel-Split Spatial Feature Transform (CSSFT).

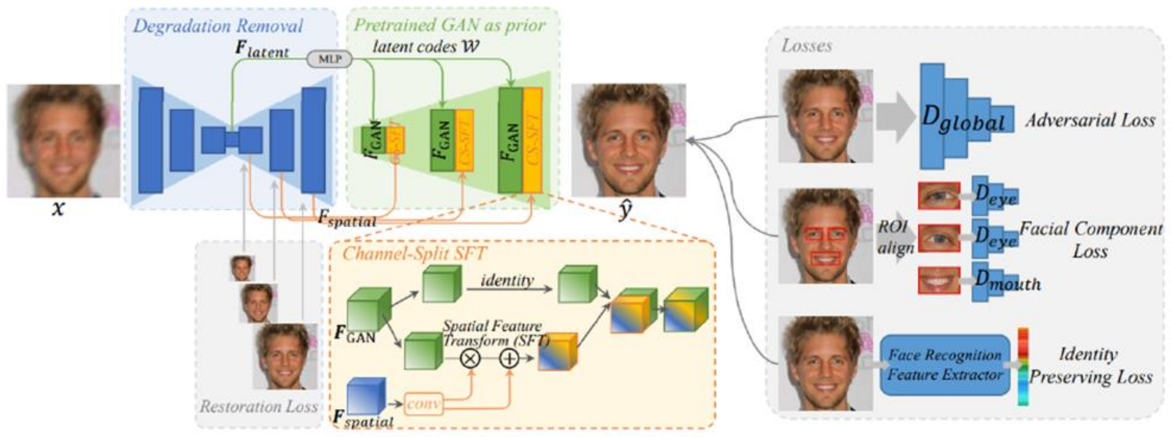


Figure 52: Overview of the GFP-GAN framework overview.

During training, the following processes are employed:

- Intermediate restoration losses are used to remove complex degradation.
- Facial component losses with discriminators to enhance the facial regions.
- Losses for preserving identity are applied to maintain the identity of the face.

This technique can be used in a variety of applications, including 3D avatar creation, photo restoration, and archive restoration (Figure 53).

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

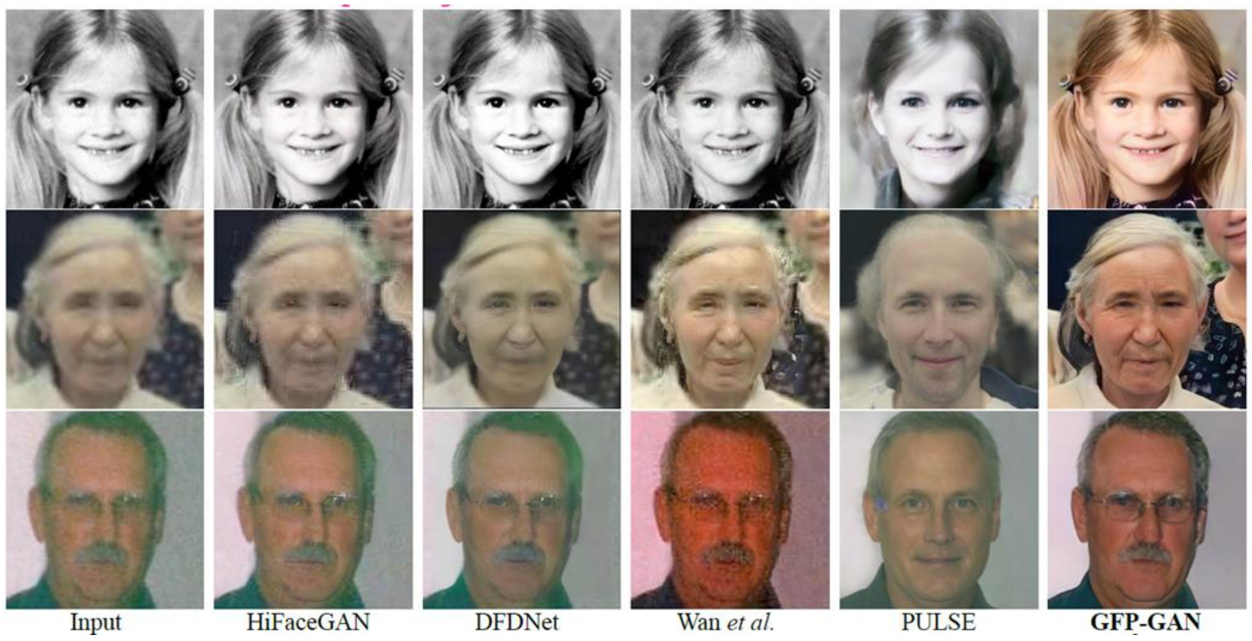


Figure 53: Comparisons with state-of-the-art face restoration methods.

Moreover, another super-resolution algorithm called Densely Residual Laplacian Network (DRLN[89]) was integrated that employs cascading residual on the residual structures, which can assist in training deep networks. Additionally, a core component called Laplacian pyramid attention that weights the features according to their relative importance is incorporated in the architecture. The algorithm achieves comprehensive quantitative and qualitative evaluations on low-resolution, noisy low-resolution, and real historical image benchmark datasets, demonstrating favourable performance on content from RAI's archive.

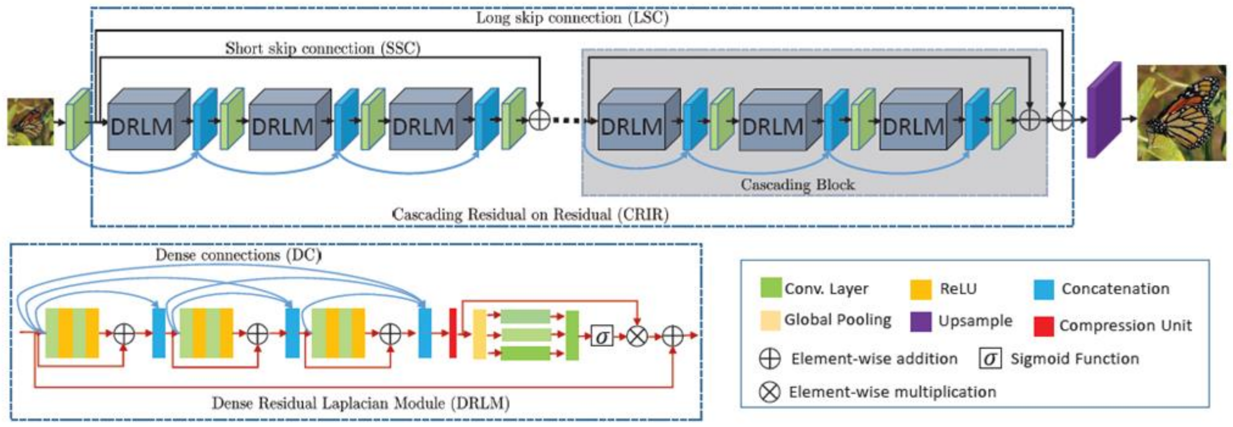

Long skip connection (LSC)
Short skip connection (SSC)
DRLM
Cascading Block
Cascading Residual on Residual (CRIR)
Dense connections (DC)
σ
Dense Residual Laplacian Module (DRLM)
Conv. Layer
ReLU
Concatenation
Global Pooling
Upsample
Compression Unit
Element-wise addition
Sigmoid Function
Element-wise multiplication

Figure 54: Top: the overall architecture of the network with cascading residual connections on the model, i.e., a long skip connection, short skip connections, and cascading structures. Bottom: the Dense Residual Laplacian Module (DRLM).

[89] https://arxiv.org/abs/1906.12021

Figure 54 illustrates the model architecture diagram of the Densely Residual Laplacian Network (DRLN) for super-resolution. The model consists of four integral components: feature extraction, cascading over residual on the residual, upsampling, and reconstruction. The feature extraction component uses a convolutional layer to extract primitive features from the low-resolution input. The cascading over residual on the residual structure allows the flow of low-frequency information to focus on learning high and mid-level features. The upsampling component is responsible for increasing the resolution of the image. Finally, the reconstruction component reconstructs the high-resolution image from the upsampled features.

6 XR volumetric and free-viewpoint videos services

6.1 Overview

XReco considers a variety of different solutions to provide volumetric services for offline and real time applications to be used in XR productions and AR experiences: Free Viewpoint Video (FVV) which allows the user to navigate freely around an XR scene controlling the position of a virtual camera; holoportation for live transmission of 3D volumetric video captures of humans based on pointclouds.

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

6.2.1 Overview and architecture

The FVV functionality provided by XReco will be based on the FVV Live system. This technology allows the user 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: capture, formed by the cameras and the Capture Servers, Stream Selector, and the rendering module, consisting of multiple Production Consoles and View Renderers. It also offers a replay module, which allows the playback of pre-recorded FVV content. They are all shown in Figure 55.

In the capture stage, the cameras acquire RGB and depth information from the scene, which will be used by the View Renderer to synthesise a virtual view from the desired viewpoint. Users can manipulate this viewpoint and visualize the rendered views using the Production Console. The Stream Selector enables rendering several simultaneous virtual views using the same source content, receiving the RGB-D streams from capture and re-distributing them to several instances of the view renderer.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

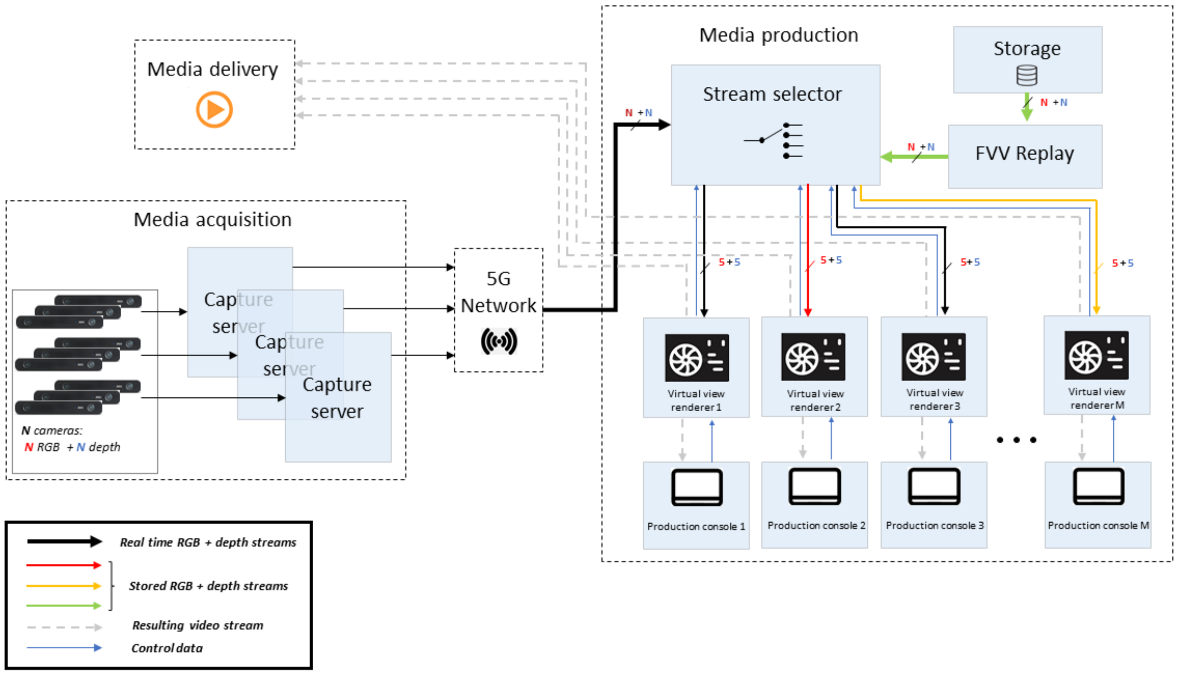


Figure 55: FVV Live block diagram presenting its main modules.

6.2.2 Capture Module

The capture module is the volumetric video acquisition stage. Several stereo cameras, usually nine (Figure 56), take care of capturing the scene and computing its geometrical information using stereo matching techniques. This geometrical information is given in the shape of depth images.

The RGB-D streams for each camera are encoded as videos using AVC/H.264 with a low latency profile and transmitted over the IP network using the RTP protocol. Depth information is fragile, encoding artifacts will greatly affect the quality of the synthesized virtual views, so depth is encoded into a separated video stream with lossless compression. Depth values are quantized with 12 bits using a 4:2:0 scheme.

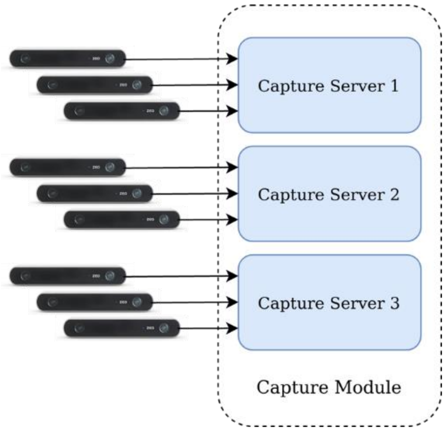


Figure 56: Capture Module diagram

Lossless encoded video yields a huge bitrate, so foreground segmentation is performed to only transmit depth information from the foreground, greatly reducing the output bitrate. This is very interesting since background information is redundant and the real-time depth estimation on far away objects is very noisy.

The FVV Live system also offers a replay module that mimics the behaviour of the capture module using pre-recorded content. It is completely transparent to the rest of components of the system, so it can be used to test them without the need for a real-time capture, but it also allows to render new virtual paths using past recordings or synthetic data.

6.2.2.1 FVV Live new segmentation approaches

Foreground segmentation is very relevant in this FVV scenario since depth information needs to be encoded with a lossless scheme, which means the output bitrate for each camera is concerningly high. To mitigate this problem, foreground segmentation is performed on the sequences so only relevant information is transmitted. Additionally, this foreground segmentation must be very accurate when FVV footage is mixed with virtual scenes.

Green Screen Cyclorama setup and segmentation algorithm

A green screen cyclorama setup will be used for the XReco use cases. Segmentation tests were performed on such setup to validate the current real-time foreground segmentation algorithm and results were satisfactory but not perfect. The implementation of a real-time green screen removal algorithm[90] is proposed to obtain better results on this specific scenario.

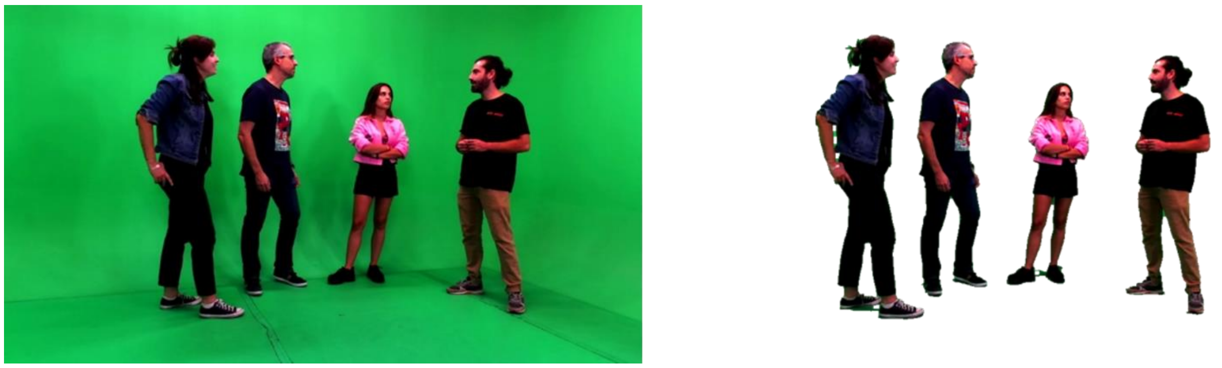


Figure 57: Example of green screen removal using the real-time FVV Live segmentation algorithm.

This green screen removal is performed in RGB space, since the background presents a contribution of green greater than the other primary colours. The algorithm performs two checks to decide if a pixel is part of the background:

- Checking if the normalized green component is bigger than a threshold T_g:

$$g(x,y) = \frac{G(x,y)}{R(x,y)+G(x,y)+B(x,y)};\ Background\ if\ g(x,y) > Tg$$

[90] Cuevas, C., Berjón, D. & García, N. A fully automatic method for segmentation of soccer playing fields. Sci Rep 13, 1464 (2023). https://doi.org/10.1038/s41598-023-28658-1

- Comparing the RGB value from the input image (u) to a reference image of the background (v). The RGB vector of a pixel from u is projected into v yielding u_v. The length of the perpendicular component of u_v is compared to a threshold T_c and the pixel is considered as background if the length is lower. Considering RGB as a 3D space, this restriction corresponds to defining a cone around v and selecting the pixels which RGB vector lie inside of said cone, as shown by Figure 58.

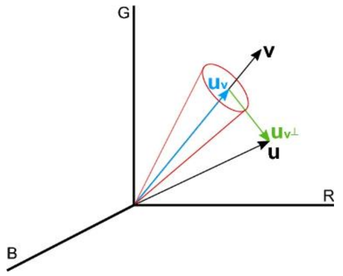


Figure 58: 3D representation of an example the cone that restricts the acceptable colours.

Deep Learning algorithms for foreground segmentation

To improve the quality of foreground segmentation on scenarios without a green cyclorama setup, Deep Learning segmentation approaches are being studied for both, real-time and offline operation. Following this strategy, a semi-automatic foreground segmentation pipeline based on Segment Anything (SAM)[91]was developed to obtain a more precise foreground segmentation that can be used in the FVV playback of the content.

Figure 59 shows a semi-automatic foreground segmentation pipeline based on SAM and object detection. The orange blocks are the only ones which require manual intervention.

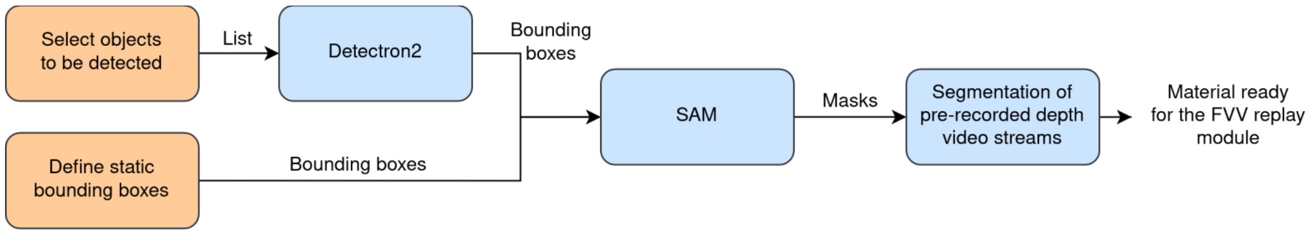

Select objects to be detected
List
Detectron2
Bounding boxes
Define static bounding boxes
Bounding boxes
SAM
Masks
Segmentation of pre-recorded depth video streams
Material ready for the FVV replay module

Figure 59: Semi-automatic foreground segmentation pipeline based on SAM and object detection.

SAM is a very powerful state-of-the-art semantic segmentation algorithm that yields masks of any object in an image. In the proposed pipeline, the user defines the objects that belong to the foreground and object detection is used to find them on each frame of the sequence, and SAM is used to obtain their segmentation masks (Figure 60Figure 59). This is neither a fully automated process, nor capable of real-time execution, so it is meant to be used to post process pre-recorded content.

[91] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollˇar, and Ross Girshick, “Segment anything,” arXiv:2304.02643, 2023.

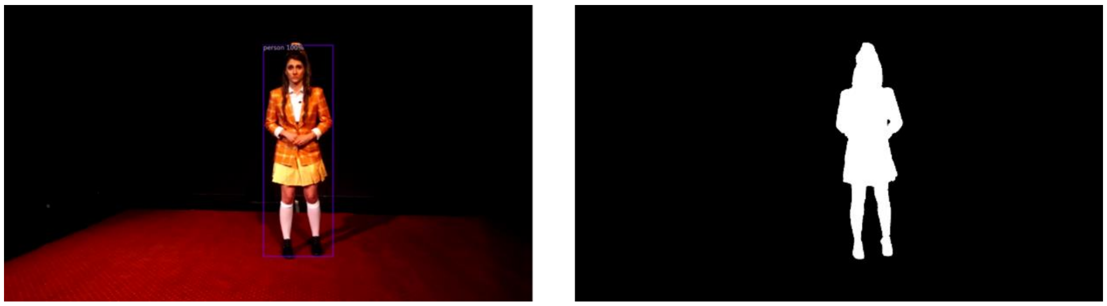


Figure 60: Example of the semi-automatic foreground segmentation pipeline using object detection and SAM.

Moreover, work has started to take on the real-time foreground segmentation problem. We propose using the presented segmentation pipeline to generate a dataset for foreground segmentation. The resulting dataset will be used to train a light deep neural network. Suitable architectures are being studied trying to archive satisfactory segmentation results while fulfilling real-time restrictions.

6.2.3 Rendering Module

The main component from this module is the View Renderer, which receives the RGB-D streams and processes them to synthesize a virtual view of the scene. When the stream selector is active, several instances of the View Renderer can be deployed (usually as Docker containers) to render multiple simultaneous camera paths.

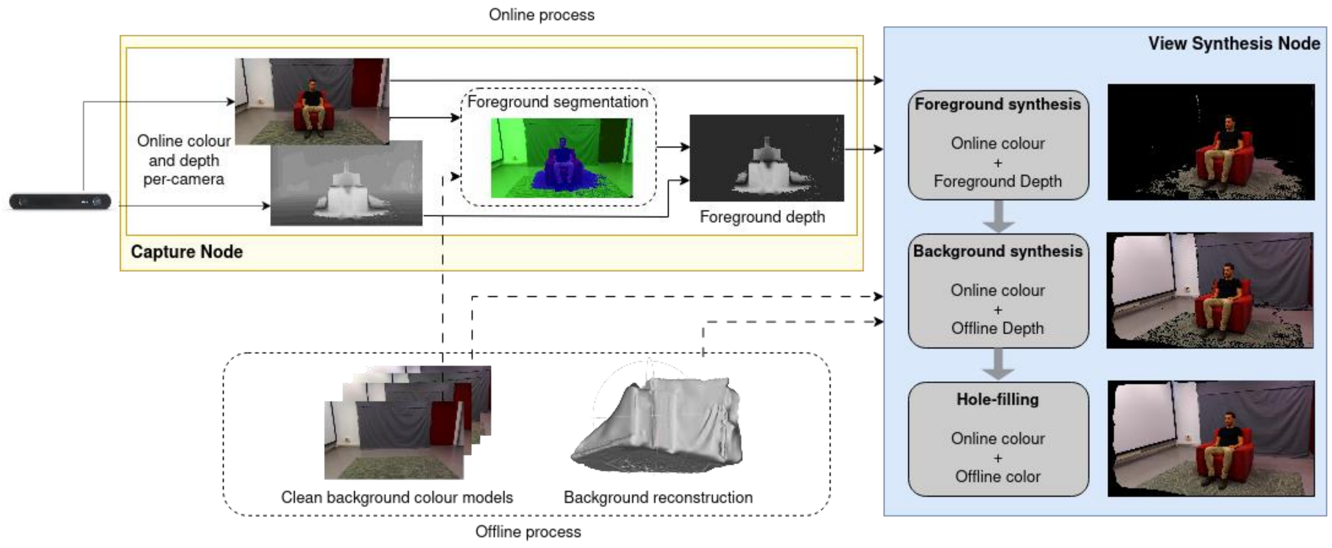

Online process
View Synthesis Node
Online colour and depth per-camera
Foreground segmentation
Foreground depth
Capture Node
Foreground synthesis
Online colour + Foreground Depth
Background synthesis
Online colour + Offline Depth
Hole-filling
Online colour + Offline color
Clean background colour models
Background reconstruction
Offline process

Figure 61: FVV Live basic synthesis process.

Figure 61 represents the basic rendering process of the virtual view[92]. Each camera transmits real-time colour and depth images to the capture node. The colour image is compared to a clean background one captured offline

[92] Teresa Hernando, Daniel Berjón, Francisco Morán, Javier Usón, Cesar Díaz, Julián Cabrera, and Narciso García. 2023. Real-Time Layered View Synthesis for Free-Viewpoint Video from Unreliable Depth Information. In Proceedings of the 15th International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE '23). Association for Computing Machinery, New York, NY, USA, 7–11. https://doi.org/10.1145/3592834.3592881

before the transmission starts to perform the foreground segmentation that is applied to the real-time depth to only transmit foreground information to the synthesis node. The synthesis node uses this foreground depth to render the foreground in the final view with the online colour images.

The virtual view synthesis process is based on depth-image-based rendering (DIBR), which involves projection of the RGB data from the known cameras to the virtual view according to the depth information available, as shown by Figure 62.

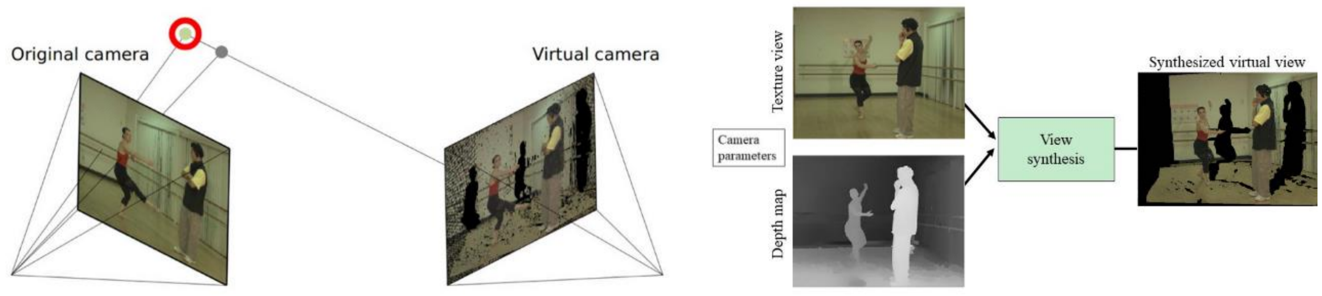


Figure 62: Profile of depth image-based rendering (DIBR) virtual view synthesis.

The view renderer performs a layered synthesis by separating the rendering of the foreground and the background. The background is reconstructed as a 3D model before real-time execution, free of real-time constraints, employing more compute-intensive techniques that provide good quality depth estimations of the clean background for each reference camera to be used along the real-time colour images to render the background. The holes left after rendering the foreground and background are filled using both the real-time and colour images and the set of clean background colour ones captured before transmission.

The resulting virtual views can be visualized in two ways: a simple mode where they are displayed on the computer screen, or a cloud server approach where they are encoded as video and sent to the user so they can play it on their machine.

6.2.3.1 Renderer improvements

The main goal is to reduce the rendering time and achieve better results in the final synthesized view delivered to the user.

Handling the background of the scene as a 3D mesh

The FVV Live system follows the same basic process for rendering the background of the scene as it does for the foreground, which is based on the DIBR technique described above. This implies that the 3D background mesh obtained upon the calibration process must be transformed into a set of depth maps, one for each of the reference cameras, to perform the projections, which involves a loss of depth accuracy. To avoid this unnecessary procedure and taking advantage of the static background the DIBR process has been replaced by a rendering of this 3D background mesh using OpenGL.

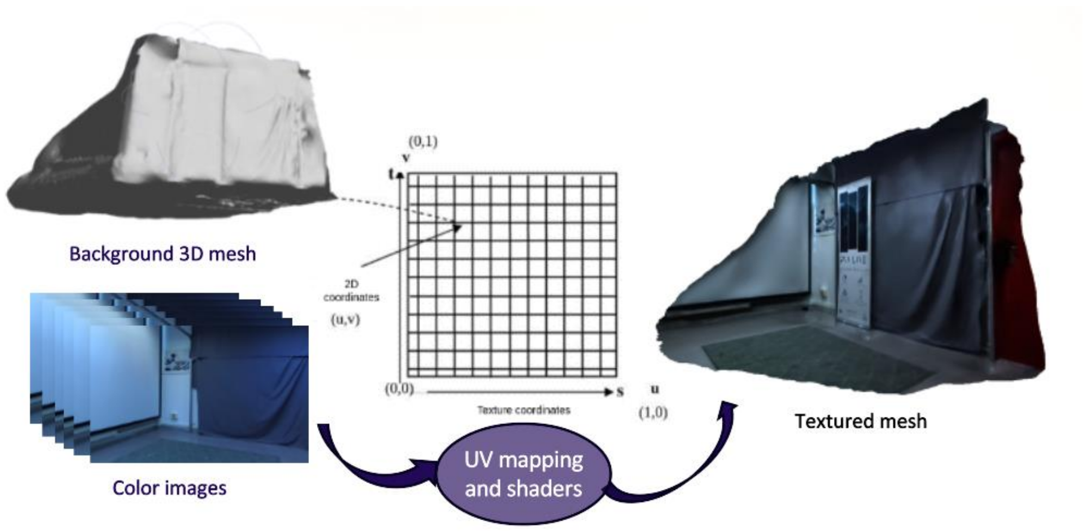


Figure 63: 3D background model texturing diagram.

Figure 63 illustrates the process of colouring the 3D mesh using the reference camera images. This process will be carried out once the foreground layer is rendered, it is therefore necessary to detect where those pixels are and delete the corresponding mesh points, so the background does not cover it.

It should also be taken into consideration that the presence of foreground objects subtly modifies the colour of the background as compared to the one captured when the scene was empty due to shadows or diffuse reflections. Hence, to avoid colour discrepancies between both layers that result in an unnatural effect on the final view, the colour images used to texture the 3D mesh are the real-time ones. This implies that the pixels of these colour images in which foreground is present must be detected to avoid colouring the background mesh with them, as well as the ones on the edge of foreground objects, since they have a colour that is a mix of contributions from both foreground and background areas, so they cannot be used to colour either of the mentioned layers. Once both layers are rendered the holes are filled the same way it was done before.

This modification in the background layer synthesis process results in a significant reduction of the rendering process time. The following graphs (Figure 64) show the synthesis process time of two different sequences with different levels of complexity measured with a GPU *GeForce GTX 1080*:

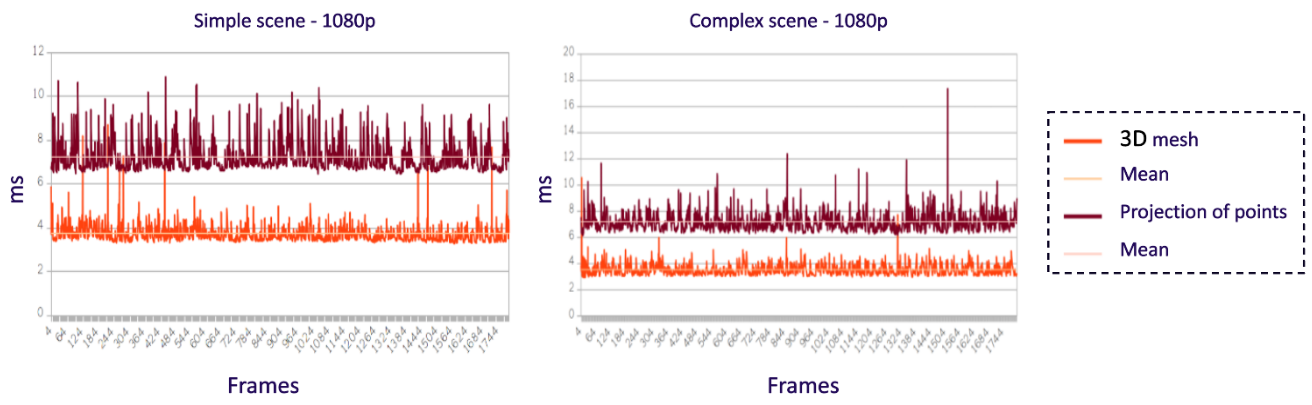


Figure 64: Rendering time of two different sequences using 3D mesh and DIBR for rendering the background.

As it can be observed, in both sequences the rendering time is lower when rendering the 3D mesh using OpenGL than when the DIBR method is used for the background synthesis. The average difference time between both techniques is in both sequences 3.6 ms, being a reduction of almost half of the rendering time.

In addition, the results of handling the background as a 3D model shows to be more stable, presenting much less difference in the rendering time between the different frames.

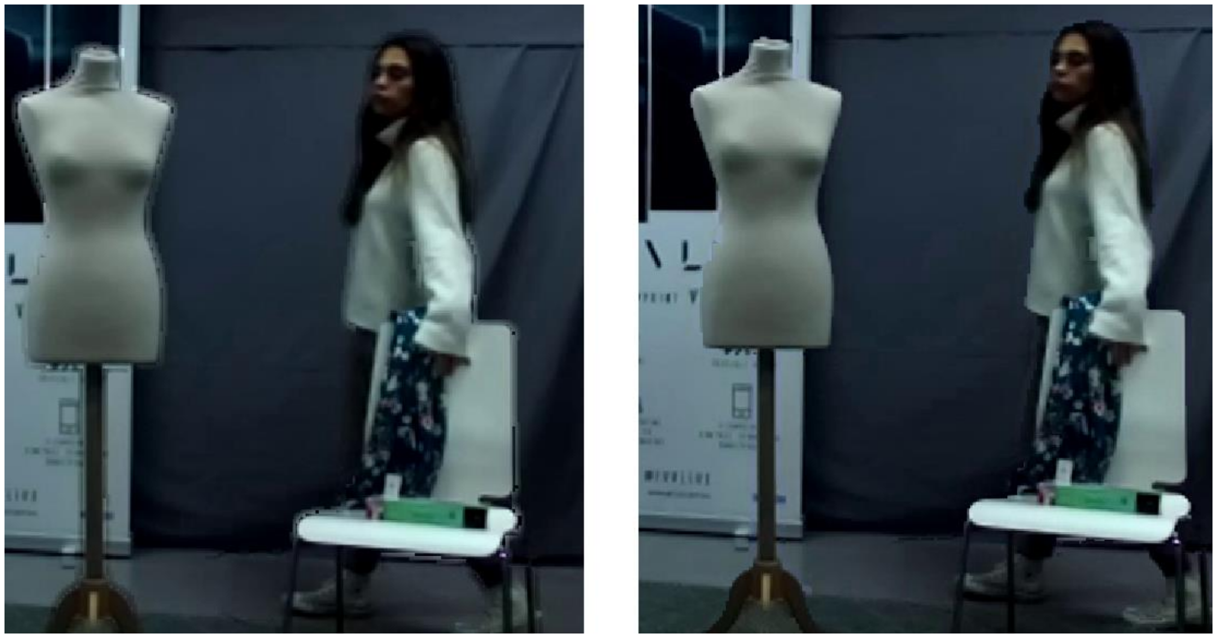


Figure 65: left: Rendered image using DIBR technique. Right: Image rendered using the 3D model.

Another outcome of this process is a better result on the foreground edges, as it can be observed in Figure 65, in addition to a more realistic and pleasant result when navigating through the scene, since by being static the background does not suddenly disappear when changing the reference cameras used for rendering the virtual view.

6.2.4 Production Console

The virtual camera path is controlled by a Production Console, which is a light application that the user runs on their machine. It is capable of reading user commands from a keyboard or gamepad/joystick and transmitting the corresponding UDP messages to the View Renderer to manipulate the virtual camera. It also allows visualization of the rendered view using FFmpeg player, FFplay.

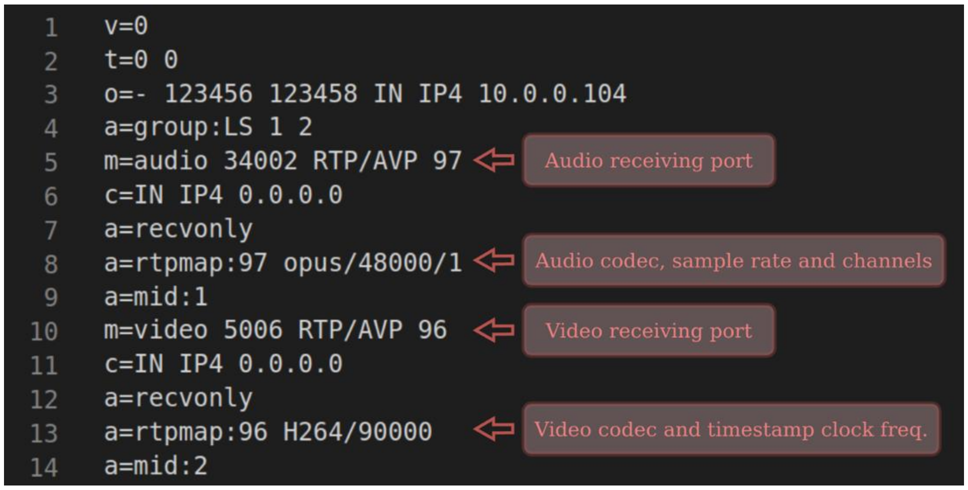
6.2.4.1 New virtual camera controls

New functionalities have been developed for the FVV Live Production Console based on the different use cases requirements, such as integrating the renderer output stream in a Unity scene.

FVV Live integration with Unity scenes

The first step of the FVV and Unity integration was a Production Console built in Unity. The key difference with the other approaches is that the FVV video stream is decoded by Unity, so the actual frames can be processed and visualized in the actual Unity application.

Video decoding is performed by the VLC (VideoLan Client) Unity plug-in. To receive the stream, an SDP (Session Description Protocol) file detailing the transmission parameters must be fed to the application. The parameters involved are mainly the codecs used for video and audio, and their respective receiving port. An example is provided by Figure 66:



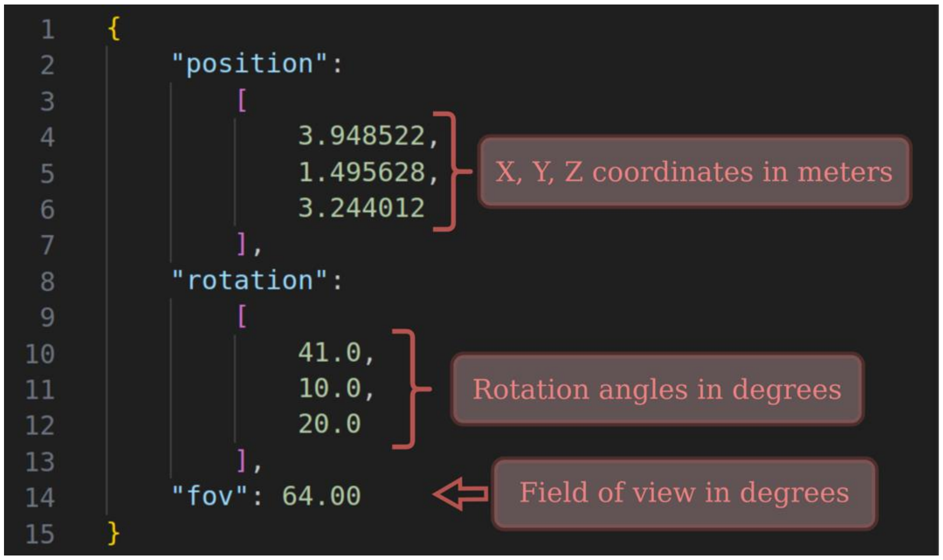
```
v=0
t=0 0
o=- 123456 123458 IN IP4 10.0.0.104
a=group:LS 1 2
m=audio 34002 RTP/AVP 97
c=IN IP4 0.0.0.0
a=recvonly
a=rtpmap:97 opus/48000/1
a=mid:1
m=video 5006 RTP/AVP 96
c=IN IP4 0.0.0.0
a=recvonly
a=rtpmap:96 H264/90000
a=mid:2
```

Audio receiving port
Audio codec, sample rate and channels
Video receiving port
Video codec and timestamp clock freq.

Figure 66: Example of SDP file to receive the FVV Live video stream in the Unity Production Console.

VLC parameters can be modified before playing the stream and their values have been finetuned to reduce latency as much as possible without sacrificing playback quality.

The current application handles the virtual camera by defining a camera object in Unity that the user can manipulate by clicking and dragging. A visual representation of the camera is shown in the user interface to give feedback about its position and orientation with respect to the scene. The camera transmits all its relevant parameters (position, rotation and field of view) to the FVV renderer in a simple JSON message such as the one presented in Figure 67.



```
{
    "position":
        [
            3.948522,
            1.495628,
            3.244012
        ],
    "rotation":
        [
            41.0,
            10.0,
            20.0
        ],
    "fov": 64.00
}
```

Figure 67: JSON message carrying camera information to control the FVV Live virtual camera.

The FVV Live View Renderer has been adapted to be able to receive these messages, understand the information that they carry, and to place the virtual camera accordingly. This way, the virtual camera from the View Renderer perfectly coincides with the virtual camera in the Unity application.

Web based console using WebRTC

To allow the Production Console to be multiplatform, the video streaming protocol WebRTC was used to develop a web application for FVV feed visualization and camera control. It enables FVV production from any device and removes the need for software installation. The implementation involves a WebRTC server that receives the rendered view stream and forwards it to all the users who access the server (Figure 68). Figure 69 shows an example of the Production Console via a web interface. Future steps involve the integration of the client with Unity and the development of a solution for HMDs.

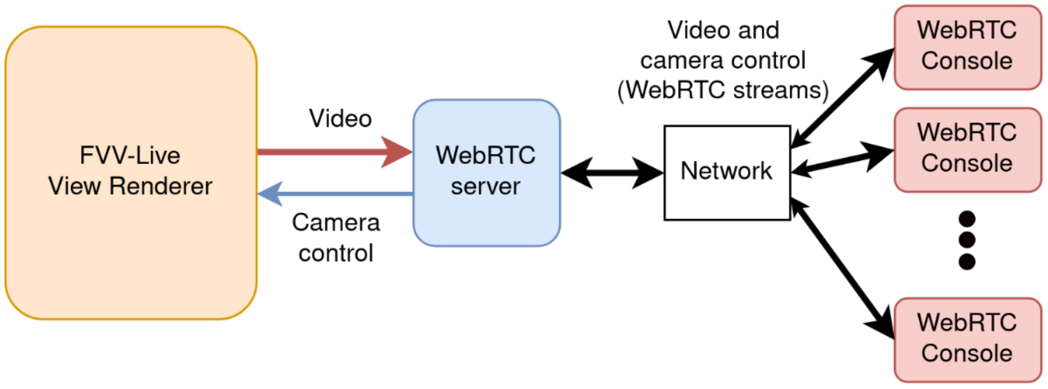


Figure 68: Deployment of the WebRTC server to allow users to control the virtual camera from a web console.

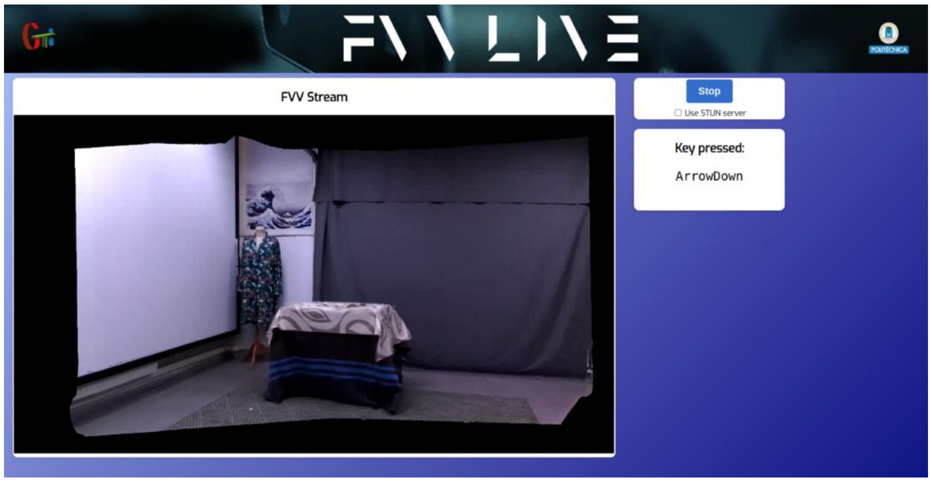


Figure 69: User interface of the web-based FVV Live Production Console.

6.3 RGB-D based real-time holoportation

Streaming technologies are quickly evolving towards 3D video paradigms, giving particular attention to novel solutions for compression, transmission and representation of volumetric video; this is especially true for natural content. The first compression standardisation efforts started within the Moving Pictures Experts Group (MPEG). However, dealing with volumetric video is not a trivial task, there are several challenges that need to be addressed. For instance, bandwidth occupancy, poor scalability and difficult viability in the distribution to high volumes of users and viewers.

Being able to capture and accurately reconstruct detailed volumetric representations would have a major impact on both augmented reality and virtual reality applications. This is particularly relevant for telepresence solutions where participants are required to create a highly detailed and accurate scan of themselves and depending on the technologies involved, the resulting point clouds will have varying degrees of quality. However, Volumetric media transmission requires an enormous amount of data to accurately represent 3D objects or scenes, for example point cloud videos with one million points requires up to 5 Gbps.

This problem becomes more complex when real-time constraints are considered, for that reason, it is mandatory to have efficient mechanisms to compress, deliver and render point clouds aiming at reducing memory and bandwidth requirements. MPEG has been developing compression standards, such as Video based PCC (V-PCC), obtaining good efficiency in the bandwidth reduction of volumetric data, however of subpar performance. The only real time application available is the V-PCC standard-compliant decoder developed by Nokia, which performs well with offline compressed point clouds representing a single person.

In real time applications, mapping a 3D volume over an image-based representation, has the big advantage of exploiting 2D video codecs to compress, stream and distribute the content efficiently. To achieve this, each element of the point cloud must remap its geometric attributes to a 2D Matrix, representing a component of the projected XYZ coordinates of a volumetric video frame for each pixel. However, there are some important considerations: first, video codecs are specifically designed for colour images which have a strong spatial and temporal correlation within the 2D space. Second, the available range of values is limited to 8-bit integers, this range of values is not enough to accurately represent volumetric data leading to loss of geometry information before compressing and geometry compression artefacts. To that end, the following sections describe 2D-based compression of 3D volumetric data, under a human-centred volumetric reconstruction perspective.

6.3.1 Real Time Point Cloud Compression Pipeline using Hardware Acceleration

In order to deal with the challenge of mapping a 3D volume over 2D matrices, but also to cover real time applications where 3D videos are captured by RGB-D sensors (e.g., Kinect 4 Azure), a real time point cloud compression pipeline has been developed. Figure 70 shows all the major steps, starting with an RGB-D based capturing system, followed by the 3D to 2D mapping and the consequent compression. The pipeline continues with the transmission and reception system and the following decoding, 3D remapping and rendering. First, using three RGB-D cameras, different points of view are captured. RGB-D data are very useful as mapping algorithm is easily parallelizable on modern GPUs for offloading a significant amount of work from the CPU.

The different RGB-D viewpoints are then fused together in a single volumetric representation. Next, the volume is projected into a 2D space which is streamed using traditional video compression. After the content is distributed, the target client decompresses the received volume and it is rendered interactively in an immersive scenario.

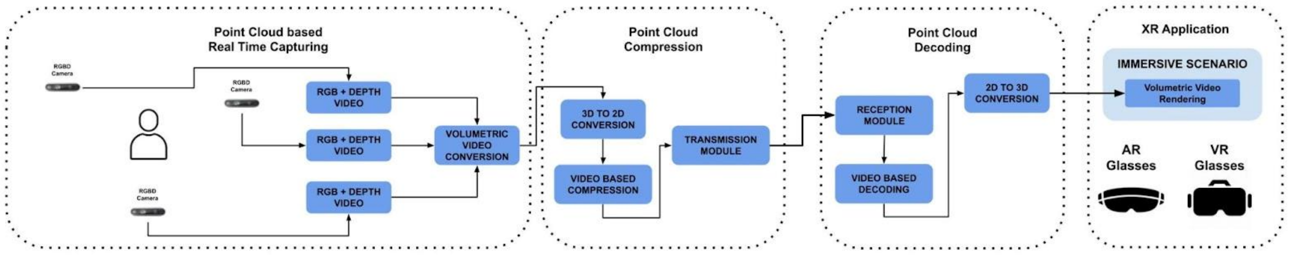


Figure 70: 2D-based point cloud compression pipeline.

The general approach of using graphics hardware has several advantages for the implementation of a real-time point cloud compression pipeline. GPUs are designed for parallel computations with different working threads for each pixel on a screen. This scheme can be leveraged to perform per-pixel operations in input images.

First, the input RGB-D data (an RGB image that stores the colours of any given frame in three 8-bit channels and a single-channel depth map) are stored using a single channel and 16 bits. Additionally, using the intrinsic parameters of the camera an additional map is created, which is used to transform the positions of the point clouds from 2D to 3D space. Finally, a position map is calculated by combining the depth map and the intrinsics projection table, giving as a result the 3D positions of the point cloud's vertices (Figure 71).

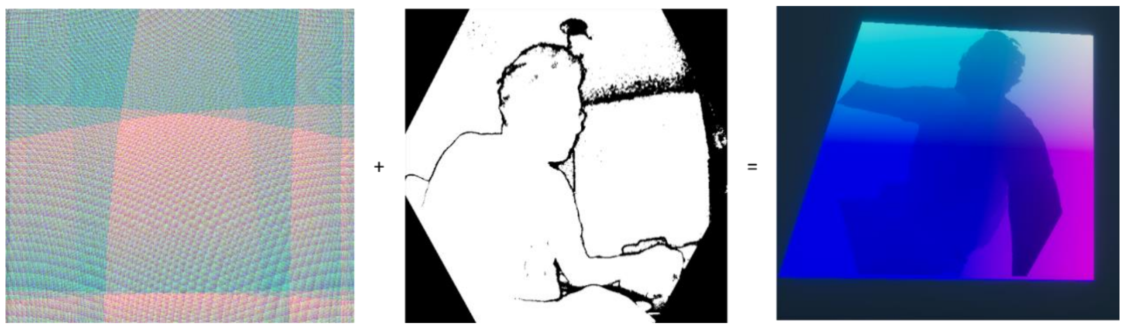


Figure 71: 2D to 3D RGB-D image mapping. From left to right: Intrinsics map, depth map, the resulting position map.

The position map is represented using floating point values, in this case, it ranges from -1.0 to 1.0 and 32 bits are needed to accurately represent the coordinates in 3D space which is not compatible with 2D video codecs. For that reason, the 32-bit values are split into two 16-bit images making it is possible to take advantage of existing video pipelines. To achieve this, each geometrical value is represented as a 16-bit chain, the first 8 bits are placed in one channel of the auxiliary image, then the second part is stored in the following channel until all three channels are remapped to the auxiliary images. The Figure 72 presents visually how this process is performed.

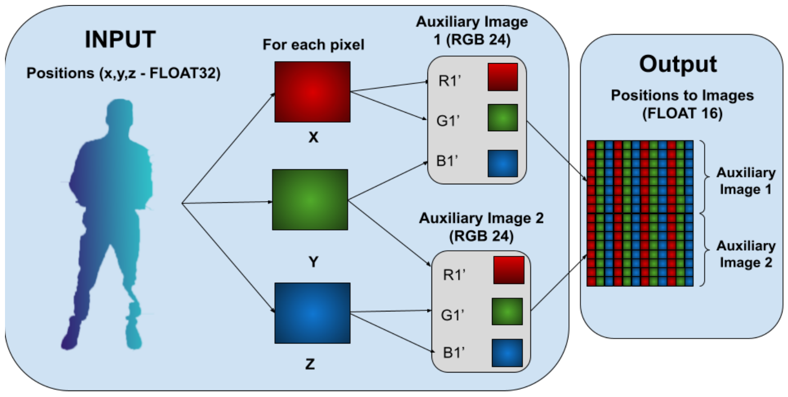

INPUT
Positions (x,y,z - FLOAT32)
For each pixel
X
Y
Z
Auxiliary Image 1 (RGB 24)
R1'
G1'
B1'
Auxiliary Image 2 (RGB 24)
R1'
G1'
B1'
Output
Positions to Images (FLOAT 16)
Auxiliary Image 1
Auxiliary Image 2

Figure 72: Data splitting for double precision image-based compression.

6.3.2 Experimental Setup and Evaluation

An experimental environment was developed for testing. This setting was built around a point cloud sequence with each frame consisting of 800k points. Figure 73 presents how the input is transformed and reconstructed. First, the input point cloud is stored in two different images, one has all the colour information and the other represents the 3D positions of the point cloud. Next, the position map is computed and stored in an auxiliary image that is prepared to be compressed and streamed over the network. Finally, by doing the inverse process, it is possible to reconstruct the point cloud without losing information.

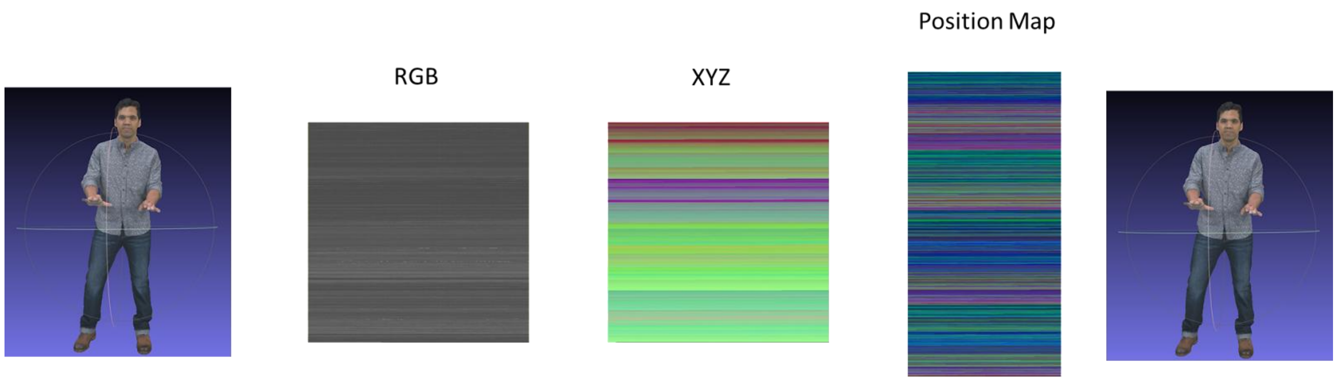


Figure 73: Input point-cloud transformation to 2D representation and 3D remapping.

6.3.3 Real Time Setup

For real-time applications three RGB-D cameras are used to capture the volumetric representation of a person. The capture is performed at 15 frames per second, rendering volumes with approximately 700k points. By offloading most of the computations to the GPU it is possible to reach interactive frame rates and dense point cloud representation. At this moment, 120 Mbps are required to distribute the volumetric content.

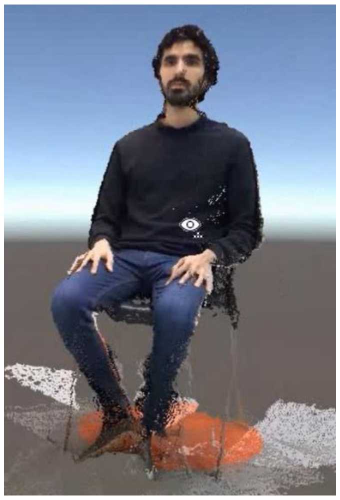


Figure 74: Real-time point cloud capture at 15 fps with 750K points per frame.

Table 4: Test parametres

CRF	0	10	17
Frame rate (fps)	15	15	15
Resolution (points)	~700k	~700k	~700k
Average latency (ms)	300	250	250
Bandwidth (Mbps)	120	80	60
Processing time (ms)	20	20	20

6.4 3D Face Reconstruction (RAI)

Creating realistic 3D models of human faces is a challenge that requires specific skills and the use of appropriate tools to ensure high quality results. We have worked on automating this task with a focus on human faces. In particular, the task consists in building a 3D model of the face of a reference character starting from 2D images.

To simplify and automate this complex process, we have developed a Python script which leverages some of the powerful features of FaceBuilder[93], a Blender plugin created by KeenTools. This allows for a highly automated 3D face reconstruction process, ensuring the creation of a high-quality base mesh of the face.

FaceBuilder is a plugin designed for Blender that simplifies the process of creating 3D models of human faces from a set of reference photos. This plugin stands out for its ability to automatically analyse the information present in the photos, leveraging artificial intelligence to configure camera virtual cameras. This makes it possible to work with photos of any size and to handle non-neutral facial expressions with ease. The choice of FaceBuilder for 3D face reconstruction is motivated by several key reasons, such as the automatic alignment of facial points. To further optimize this mesh creation task, we have developed a script that automates the manual FaceBuilder steps, eliminating the need to use Blender and greatly simplifying the process. Script functionality includes creating the head, inserting images, aligning virtual cameras for each image, and creating a texture. The main goal of the script is to offer a fully automated process for generating high-quality 3D meshes. This approach makes the entire face reconstruction process accessible to users of any experience level, thus helping to further simplify the overall workflow.

6.4.1 Stages of script execution

The process of building a 3D mesh of a face consists of several key steps. The script begins creating a new scene in Blender and removing the default elements to prepare the scene for the 3D head reconstruction automatized stage (Figure 75):

1. Adding a "New Head": The script begins with the addition of a "New Head," and it calls the function provided by FaceBuilder to begin the modelling process.
2. Loading Images: The script handles the loading of the images used as reference for the creation of the model.
3. Aligning Virtual Cameras: In this step, the virtual cameras associated with each image are aligned so that they are oriented, positioned and set with the necessary focal length to achieve a perfect match between the images and the 3D model mesh. Alignment is a critical process to ensure the quality of the final result, so we have developed a function to perform the alignment for each uploaded image.
4. Creation of Texture: The texture creation process is implemented in a function that, after performing the steps outlined above, extracts visual information from all the images used during the alignment phase. These images are "mixed" by a bake process on the mesh, generating a detailed texture that will be applied to the model.xr
5. Saving Results: Finally, a function manages the saving of results within a dynamically created folder in the same location as the folder with the images used to make the model.

[93] https://medium.com/keentools/facebuilder-for-blender-guide-cbb10c717f7c

The output files are:

• the texture in .png format

• the head mesh in .fbx and .glb formats

• the .blend file to be able to allow editing of details on the mesh

• a .json file containing information about the virtual cameras

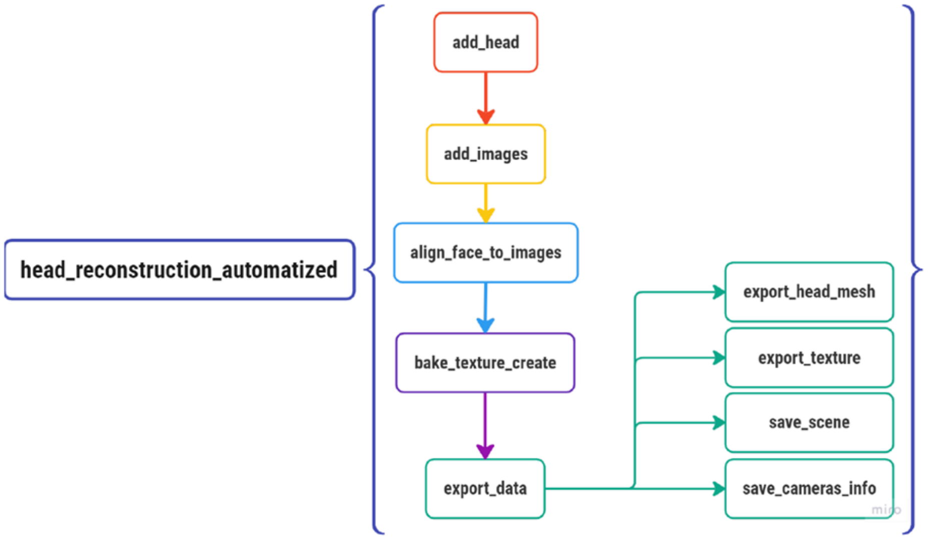


Figure 75: General scheme of the script 3D face building.

7 Authoring tool development and service communication

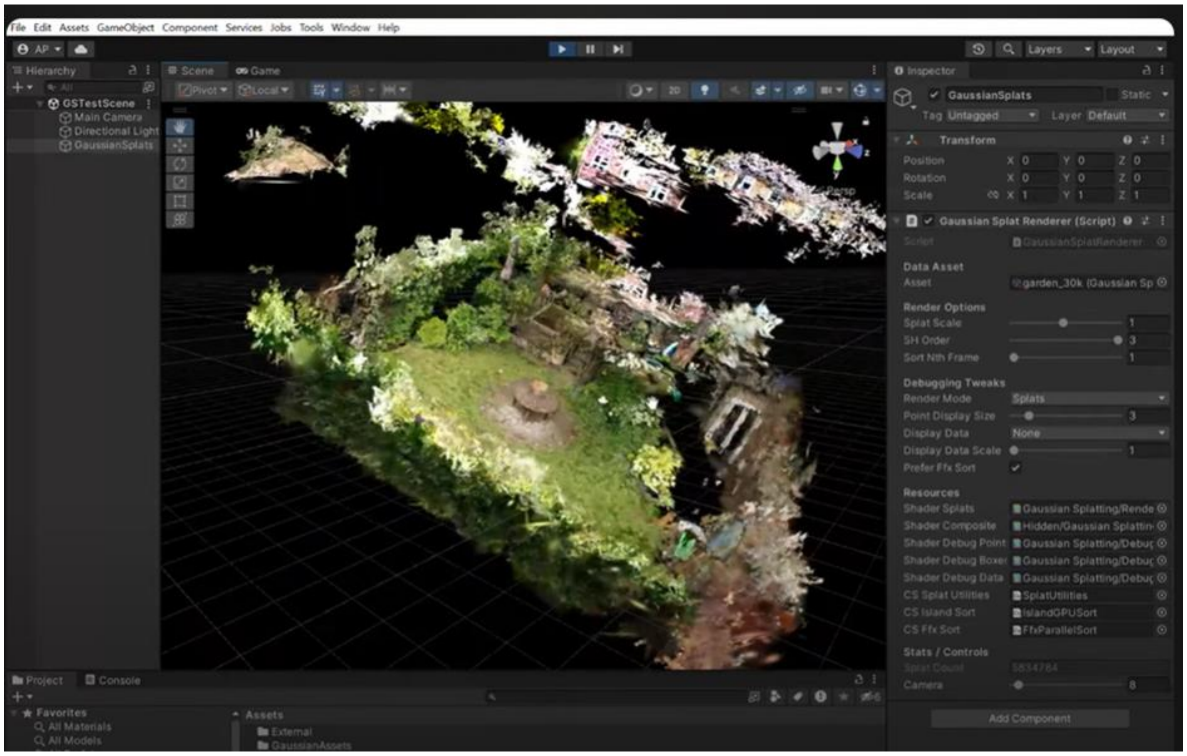
The various assets that can be found or generated on the XReco Platform can be used in a wide variety of applications. XReco considers four authoring tool experiences for compositing XR applications each requiring different sets of digital skills, taking into account professional workflows as well as more intermediate ones.

7.1 Unity-based authoring

The Unity-based authoring gives the user the most flexibility in creating very unique applications on a wide range of use cases. Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Worldwide Developers Conference as a Mac OS X game engine. The engine has since been gradually extended to support a variety of desktop, mobile, console and virtual reality platforms. It is particularly popular for iOS and Android mobile game development, is considered easy to use for beginner developers, and is popular for indie game development. The engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering, construction, and the United States Armed Forces.

For XReco we will add additional features to the editor to improve the workflow for the user. For example, we will provide an easy link to the Orchestrator Dashboard, so that the user has fast access to the assets. These assets can then easily (via Drag&Drop) be added to the scene. We will also make 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. The power of the Unity editor then allows for customization of every detail.

Also, the integration of other XReco services like FVV or Holoporation will easily be possible within our authoring tool.



GaussianSplats — Tag Untagged — Layer Default

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

Gaussian Splat Renderer (Script)
Script: GaussianSplatRenderer

Data Asset
Asset: garden_30k (Gaussian Sp

Render Options
Splat Scale: 1
SH Order: 3
Sort Nth Frame: 1

Debugging Tweaks
Render Mode: Splats
Point Display Size: 3
Display Data: None
Display Data Scale: 1
Prefer Ffx Sort: ✓

Resources
Shader Splats: Gaussian Splatting/Rende
Shader Composite: Hidden/Gaussian Splattin
Shader Debug Point: Gaussian Splatting/Debuç
Shader Debug Boxes: Gaussian Splatting/Debuç
Shader Debug Data: Gaussian Splatting/Debuç
CS Splat Utilities: SplatUtilities
CS Island Sort: IslandGPUSort
CS Ffx Sort: FfxParallelSort

Stats / Controls
Splat Count: 5834784
Camera: 8

Figure 76: NeRF scene in Unity editor

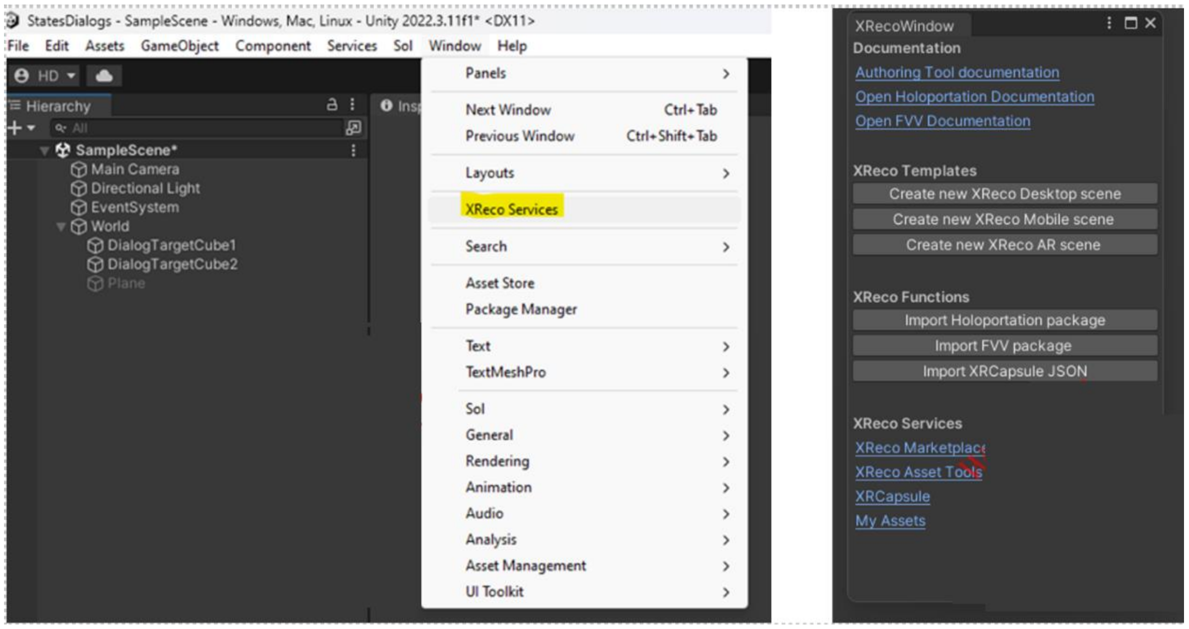


Figure 77: Quick access to XReco Services in Unity Editor

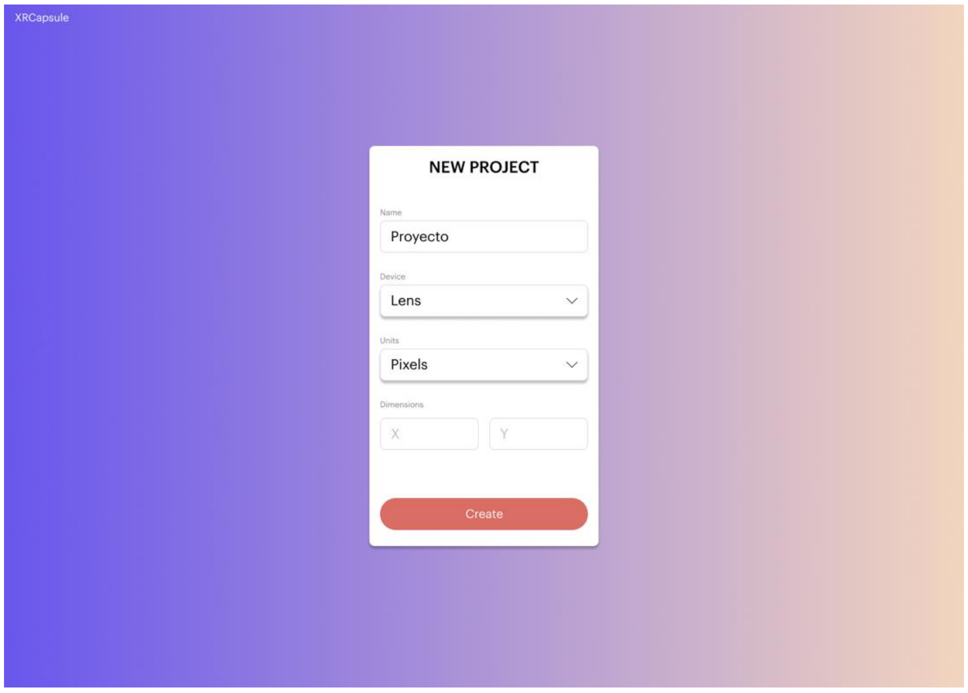
7.2 XR-Capsules concept and authoring

XR Capsules is an open web-based solution authoring tool that provides non-expert users with the ability to create their own simple XR experiences. To do this, this tool provides templates according to the interests and goals of the user. The templates ensure that the users do not need any technical knowledge in 3D skills. They serve the purpose of guiding the user within the appropriate boundaries so that they can successfully export their XR experience.

XR Capsules take their content from XReco. This way, users can work with their own assets or attain them from the marketplace. In no case XR Capsule consists of an asset-level creation solution.

7.2.1 User Journey

When a user opens XR Capsule in their web browser they log in. Then, they can choose to open a new project or load an existing project they have previously been working on. When pursuing a new project, the user is required to select a template from which to begin. Depending on the desired target device in which the experience will be consumed, the user will choose “XR”, “Smartphone” or “Workstation”. The user will select an available template. A template is technically a .json file that defines a set of triggers, and a set of constraints due to the playback device. The execution of the application defined by assets is played on the target device and represented by a .json and a series of assets. The actual application is made in Unity3D starting from the XR Capsule template.


XRCapsule

NEW PROJECT

Name: Proyecto

Device: Lens

Units: Pixels

Dimensions: X Y

Figure 78: Project selection window.

Some exemplary templates allow for:

- Creating an immersive newscast.
- Informative natural disaster experience.
- Immersive historical tour.
- Creating a virtual museum tour.
- Industrial maintenance simulation.
- Creating a tourism experience in a city.

In Section 7.2.4, the reader may find further information on exemplary use cases.

Depending on the selected template, the user will be presented with a customised workspace. A workspace is a 3D view of the assets (bounding box, cubes) in a defined 2D space. Common to all of them are the features of loading local assets or importing them from the XReco NMR. In doing so, the user can operate with the assets placing them on the workspace and according to specific triggers to the template.

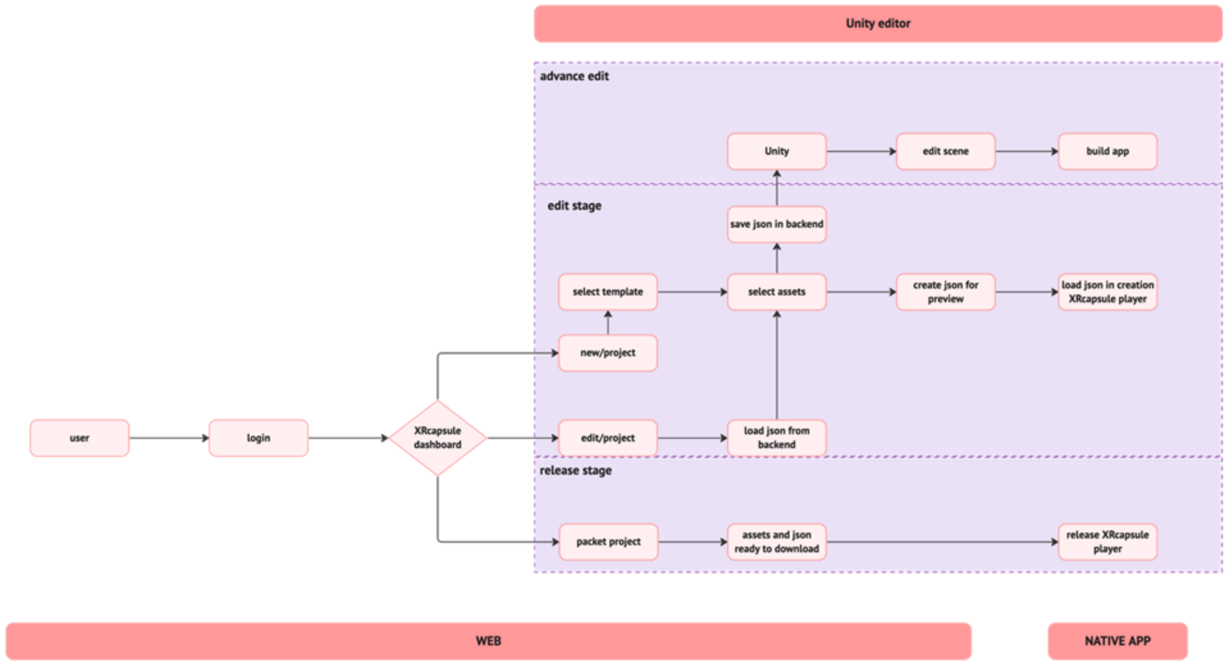


Figure 79: An XR-Capsule user journey.

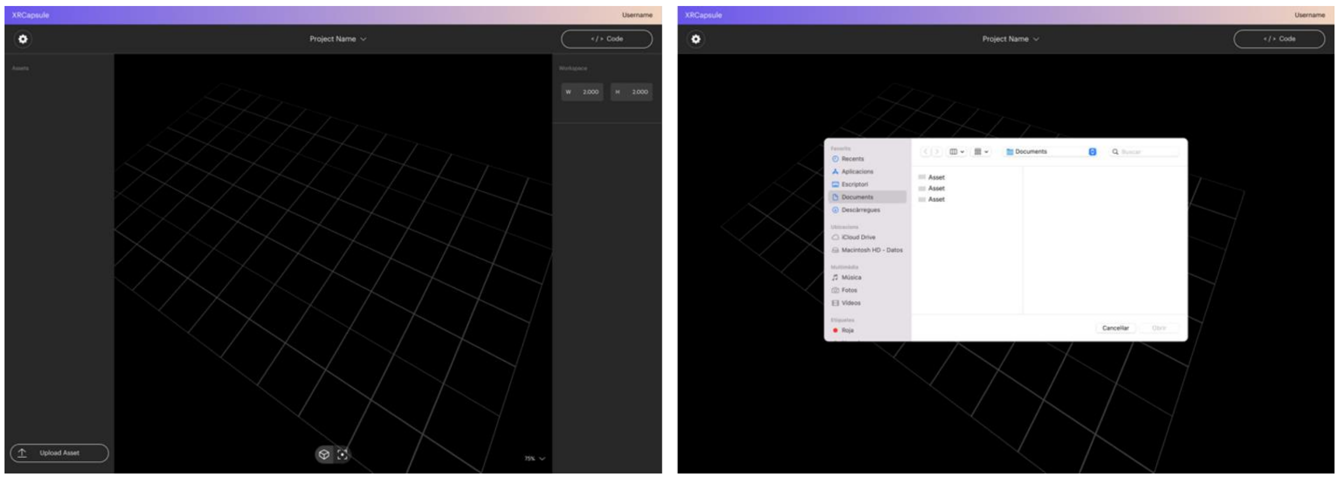


Figure 80: Left: An example of an empty custom workspace. Right: the Asset section window.

7.2.2 Examples of assets that can be opened within XR capsules

- 360º backgrounds
- 3D models
- Videos
- Trained NeRF algorithms
- Images
- Text
- Audio
- GPS/VPS location
- FVV
- Holoportation

7.2.3 Examples of triggers in the XR Capsule templates

- Time:
  - Time from start
  - Time from another trigger
- User location:
  - GPS location
  - User location/camera on specific workspace location
- External source:
  - Sensor trigger
  - Visual recognition of monument or object
  - Synchronisation between tv-streaming and mobile experience
  - User action

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

- Touch:
  - Rotation
  - Pinch
  - Keystroke

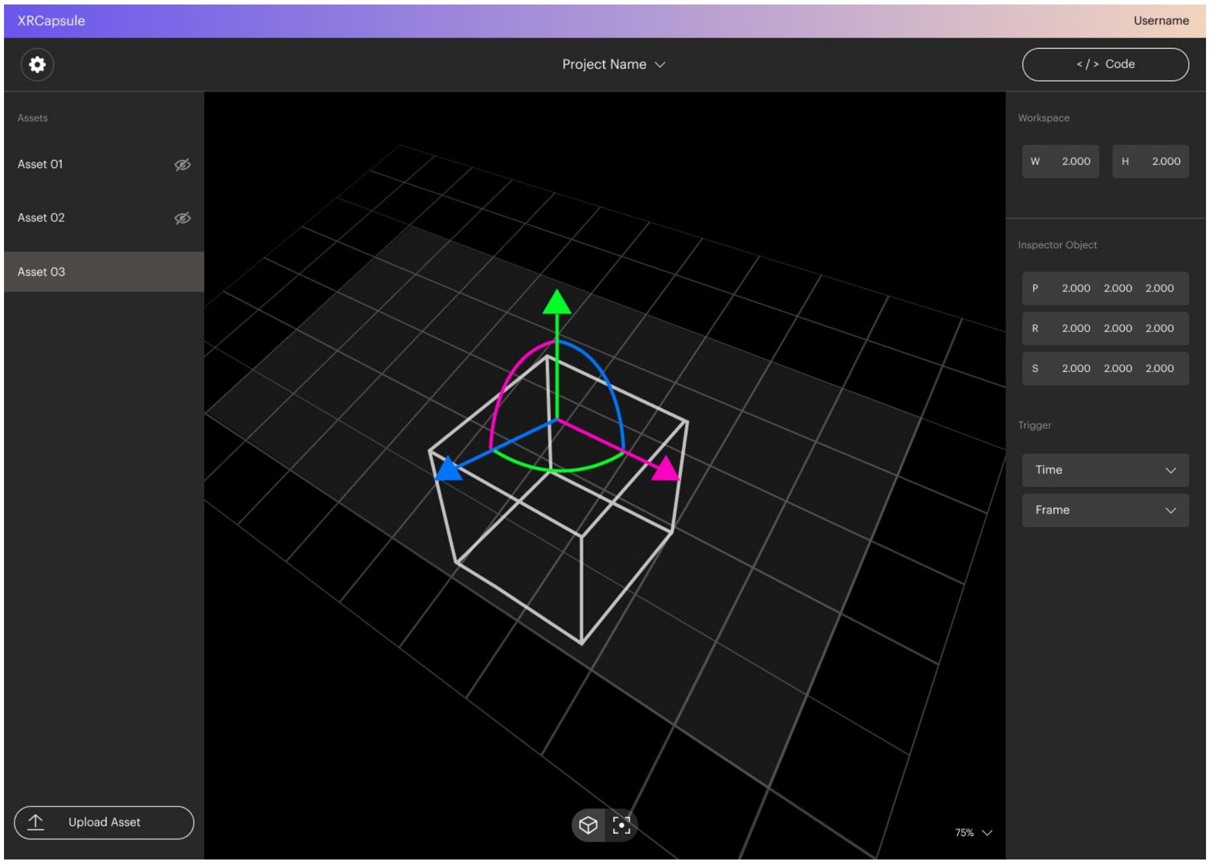


Figure 81: A sample asset in the workspace and the available controls for the given template.

When the user positions their asserts, and configures their experience, they can do two things: they can either export it to the target device and finish the process or export it into Unity.

In this last case, a more advanced user who would like to access expanded features could import their project into Unity and continue shaping their experience. The XR Capsule Unity plugin serves as a "recipe" to prepare the scenes for building across various solutions. It provides the ability to edit the scenes before compiling them. Specifically, the plugin modifies the .json exported from XR Capsule to include general Unity preset parameters required for compilation. This allows the Unity project to be built properly for different target platforms and devices.

By abstracting away these compilation details, the XR Capsule Unity plugin enables seamless exporting of projects created in XR Capsule into Unity-based XR solutions. Developers can then perform further customization and optimization in Unity while leveraging the simplicity of authoring content initially in XR Capsule. This streamlined workflow enables rapid development of XR experiences for multiple platforms.

7.2.4 Potential use case examples

7.2.4.1 Informative natural disaster experience

- The user creates a project and selects the "virtual production" template
- Imports 3D models (from XReco) and assets related to the natural disaster
- Recreates the scene of the event by positioning assets
- Configures proximity triggers to play events.

7.2.4.2 Immersive historical tour

- The user creates a project and selects the "AR smartphone" template
- Imports 3D models, images and assets from different historical periods
- Positions assets along a touch trigger
- Adds texts and audio with relevant information from each period
- Configures triggers on each asset for interacting and playing its content

7.2.4.3 Virtual museum tour

- The user creates a new project in XR Capsule and selects the "Virtual Museum" template
- Imports 3D models of the rooms and works of art from XReco
- Positions the assets replicating the actual layout of the museum
- Adds informational panels as images or text
- Configures proximity triggers so audios with additional information about the works are played

7.2.4.4 Industrial maintenance simulation

- The user creates a project and selects the "Industrial Training" template
- Imports complex 3D machinery model from XReco
- Positions the model in the workspace
- Adds interaction points on the machine associated with maintenance procedures
- Configures proximity triggers at points to display relevant information

7.2.4.5 A tourism experience in a city

- The user creates a project and selects the "Tourist Route" template
- Imports 3D models of iconic buildings and city assets
- Positions the assets replicating the real skyline of the city
- Adds 360° videos at points of tourist interest
- Configures geolocation triggers to display contextual information about places

7.2.5 Software stack

XR Capsule consists of several components that work together to enable easy XR content creation:

Frontend Interface: The frontend is a web-based interface built using Three.js. It provides the main user-facing application where templates can be selected, and assets arranged into scenes. The goal is to support execution on different platforms through web technology.

Backend Services: The backend connects the frontend interface to the other services. Key functions include:

- Handling action from the frontend.
- Managing user access and sessions.
- Storing/retrieving user projects.
- Importing assets from XReco's repository.
- Executing media transcoding and processing.
- Generating Unity-compatible .json scene definitions.
- XR Capsule Player: This is a Unity application that runs the experiences exported from XR Capsule projects on the target devices. It interprets the .json scene definition along with the packaged assets. The player app would be customized for each partner's demonstration applications.
- XR Capsule Unity Plugin: This plugin allows exporting XR Capsule projects directly into Unity for further editing before building. It essentially prepares the scene for compilation across different XR platforms/devices.
- NMR: An area for partners to store their compiled demonstration applications built using the XR Capsule Player and customized for their solution.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

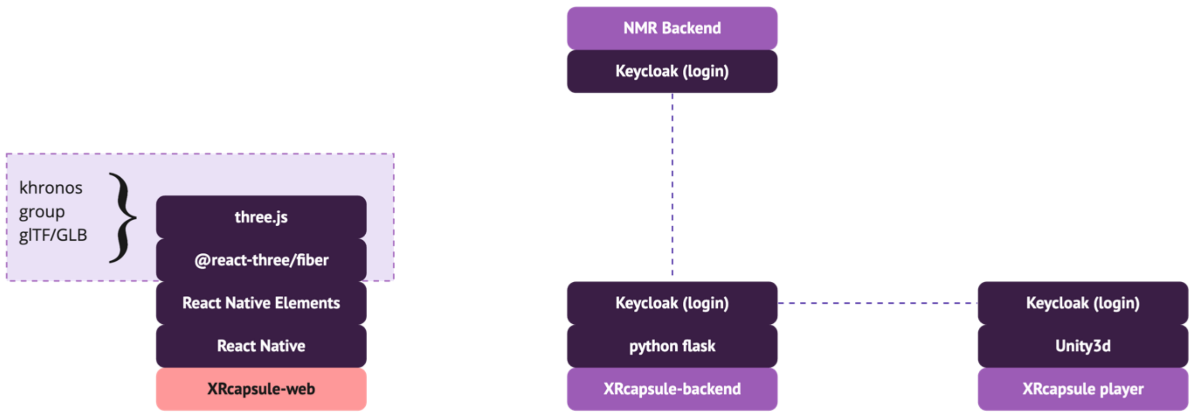


Figure 82: XR Capsule Software Stack

XR Capsule provides a valuable solution to simplify immersive content creation for non-technical users. Its web-based templates and integration with the XReco asset repository empower users to craft their own XR experiences. Meanwhile, the ability to export to Unity ensures the full capabilities of a leading XR engine are available for more advanced development. By leveraging pre-built templates on the frontend and seamless interoperability with Unity on the backend, XR Capsule successfully bridges the gap between ease-of-use and customizability. It democratizes access to XR technology while providing an on-ramp for users to learn and iterate. As immersive content creation becomes pivotal across industries, XR Capsule represents an important step in putting simple yet powerful tools into the hands of everyday users. Its approach of balancing simplicity and customizability is key to wider XR adoption.

As XR Capsule evolves its features and capabilities, it will continue leveraging integration with the robust Neural Media Repository underpinning the XReco platform. By giving users access to this ever-growing asset database and media processing engine, XR Capsule is positioned to scale not just in terms of functionality but also content volume and diversity. The value of XRCapsule as an easy on-ramp for consumers to craft custom immersive worlds will be greatly amplified through this tight coupling with XReco's enterprise-grade media management solution. Democratizing both technology and content in tandem is central to achieving mainstream XR adoption.

7.3 Orchestrating between user interfaces and services

The Orchestrator Dashboard plays a crucial role in seamlessly integrating various XReco services, ensuring users' access to a diverse array of UI components and services within the XReco platform. Its primary goal is to empower users to browse and search for assets, mainly images and videos, both within and outside the XReco repository, to serve as inputs for reconstruction services. Simultaneously, the Orchestrator Dashboard provides access to supplementary content creation tools within the XReco context, including the Authoring Tool. The high-level architecture of the Orchestrator Dashboard can be seen in Figure 83.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

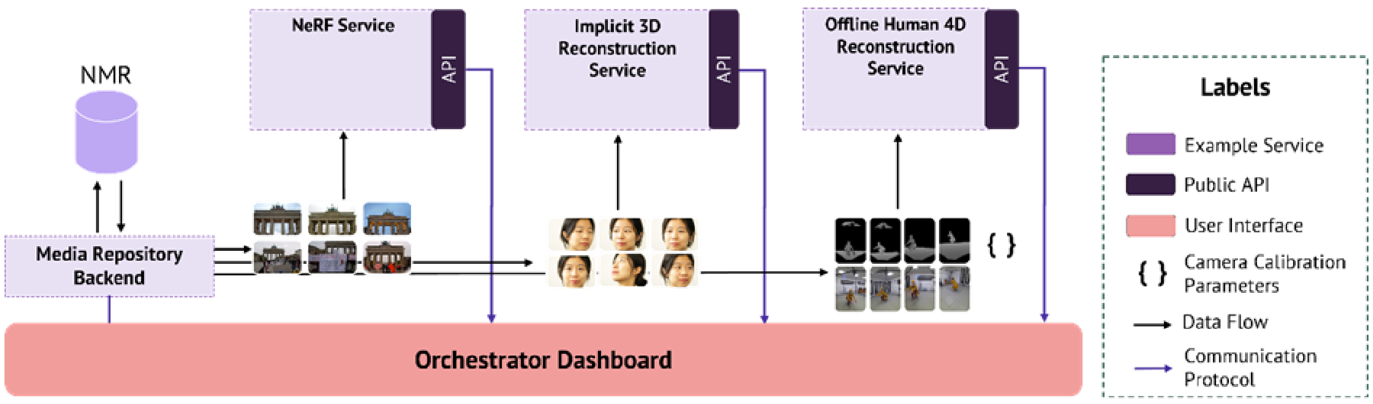


Figure 83: Orchestrator Dashboard high-level architecture.

Beyond the functionality of content browsing and search, users can also organize assets into content baskets. This feature empowers users to centralize their assets in one location, streamlining the process of utilizing the XReco reconstruction services.

Key functionalities of the Orchestrator Dashboard include:

- **Browsing and searching content:** Users can explore and search content from organizational repositories (e.g., RAI's repository) or external repositories (e.g., Wikimedia Commons). Assets can be grouped by adding them to content baskets.
- **Managing content baskets:** The Orchestrator allows users to handle multiple content baskets, which can be either discarded or edited.
- **XReco reconstruction services:** Users have the option to leverage XReco's reconstruction services for creating new 3D assets with customized configurations.
- **Access to other XReco applications:** The Orchestrator Dashboard provides users with access to other XReco applications for additional asset editing or scene creation, such as Unity3D.

The initial version of the Orchestrator Dashboard is currently under implementation, and its comprehensive workflow will be detailed subsequently. Upon accessing the Orchestrator Dashboard, users are required to log in with their credentials on the Login Page, as depicted in Figure 84.

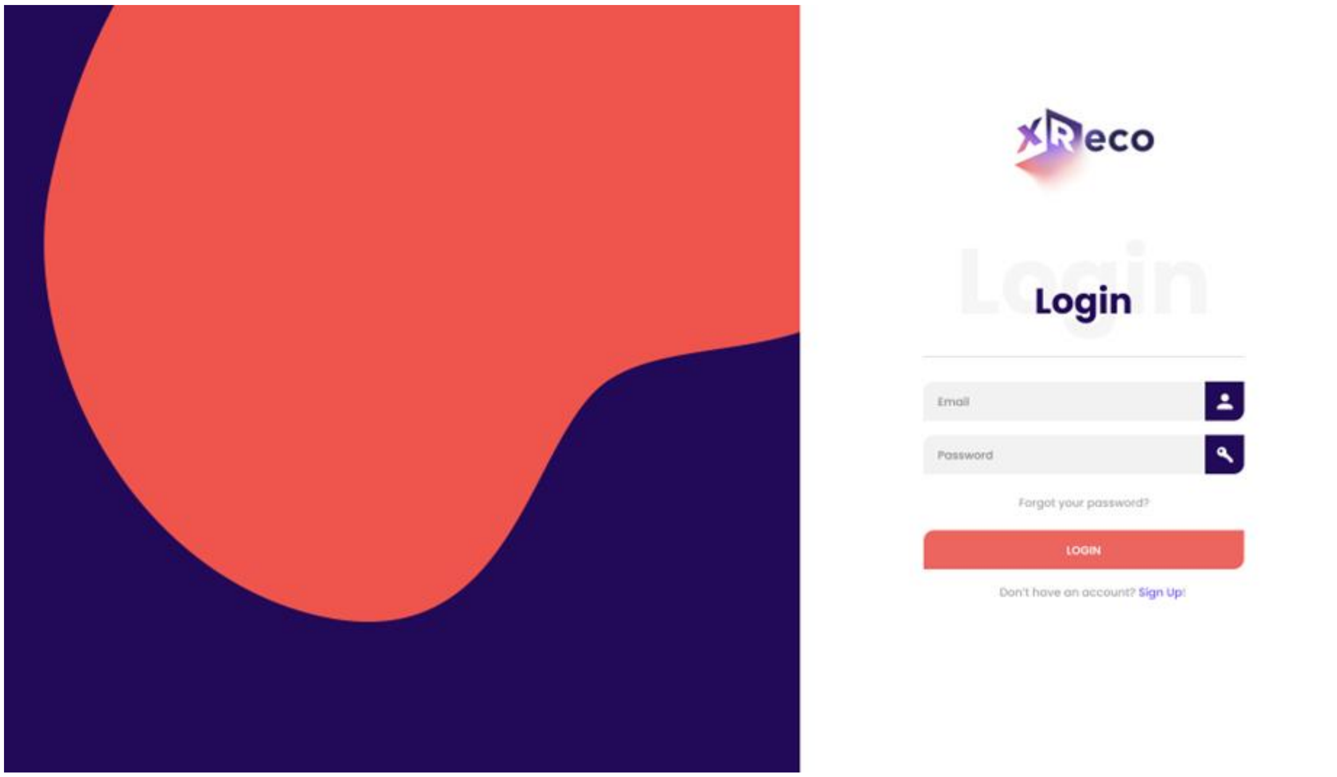


Figure 84: Orchestrator Dashboard: Login Page

Once logged in, users will have access to the main page of the Orchestrator Dashboard. On this page, users have access to most of the key functionalities of this component, with relevant sections highlighted in Figure 85, including the content basket management, services configuration, and asset browser sections.

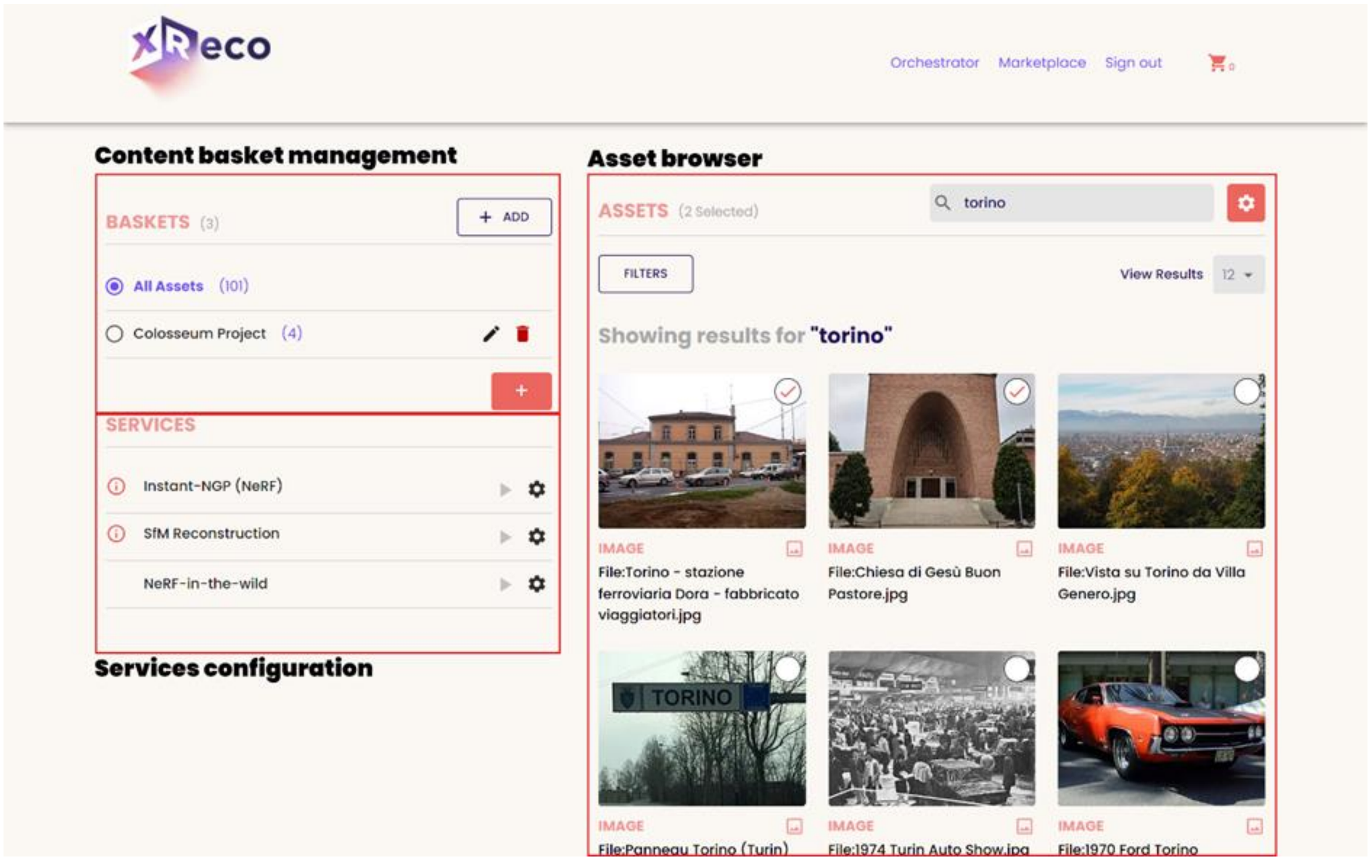


Figure 85: Orchestrator Dashboard: Main page overview.

In the asset browser section, users can search for assets inside or outside the XReco repository. In the content basket management section, users can create, edit, or delete content baskets. Here, it is also possible to add items to an already created basket; to do that, users only need to select assets on the asset browser and click the "ADD" button in the content basket management section.

In the content basket management area, users can also select which content basket should be visible on the asset browser. Selecting a basket in the content basket management area will display the corresponding items in the asset browser, as shown in Figure 86. Users can also remove items from a selected content basket.

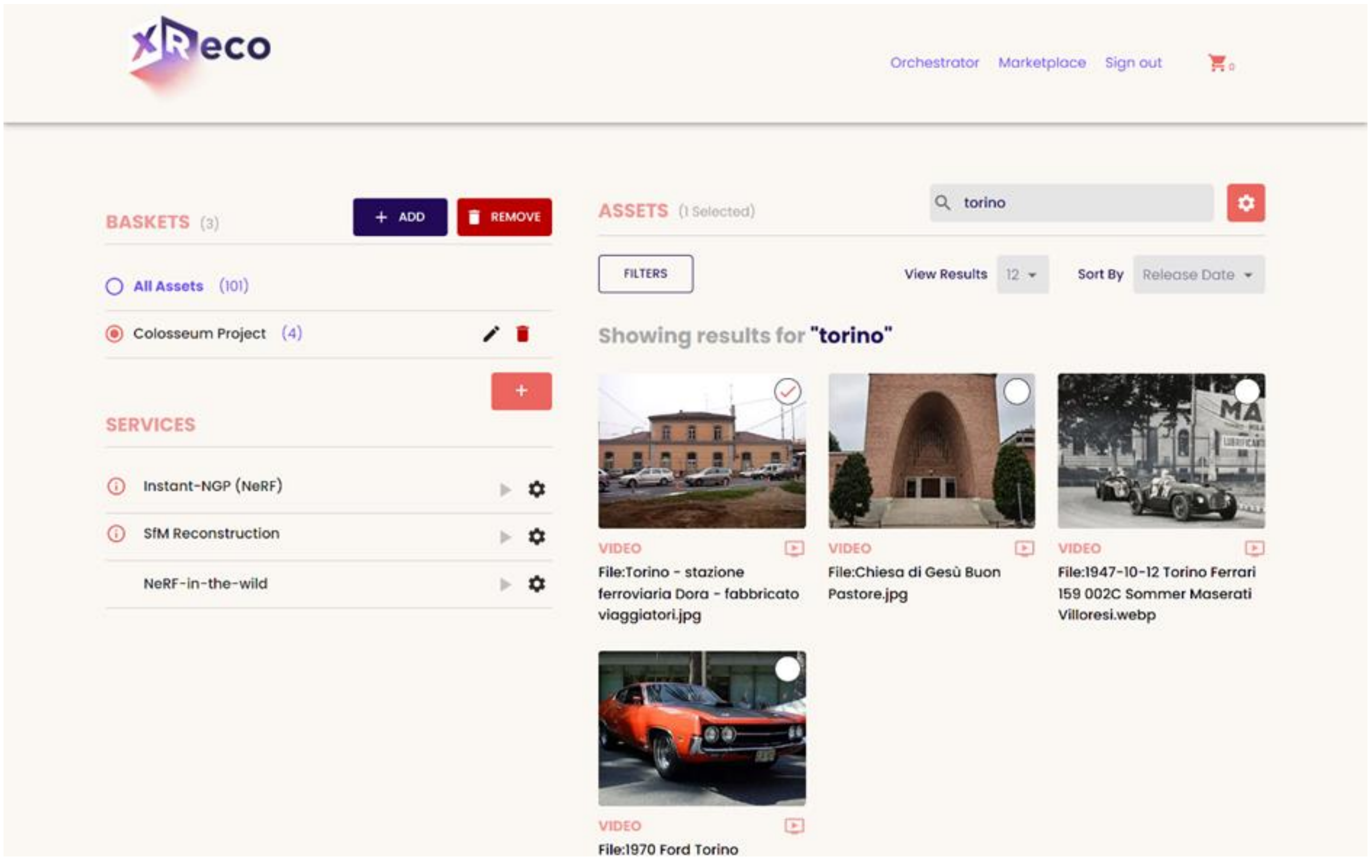
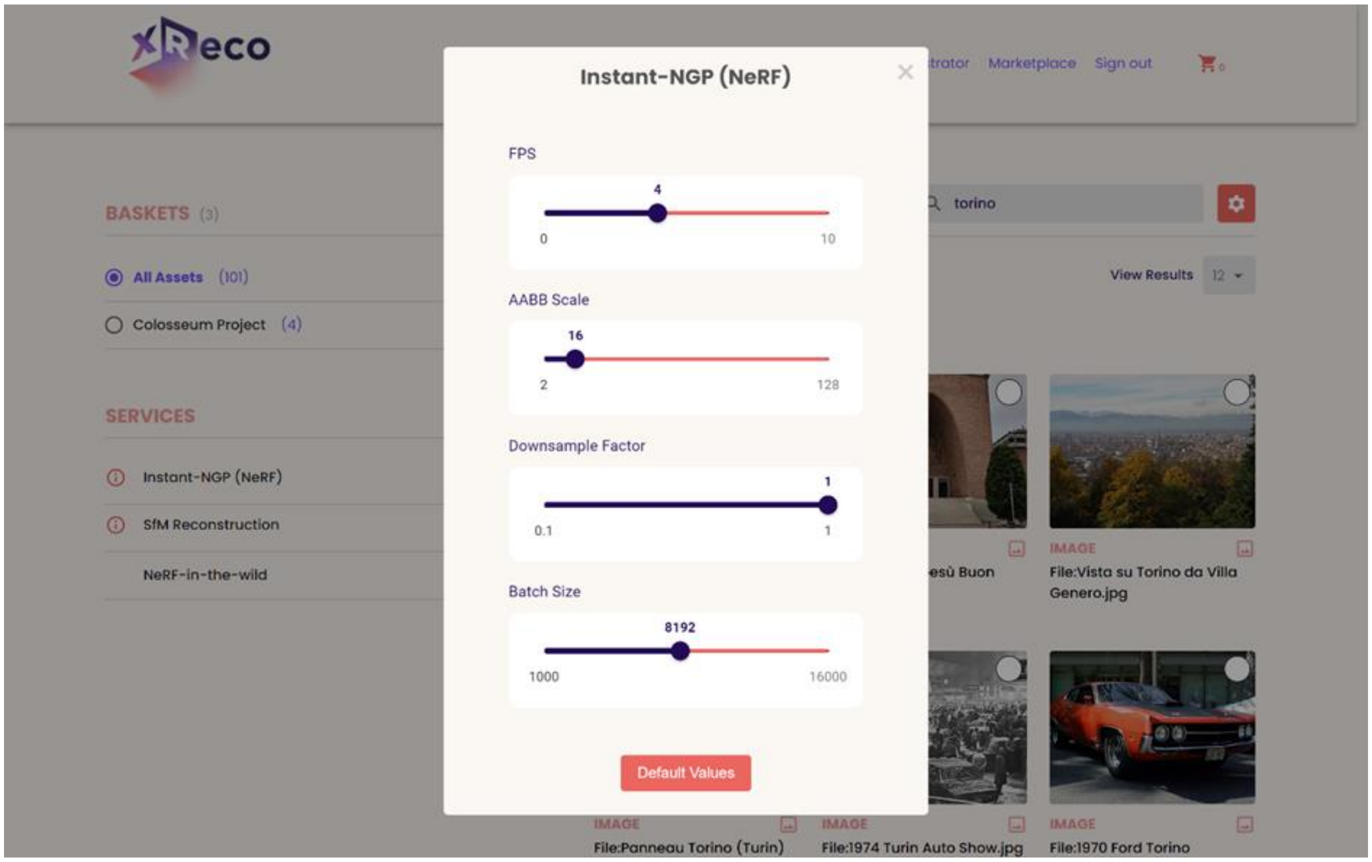


Figure 86: Orchestrator Dashboard: Selected basket.

Within the services configuration section, users have visibility into the XReco services available for invocation on the Orchestrator Dashboard. In this space, users can tailor the parameterization of each service, as illustrated in Figure 87, or opt to execute the services using default values. Following the selection of a content basket, users only need to click the "play" button. If the service inputs are accurate, the service will execute successfully, producing a newly generated 3D asset. In the event that the service inputs are incorrect, the Orchestrator will display an error message indicating the issues with the inputs.

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.



Instant-NGP (NeRF)

FPS: 4 (0–10)

AABB Scale: 16 (2–128)

Downsample Factor: 1 (0.1–1)

Batch Size: 8192 (1000–16000)

Figure 87: Orchestrator Dashboard: Service configuration.

7.4 CMS-based authoring

Location-based XR experience authoring in XReco is realised by ZAUBAR's editor, which represents a cutting-edge integration of augmented reality (AR) and artificial intelligence (AI) within a content management system (CMS) framework. This editor, part of the ZAUBAR platform, is designed to facilitate the creation and management of AR experiences with a focus on geospatially anchored content.

Central to the editor's functionality is its ability to seamlessly blend AR with real-world locations. This is achieved through a sophisticated geospatial 3D engine viewer and a 3D creator. These tools enable users to visualize and craft AR experiences that are intricately tied to specific physical locations, enhancing the sense of immersion and engagement.

The CMS aspect of the editor plays a pivotal role in the planning and execution of AR tours. Users can employ a web-based CMS to meticulously plan the AR experience, ensuring that each location within the tour is enriched with unique, interactive content. This content is not only specific to each location but is also tailored to enhance the storytelling and experiential aspects of the AR tour.

Moreover, the editor boasts a web-based AI mural maker. This innovative feature allows for the artistic and creative augmentation of physical spaces within the AR experience. It leverages AI to transform images into captivating murals, adding a layer of artistic expression to the AR content.

In terms of content creation and customization, the editor provides a robust and intuitive interface. This interface facilitates the fine-tuning of AR content directly on location, allowing creators to adjust and perfect their experiences in real-time. This level of control and customization is key to delivering high-quality, engaging AR experiences that resonate with users.

Additionally, the document touches on the generative AI pipeline integral to the editor. This pipeline encompasses a series of steps such as image upload, colorization, and canvas expansion, culminating in the creation of dynamic and visually striking AR elements. These elements are crucial for crafting immersive and interactive AR experiences.

In summary, the XReco editor, as part of the ZAUBAR platform, stands out for its sophisticated integration of AR, AI, and CMS capabilities. Its focus on geospatial anchoring, coupled with an advanced set of tools for content creation and customization, positions it as a powerful tool for crafting immersive AR experiences that are deeply connected to real-world locations. The editor's user-friendly interface and AI-enhanced features facilitate a creative and engaging process for both creators and end-users, setting a new standard in the realm of AR content creation and management.

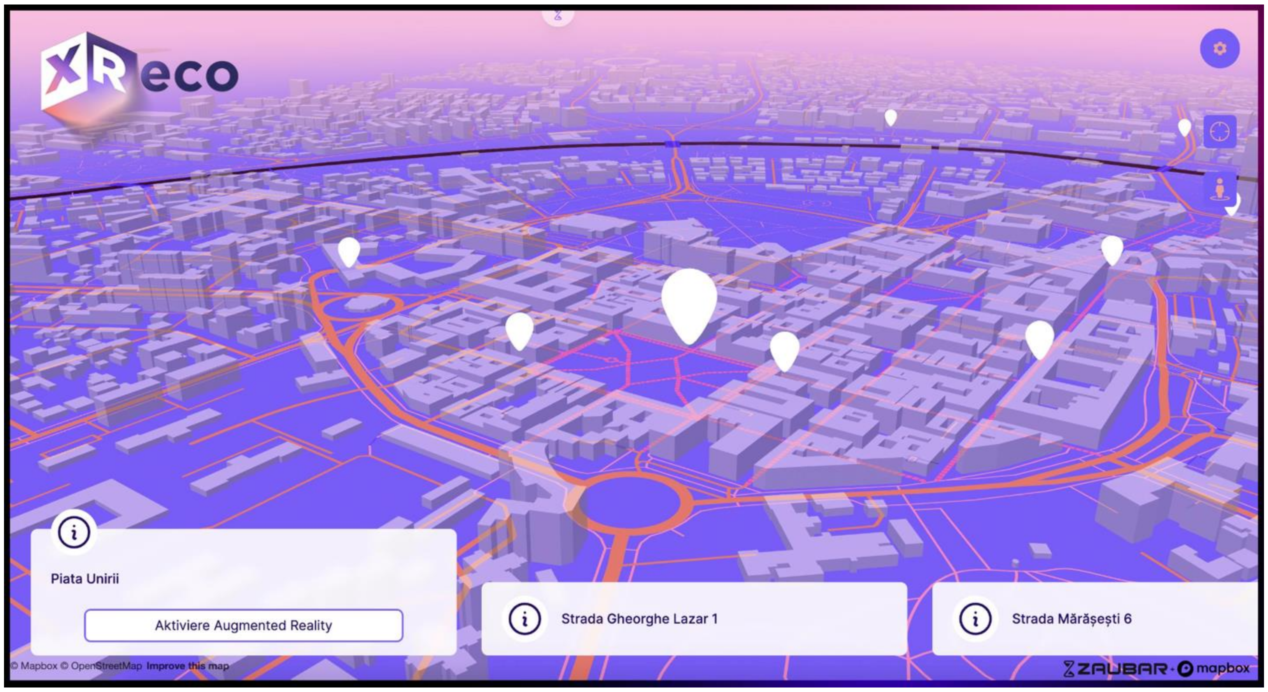


Figure 88: CMS-Editor: 3D Map overview (ZAUBAR)

XReco is a Horizon Europe Innovation Project co-financed by the EC under Grant Agreement ID 101070250.
The content of this document is © by its authors. For further information, visit xreco.eu.

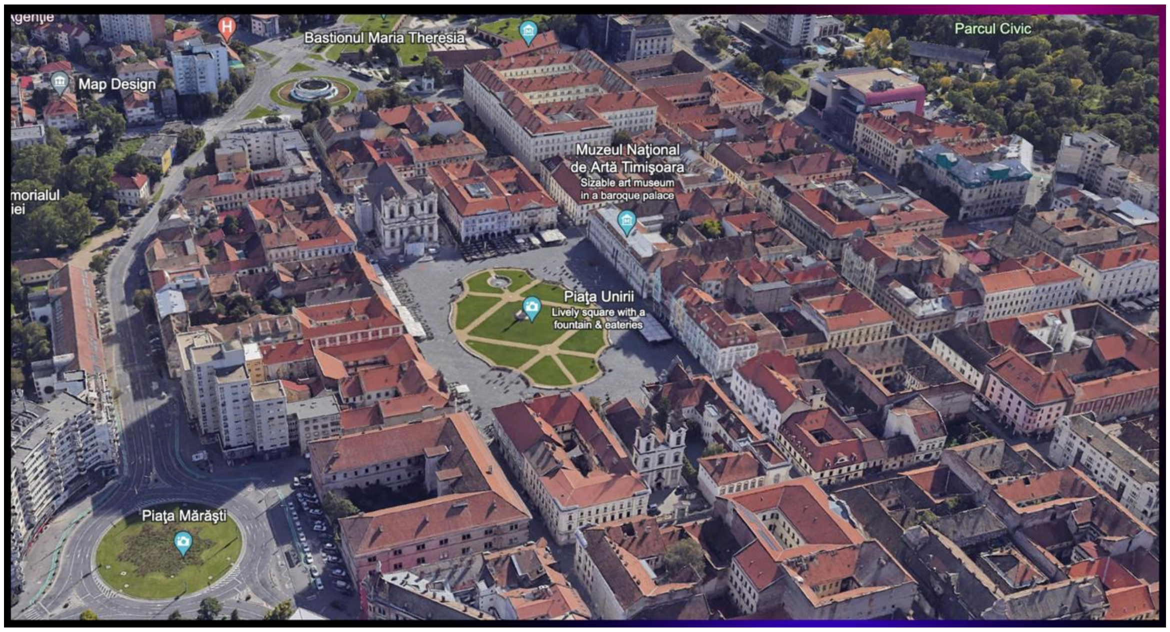

Bastionul Maria Theresia
Parcul Civic
Map Design
Muzeul Național de Artă Timișoara
Sizable art museum in a baroque palace
morialul iei
Piața Unirii
Lively square with a fountain & eateries
Piața Mărăști

Figure 89: CMS-Editor: 3D map overview with Google Tiles (ZAUBAR)

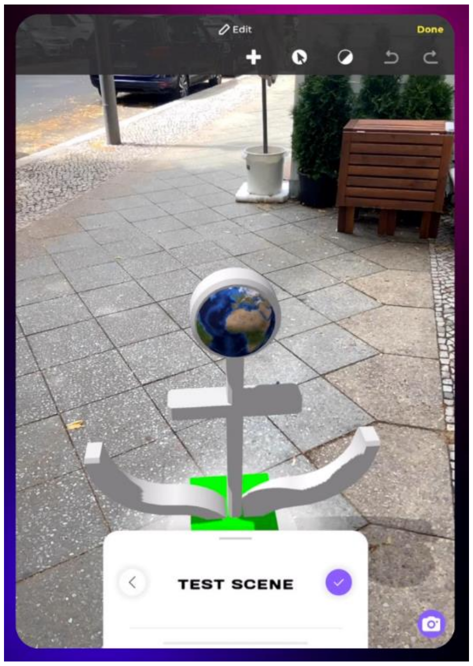


Figure 90: AR editor for placing an anchor.

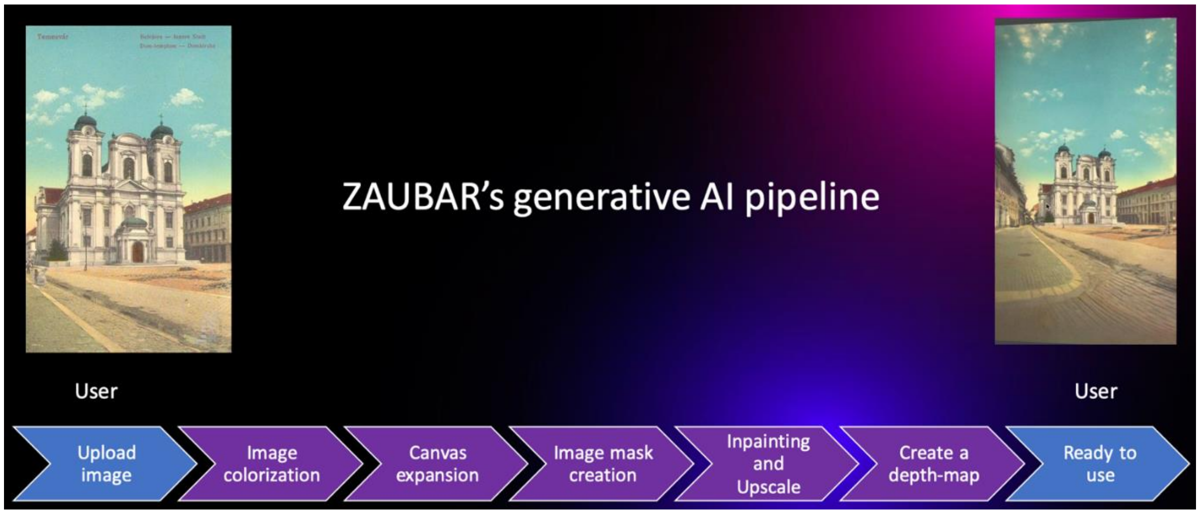
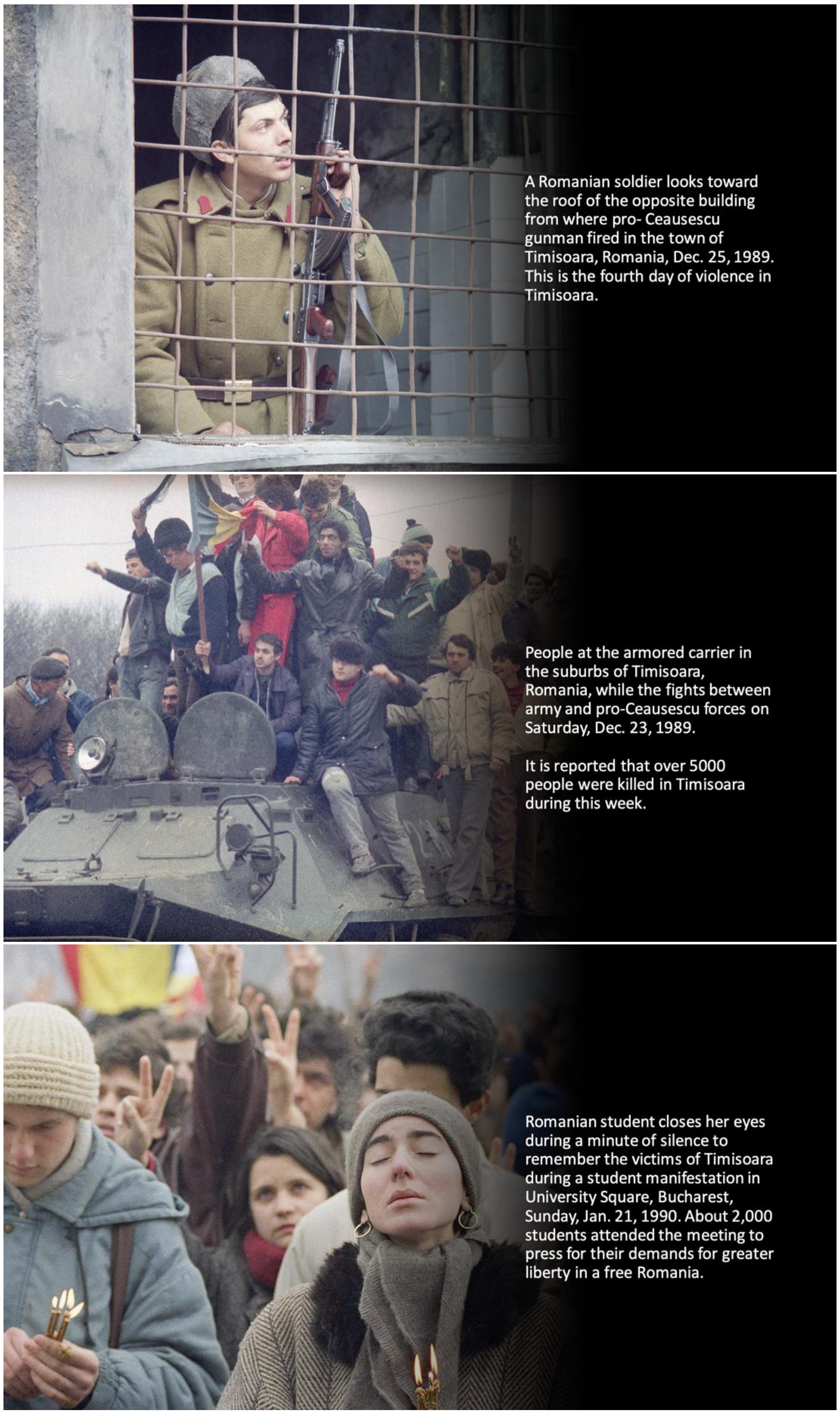


Figure 91: ZAUBAR's Generative AI pipeline to create AR portals.


A Romanian soldier looks toward the roof of the opposite building from where pro- Ceausescu gunman fired in the town of Timisoara, Romania, Dec. 25, 1989. This is the fourth day of violence in Timisoara.

People at the armored carrier in the suburbs of Timisoara, Romania, while the fights between army and pro-Ceausescu forces on Saturday, Dec. 23, 1989.

It is reported that over 5000 people were killed in Timisoara during this week.

Romanian student closes her eyes during a minute of silence to remember the victims of Timisoara during a student manifestation in University Square, Bucharest, Sunday, Jan. 21, 1990. About 2,000 students attended the meeting to press for their demands for greater liberty in a free Romania.

Figure 92: Mock-up screens of an AR experience about Romania in 1989.

The concept of historical time travels in AR, especially focusing on Romania in 1989, can be a riveting application of the XReco editor. Envision an AR experience that transports users back to the significant events of the Romanian Revolution of 1989, a pivotal moment in the country's history. Using historical photographs and geospatial anchoring, this AR journey could recreate scenes from key locations in cities like Timisoara and Bucharest.

As users navigate these historical sites with their AR-enabled devices, they would see overlaid images from 1989, bringing to life the intense atmosphere of the revolution. The photos, possibly sourced from archives or personal collections, would not only add a visual dimension but also an emotional and educational layer to the experience. Through this immersive technology, users could witness the unfolding of the revolution, understand the context, and feel a deeper connection to Romania's past.

Incorporating narratives and testimonials from those who lived through these events could further enrich the experience. These stories, embedded within the AR environment, would provide personal insights and reflections, offering a more comprehensive understanding of the historical significance of the Romanian Revolution. This fusion of technology and history would not only serve as a powerful educational tool but also as a poignant reminder of the country's journey to freedom and democracy.

8 Outlook

This deliverable presented the technical developments of XReco within WP4. It presented a detailed list of backend and frontend solutions for content searching, filtering, monitoring, as well as querying from XR interfaces. Additionally, NeRF algorithms that consider the content centralisation of assets in the XReco repositories were described, for general and in-the-wild scene fitting, as well as in a human-centred context. Moreover, SfM and data enhancement methodologies are presented as part of the asset aggregation and optimisation task (T4.3). Furthermore, human-centred reconstruction and volumetric streaming approaches were described for 3D capturing and streaming of humans. Finally, the authoring tools that will be used for realising XReco’s use cases were presented.

The first iterations of the technologies described in this deliverable at this point in the project’s lifetime, are finalized to be deployed as container applications in a distributed micro-service environment. Current efforts are focused on deploying them into docker containers in order to realise a first integrated version of XReco and to be evaluated by the end-users of the project by creating XR content and use case applications.

The next steps after their integration will involve further extending them, as well as adding other technologies that will further aid in content creation, targeting faster workflows that enhance further the user experience. More specifically, asset aggregation algorithms (such as NeRF and 3D reconstruction algorithms that aggregate data from different sources) will be enhanced with data evaluation algorithms for linking to T3.4 via feedback loops, calculating the value of each training sample used. This will provide valuable information for distributing royalties to each different content source. In addition, other encoding schemes and processing solution will be investigated for achieving faster algorithm convergence. Furthermore, APIs for Orchestrator and service communication will be finalized to realise a first version of the XReco platform.