electronics

Article

Towards Hyper-Personalized Travel Planning: A Multimodal AI Agent with Integrated Neural Rendering for Immersive Itineraries

José Márquez-Algaba [1,*], Pablo Vicente-Martínez [1,*], Emilio Soria-Olivas [2], Manuel Sánchez-Montañés [3], María Ángeles García-Escrivà [4] and Edu William-Secin [5]

[1] SPV Scala, 35100 Gran Canaria, Spain
[2] Intelligent Data Analysis Laboratory, Department of Electronic Engineering, Universitat de València, 46100 Valencia, Spain; emilio.soria@uv.es
[3] GNB, Department of Computer Science, Universidad Autónoma de Madrid, 28049 Madrid, Spain; manuel.smontanes@uam.es
[4] Fundación Canaria Living Lab, 35007 Gran Canaria, Spain
[5] Department of Economics and Business Management, Institute of Tourism and Sustainable Development, TIDES, Universidad Las Palmas de Gran Canaria, 35001 Las Palmas, Spain
* Correspondence: c.datos9@salascala.com (J.M.-A.); c.datos12@salascala.com (P.V.-M.)

Abstract

The digital transformation of the tourism industry faces a dual challenge: the fragmentation of data across platforms and the lack of immersive "try-before-you-buy" experiences. While Large Language Models (LLMs) have revolutionized information synthesis, they typically lack real-time visual verification capabilities. This paper proposes a novel, multimodal AI Agent architecture that integrates advanced natural language planning with photorealistic 3D visualization. We present a system where a conversational agent, powered by Gemini 2.5 Flash, orchestrates a suite of dynamic tools to build structured travel itineraries (flights, hotels, activities) while simultaneously deploying a neural rendering engine. This engine utilizes a modular Structure-from-Motion (SfM) pipeline feeding into 3D Gaussian Splatting (3DGS) to render navigable, high-fidelity digital twins of hotel facilities directly within the chat interface. Positioned as a Technology Readiness Level 4 (TRL 4) proof of concept (PoC), this work demonstrates the technical feasibility of the multimodal integration between conversational logic and automated visual synthesis. The results demonstrate the technical feasibility of a pipeline that dynamically binds LLM inference to 3D spatial data, providing a foundation for high-fidelity, interactive travel consultancy.

Keywords: AI agents; Large Language Models; tool calling; 3D Gaussian Splatting; neural rendering; digital tourism; 3D reconstruction

Check for updates

Academic Editor: Dimitris Apostolou

Received: 2 February 2026
Revised: 25 February 2026
Accepted: 25 February 2026
Published: 10 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The tourism industry is undergoing a paradigm shift driven by the convergence of Artificial Intelligence (AI) and immersive technologies. Traditionally, travel planning has been a fragmented process, requiring users to navigate multiple platforms to align flights, accommodations, and activities [1]. While online travel agencies (OTAs) have streamlined bookings, they often lack the personalized, consultative touch of a human agent. Crucially, the visual aspect of booking—validating the quality of a hotel room or facility—remains dominated by static 2D grid galleries, which frequently fail to convey spatial context or realistic lighting.This highlights the critical need for navigable 3D digital twins; unlike 2D

images, 3D visualizations allow for spatial verification of layouts and ambiance, offering a true "try-before-you-buy" experience essential for high-stakes hospitality decisions.

The emergence of **Large Language Models (LLMs)** has enabled the creation of **Intelligent Agents** capable of reasoning, planning, and executing complex tasks through natural language [2]. These agentic systems move beyond passive chatbots, leveraging **dynamic tool calling** and structured schemas to actively query databases and construct complex travel itineraries. However, a significant gap remains: current text-based agents operate in a visual vacuum, capable of recommending a "luxury suite with a sea view" but unable to provide real-time, immersive visual validation to the user.

To address this visual deficiency, we turn to advancements in 3D reconstruction. While Neural Radiance Fields (NeRF) [3] offered revolutionary photorealistic view synthesis, their high computational cost made real-time interaction prohibitive. **3D Gaussian Splatting (3DGS)** [4] emerged as a groundbreaking alternative, achieving state-of-the-art visual quality with real-time rendering speeds. However, the quality of a 3DGS reconstruction is highly dependent on the initial camera poses provided by a Structure-from-Motion (SfM) algorithm. However, the deployment of 3DGS in the hospitality sector requires a seamless workflow from initial camera estimation to final volume synthesis. Establishing this integrated pipeline for complex indoor environments remains a considerable engineering challenge, demanding a framework that automates the **Structure-from-Motion (SfM) to 3DGS transition**.

To address the critical trade-off between Agent capabilities and visual immersion, we propose a unified, multimodal system that bridges the gap between **Agentic AI** and **Neural Rendering**. Our contribution is a hyper-personalized itinerary creator centered around a conversational agent, fully integrated with a novel, modular **SFM-3DGS Pipeline**. Positioned as a TRL 4 proof of concept, this system is powered by the following two main components:

1. A robust Agentic System Architecture that utilizes internal tool-calling and Pydantic schemas to orchestrate the transformation of unstructured user requests into structured travel frameworks. This layer acts as a logic-gate, ensuring that destination data and facility requirements are formatted correctly to trigger the subsequent visual synthesis pipeline.
2. A modular **SfM-3DGS Pipeline** designed to automate the 3D reconstruction process. This pipeline integrates a Structure-from-Motion (SfM) component to provide initial geometry for **3D Gaussian Splatting (3DGS)**, enabling the generation of digital twins for hotel facilities.

This integration provides a **Multimodal Interface** where the conversational Agent can contextually trigger and embed interactive 3DGS viewers within the conversation flow, enabling users to explore immersive representations of recommended facilities as the itinerary is being built. This work provides a functional validation of the integrated multimodal workflow, demonstrating the technical feasibility of combining agentic reasoning with automated visual synthesis for the future of digital tourism. The purpose of this study is not to measure UX improvements, but to prove that a pipeline that dynamically binds LLM inference results to 3D space works without failure. We also acknowledge the system's current boundaries, particularly the dependency of SfM on texture-rich environments versus modern minimalist or mirrored architectural settings.

The remainder of this paper is organized as follows: Section 2 reviews LLM agents, 3DGS, and related computer vision techniques. Section 3 details the dual-core architecture of the conversational Agent and the modular 3D Pipeline. Section 4 presents the qualitative results of the reconstruction and the agent's performanceand discusses environmental limitations. Finally, Section 5 concludes the paper and outlines future research directions.

2. Related Work

Our work sits at the intersection of two rapidly evolving fields: Autonomous **AI Agents** in tourism and **Neural Rendering** for **3D reconstruction**.

2.1. LLMs and Agents in Tourism

The application of AI in tourism has evolved from simple recommender systems to complex conversational interfaces. Early implementations relied on rule-based chatbots with limited context retention [5]. The advent of Transformer-based **Large Language Models (LLMs)** changed this landscape, enabling models to understand user intent and context over long conversations. However, standard, isolated LLMs often lack the ability to handle structured constraints or interface with specialized visualization pipelines [6].

To overcome these limitations, the concept of the **AI Agent** has emerged—a system that uses an LLM as its "brain" for reasoning, but integrates it with a set of external functionalities, referred to as **Tools** (or functions) [2]. This architecture allows the agent to break down complex queries (e.g., "Plan a week in Spain under 2000 euros") into sequential, actionable steps. In this work, the **AI Agent** acts as a structural orchestrator, translating natural language intent into a rigid itinerary framework (destinations, dates, activities, and accommodations) that serves as the foundation for multimodal synthesis.

The core mechanism enabling this precision is **Tool Calling** [7]. In this PoC, **Tools** are implemented as internal deterministic logic functions and Pydantic schemas rather than external API connectors. This design choice prioritizes the architectural integrity of the multimodal loop; the LLM invokes these tools to map unstructured user input into standardized JSON schemas [8]. The agent operates through a structured conversational loop: when the LLM identifies a need for formalization, it generates a structured **Tool Call**. The internal tool processes the input and returns a **Tool Response** containing the validated schema, which is then used to parameterize the neural rendering engine if needed. This approach ensures that the subsequent 3D visual verification is contextually grounded in the user's specific stay requirements.

Beyond the functional execution provided by Tool Calling, the field of intelligent tourism also frequently employs **Retrieval-Augmented Generation (RAG)** as a parallel strategy for knowledge enhancement. While Tool Calling focuses on invoking specific functions or APIs to perform actions, RAG focuses on grounding the LLM's responses by retrieving relevant excerpts from massive, unstructured datasets—such as travel blogs, historical archives, or local regulations—before generating text [9,10]. This technique is particularly effective for ensuring that travel agents provide factually accurate descriptions of landmarks or cultural nuances without requiring the information to be hard-coded into the model's weights. However, as the primary objective of this work is to validate the architectural synergy between agentic planning and real-time visual synthesis, the implementation of RAG-based systems is considered outside the scope of this research. This study prioritizes the deterministic data flow between internal logic tools and the neural rendering pipeline to establish the feasibility of the multimodal user experience.

A key novel contribution of our system is the development of a specific tool that enables the agent to display real-time, immersive **3D Gaussian Splatting (3DGS)** representations of hotels and tourist facilities directly within the chat interface, offering an immersive visual experience. This capability moves the agent beyond pure textual planning and exemplifies how advanced **Tool Calling** extends an agent's function to dynamic, visual, and personalized interaction.

2.2. Structure from Motion Methods

Structure-from-Motion (SfM) is a photogrammetric technique that simultaneously recovers the 3D structure of a scene and the 6-DoF (Degrees of Freedom) camera poses from a collection of unordered 2D images [11,12]. The quality of the camera poses estimated by SfM is a critical prerequisite for high-fidelity neural rendering methods, including 3DGS.

Modern SfM pipelines can be broadly categorized. Incremental SfM, popularized by systems like COLMAP [11], starts from a minimal reconstruction and iteratively adds more views through feature matching, triangulation, and Bundle Adjustment (BA). While robust and highly accurate, this incremental approach can suffer from computational cost and potential drift on large-scale sequences. Global SfM methods, in contrast, attempt to solve for all camera poses simultaneously, often showing better scalability [13].

Hierarchical approaches, such as Hloc [14], have gained prominence by combining the strengths of deep learning-based local features (e.g., SuperPoint [15]) and global image retrieval (e.g., NetVLAD [16]) to achieve robust matching, especially in challenging conditions with repetitive textures or viewpoint changes, which are common in indoor environments. Other methods, like GLOMAP [17] and PixSfM [18], focus on improving accuracy and robustness through novel optimization strategies or by leveraging dense pixel-level correspondences. Table 1 provides a conceptual comparison of the SfM methods utilized in our pipeline.

Table 1. Conceptual comparison of the five SfM methods integrated into our pipeline. For our work purposes, Glomap and OpenMVG were selected given their robustness and scalability in general environments.

Method	Primary Author(s)	Key Feature
COLMAP [11]	Schönberger & Frahm	High accuracy, robust BA
OpenMVG [13]	Moulon et al.	Modular, efficient library
GLOMAP [17]	Schönberger et al.	Scalable optimization
Hloc [14]	Sarlin et al.	Deep-learning features
PixSfM [18]	Lindenberger et al.	Pixel-level accuracy

Despite the maturity of these methods, no single SfM solution is universally optimal. Their performance varies significantly based on scene geometry, texture, and image capture patterns. A systematic comparison of their impact on 3DGS reconstruction quality for indoor tourism scenes remains an open research question.

2.3. 3D Gaussian Splatting

Introduced by Kerbl et al. in 2023, 3D Gaussian Splatting (3DGS) [4] represents the scene explicitly as a set of 3D anisotropic Gaussians. This approach combines the best of point-based rendering (efficiency) and volumetric rendering (differentiability). Each Gaussian is defined by its position $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, opacity α, and color represented by Spherical Harmonics (SH).

The core idea is to project these 3D Gaussians onto the 2D image plane, resulting in 2D "splats" which are then composited using α-blending. The rendering process is fully differentiable, allowing for efficient optimization. The covariance matrix $\boldsymbol{\Sigma}$ is decomposed into a scaling matrix $\boldsymbol{S}$ and a rotation matrix $\boldsymbol{R}$ (represented as a quaternion $\boldsymbol{q}$), $\boldsymbol{\Sigma} = \boldsymbol{RSS}^T\boldsymbol{R}^T$, which are optimized directly. The projection of a 3D Gaussian onto the image plane involves multiplying its covariance matrix by the Jacobian $\boldsymbol{J}$ of the affine approximation of the projective transformation:

$$\boldsymbol{\Sigma}' = \boldsymbol{J\Sigma J}^T \tag{1}$$

This explicit, differentiable representation, combined with a highly optimized tile-based CUDA rasterizer, allows 3DGS to achieve real-time rendering ($\geq$30 fps) at high resolution while matching or exceeding NeRF's visual quality, all with significantly faster training times (often under an hour). Our work leverages the open-source `gsplat` library [19], which provides an efficient PyTorch 2.0.0 and CUDA 11.8 based implementation of these principles.

3. Methodology

Our system is composed of two distinct but interconnected subsystems: the **Conversational AI Agent** (responsible for logic, planning, and user interaction) and the **3D Reconstruction Pipeline** (responsible for generating the immersive assets). The architecture is summarized in Figure 1.

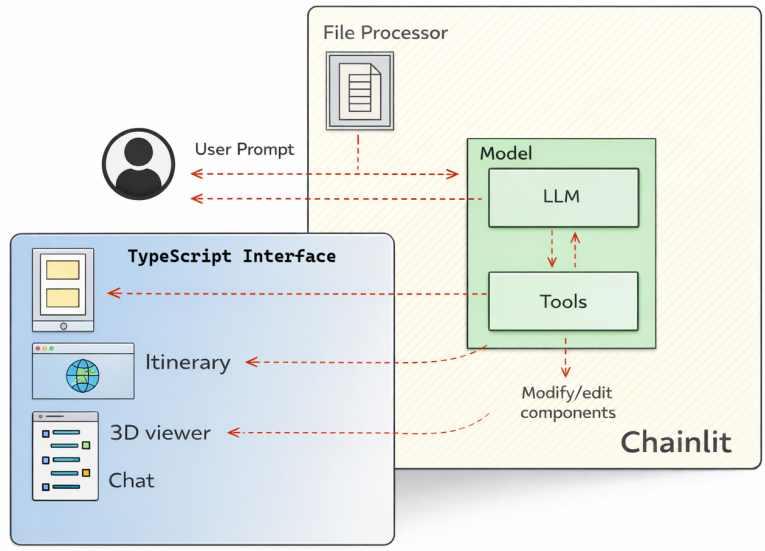

Figure 1. System Architecture. The Engine manages the conversation state and routes requests to specific Tools. The TypeScript interface allows us to display the conversation, the 3D Viewer and the user itinerary all at once. The Viewer Tool dynamically loads 3DGS models generated by our reconstruction pipeline.

3.1. The AI Agent Architecture

The core of our system is an autonomous agent designed to assist travel agents in creating personalized travel itineraries through natural language interaction. The Agent's primary function is to orchestrate the transition from unstructured user intent into a standardized travel framework (encompassing destinations, flights, dates, activities, and accommodations). The system is implemented in Python 3.11, leveraging **LangChain** for agentic orchestration and **Chainlit** [20] for the multimodal chat interface. The architecture follows a modular design pattern, strictly separated into **Engine** and **Structural Tool** classes.

3.1.1. Engine and Tool Classes

We adopted an Object-Oriented Programming (OOP) approach to define the system's behavior, decoupling the reasoning logic (Engine) from the functional execution layer (Tools).

- **The Engine:** This class initializes the base LLM (Google's **gemini-2.5-flash**). It maintains the **System Prompt**, which defines the agent's role as a professional, empathetic travel consultant, and manages the conversation history. The Engine serves as the central controller, processing user inputs to determine whether a direct textual response is sufficient or if a transition to a structured state is required via a tool call. It handles

the recursive reasoning loop (analogous to the "thought–action–observation" pattern) that guides the itinerary creation process.

- **The Tools:** Rather than interfacing with live external databases, the system integrates a suite of **Internal Logic Tools**. These tools function as deterministic mapping functions that enforce structural integrity on the information extracted by the LLM. Each tool inherits from a base `Tool` class, ensuring a uniform interface for the Engine:
    1. **Itinerary Structuring Tools (Hotels, Flights, Rental Car & Activities):** These tools are responsible for extracting specific parameters from the conversation and mapping them into a predefined JSON schema. They ensure that essential trip components are formalized without the ambiguity of natural language.
    2. **Itinerary Compiler:** A utility tool that aggregates the session's validated data into a final, coherent document, facilitating the transition from a conversational state to a fixed travel plan.
    3. **3D Visualization Bridge:** A critical multimodal tool that acts as the interface to the neural rendering pipeline. It allows the Engine to contextually trigger the display of interactive 3DGS digital twins based on the identified facility requirements.

3.1.2. Dynamic Tool Calling with Pydantic

The core mechanism enabling the agent's transition from natural language to a machine-readable format is **Tool Calling**. In this architecture, tools act as deterministic logic gates that enforce structural integrity on the information extracted by the LLM.

To ensure the LLM interacts reliably and extracts the necessary data for the itinerary, we utilize **Pydantic** schemas to define the precise input and output structure for each tool. This ensures that the agent does not just "chat," but actively populates a formal data model. A critical advantage of this approach is the mitigation of model hallucinations: instead of presenting raw LLM outputs directly to the user, the system undergoes strict validation via Pydantic. This structure blocks non-existent or logically invalid types, such as negative night lengths, invalid date formats, or out-of-range prices, forcing a re-prompting cycle if the schema is not satisfied. This structured conversational loop, shown as well in Algorithm 1 and Figure 2, proceeds as follows:

1. **User Message:** The user submits an unstructured request, such as "Find me a hotel in Gran Canaria with a sea view."
2. **Schema Analysis & Tool Call:** The LLM analyzes the intent and determines it must invoke any tool. It generates a structured **Tool Call** identifying the required parameters based on the associated Pydantic model (e.g., `{''location'': ''Gran Canaria'', ''preferences'': ['sea view']}`).
3. **Constraint Satisfaction:** If the LLM identifies that mandatory fields defined in the Pydantic schema (e.g., `check_in_date`) are missing, it initiates a clarifying dialogue. The agent generates a message asking the user for the missing data, ensuring the data object is complete before moving to the synthesis phase.
4. **Internal Execution:** Once all required arguments are satisfied, the tool executes its internal mapping logic. This involves registering the user's choices into the session's stateful itinerary object and retrieving the corresponding `Facility_ID` for 3D visualization, if needed.
5. **Multimodal Integration:** The Agent processes the structured observation to formulate a natural language response. If asked by the user, it simultaneously triggers the rendering of interactive UI components (cards) and 3DGS viewers, contextually linked to the validated schema.

This process allows the agent to iteratively and reliably build a complex JSON object representing the user's trip, which is dynamically rendered in the UI as interactive components (cards) alongside the text chat.

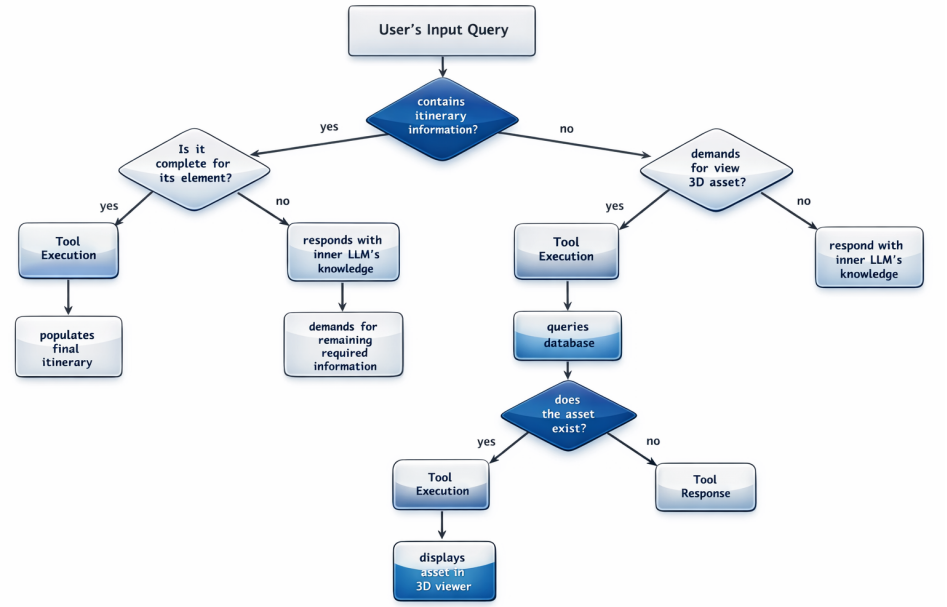

Figure 2. The system's operational workflow is divided into three distinct phases. First, the LLM agent performs an intent analysis to determine if the user's unstructured request demands for (1) visualizing 3D assets or, if belongs to itinerary concerns, (2) its information is complete. In (1) the agent queries a database in order to find and display the 3D asset in an embedded viewer. In (2) it proceeds to formalization by internal orchestration tools to map natural language into a Pydantic schema or demands for extra information if required.

Algorithm 1 Multimodal Agent Orchestration

Input: User Natural Language Query Q, User Profile P
Initialize: Conversation Context $C \leftarrow [P]$
Phase 1: Intent Analysis & Deterministic Structuring
$A_t \leftarrow \text{LLM\_Policy}(Q, C)$ (Agent reasons if structuring is required)
if A_t is *Action(structure_itinerary)* **then**
    $S \leftarrow \text{Invoke\_Internal\_Logic\_Tool}(Q)$ (Apply Pydantic Schema)
    $S_{json} \leftarrow \{\text{Destination, Dates, Facility\_Type, Preferences}\}$
    $C \leftarrow C \cup \{S_{json}\}$ (Update context with structured data)
end if
Phase 2: Multimodal Verification Trigger
if S_{json} contains *Facility_ID* with available 3D assets **then**
    $Splat \leftarrow \text{Retrieve\_3DGS\_Splat}(S_{json}.\text{Facility\_ID})$
    $View \leftarrow \text{Display\_Splat\_Viewer}(Splat)$
    $C \leftarrow C \cup \{View\}$
end if
Phase 3: Final Response Synthesis
$R_{final} \leftarrow \text{LLM\_Generate}(C)$ (Consolidate text plan with embedded 3D visuals)
return R_{final}

3.1.3. The 3D Visualization Bridge: Contextual Neural Rendering

A key architectural contribution of this work is the `Viewer` tool, which functions as the multimodal interface between the agentic reasoning and the neural rendering pipeline. This tool enables the agent to embed real-time, immersive 3DGS representations of hospitality facilities directly within the conversational flow.

For the scope of the PoC, the system utilizes a specialized local database consisting of reconstructions from selected hotel facilities in Gran Canaria. This constrained dataset allows for the validation of the multimodal loop in a controlled, real-world environment. The operational logic follows a deterministic retrieval pattern:

- **Intent & Entity Recognition:** When a user expresses interest in a specific facility attribute (e.g., "Show me the ocean-view suite" or "What does the pool area look like?"), the Agent extracts the facility type and the specific asset requirement.
- **Localized Asset Mapping:** The tool queries the local repository using the structured `Facility_ID` generated in the previous logic step. This database stores pre-processed 3DGS scenes (.ply files) specifically optimized for the Gran Canaria pilot study (e.g., `GC_Hotel_Dunamar_Pool.ply`).
- **Frontend Orchestration:** Upon a successful match, the tool issues a specific execution command to the **Chainlit** interface. This triggers the mounting of a specialized WebGL-based Gaussian Splatting viewer, which streams the volumetric data directly into the chat interface as an interactive video or navigable 3D scene.

By integrating this localized database, the system moves beyond traditional textual planning. It establishes a "try-before-you-buy" verification layer where the agent's textual recommendations are immediately grounded by the physical reality of the destination, demonstrating the technical feasibility of hyper-personalized, visual travel consultancy.

3.2. The 3D Reconstruction Pipeline

This subsection details the architecture and components of our proposed multimodal 3D reconstruction pipeline. Our approach is divided into several stages, as illustrated in Figure 3: (1) Image preprocessing from video sources, (2) 3D sparse reconstruction using multiple interchangeable Structure from Motion (SfM) methods, (3) Photorealistic scene representation using 3D Gaussian Splatting (3DGS), and (4) Integration with an LLM-based conversational agent for interactive visualization.

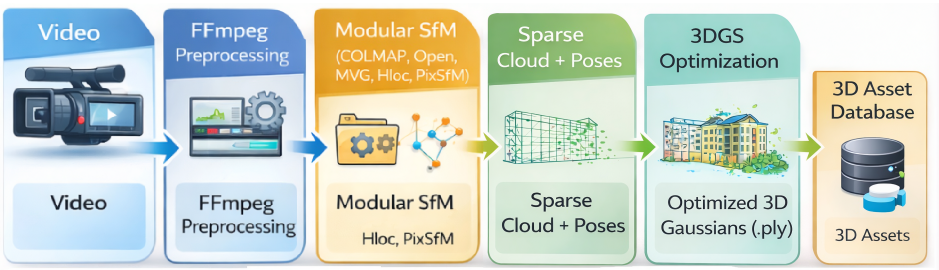

Figure 3. The proposed modular 3D reconstruction pipeline. Raw video is processed into high-quality frames. These frames are fed into one of five interchangeable SfM methods (COLMAP, OpenMVG, GLOMAP, Hloc, PixSfM) to produce a sparse point cloud and camera poses. This sparse representation initializes the 3D Gaussian Splatting optimization using gsplat. The final 3D model is saved in a local database and then rendered in an interactive web viewer, which is contextually controlled by an LLM-based conversational agent.

3.2.1. Data Acquisition and Preprocessing

The quality of the input images is critical for the success of both SfM and 3DGS. As our primary data source is video footage captured from consumer-grade cameras (smartphones), a robust preprocessing step is required. We utilize the **FFmpeg** [21] toolkit for automated frame extraction.

We extract frames at a rate of 2 frames per second (fps). This rate was determined empirically; a lower rate (e.g., 1 fps) often fails to provide sufficient parallax and overlap for robust feature matching in texture-poor indoor environments, while a higher

rate (e.g., >5 fps) introduces significant data redundancy and motion blur, increasing SfM processing time without proportional gains in quality. The FFmpeg command used for extracting images from a video is:

```
ffmpeg -i input.mp4 -qscale:v 2 -vf ‘‘fps=2’’ output/frame_%04d.png
```

Here, `-qscale:v 2` ensures high-quality (low compression) PNG image output. Following extraction, we apply a filtering process based on Laplacian variance for each frame, a common metric for image blur estimation. Frames below a predefined variance threshold (indicating significant motion blur) are discarded. This ensures that only visually distinct frames are passed to the SfM pipeline. Algorithm 2 outlines this process.

Algorithm 2 Frame Extraction and Blur Filtering from Video

Input: V_{in} (Input Video), F_{rate} (Frame Rate, e.g., 2), T_{blur} (Blur Threshold)
Output: $I_{filtered}$ (Set of filtered, non-blurry images)
$I_{raw} \leftarrow \text{FFmpegExtract}(V_{in}, F_{rate}, \text{Quality='High'})$
$I_{filtered} \leftarrow \emptyset$
for each image I in I_{raw} **do**
    $L_{var} \leftarrow \text{LaplacianVariance}(I)$
    if $L_{var} > T_{blur}$ **then**
        $I_{filtered} \leftarrow I_{filtered} \cup \{I\}$
    end if
end for
return $I_{filtered}$

3.2.2. Modular Structure-from-Motion (SfM) Integration

The 3D reconstruction pipeline is built upon a modular SfM component designed to be agnostic to the specific camera pose estimator. This module processes the image set $I_{filtered}$ to produce the camera intrinsics, extrinsics, and the sparse 3D point cloud required for 3DGS initialization. The architecture is designed to support multiple state-of-the-art SfM backends through a unified interface.

- **Geometry-based Estimators:** The system supports traditional incremental and global SfM frameworks, such as **COLMAP** [11] and **OpenMVG** [13], which provide high-accuracy baselines for feature-rich environments.
- **Learning-based Matchers:** To handle the challenging lighting conditions and repetitive textures common in indoor hotel facilities (e.g., corridors or plain walls), the pipeline integrates deep-learning-based matching via **Hloc** [14]. This utilizes SuperPoint features and SuperGlue matching [22] to ensure reconstruction stability where traditional hand-crafted descriptors may fail.
- **Global and Refined Solvers:** The architecture also accommodates hybrid solvers like **GLOMAP** [23] for computational efficiency and refinement layers such as **PixSfM** [18] for pixel-accurate pose optimization.

A critical implementation detail is the **Data Format Harmonization Layer**. Because each SfM tool produces camera and point data in distinct proprietary formats, our pipeline implements a parsing stage that standardizes these outputs into a unified `json` schema. This schema contains the camera intrinsic parameters, extrinsics, and the sparse point cloud, alongside a structured `zip` archive. By standardizing this data flow, the pipeline ensures a seamless transition to the 3DGS training stage, regardless of the underlying SfM backend selected for a specific reconstruction task. While the pipeline is modular, **OpenMVG** and **Glomap** were utilized as the primary backend for the Gran Canaria pilot due to its robustness in camera poses for general environments.

3.2.3. 3D Gaussian Splatting Implementation

We adopt the 3D Gaussian Splatting (3DGS) method [4] for photorealistic scene representation. The scene is modeled as a set of N anisotropic 3D Gaussians. Each Gaussian G_i is defined by a set of optimizable parameters:

- **Position:** $\mu_i \in \mathbb{R}^3$.
- **Covariance:** $\Sigma_i \in \mathbb{R}^{3\times3}$. For stable optimization, this is decomposed into a scaling vector $s_i \in \mathbb{R}^3$ and a rotation quaternion $q_i \in \mathbb{R}^4$. The covariance matrix is constructed as $\Sigma_i = R_i S_i S_i^T R_i^T$, where R_i is the rotation matrix derived from q_i and S_i is a diagonal scaling matrix derived from s_i.
- **Opacity:** $\alpha_i \in [0,1]$, controlled via a sigmoid function to ensure it remains in the valid range.
- **Color:** Represented by Spherical Harmonics (SH) coefficients $c_i \in \mathbb{R}^{k\times3}$. We use degree 3 SH ($k = 16$ coefficients), which allows for modeling view-dependent effects.

Our implementation is built using **PyTorch** [24] and the open-source **gsplat** [19] library. gsplat provides highly optimized, differentiable CUDA kernels for the 3DGS forward and backward passes (rasterization), which is essential for achieving fast training times.

Initialization: The 3DGS optimization is initialized using the sparse point cloud generated by one of the SfM methods. Each 3D point p_j from the SfM output is used to initialize the mean μ_j of a corresponding Gaussian. The initial opacity α_j is set to a small value (e.g., 0.1), and the color c_j is initialized using the RGB value of p_j. The covariance is initialized as an isotropic Gaussian based on the mean distance to the k-nearest neighboring points in the sparse cloud.

3.2.4. GS Training and Optimization

The parameters of all Gaussians are optimized jointly to minimize the difference between the rendered images and the ground-truth input training images. The optimization process is driven by a composite loss function $\mathcal{L}$ that combines a robust L_1 loss and a perceptual D-SSIM (Structural Dissimilarity) loss [25]. This combination is widely adopted in neural rendering as it preserves high-frequency details and structural information more effectively than a simple L_2 (MSE) loss.

The loss function is defined as:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{L1} + \lambda\mathcal{L}_{D-SSIM} \tag{2}$$

with $\mathcal{L}_{L1} = ||I_{gt} - I_{render}||_1$ and $\mathcal{L}_{D-SSIM} = \frac{1-\text{SSIM}(I_{gt}, I_{render})}{2}$. I_{gt} is the ground-truth training image and I_{render} is the image rendered from the current set of Gaussians. We follow common practice and set the weighting factor $\lambda = 0.2$.

A critical component of 3DGS training is **adaptive density control**, which adjusts the number of Gaussians during optimization. This process, executed periodically (e.g., every 100 iterations), involves:

- **Pruning:** Gaussians with an opacity α below a small threshold ϵ are removed.
- **Densification (Cloning):** For Gaussians in under-reconstructed areas (identified by a large view-space position gradient), the Gaussian is duplicated (cloned) and moved slightly in the direction of the gradient.
- **Densification (Splitting):** For large Gaussians in over-reconstructed areas, the Gaussian is split into two smaller Gaussians, with their scales divided.

This adaptive control allows the model to start from a sparse initialization (typically 5k–50k points) and automatically allocate geometric complexity, growing to ∼500k–1M Gaussians for our indoor scenes. We use the Adam optimizer [26] with the learning rate schedules specified by Kerbl et al. [4].

3.2.5. Rendering and Visualization

For real-time visualization, we utilize a custom web-based viewer. The optimized 3D Gaussians (position, rotation, scale, opacity, SH coefficients) are exported to a compact `ply` file. This file is loaded into a web application built with **TypeScript 5.7.2** and **React 19.0.10**. The rendering is performed client-side using a WebGL-based Gaussian Splatting renderer, enabling interactive exploration (orbit, pan, zoom) at high frame rates (>30 fps) directly in the browser.

4. System Validation and Results

The multimodal agent was deployed as a functional TRL 4 prototype using a decoupled full-stack architecture. The core backend orchestration—comprising the LLM reasoning loop, the tool-calling logic, and the structural itinerary generation—was managed by a **FastAPI** server. This server acted as the middleware between the agent's cognitive layer and the **Chainlit** framework, which served as the primary **multimodal frontend** layer. By integrating **React**-based components within the chat interface, the system enabled the seamless transition between textual planning and the mounting of the WebGL-based 3D viewer. The validation focused on the system's ability to maintain state across multi-turn conversations, enforce schema constraints, and contextually trigger the neural rendering pipeline.

4.1. Structural Itinerary Synthesis and Constraint Enforcement

The system successfully demonstrated robust **constraint enforcement** through a conversational "slot-filling" workflow. As formalized in the Methodology (see Algorithm 1), this process relies on the Agent's ability to map unstructured user intent to the mandatory fields defined by **Pydantic schemas**.

The functional validation revealed the following key behaviors:

- **Dynamic Argument Completion:** When presented with partial inputs (e.g., "Find me a flight from Valencia to Gran Canaria"), the Agent correctly identified missing mandatory parameters (such as departure dates or traveler count) based on the internal tool schema. It successfully paused the execution to initiate a clarifying dialogue, ensuring the data object was complete before finalization (see Figure 4).
- **Stateful Component Building:** Unlike traditional chatbots that generate monolithic text, this architecture built the itinerary as an ensemble of structured dictionaries. This allowed the Agent to maintain a persistent session state, enabling the user to add, modify, or refine specific trip components (e.g., changing a hotel selection) across multiple turns without losing the context of the broader itinerary.
- **Deterministic Mapping:** Once the required constraints were satisfied, the internal logic tool successfully mapped the validated parameters into structured UI cards and registered the requirements for the subsequent 3D visualization phase.

Finally, for a completed itinerary trip, we can leverage the use of the Tool calling to retrieve the information displayed in the right column in an exportable pdf file as shown in Figure 5.

Multimodal Input and Tool Calling

A significant finding was the successful integration of multimodal input processing. The Agent was instructed via the System Prompt to process information from **image screenshots** (e.g., of flight search results or TripAdvisor pages) to complete the itinerary fields. This was achieved by integrating the **Gemini 2.5 Flash** model's multimodal capabilities for image and PDF parsing. The Agent was able to correctly parse complex tabular data from a screenshot of a flight search and populate the corresponding Flight Tool's Pydantic

schema (see Figure 6). Furthermore, if the total price was provided but not individual prices, the Agent correctly reasoned to divide the price among the detected flights.

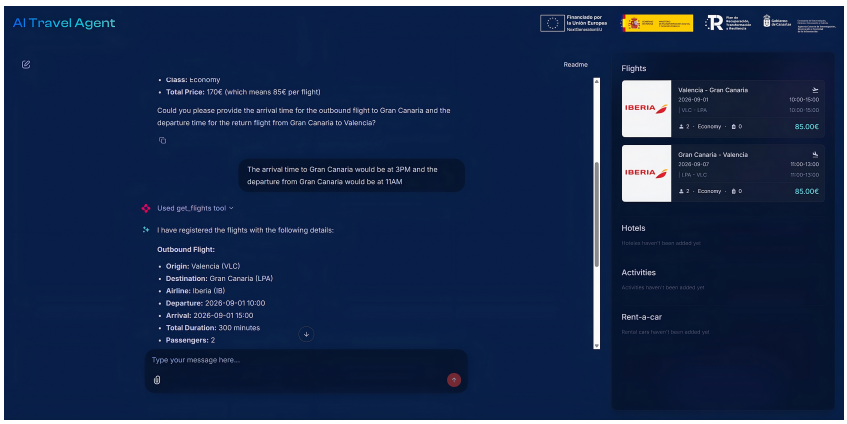

Figure 4. The conversational flow demonstrating slot-filling for hotel booking. The Agent asks for mandatory Pydantic fields before successfully rendering the structured Hotel Card UI.

Figure 5. Itinerary information retrieve based on Tool calling. Our Agent retrieves the itinerary information added by the user to create an exportable document.

The integration of the **3D Viewer Tool** proved seamless and contextual. When the user expresses interest in a specific facility ("Can I see the pool?"), the Agent queries its local database of available reconstructed scenes based on the hotel name. If a match is found, the `get_3dview` tool is invoked. The tool returns a command that triggers the frontend to mount a WebGL-based **Gaussian Splatting** viewer directly within the chat stream. This confirms the viability of mixing text generation with asset retrieval in a single agentic workflow. The viewer is deliberately constrained to limited mouse controls to prevent the user from getting lost, prioritizing a guided, immersive inspection experience.

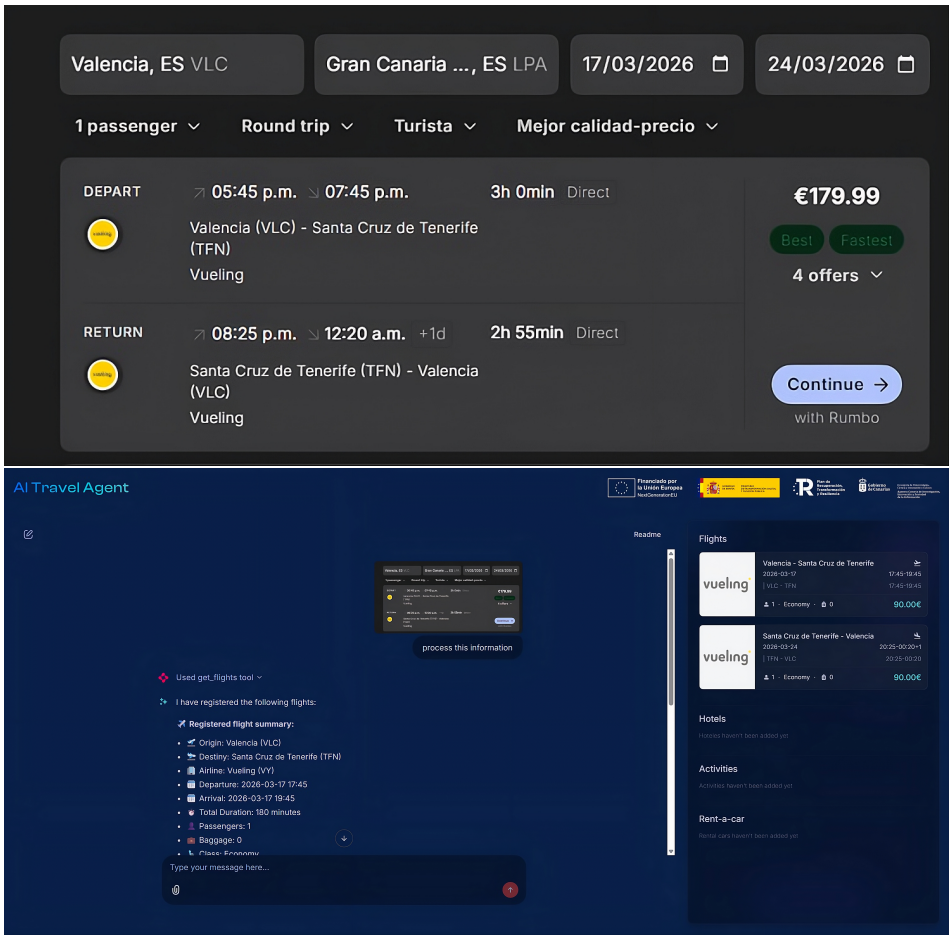

Figure 6. Use of the image parser processor for our agentic system. We feed our Agent with Gemini 2.5 Flash to extract the required fields for the flights tool calling.

4.2. Visual Validation: Asset Integrity and Synthesis

To validate the technical feasibility of the immersive assets delivered by the AI Agent, the **SfM → 3DGS pipeline** was executed on datasets captured across three pilot locations in Gran Canaria: **Lopesan Corallium Dunamar**, **Kumara Serenoa**, and **HD Parque Cristóbal**. As a **TRL 4 proof of concept**, the evaluation was centered on the qualitative coherence and the functional integrity of the data flow, from natural language intent to 3D visualization (see Figure 7), rather than a comparative benchmarking of rendering speeds or point-cloud densities.

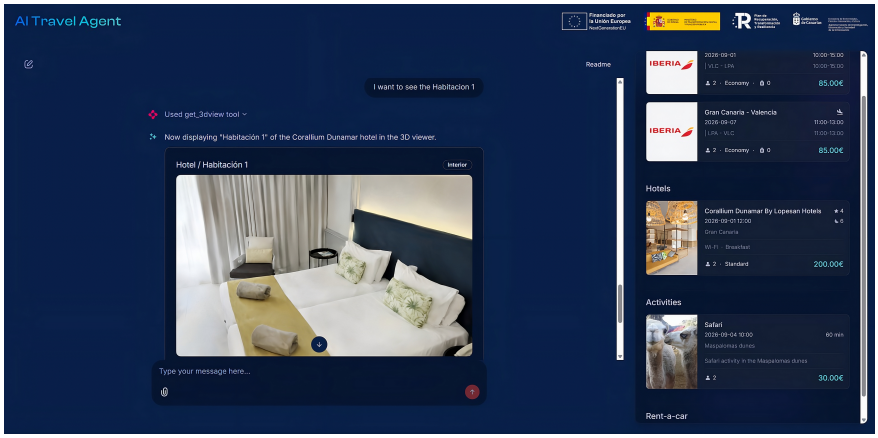

Figure 7. Visual verification tool. The Agent calls the 3D Viewer tool, allowing the user to inspect the *Lopesan Corallium Dunamar* room interactively, without leaving the chat conversation.

4.2.1. Environmental Bias and Technical Limitations

A critical observation during the validation process concerns the dependency of the SfM (Structure-from-Motion) backend on environmental textures. The successful reconstructions in this study were facilitated by the abundant visual features present in the selected Gran Canaria resorts. However, the system's reliability is subject to significant environmental bias:

- **Featureless Surfaces:** In modern minimalist architecture characterized by pure white walls or uniform "flat" surfaces, SfM algorithms often fail to find sufficient feature points. This can lead to a collapse of the sparse reconstruction, preventing the agent from presenting any visual information to the user.
- **Specular and Transparent Materials:** While the 3DGS refinement helps mitigate artifacts, initial camera pose estimation remains highly sensitive to mirrored surfaces and glass walls common in urban luxury hotels. In such environments, the pipeline may produce "hallucinated" spatial depths or fail to localize the camera poses entirely.
- **Spatial Density:** The current work was limited to resort-style spaces. The accuracy and success rate of the pipeline may vary significantly when transitioning to densely populated, narrow urban hotel corridors where the lack of wide-angle perspective can hinder the triangulation of 3D points.

4.2.2. Qualitative Observations on Neural Scene Synthesis

The functional validation of the generated scenes provided insights into the behavior of the **3DGS** optimization process when applied to real-world hospitality environments:

- **Convergence and Structural Fidelity:** The pipeline demonstrated rapid convergence of the global geometry. Basic structural elements reached a stable state early in the optimization process (approx. 1k iterations). Subsequent training phases (up to 10k iterations) focused on the refinement of view-dependent effects through the adaptive densification of **3D Gaussians**. Figure 8 illustrates this progressive refinement toward visual high-fidelity.
- **Heuristic Artifact Mitigation:** A key technical observation was the effectiveness of the adaptive control loop in managing **floaters**. By utilizing density-based pruning—where Gaussians with low opacity or excessive scale are removed—the pipeline maintained high visual clarity even in scenes with challenging reflections (e.g., pool water), ensuring the output remained suitable for a user-facing interface.
- **Spatial Complexity and Resource Scaling:** The validation confirmed a correlation between scene typology and Gaussian density. Contained interior environments achieved functional visual quality with approximately 500k Gaussians, whereas large-scale exterior facilities required a significantly higher volume. This suggests that for a scalable implementation, the system must adaptively adjust training iterations based on the estimated spatial volume.
- **Coverage-Dependent Fidelity:** The visual fidelity of the final render is intrinsically linked to the completeness of the initial video trajectory. While the agent successfully provides immersive "try-before-you-buy" views, the navigable zone is constrained to the primary capture paths. This highlights the importance of standardized capture protocols to ensure consistent asset quality across different hotel facilities.

Figure 8. Visual progression of the 3DGS training process, showing the convergence from rough structure to fine photorealistic detail. (**Top**): reconstruction after 1k iterations. (**Bottom**): reconstruction after 10k iterations.

4.3. The Agent as Orchestrator: An LLM-Assisted Engineering Perspective

Beyond the end-user experience, the architecture of the proposed system serves as a case study in LLM-assisted software engineering and meta-programming. As noted in recent literature regarding AI Copilots [27], the LLM acts not merely as a generator of text but as an orchestrator capable of compiling natural language intent into executable sequences of function calls.

In our system, the transition from an unstructured query to a validated Pydantic schema represents a form of dynamic code generation. The agent must "reason" about the necessary function parameters and "program" the state of the itinerary in real-time. This aligns with the paradigm where models manage and generate code contextually to solve complex, multi-step tasks [27]. By viewing the Tool Calling mechanism as an orchestration layer, we position the system as a precursor to more complex, multi-modal agents in domains like real estate or urban planning, where natural language must be translated into rigorous spatial and logistical data.

Furthermore, while this PoC relies on the LLM's parametric memory for initial reasoning, the integration of Retrieval-Augmented Generation (RAG) represents the clear pathway for industrial-grade reliability. By adopting RAG-based strategies—such as those utilizing vector databases to store and retrieve real-time pricing, policy updates, and verified hotel reviews [28]—the system can ground its "meta-programming" in verified data. This significantly reduces the risk of hallucinations and ensures that the "compiled" itinerary is not only structurally sound but factually accurate.

4.4. Future Work

Based on the functional validation and the architectural boundaries established in this work, several avenues for future research are identified to advance the system toward higher readiness levels:

1. **Comparative Evaluation and Benchmarking:** upon this architectural design, future iterations could include a comprehensive quantitative study. This involves benchmarking the agent's planning efficiency against traditional rule-based systems and conducting structured user studies to measure the impact of 3DGS visualization on booking confidence and user retention.

2. **SfM Pipeline Optimization and Benchmark Analysis:** A critical research direction involves the systematic benchmarking of the integrated Structure-from-Motion (SfM) methods. Future studies will conduct a multi-metric comparative analysis of the implemented methods in our pipeline, specifically within the context of indoor hospitality environments. By evaluating these backends against known metrics (such as PSNR, SSIM, etc.), we aim to establish a decision-making heuristic that automatically selects the optimal SfM initialization based on the specific architectural features and lighting conditions of the target facility.
3. **Integration of Retrieval-Augmented Generation (RAG):** to move beyond deterministic internal tools, the system will incorporate RAG layers. This will enable the agent to ground its responses in unstructured external knowledge—such as local cultural insights or real-time travel regulations—thereby eliminating potential hallucinations while maintaining the structural integrity of the itinerary.
4. **Commercial-Scale Automation and API Integration:** transitioning from a local database to live commercial environments will require the integration of Global Distribution System (GDS) APIs for real-time validation of flights and hotels. Additionally, the selection of SfM backends could be automated using machine-learning heuristics to optimize reconstruction quality for diverse hospitality typologies.

5. Conclusions

This work successfully designed and developed a modular, multimodal system that bridges the gap between conversational AI planning and real-time immersive visualization, specifically tailored for the tourism and hospitality sector. Our core contribution is the synergistic integration of an **AI Agent** based on the **Gemini 2.5 Flash** model with a novel **3D Gaussian Splatting (3DGS) pipeline**.

The **Agentic System** proved effective in its primary task: guiding a conversational flow to generate structured travel itineraries. The implementation of **dynamic Tool Calling** alongside strict **Pydantic schemas** was paramount to this success. This mechanism functioned as a physical logic-gate, blocking non-existent or invalid data types (e.g., negative night lengths or malformed dates) at the schema level. This ensures that the agent does not merely "chat" but actively populates a formal, validated data model, mitigating the risk of model hallucinations in the final itinerary. Furthermore, the Agent's multimodal capability to parse data from image screenshots demonstrated its robustness in handling unstructured real-world inputs.

Crucially, the system validated the benefit of using **embedded 3DGS viewers** over traditional 2D grid galleries for enhancing the user experience. While 2D images often fail to convey spatial context, the integration of navigable 3D digital twins provides a "try-before-you-buy" verification layer, allowing users to interactively explore the layout and ambiance of recommended facilities. Positioned as a **Technology Readiness Level 4 (TRL 4)** proof of concept, the primary purpose of this study was not to measure user-centric UX improvements, but to prove the technical feasibility of a pipeline that dynamically binds LLM inference results to 3D spatial data without failure.

The **modularity** of the system was demonstrated through the integration of five interchangeable SfM methods, with **Glomap** and **OpenMVG** utilized as the primary backend for the pilot study due to its robust performance for general environments. However, a notable limitation is the system's reliance on feature-rich environments. While it performed well in the textured interiors of the Gran Canaria pilot, the SfM pipeline may encounter failures in modern architectural settings characterized by mirrored surfaces or featureless white walls, where the agent would be unable to present accurate visual information. Future work will focus on expanding the pilot to diverse urban environments and conducting success rate

evaluations for generated itineraries to further establish the system's practical effectiveness beyond the laboratory environment.

Author Contributions: Conceptualization, E.S.-O., M.S.-M. and E.W.-S.; methodology, J.M.-A. and P.V.-M.; software, J.M.-A.; validation, M.Á.G.-E. and P.V.-M.; writing—original draft preparation, E.S.-O., M.S.-M. and J.M.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been carried out within the framework of the Spain Living Lab project (Grant Reference 1/1/2024-0412093852—SLLC16-01), funded by the Canarian Agency for Research, Innovation and the Information Society (ACIISI), Department of Universities, Science, Innovation and Culture of the Government of the Canary Islands, under the RETECH Programme, contributing to milestones 251, 252 and 253 of Component 16 of the Recovery, Transformation and Resilience Plan (PRTR), and co-funded by the European Union—Next Generation EU.

Data Availability Statement: The data and code supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: Authors José Márquez-Algaba and Pablo Vicente-Martínez were employed by the company SPV Sala. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Buhalis, D.; Leung, R. Smart Hospitality—Interconnectivity and Interoperability Towards an Ecosystem. *Int. J. Hosp. Manag.* **2020**, *71*, 41–50. [CrossRef]
2. Zou, R.; Cheng, S.; Feng, R.; Deng, W.; Zhang, M.; Sun, W. Achieving Tool Calling Functionality in LLMs Using Only Prompt Engineering Without Fine-Tuning. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), Miami, FL, USA, 12–16 November 2024.
3. Mildenhall, B.; Srinivasan, P.P.; Tancik, M.; Barron, J.T.; Ramamoorthi, R.; Ng, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020.
4. Kerbl, B.; Kopanas, G.; Leimkühler, T.; Drettakis, G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* **2023**, *42*, 139:1–139:14. [CrossRef]
5. Buhalis, D.; Harwood, T.; Bogicevic, V.; Viglia, G.; Beldona, S.; Hofacker, C. Technological disruptions in Services: Lessons from Tourism and Hospitality. *J. Serv. Manag.* **2019**, *30*, 484–506. [CrossRef]
6. Schick, T.; Schütze, E.; Dwivedi, V.; de GTella, J.; Aitchison, E.; Wermeß, N.; Zoph, B.; Le Scao, T.; Schwenk, H. Toolformer: Language Models Can Teach Themselves to Use Tools. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 68539–68551.
7. Yao, S.; Cui, D.Y.; Li, K.; Li, Z.; Yan, J.; Karypis, G.; Cong, Z.; Zhang, Y. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 11809–11822.
8. Li, M.; Zhao, Y.; Yu, B.; Song, F.; Li, H.; Yu, H.; Li, Z.; Huang, F.; Li, Y. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), Singapore, 6–10 December 2023.
9. Ni, H.; Liu, F.; Ma, X.; Su, L.; Wang, S.; Yin, D.; Xiong, H.; Liu, H. TP-RAG: Benchmarking Retrieval-Augmented Large Language Model Agents for Spatiotemporal-Aware Travel Planning. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP), Suzhou, China, 4–9 November 2025.
10. Song, S.; Yang, C.; Xu, L.; Shang, H.; Li, Z.; Chang, Y. TravelRAG: A Tourist Attraction Retrieval Framework Based on Multi-Layer Knowledge Graph. *Isprs Int. J. Geo-Inf.* **2024**, *13*, 414. [CrossRef]
11. Schönberger, J.L.; Frahm, J.M. Structure-from-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 4104–4113.
12. Snavely, N.; Seitz, S.M.; Szeliski, R. Photo tourism: Exploring photo collections in 3D. *ACM Trans. Graph.* **2006**, *25*, 835–846. [CrossRef]
13. Moulon, P.; Monasse, P.; Perrot, R.; Marlet, R. OpenMVG: Open Multiple View Geometry. In *Proceedings of Reproducible Research in Pattern Recognition (RRPR)*; Springer: Cham, Switzerland, 2016.
14. Sarlin, P.E.; Cadena, C.; Siegwart, R.; Dymczyk, M. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2019.
15. DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperPoint: Self-Supervised Interest Point Detection and Description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*; IEEE: Piscataway, NJ, USA, 2018.

16. Arandjelovic, R.; Gronat, P.; Torii, A.; Sivic, J. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition*; IEEE: Piscataway, NJ, USA, 2016.
17. Schönberger, J.L.; Zheng, E.; Pollefeys, M.; Frahm, J.M. Globally-Optimal Structure-from-Motion with Scalable Robust Validation. In *Proceedings of the International Conference on 3D Vision (3DV), Stanford, CA, USA, 25–28 October 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 384–393.
18. Lindenberger, P.; Sarlin, P.E.; Larsson, V.; Pollefeys, M. Pixel-Perfect Structure-from-Motion with Pixel-Accurate Pointclouds. In *Proceedings of the International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2021.
19. Sharma, Y.; Fan, J.Z.; Kanazawa, A. gsplat: Efficient and Scalable Gaussian Splatting Using CUDA. 2024. Available online: https://github.com/nerfstudio-project/gsplat (accessed on 1 December 2025).
20. Chainlit Inc. Chainlit: The Open-Source Platform for LLM Apps. 2023. Available online: https://chainlit.io (accessed on 1 December 2025).
21. FFmpeg Developers. FFmpeg: A Complete, Cross-Platform Solution to Record, Convert and Stream Audio and Video. FFmpeg Project. 2000. Available online: https://ffmpeg.org/ (accessed on 24 October 2025).
22. Sarlin, P.E.; DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperGlue: Learning Feature Matching with Graph Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 4938–4947.
23. Schönberger, J.L.; Cao, Y.; Jauert, T.; Larsson, V.; Pollefeys, M. GLOMAP: Global Optimization for Structure-from-Motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 4825–4834.
24. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8026–8037.
25. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
26. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
27. Wang, Y.; Guo, S.; Tan, C.W. From Code Generation to Software Testing: AI Copilot with Context-Based RAG. *IEEE Softw.* **2025**, *42*, 34–42. Available online: https://arxiv.org/abs/2504.01866 (accessed on 2 February 2026). [CrossRef]
28. OpenAI. OpenAI Cookbook: Question Answering Using Retrieval-Augmented Generation with Qdrant. 2024. Available online: https://developers.openai.com/cookbook/examples/fine-tuned_qa/ft_retrieval_augmented_generation_qdrant (accessed on 2 February 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.