3D SCENE VIRTUALIZATION FOR FLOOR PLANNING

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

by

ALLAM SAI SRAVANA KUMARA VIGNESH (REG NO - 41611009)

GANGADHARA RAVI TEJA (REG NO - 41611049)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category – 1 University by UGC

Accredited with Grade "A++" by NAAC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119

April-2025

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Category - I University by UGC
Accredited "A++" by NAAC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ALLAM SAI SRAVANA KUMARA VIGNESH (REG NO - 41611009) and GANGADHARA RAVI TEJA (REG NO - 41611049),** who carried out Project entitled **"3D SCENE VIRTUALIZATION FOR FLOOR PLANNING"** under my supervision from November 2024 to April 2025.

04/04/25

Internal Guide

Ms. B. BALA SAI GAYATHRI, M.E.

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.

Submitted for Viva voce Examination held on 05/04/2025

Anubarathi 5/4/25

Internal Examiner

5/4/25

External Examiner

DECLARATION

I, **ALLAM SAI SRAVANA KUMARA VIGNESH (REG NO - 41611009)**, hereby declare that the Project Report entitled **"3D SCENE VIRTUALIZATION FOR FLOOR PLANNING"** done by me under the guidance of **Ms. B. BALA SAI GAYATHRI, M.E.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning.**

DATE: 05/04/2025

Vignesh Allam

PLACE: Chennai

Signature of the Candidate

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph. D.**, **Dean**, School of Computing, and **Dr. S. VIGNESHWARI, M.E., Ph.D., Head of the Department** of Computer Science and Engineering with Specialization in Artificial intelligence and Machine Learning for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. B. BALA SAI GAYATHRI, M.E.**, for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning** who were helpful in many ways for the completion of the project.

ABSTRACT

Traditional 2D floor plans often limit spatial understanding and design accuracy, making it challenging for architects and clients to visualize spaces effectively. This project, "3D Scene Virtualization for Floor Planning," introduces an automated pipeline to convert 2D blueprints into interactive 3D models. By leveraging edge detection and vectorization techniques, the system extracts structural elements and translates them into a digital 3D representation, enhancing clarity and design precision. The back end is built using Django, handling user uploads, image processing, and 3D model generation. A Blender-based headless script processes the extracted data to create GLTF models, which can be explored in augmented reality (AR) or interactive web-based 3D viewers. This enables users to interact with the design, modify layouts, and gain a realistic sense of spatial proportions before construction begins. Key challenges include accurate edge detection, seamless software integration, and computational efficiency, which are addressed through optimized algorithms and structured data workflows. The system ensures a smooth transition from 2D to 3D while maintaining scalability and responsiveness. By automating floor plan virtualization, this project reduces design errors, improves collaboration, and enhances decision-making for architects, designers, and clients. This approach demonstrates the potential of AI-driven 3D modeling in architectural design, paving the way for future advancements in automated floor plan visualization. The integration of 3D and AR technologies makes the planning process more immersive, efficient, and adaptable to real-world applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations		Description
AI	-	Artificial Intelligence
AIML	-	Artificial Intelligence and Machine Learning
DL	-	Deep Learning
AR	-	Augmented Reality
GLTF	-	Graphics Library Transmission Format
VR	-	Virtual Reality
VET	-	Virtual Experience Toolkit
CNN	-	Convolutional Neural Networks
VRML	-	Virtual Reality Modeling Language
CT	-	Computed Tomography
CAD	-	Computer-Aided Design
CV	-	Computer Vision

CHAPTER 1

INTRODUCTION

Two-dimensional blueprints have long been used in the floor planning industry to represent spatial designs. Even though 2D drawings have been crucial to the fields of architecture and interior design, they usually fall short of capturing the intricate detail of a given space. The use of three-dimensional (3D) scene virtualization is changing how spaces are perceived, created, and experienced because of technological advancements.

By creating detailed three-dimensional models of environments, 3D scene virtualization allows designers and architects to work with their designs more accurately and immersive than they could with only 2D sketches. Experts can produce realistic representations of floor plans using advanced modeling software and virtual reality (VR) technologies, which improves understanding of spatial relationships and design elements.

This approach improves decision-making and design accuracy in addition to visual clarity. Users can virtually navigate and examine spaces, assess the impact of different design choices, and make instantaneous changes. This interactive procedure enhances teamwork and offers a more intuitive and engaging method to present and improve design ideas, fostering better collaboration within the team.

As design moves into a new era where digital tools are becoming more important, 3D scene virtualization is highlighted as a significant advancement in floor planning. This overview will look at the role that VR and 3D modeling play in modern design processes, highlight their benefits, and talk about the challenges and opportunities that these technologies present for changing the face of interior and architectural design.

1.1 Problem Description

In architecture, real estate, and interior design, traditional methods of floor planning, which primarily rely on 2D representations, often fall short in effectively conveying the true nature of a space. These limitations are especially apparent when trying to understand depth, scale, and spatial relationships within a building.

As a result, clients and stakeholders who are not well-versed in interpreting architectural drawings may struggle to fully grasp how a space will look and feel. This disconnect can lead to misunderstandings and misaligned expectations, which, in turn, contribute to errors during the construction phase.

Moreover, the current design process is often inflexible, making it difficult to iterate and explore different design options. Revisions to the design after construction has commenced are typically costly and time-consuming, and traditional tools do not offer the dynamic, interactive capabilities needed to experiment with and refine designs before they are finalized. This rigidity also limits client involvement, as many clients find it challenging to provide meaningful input when they cannot easily visualize the proposed designs. Consequently, clients may feel less engaged in the process, which can lead to dissatisfaction with the outcome.

Construction planning also suffers from the limitations of 2D floor plans. Misinterpretations of these plans can cause significant coordination issues on-site, leading to mistakes that require rework and cause delays. Additionally, without a clear and accurate understanding of space, it becomes difficult to plan resources and materials efficiently, potentially leading to both waste and shortages.

While powerful 3D visualization tools do exist, they are often complex, costly, and not easily integrated into the standard workflow of architects and designers. This technological gap means that smaller firms or individual designers may find these tools inaccessible, further exacerbating the challenges they face in delivering high-quality, accurate visualizations of their designs.

The consequences of these issues are significant. Increased costs, primarily due to rework, material wastage, and extended project timelines, are a common outcome. Clients may also experience dissatisfaction, as their expectations, shaped by inadequate visualizations, are not met by the final product. Furthermore, the inefficiencies inherent in the traditional design process can slow down the entire workflow, requiring multiple rounds of revisions and approvals before plans are finalized.

Given these challenges, there is a clear need for a more effective solution that can enhance the accuracy and interactivity of the design process. A 3D scene virtualization system for floor planning presents a compelling answer, offering a

way to bridge the gap between design intent and final construction. This system would enable real-time interaction with the space, allow for dynamic customization, and improve communication between all stakeholders, ensuring that the envisioned space is accurately understood and executed from the outset.

1.2 Solution Overview

To address the limitations of traditional floor planning and space visualization methods, the proposed solution is a 3D scene virtualization system designed specifically for floor planning. This system leverages advanced 3D modeling, real-time rendering, and virtual reality (VR) technologies to provide an interactive and immersive experience for both designers and clients.

The core of the solution is a robust platform that allows architects, designers, and clients to explore and interact with a virtual representation of a space in real-time. By converting 2D plans into detailed 3D models, users can walk through the space, assess spatial relationships, and gain a better understanding of the design before any physical construction begins. This immersive experience is further enhanced through VR integration, enabling users to experience the space at a human scale and make informed decisions about design elements.

In addition to improving visualization, the 3D scene virtualization system also streamlines the workflow by integrating with existing design tools and project management software. This integration facilitates seamless data exchange, reducing manual data entry and the risk of errors. The system’s cloud-based architecture ensures that large 3D models and project data are securely stored and easily accessible, enabling real-time collaboration among team members, regardless of their location.

By providing a more intuitive, interactive, and accurate way to design and visualize spaces, this solution aims to bridge the gap between concept and construction. It enhances communication, reduces the likelihood of errors, and ultimately leads to more efficient project delivery and higher client satisfaction.

CHAPTER 2
LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

The Virtual Experience Toolkit (VET) uses computer vision and deep learning to quickly and accurately create 3D models from real indoor environments. This automated process improves the efficiency and precision of 3D visualization, enhancing VR applications. They integrated a single application to automate virtualization of 3D objects using deep learning methods. It can handle indoor objects and can work with around 200 classes. This framework was used to analyze scenes from the ScanNet dataset and some custom indoor scenes. This approach is set to be used for mental treatment by creating a safe virtual environment created using VET. The user will experience new realities without phobic situations. The VET can be highly scaled and is an automatic framework.[1]

In this paper, the authors attempt to show how virtual environments can be used in training as well as in the maintenance and repair of manufacturing facilities and industrial plants. The authors describe a software framework designed to improve industrial system assembly, operation, and maintenance tasks in the context of 3D modeling, animation, and simulation. Because the platform provides trainees with realistic and interactive experiences, it lowers training costs and increases training efficiency. By simulating their operation in an interactive setting, it also aids in the maintenance of industrial installations. Two applications illustrate the framework's adaptability and show how it can enhance industrial procedures. Additionally, this represents a significant advancement in the application of virtual environments for industrial maintenance and training. The system's performance improvement during real-world applications is encouraging because it points to a wide range of potential uses for this technology.[2]

Using image processing and augmented reality (AR), this paper presents a novel approach to reconstructing three-dimensional models from two-dimensional floor plans. Computer vision algorithms are used to determine the arrangement and measurements of a 2D paper plan that was taken with a smartphone camera.

Interactive 3D representations of the 2D models are produced using AR tools, allowing for dynamic exploration. Tested on real building floor plans, the approach produced accurate and efficient results. This development provides a new generation of augmented reality tools that help designers or architects collaborate and communicate more effectively. The improved visualization of the designs helps clients comprehend them more clearly.[3]

Although the Virtual Repository is still under development, this paper outlines its development with the goal of serving as a research and teaching tool for museum collection access. The repository improves the research and education outreach of distant researchers, students, and the general public by filling in the gaps in remote access. For many decades, it builds a permanent online library of artifacts, fossils, locations, and information for use in research and education. The goal is to produce researchers who are focused on museums and conduct analyses that were previously impossible. Additionally, the repository offers researchers, educators, and students new ways to engage with virtual collections, which has the potential to change how science is conducted and taught. It addresses the challenges researchers face with data aggregation in various scientific fields, particularly in natural history and archaeology. It points out that limited access to collections has led to issues with data comparability and reliance on published conclusions. To overcome these challenges, it advocates for the creation of virtual repositories using advanced 3D technologies and image-based database systems. These virtual collections, accessible online, can significantly enhance scientific analysis and make collections available to researchers, students, educators, and the public globally.[4]

This work focuses on low-cost methods of producing and viewing 3D objects by developing a novel architecture for 3D acquisition, transmission, and display. It is cloud-based and provides explanations for all 3D tasks, making it an extensible and user-friendly interface. In order to increase the depth of 3D materials, the system has a crucial 2D to 3D video conversion technology that is supplied both in the synthesis stage and in the real-time processor. Image-based rendering techniques with replication structure are used to enable thin penetrability and guarantee the efficient and high-quality representation of 3D content. The quality of the 3D presentation can be improved by anti-aliasing pixel rendering. By using

integrated 2D/3D windows (i2/3DW) with a micro retarder, the user can interact with the 3D environment by viewing both 2D and 3D images on the same screen. Such a localized 2D/3D display can be used in web services, 3D TVs, movies, and games. The paper also lists a few 3D image formats that may be utilized in the future for 3D technologies. Generally speaking, this strategy should make it easier to create an operational breakthrough that will enable the deployment of more sophisticated 3D algorithms for wider applications.[5]

The combination of virtual reality (VR) and 3D animation has revolutionized effects in the media industry, resulting in more expressive and captivating work. VR adds realism to user experience by integrating physical elements and character interactions. Motion capture and 3D or aerial photos enable real-time physical interaction with 3D models. However, advanced workflows and data loads are issues. But, innovations in visual media do offer new levels of realism and creativity that were never experienced before. They gave a new 3D technology method for designing animation scenes that is faster than the binocular vision method for point cloud computation. By using a grid node for depth information, the new method significantly reduces computation time. For example, at 12 experiments, it takes 48.97 minutes compared to 128.97 minutes for the binocular vision method.[6]

This study aims to explore the use of deep learning and 3D visualized scenes in wireless identifying anomalies in a sensor image. Techniques for detecting and classifying anomalies are centered on a number of issues, such as equipment malfunctions, signal interference, and an overall amount of pattern data. CNN-based deep learning algorithms are ideal for these kinds of anomaly identification methods. One only needs to produce a collection of training images that can be used as an adequate representation of common phenomena in order to correctly identify instances of a 3D visualized complex scene that also contain abnormalities. Because it enables a much more dependable system by reacting in real time to damage or interference patterns, this application is very helpful when analyzing sensor data on a larger scale. Furthermore, it facilitates a clearer understanding of how sensor anomalies function, leading to a more positive method of operating and evaluating sensor systems. In conclusion, the techniques enable the creation of a thorough strategy for tracking sensor network performance and identifying

issues, expanding the range of their uses and dependability.[7]

The authors of this paper discuss a surface transformer-based, one-stage 3D object detection method that incorporates a transformer 3D encoder and a new local umbrella surface description. Finely detailed local features are captured by the local umbrella surface description, which encodes first order gradient slices of each object point as feature vectors. In order to better describe the object, the transformer 3D encoder combines local surface features with global long-term features extracted from the input point cloud. The network can focus on the central regions of the point cloud scenes by using an instance- aware down sampling technique. The results show that this technique outperforms the other one-stage detectors among the current methods.[8]

This study introduces an improved image matching method that addresses the registration of three-dimensional CT and two-dimensional x-ray images. It is important to register these images so that the area of interest is precisely maintained within the radiation zone when the patient is exposed to radiation. The authors examine the application of various matching criteria to attain proper image registration and observe that partial volume interpolation- based mutual information (MI) appears to be the most effective. Three search strategies have been tested for this purpose, and the best one for in-plane refined search is a modified Powell method. The average mean square error for translation and rotation using the suggested algorithm is as low as 0.26 mm and 0.38 degrees, respectively, according to qualitative and quantitative analysis of the results obtained. The simulations validate the suggested approach, which asserts that it is clinically viable. The correspondence problem solution in medicine is improved by this study.[9]

The purpose of this study is to learn how to use three-dimensional modeling technology to create virtual environments for interactive teaching. In addition, the data- driven approach is used to model the equipment's dynamic states, and VRML enables user-equipment interaction in a virtual environment.Additionally, the inclusion of multi-role interaction technology in the virtual environment promotes collaboration between students and teachers as well as between students. Students' practical skills, teamwork abilities, and participation in the learning activities are all developed through this kind of active and participatory

learning. Additionally, the virtual operation environment increases learners' motivation in addition to improving training outcomes. It discusses using 3D virtual reality technology to create training environments for electrical equipment in higher education. Using 3ds Max, virtual components are statically modeled, rendered to match standard experimental equipment, and then exported. This approach facilitates comprehensive sensory simulations and realistic training scenarios.[10]

2.2 EXISTING SYSTEM AND PROPOSED SYSTEM

Traditional floor planning primarily relies on 2D blueprints and manual drafting techniques, which have several limitations. These blueprints are often difficult to interpret, especially for clients without architectural expertise. Converting 2D plans into 3D models is typically done manually using CAD software, which is a time-consuming and labor-intensive process.

Additionally, traditional methods lack interactivity and real-time collaboration, making it challenging to visualize spatial relationships effectively. While some modern CAD tools provide 3D visualization, they often require specialized training and expensive licenses, limiting accessibility. Furthermore, existing approaches struggle with automating the conversion of 2D layouts into accurate 3D models, leading to inefficiencies in the design process.

The proposed system introduces an automated pipeline to transform 2D floor plans into interactive 3D models. This process involves edge detection and vectorization to extract structural elements, followed by a Blender-based headless script that generates a GLTF 3D model. A Django-based web application allows users to upload floor plans, process them, and visualize the generated 3D models. These models can be explored in augmented reality (AR) or web-based 3D viewers, enhancing spatial understanding and decision-making. The system significantly reduces manual effort, improves accuracy, and enables real-time collaboration between architects, designers, and clients.

By leveraging Computer Vision (CV) and automation, this solution provides a faster, more scalable, and user-friendly alternative to traditional floor planning methods, ensuring better design precision and cost efficiency.

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

Author & Journal name	Title	Existing Techniques	Drawbacks
Clara Garcia, Pau Mora, Mario Ortega, Eugenio Ivorra & Gaetano Valenza	Virtual Experience Toolkit: Enhancing 3D Scene Virtualization from Real Environments Through Computer Vision and Deep Learning Techniques	A novel framework called Virtual Experience Toolkit (VET) has been proposed. It employs CV and Deep Learning (DL) techniques to swiftly and seamlessly visualize any 3D scenario from real indoor environments.	High computational demands, potential data privacy issues, and accuracy limitations.
Lei Li	Application of Computer 3D Technology in Graphic Design of Animation Scene	virtual reality technology, the grid node representing the depth information of the animation field is first established in space, and the animation scene is obtained according to the relationship between the feature points of the animation scene image and the 3D point cloud.	High computational and rendering costs, a steep learning curve for complex software
Guangwei Li	Abnormal Recognition Technology of 3D Virtual Scene Image Based on Wireless Network Sensor	Deep learning techniques were used, combined with image processing and pattern recognition methods, to achieve accurate identification of anomalies in wireless network sensor images.	Inaccuracies due to sensor limitations and network interference, high data transmission demands.
Shanshan Zhang	Application of 3D Digital Technology in Virtual Laboratory Training	This paper uses three dimensional visual virtual reality to realize this training environment. Based on the application of 3D visual virtual reality technology. The static modeling of virtual components is carried out through 3ds Max.	It can have limitations in replicating the tactile and nuanced aspects of physical experiments
Yanfei Liu, Kanglin Ning	Surface Transformer for 3D Object Detection	A novel framework called Virtual Experience Toolkit (VET) has been proposed. It employs CV and Deep Learning (DL) techniques to swiftly and seamlessly visualize any 3D scenario from real indoor environments.	Handling occlusions and complex object shapes, and performance be sensitive to the quality of input data.
Pau Mora, Clara Garcia, Eugenio Ivorra ,Mario Ortega & Mariano L. Alcañiz	An End-to-End Automated 3D Scene Virtualization Framework Implementing Computer Vision Techniques	Virtual reality technology, the grid node representing the depth information of the animation field is first established in space, and the animation scene is obtained	High computational and memory requirements, potential inaccuracies

		according to the relationship between the feature points of the animation scene image and the 3D point cloud.	due to data quality and algorithm limitations, and challenges with handling complex or occluded scenes.
J. El-Chaar, C. R. Boer, P. Pedrazzol, S. Mazzola, G. Dal Maso	Interactive 3D virtual environments for industrial operation training and maintenance	This work introduces the main characteristics of the Virtual Environments in contexts of education-centred and training-centred.	Costly and resource intensive, and may struggle with realistic scenario replication and user proficiency.
Herbert D. G. Maschner, Corey D. Schou, Jonnathan Holmes	Virtualization and the democratization of science: 3D technologies revolutionize museum research and access	Using 3D technologies, emerging image-based database architectures, online measurement and analysis tools, and related methods of virtualization enhance science by bringing collections to any scientist, student, educator, or layperson, anywhere in the world.	Implementation costs, potential loss of tactile experiences, and technical challenges in creating accurate representations.
Yang Lei, Yan Zhang	An improved 2D-3D medical image registration algorithm based on modified mutual information and expanded Powell method	Using approximate geometric relationship and the results in the two projections are then combined and converted to a 3D rigid transformation by 2D-3D geometric transformation.	Potential inaccuracies in complex or noisy images
Tzuan-Ren Jeng, Der-Ray Huang, KaiChe Liu, Fu-Jen Hsiao	New 3D Image Technologies Developed in Taiwan	The obtained multiview images can be transmitted through popular streaming channel to the cloud computing center that handles huge tasks of 2D/3D image processing.	Limited integration with existing systems, and potential challenges in achieving global compatibility and standardization

Table 2.3.1 Drawbacks on existing systems

CHAPTER 3
REQUIREMENTS ANALYSIS

3.1 RISK ANALYSIS FOR THE PROJECT

3.1.1 Technical Risks

Edge Detection Errors: Inaccurate detection of walls and structural elements may result in flawed 3D models.

Resolution: Enhance image processing with advanced algorithms and machine learning for higher precision.

High Computational Load: Converting 2D floor plans into 3D models can be resource-intensive.

Optimization: Improve algorithm efficiency and implement streamlined rendering techniques for faster processing.

Software Compatibility Issues: Integrating multiple tools (Django, Blender, GLTF) may lead to data format inconsistencies.

Standardization: Use universally accepted file formats and establish seamless data conversion pipelines.

3.1.2 Operational Risks

Server Downtime & Performance Bottlenecks: High user traffic could slow down or disrupt the system.

Enhancement: Deploy a scalable cloud-based infrastructure with load balancing and caching mechanisms.

Security & Data Privacy Concerns: User-uploaded floor plans may contain sensitive architectural data.

Protection: Implement encryption, secure access controls, and robust data handling policies.

3.1.3 User-Related Risks

User Adoption & Learning Curve: Users unfamiliar with 3D visualization tools may struggle with navigation.

Accessibility: Design an intuitive interface, simplify interactions, and provide guided tutorials.

Misinterpretation of 3D Models: Users may face challenges in understanding spatial proportions.

Clarity: Incorporate real-world scaling references, annotations, and interactive measurement tools.

3.2 SOFTWARE AND HARDWARE REQUIREMENTS SPECIFICATIONS DOCUMENT

3.2.1 Software Requirements

3.2.1.1 Backend Technologies

Programming Language: Python

Framework: Django

3D Processing Engine: Blender (running in headless mode)

Database: SQLite

3.2.1.2 Frontend Technologies

Web Framework: HTML, CSS, JavaScript

3D Model Viewer: Three.js

AR Support: WebXR

3.2.1.3 Development & Deployment Tools

Version Control: GitHub

Server Deployment: AWS

Blender Automation: Python scripting for 3D model generation

3.2.2 Hardware Requirements

3.2.2.1 Minimum System Requirements

Processor: Intel Core i5 or equivalent

RAM: 8 GB

Storage: 256 GB SSD

GPU: Integrated GPU (for basic testing)

Operating System: Windows / Linux

3.2.2.2 External Devices

High-resolution monitor

Android / Iphone

VR headset (optional, for immersive AR/VR exploration)

CHAPTER 4
DESCRIPTION OF PROPOSED SYSTEM

4.1 FLOW CHART OF THE PROCESS USING MACHINE LEARNING

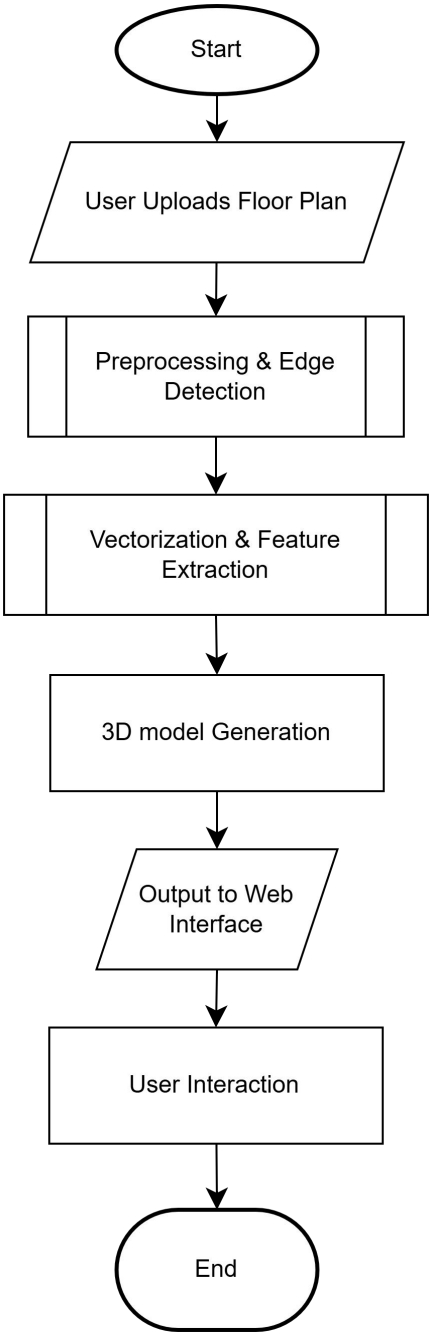

Fig 4.1.1 Flow Diagram

4.2 SELECTED METHODOLOGY OR PROCESS MODEL

For this project, the Incremental Model is the best approach. It allows us to develop the system in small, manageable steps, ensuring steady progress. Each part of the project is tested as soon as it's completed, making it easier to identify and fix issues early. This step-by-step method also allows us to improve and refine features based on feedback. Overall, it ensures a smooth development process with flexibility for future enhancements.

4.2.1 Development Phases

4.2.1.1 Processing the 2D Floor Plan

- Users upload a 2D floor plan image to the system.
- The system processes the image, detecting edges, walls, and key structural elements.
- Advanced image processing techniques, such as OpenCV-based edge detection, are used to extract relevant data.
- The extracted information is then converted into vector data for further processing.

4.2.1.2 Creating the 3D Model

- The vectorized floor plan is transformed into a 3D model using Blender.
- Walls, doors, and other architectural elements are accurately placed based on detected edges.
- The 3D model is optimized for better performance and visualization, ensuring smooth rendering.

4.2.1.3 Building the Web Application

- A user-friendly web interface is developed using Django, HTML, CSS, and JavaScript.
- The 3D model is integrated into the web application using Three.js.
- Users can interact with the model by rotating, zooming, and viewing different angles.
- Features like saving, downloading, and modifying the model are implemented for enhanced user control.

4.2.1.4 Adding Augmented Reality (AR)

- AR features are integrated using WebXR to allow real-world visualization of the 3D model.

- Users can project the model onto real surfaces using their mobile devices for a more immersive experience.
- Interactive tools like real-time measurements and annotations enhance usability.

4.3 ARCHITECTURE OF PROPOSED SYSTEM

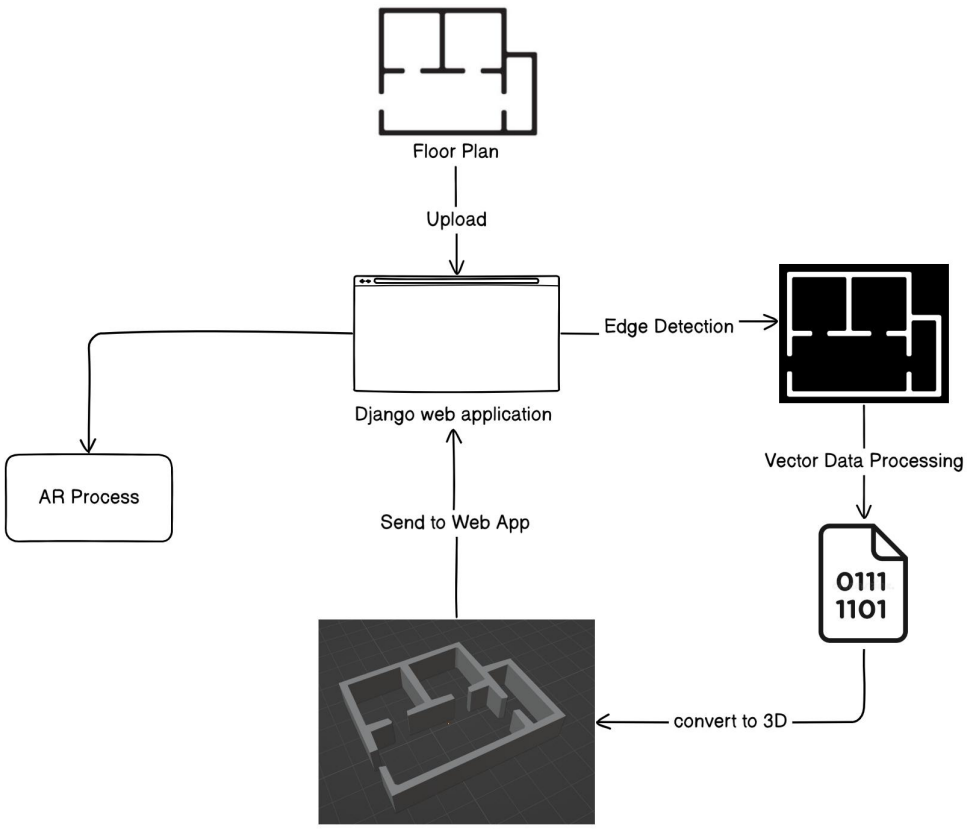

Fig 4.3.1 Architecture Diagram

The 3D Scene Virtualization for Floor Planning system follows a modular architecture that integrates Computer Vision, 3D Modeling, and Augmented Reality (AR) into a seamless workflow. Below is the architectural breakdown:

4.3.1 User Interface

- Users upload a 2D floor plan in JPEG or PNG.
- View and interact with the 3D model in the browser.
- Option to preview the model in AR using WebXR.

4.3.2 Backend Processing using Django based Server

- Processes uploaded images and extracts features.

- Applies edge detection to identify walls, doors, and structures.
- Converts the processed image into vectorized data.
- Passes structured data to Blender for 3D model generation.

4.3.3 3D Model Generation in Blender Headless Mode

- Converts vector data into a 3D representation.
- Creates walls, rooms, and layout features.
- Outputs a GLTF/GLB 3D model file for rendering.

4.3.4 Storage and Data Management

- Stores user-uploaded floor plans.
- Saves vectorized data & 3D models for future retrieval.

4.3.5 Rendering and AR Integration

- Renders the 3D model inside the web app.
- Provides AR visualization for real-world placement.
- Allows users to explore & adjust models interactively.

4.4 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL / SYSTEM

3D Scene Virtualization for Floor Planning is built using a combination of computer vision, 3D modeling, and web-based rendering technologies to enhance architectural visualization. The software is structured into different components, each responsible for a specific function in the workflow, from image processing to 3D model rendering and augmented reality (AR) integration.

The back-end of the system is developed using Django, which manages user interactions, handles file uploads, and facilitates communication between different modules. The back-end also integrates SQLite for efficient data management, ensuring smooth processing and retrieval of floor plans and 3D models. The front-end utilizes HTML, CSS, JavaScript, and Django templates, providing an intuitive interface for users to upload images, interact with 3D models, and explore floor plans in a virtual environment.

The image processing modules, powered by OpenCV and NumPy, play a crucial role in converting 2D blueprints into structured data. This module detects edges, walls, and structural elements from the uploaded images and transforms them into a vector representation. The extracted data is then passed to the 3D modeling module, which runs Blender in headless mode, automating the generation of

GLTF/GLB 3D models. These models retain architectural accuracy, ensuring a faithful representation of the original floor plan.

To enable real-time interaction, the generated 3D models are rendered using Model Viewer, a lightweight web-based render-er designed for GLTF/GLB models. This integration allows users to explore the 3D scene with interactive controls such as rotation, zooming, and AR visualization. Model Viewer also supports WebXR, enabling users to view their floor plans in augmented reality directly from their web browsers.

Through this structured implementation, the 3D Scene Virtualization for Floor Planning system delivers a powerful and efficient tool for architects, designers, and engineers. By leveraging Django, OpenCV, Blender, and Model Viewer, the software provides a seamless transition from 2D blueprints to immersive 3D environments, revolutionizing the way floor plans are visualized and explored.

4.4.1 Testing Plan for the Proposed Model/System

The 3D Scene Virtualization for Floor Planning project follows a structured testing approach to ensure that each component functions correctly and delivers an optimal user experience. The system is tested across multiple phases, covering functional testing, performance testing, usability testing, security testing, and integration testing. The primary objective is to validate the accuracy of the conversion from 2D floor plans to 3D models, ensure proper edge detection, confirm the correctness of 3D reconstruction, and assess the efficiency of Model Viewer in rendering the final models seamlessly.

The unit testing phase focuses on verifying the correctness of individual components. The image processing module, which extracts edges from the uploaded floor plans using OpenCV, is tested to ensure that it correctly identifies structural elements such as walls and doors. The 3D model generation process in Blender is also tested to validate the transformation of vectorized floor plans into 3D representations, ensuring that the models maintain architectural accuracy. Additionally, the rendering process using Model Viewer is examined to verify that models are displayed correctly, interactive controls function smoothly, and augmented reality (AR) visualization aligns with real-world scaling.

Following unit testing, integration testing is conducted to ensure smooth data flow between different modules. The transition from edge detection to vector conversion, 3D model generation, and rendering is tested for consistency and

accuracy. This phase ensures that extracted edges align correctly in the final 3D output and that the user experience remains intuitive when navigating between different views. The integration of Model Viewer with Django is validated to confirm that the rendering process is smooth, responsive, and compatible with various devices and browsers.

Performance testing plays a crucial role in optimizing the system's responsiveness and efficiency. The time taken to process and convert floor plans into 3D models is analyzed to identify potential bottlenecks. The server load capacity is also tested to ensure that multiple users can simultaneously upload floor plans and view their respective 3D models without system slowdowns. Additionally, the WebXR and AR functionalities are evaluated for real-time responsiveness, ensuring that models load correctly in augmented reality environments without lag or distortion.

To enhance usability and accessibility, the system undergoes user experience testing to evaluate how intuitive the interface is for uploading floor plans and interacting with the 3D models. Users are observed while navigating the system to identify areas of improvement in design and usability. Feedback is collected and incorporated to refine interface elements, improve loading times, and enhance overall interaction.

Security testing is also conducted to protect user data and prevent unauthorized access. Uploaded floor plans and generated 3D models are secured against potential breaches, ensuring that sensitive design data remains confidential. Measures such as secure authentication, encrypted storage, and access control mechanisms are implemented and tested to safeguard user information.

The final phase involves system validation and acceptance testing, where the entire workflow is assessed under real-world conditions. Test cases simulate various scenarios, such as uploading complex floor plans, interacting with detailed 3D models, and accessing the system from different devices. The system is refined based on the results, ensuring that all functionalities perform as expected. With a comprehensive testing strategy, the project guarantees a seamless, interactive, and highly efficient 3D visualization experience for users.

CHAPTER 5
IMPLEMENTATION DETAILS

5.1 SYSTEM STUDY / TESTING

The 3D Scene Virtualization for Floor Planning project is developed to bridge the gap between traditional 2D floor plans and interactive 3D visualizations. The primary goal is to help users, including architects, interior designers, and real estate professionals, better understand spatial arrangements before construction. By utilizing computer vision, machine learning, and 3D rendering technologies, this system automates the process of converting flat blueprints into immersive 3D models.

5.1.1 Working Mechanism

The system follows a structured approach:

- Image Upload – Users upload 2D floor plans, which are processed in the back-end.
- Edge Detection & Vectorization – OpenCV-based edge detection identifies walls, doors, and structures, converting them into vector data.
- 3D Model Generation – A headless Blender script takes the processed data and converts it into a GLTF/GLB 3D model with proper scaling and architecture.
- Rendering & AR Integration – The generated models are displayed using Model Viewer, allowing users to rotate, zoom, and view them in augmented reality (AR).
- User Interaction & Modifications – Users can explore, analyze, and adjust the designs for better visualization and decision-making.

5.1.2 Technology Stack

- Backend – Django (for request handling, processing, and user authentication).
- Frontend – HTML, CSS, JavaScript (for UI and user interactions).
- Database – SQLite (for secure data storage and management).
- Computer Vision – OpenCV, NumPy (for edge detection and feature

extraction).

- 3D Processing – Blender (for generating 3D models in headless mode).
- Rendering – Model Viewer (for web-based visualization and AR integration).

5.1.3 System Testing

The primary objective of testing is to validate the accuracy, performance, and usability of the system under different conditions. The testing process ensures that:

- The system correctly extracts and processes 2D floor plans without errors.
- The 3D model generation process maintains precision in measurements and structures.
- The rendering and AR visualization work smoothly across different devices and browsers.
- The system can handle multiple user requests without lag or crashes.

5.1.4 Functional Testing

- Image Processing – Tests whether edge detection correctly identifies walls, doors, and openings.
- 3D Model Generation – Verifies if Blender correctly converts the processed data into realistic and well-scaled 3D models.
- Rendering & AR Integration – Ensures Model Viewer displays models correctly with smooth interaction.

5.1.5 Integration Testing

- The Django Backend should seamlessly process uploaded images and return 3D models.
- The pipeline from OpenCV processing to Blender-based 3D modeling should function without data loss.
- The Model Viewer should properly render 3D objects without distortion.

5.1.6 Performance Testing

- The server’s ability to handle multiple users uploading and rendering floor plans simultaneously is evaluated.
- Memory consumption and processing efficiency are optimized to avoid crashes.

5.1.7 Usability Testing

- Ease of uploading images and interacting with 3D models.
- Clarity and accuracy of generated 3D visualizations.
- Effectiveness of the AR feature in providing real-world perspective.

5.1.8 Security Testing

- User authentication and secure storage of uploaded floor plans and 3D models.
- Protection against unauthorized access and data breaches.
- Ensuring the system follows secure file-handling practices to prevent exploitation.

5.2 OVERALL DESIGN FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSEDMODEL/SYSTEM

The implementation of this system follows a layered architecture, ensuring modularity and efficient processing. The backend, powered by Django, handles the core functionalities, including image processing, model generation, and data management. The frontend, developed using React, provides a user-friendly interface where users can upload floor plans and explore 3D models. The Model Viewer framework is integrated for real-time rendering, allowing users to interact with the generated 3D structures.

The process begins with image preprocessing using OpenCV, where edges and structural elements are detected. The extracted data is then processed using computer vision algorithms, identifying walls, doors, and other key architectural components. Once the floor plan is analyzed, a 3D model is generated using Blender in headless mode, producing a GLTF/GLB file. This model is then made accessible to users via Model Viewer, where they can explore it in both standard and augmented reality views.

To ensure smooth execution, the implementation is divided into phases:

- Data Collection & Preprocessing – Users upload floor plans, and images are processed to detect architectural features.
- Feature Extraction & 3D Model Generation – Walls, doors, and structural elements are identified and converted into a 3D structure.
- Rendering & Visualization – The generated model is rendered using Model

Viewer, enabling interactive exploration.

- AR Integration & Optimization – Enhancements are applied for real-world AR visualization, improving model accuracy and user experience.

5.2.1 Testing Plan

The testing phase ensures that the generated 3D models are structurally accurate and visually consistent with the original floor plan. The primary validation method involves comparing the original 2D floor plan with the generated 3D top-view image to verify that walls, doors, and room layouts are accurately represented. Performance testing ensures that the system processes images quickly and renders models efficiently, while usability testing focuses on user interaction and ease of navigation.

Unit testing is conducted to validate individual components such as edge detection accuracy, feature extraction, and model generation. Integration testing ensures that the backend, rendering module, and AR interface function together without issues. Performance testing evaluates the processing time and rendering speed, ensuring that the system can handle multiple floor plans efficiently. Finally, user acceptance testing is performed to gather feedback and make necessary improvements for a smoother experience.

5.3 PROJECT PLAN

The implementation of the 3D Scene Virtualization for Floor Planning project is structured into multiple phases to ensure a systematic and efficient development process. Each phase is tailored to the core functionalities of the system, including image processing, 3D model generation, rendering, and validation.

5.3.1 Phase 1: Requirement Analysis and System Planning

- Define the project scope: Converting 2D floor plans into interactive 3D models.
- Identify required technologies: Django (backend), React (frontend), Blender (3D modeling), Model Viewer (rendering), and OpenCV (image processing).
- Plan the workflow, specifying how users will upload images, process them, and obtain a 3D model.
- Set hardware and software requirements, ensuring the server can handle Blender headless processing.
- Establish a timeline for each phase to meet implementation deadlines.

5.3.2 Phase 2: System Architecture Design

- Design the backend using Django, structuring APIs for processing user inputs and managing 3D model generation.
- Implement the frontend with React, creating a UI where users can upload floor plan images and interact with the generated 3D model.
- Define a workflow for image processing, feature extraction, and model generation.
- Plan server-side execution for running Blender scripts in headless mode to generate GLTF models.
- Integrate Model Viewer for interactive visualization of the 3D floor plan.

5.3.3 Phase 3: Image Processing and Edge Detection

- Implement OpenCV-based image preprocessing to enhance floor plans by converting them to grayscale, applying thresholding, and noise reduction.
- Develop edge detection algorithms to accurately detect walls and boundaries in the floor plan.
- Convert detected edges into vector-based line drawings that can be translated into 3D models.
- Validate the accuracy of detected walls and openings by comparing extracted features with the input image.

5.3.4 Phase 4: 3D Model Generation Using Blender

- Develop Blender Python scripts to process extracted wall edges and convert them into 3D structures.
- Define height, thickness, and material properties for walls based on the input blueprint.
- Generate the GLTF model dynamically, ensuring it aligns accurately with the floor plan.
- Optimize the 3D model generation process to handle different floor plan sizes and complexities efficiently.

5.3.5 Phase 5: Rendering and Visualization with Model Viewer

- Integrate Model Viewer into the frontend to display interactive 3D models.
- Implement camera controls, allowing users to zoom, rotate, and explore the generated 3D floor plan.
- Optimize rendering performance to ensure smooth visualization on both desktop and mobile devices.

- Enable lightweight web-based interactions, making the system accessible without requiring powerful hardware.

5.3.6 Phase 6: Testing and Validation

- Accuracy Testing: Compare the original 2D floor plan with the 3D top-view output to ensure wall alignment.
- Performance Testing: Measure the time taken for image processing, model generation, and rendering.
- Usability Testing: Validate the user interface and workflow, ensuring intuitive navigation.
- Error Handling: Test cases for distorted images, incomplete blueprints, and irregular structures.
- Ensure the generated 3D structure is dimensionally correct and matches real-world measurements.

5.3.7 Phase 7: Deployment and Optimization

- Deploy the Django backend on a server with GPU acceleration to enhance processing speed.
- Optimize Blender scripts to reduce processing time and handle large-scale floor plans efficiently.
- Improve Model Viewer performance, ensuring smooth rendering even on low-end devices.
- Implement security measures for safe file uploads and data processing.

CHAPTER 6
RESULTS AND DISCUSSIONS

The 3D Scene Virtualization for Floor Planning system successfully converts 2D architectural blueprints into interactive 3D models, enabling a more comprehensive visualization of floor layouts. By integrating computer vision for edge detection, Blender for 3D model generation, and Model Viewer for real-time rendering, the system provides an automated, efficient, and user-friendly approach to floor planning. This transformation eliminates the limitations of traditional 2D blueprints, offering architects, designers, and users an immersive experience to explore and analyze spatial arrangements before physical construction begins.

One of the key achievements of the system is its high accuracy in detecting walls, doors, and other structural elements from floor plan images. Using OpenCV, the system applies edge detection, contour extraction, and vectorization techniques to process the input image and identify key architectural features. The extracted structural elements are then processed through Blender's headless mode, where Python scripts generate GLTF based 3D models that maintain correct proportions and spatial relationships. The accuracy of this process is verified by comparing the original 2D blueprint with the top-down view of the generated 3D model to ensure correct alignment.

Rendering is efficiently handled using Model Viewer, which allows users to interact with the generated 3D floor plan in real-time. The smooth rendering performance ensures that users can rotate, zoom, and navigate through the model seamlessly, providing a detailed perspective of the floor plan. Since the system is optimized for web-based usage, it is accessible on a wide range of devices without requiring high-end computational power.

Performance testing confirms that the system processes and generates 3D models in a reasonable time frame, maintaining an optimal balance between speed, accuracy, and computational efficiency. The image processing pipeline effectively handles different floor plan complexities, ensuring scalability across various architectural designs. Testing also includes error handling for incomplete or distorted blueprints, ensuring robustness in different real-world scenarios.

Validation is performed by comparing the original 2D blueprint with the top-view of the generated 3D model, checking for consistency in wall placements and spatial proportions. The results confirm that the system effectively transforms 2D plans into accurate 3D representations, reducing dependency on manual 3D modeling. Future enhancements may include automated room labeling, AI-driven structural modifications, and real-time AR integration, further improving the system's capabilities and making it a powerful tool for modern architectural planning.

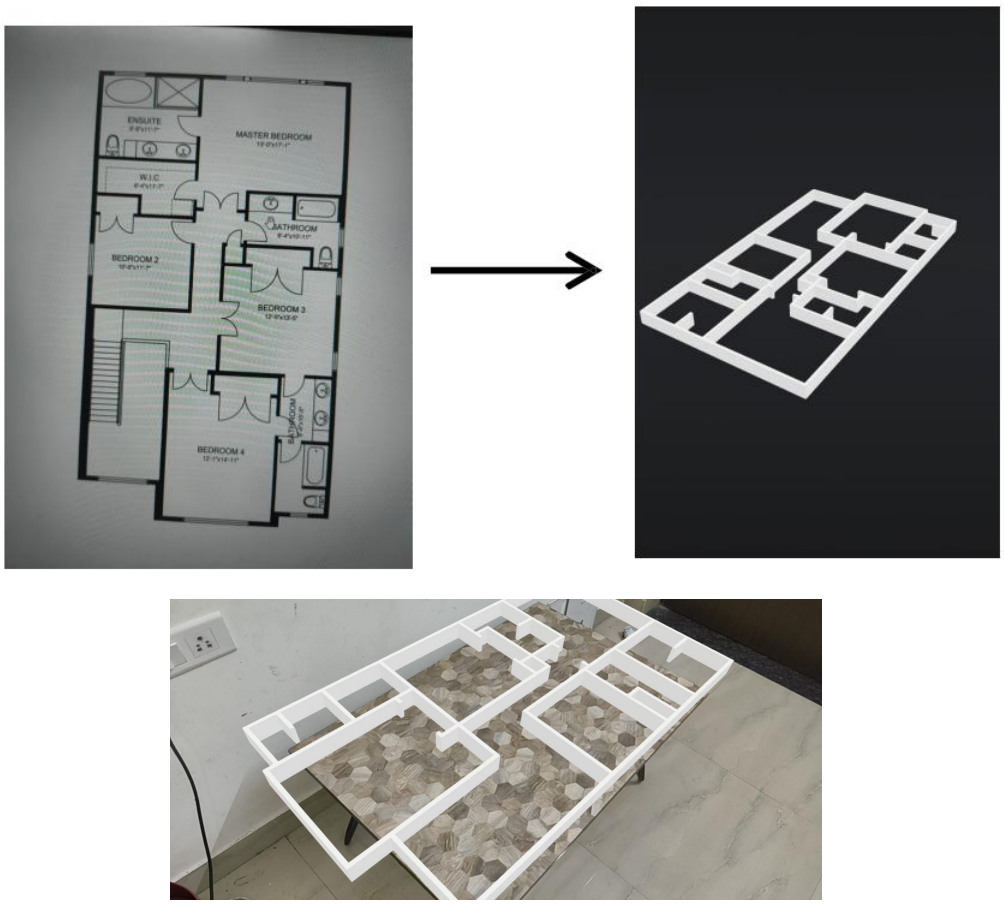

Fig 6.1 Results

CHAPTER 7
CONCLUSION

The 3D Scene Virtualization for Floor Planning system provides an advanced approach to architectural visualization by automatically converting 2D blueprints into interactive 3D models. Traditional 2D floor plans often pose challenges in understanding spatial relationships, making it difficult for architects, designers, and clients to interpret layouts accurately. This system streamlines the process by integrating computer vision for feature detection, Blender for 3D modeling, and Model Viewer for real-time rendering, creating a seamless and automated workflow.

The system ensures high accuracy in detecting walls, doors, and structural elements by applying edge detection and contour extraction using OpenCV. Once the architectural features are identified, they are converted into vectorized data that is processed in Blender's headless mode. This automated approach ensures that the generated 3D model retains the proportions and spatial relationships of the original 2D blueprint without requiring extensive manual adjustments.

To enhance user interaction, Model Viewer is used for rendering the 3D floor plan in real-time, allowing users to zoom, rotate, and navigate through the space effortlessly. This capability provides a more intuitive way to explore architectural designs, making it easier for stakeholders to understand the layout before construction begins. Since the system is web-based, it is also lightweight and accessible on multiple devices, eliminating the need for specialized hardware.

The accuracy and effectiveness of the system are validated by comparing the generated 3D top-down view with the original 2D blueprint. This comparison ensures that wall placements, proportions, and overall layout remain consistent throughout the conversion process. Additionally, performance testing across various architectural designs confirms the system's adaptability, proving its ability to handle different levels of complexity efficiently.

The 3D Scene Virtualization for Floor Planning system successfully bridges the gap between traditional architectural planning and modern 3D visualization, offering a precise, automated, and efficient method for floor plan analysis. By eliminating the limitations of 2D representations, this system provides a more

interactive and accurate way to understand spatial structures, significantly improving the planning and design process for architects, engineers, and clients.

7.1 FUTURE WORK

The 3D Scene Virtualization for Floor Planning system has demonstrated significant potential in automating the conversion of 2D blueprints into interactive 3D models. However, there are several areas where the system can be improved to enhance accuracy, efficiency, and user experience. One key area of future development is AI-driven room classification and labeling, which would allow the system to automatically recognize different spaces, such as bedrooms, kitchens, or bathrooms, based on predefined datasets. This would improve usability and provide a clearer representation of floor plans.

Another potential enhancement is the integration of real-time editing capabilities, allowing users to modify the generated 3D models directly within the application. By implementing drag-and-drop features for walls, doors, and furniture, designers and architects can make instant changes, reducing the need for external modeling software. This would make the system more interactive and user-friendly, catering to a wider range of users.

Additionally, Augmented Reality (AR) integration could further elevate the system's functionality by allowing users to project the 3D model into real-world environments using mobile devices or AR headsets. This would enable clients and designers to visualize how a floor plan would look and fit within a given space, making decision-making more immersive and efficient.

Expanding support for different file formats, such as IFC and OBJ, would also increase the system's compatibility with existing architectural and design tools. This would allow seamless data exchange between different platforms, ensuring that professionals can integrate the system into their workflows without limitations.

Lastly, optimization for performance and scalability is crucial for handling larger and more complex floor plans. Implementing more efficient algorithms for 3D rendering and data processing would allow the system to process high-resolution blueprints faster, making it suitable for large-scale architectural projects. These improvements would ensure that the system remains robust, efficient, and widely applicable in the field of architectural visualization.

REFERENCES

[1] Alcañiz M L, Garcia C, Ivorra E, Mora P, Ortega M and Valenza G, "Virtual Experience Toolkit: Enhancing 3D Scene Virtualization From Real Environments Through Computer Vision and Deep Learning Techniques," 2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE), Milano, Italy, 2023, pp. 694-699, doi: 10.1109/MetroXRAINE58569.2023.10405757.

[2] Boer C R, El-Chaar J, Mazzola S, Maso G D and Pedrazzoli P, "Interactive 3D virtual environments for industrial operation training and maintenance," The Proceedings of 2019 9th International Conference on Reliability, Maintainability and Safety, Guiyang, China, 2019, pp. 1376-1381, doi: 10.1109/ICRMS.2019.5979485.

[3] Deshmukh P et al., "2D to 3D Floor plan Modeling using Image Processing and Augmented Reality," 2023 International Conference on Recent Advances in Electrical, Electronics & Digital Healthcare Technologies (REEDCON), New Delhi, India, 2023, pp. 682-687, doi: 10.1109/REEDCON57544.2023.10151165.

[4] Holmes J, Maschner H D G and Schou C D, "Virtualization and the democratization of science: 3D technologies revolutionize museum research and access," 2023 Digital Heritage International Congress (DigitalHeritage), Marseille, France, 2023, pp. 265-271, doi: 10.1109/DigitalHeritage.2023.6744763.

[5] Jeng T R et al., "New 3D Image Technologies Developed in Taiwan," in IEEE Transactions on Magnetics, vol. 47, no. 3, pp. 663-668, March 2021, doi: 10.1109/TMAG.2021.2102344.

[6] Li L, "Application of Computer 3D Technology in Graphic Design of Animation Scene," 2022 2nd International Conference on Computer Graphics, Image and Virtualization (ICCGIV), Chongqing, China, 2022, pp. 42- 45, doi: 10.1109/ICCGIV57403.2022.00013.

[7] Li G, "Abnormal Recognition Technology of 3D Virtual Scene Image Based on Wireless Network Sensor," 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), Ballari, India, 2023, pp. 1-6, doi: 10.1109/AIKIIE60097.2023.10389837.

[8] Liu Y and Ning K, "Surface Transformer for 3D Object Detection," 2022 2nd International Conference on Computer Graphics, Image and Virtualization (ICCGIV), Chongqing, China, 2022, pp. 159-164, doi:

10.1109/ICCGIV57403.2022.00038.
[9] Lei Y and Zhang Y, "An improved 2D-3D medical image registration algorithm based on modified mutual information and expanded Powell method," 2023 IEEE International Conference on Medical Imaging Physics and Engineering, Shenyang, China, 2023, pp. 24-29, doi: 10.1109/ICMIPE.2023.6864496.
[10] Zhang S, "Application of 3D Digital Technology in Virtual Laboratory Training," 2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI), Zhuhai, China, 2022, pp. 39-42, doi: 10.1109/IWECAI55315.2022.00015.

APPENDIX

A. SOURCE CODE

A.1 views.py

```
from django.shortcuts import redirect, render
from .models import Document, HouseModel
from .forms import DocumentForm
import os
from django.conf import settings
from .wallDetect import fetch

def my_view(request):
    openModel = False
    if request.method == 'POST':
        form = DocumentForm(request.POST, request.FILES)
        if form.is_valid():
            newdoc = Document(docfile=request.FILES['docfile'])
            newdoc.save()
            name = newdoc.docfile.name
            fetch(name)
            newdoc.docfile.delete()
            newdoc.delete()
            openModel = True
    else:
        form = DocumentForm()
    documents = list_plans()

    context = {'documents': documents, 'form': form, 'openModel': openModel}
    return render(request, 'index.html', context)

def list_plans():
    Mdir = os.path.join(settings.MODELS_ROOT)
    houses = [
```

```
        os.path.join(settings.MEDIA_URL, 'models//', file)
        for file in os.listdir(Mdir)
        if file.endswith(('.glb'))
    ]
    return sorted(houses, key=lambda x: int(x.split("model")[-1].split(".")[0]))[::-1]
```

A.2 wallDetect.py

```
import os
import json
from django.conf import settings
import cv2
import numpy as np

def fetch(filename):
        print("[SERVER] Detecting Edges", os.path.join(settings.MEDIA_ROOT,
filename))
        img = cv2.imread(os.path.join(settings.MEDIA_ROOT, filename))
        imgX = np.flip(img, axis=1)
        cv2.imwrite(os.path.join(settings.MEDIA_ROOT, filename), imgX)
        gray = cv2.cvtColor(imgX, cv2.COLOR_BGR2GRAY)

        # convert gray to binary image
        _, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)

        # noise removal
        kernel = np.ones((3, 3), np.uint8)
        opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

        # dilate image
        sure_bg = cv2.dilate(opening, kernel, iterations=3)

        # sharpen image
```

```
    dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)

    # get ceter of walls
    ret,sure_fg = cv2.threshold(0.5*dist_transform, 0.2 * dist_transform.max(),
255,0)

    # remove center and keep only edges
    sure_fg = np.uint8(sure_fg)
    wall_img = cv2.subtract(sure_bg, sure_fg)

    # extract contours
    boxes = []
    contours, _ = cv2.findContours(wall_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # simplify outlines
    for cnt in contours:
        epsilon = 0.001 * cv2.arcLength(cnt, True)
        approx = cv2.approxPolyDP(cnt, epsilon, True)
        boxes.append(approx)

    # finding outer contour
    ret, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY_INV)

    contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # optimize the contour
    largest_contour_area = 0
    for cnt in contours:
        if cv2.contourArea(cnt) > largest_contour_area:
            largest_contour_area = cv2.contourArea(cnt)
            largest_contour = cnt
```

```
epsilon = 0.001 * cv2.arcLength(largest_contour, True)
contour = cv2.approxPolyDP(largest_contour, epsilon, True)

def points_inside_contour(points, contour):
    for x, y in points:
        if cv2.pointPolygonTest(contour, (int(x), int(y)), False) == 1.0:
            return True
    return False

res = []
for wall in boxes:
    for point in wall:
        if points_inside_contour(point, contour):
            res.append(wall)
            break
boxes = res

def scale_point_to_vector(boxes, height=0, scale=np.array([1, 1, 1])):
    res = []
    for box in boxes:
        for pos in box:
            res.extend([[(pos[0]) / 100, (pos[1]) / 100, height]])
    return res

def create_4xn_verts_and_faces(boxes, height=1, scale=np.array([1, 1, 1])):
    counter = 0
    verts = []

    for box in boxes:
        verts.extend([scale_point_to_vector(box, 1, scale)])
        counter += 1
    faces = []
```

```
        for room in verts:
            count = 0
            temp = ()
            for _ in room:
                temp = temp + (count,)
                count += 1
            faces.append([(temp)])

        return verts, faces, counter

print("[SERVER] Generating 3D model")
verts, faces, _ = create_4xn_verts_and_faces(boxes=boxes)

with open("scans/wall_horizontal_verts.txt", "w") as f:
        f.write(json.dumps(verts))

with open("scans/wall_horizontal_faces.txt", "w") as f:
        f.write(json.dumps(faces))

os.system("cd ..\\blender && blender -b -P ..\\floorplanapp\\upload\\3dfily.py")

from skimage.metrics import structural_similarity as ssim

kernel = np.ones((3, 3), np.uint8)  # Small kernel for noise removal
processed = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel)

if thresh.shape != processed.shape:
        thresh = cv2.resize(thresh, (processed.shape[1], processed.shape[0]))

edges = cv2.Canny(thresh, 100, 200)
reference_edges = cv2.Canny(processed, 100, 200)
matching_pixels = np.sum(edges == reference_edges)
```

```
        total_pixels = edges.shape[0] * edges.shape[1]
        edge_accuracy = (matching_pixels / total_pixels) * 100

        ssim_score = ssim(edges, reference_edges)

        print("\n")
        print(f"Wall Edge Accuracy: {edge_accuracy:.2f}%")
        print(f"SSIM Edge Similarity: {ssim_score:.4f}")
        print("\n")
```

A.3 3dfily.py

```
import os, bpy, json, numpy as np

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

bpy.ops.outliner.orphans_purge()
bpy.ops.outliner.orphans_purge()
bpy.ops.outliner.orphans_purge()

def init_object(name):
    # Create new blender object and return references to mesh and object
    mymesh = bpy.data.meshes.new(name)
    myobject = bpy.data.objects.new(name, mymesh)
    bpy.context.collection.objects.link(myobject)
    return myobject, mymesh

def average(lst):
    return sum(lst) / len(lst)

def get_mesh_center(verts):
    # Calculate center location of a mesh from verts
    x = []
```

```
    y = []
    z = []

    for vert in verts:
        x.append(vert[0])
        y.append(vert[1])
        z.append(vert[2])
    return [average(x), average(y), average(z)]

def subtract_center_verts(verts1, verts2):
    # Remove verts1 from all verts in verts2, return result, verts1 & verts2 must have
same shape!
    for i in range(0, len(verts2)):
        verts2[i][0] -= verts1[0]
        verts2[i][1] -= verts1[1]
        verts2[i][2] -= verts1[2]
    return verts2

def create_custom_mesh(objname, verts, faces, mat=None, cen=None):
    # Create mesh and object
    myobject, mymesh = init_object(objname)

    # Find center of verts
    center = get_mesh_center(verts)
    # Subtract center from verts before creation
    proper_verts = subtract_center_verts(center, verts)

    # Generate mesh data
    mymesh.from_pydata(proper_verts, [], faces)
    # Calculate the edges
    mymesh.update(calc_edges=True)

    parent_center = [int(cen[0] / 2), int(cen[1] / 2), int(cen[2])]
```

```
    # Move object to input verts location
    myobject.location.x = center[0] - parent_center[0]
    myobject.location.y = center[1] - parent_center[1]
    myobject.location.z = center[2] - parent_center[2]

    # add material
    if mat is None:  # add random color
        myobject.data.materials.append(
            create_mat(np.random.randint(0, 40, size=4))
        )  # add the material to the object
    else:
        myobject.data.materials.append(mat)  # add the material to the object
    return myobject

def create_mat(rgb_color):
    mat = bpy.data.materials.new(name="MaterialName")  # set new material to
variable
    mat.diffuse_color = rgb_color  # change to random color
    return mat

parent, _ = init_object("Floorplan")

with open("..\\floorplanapp\\scans\\wall_horizontal_verts.txt", 'r') as f:
    verts = json.loads(f.read())
with open("..\\floorplanapp\\scans\\wall_horizontal_faces.txt", 'r') as f:
    faces = json.loads(f.read())

boxcount = 0
wallcount = 0
wall_parent, _ = init_object("Walls")

for i in range(0, len(verts)):
```

```
    roomname = "VertWalls" + str(i)
    obj = create_custom_mesh(
        roomname,
        verts[i],
        faces[i],
        cen=np.array([0, 0, 0]),
        mat=create_mat((0.5, 0.5, 0.5, 1)),
    )

bpy.ops.object.select_all(action='DESELECT')

for obj in bpy.data.objects:
    if "VertWalls" in obj.name:
        obj.select_set(True)
        bpy.context.view_layer.objects.active = obj

bpy.ops.object.mode_set(mode='EDIT')
bpy.ops.mesh.extrude_region_move(MESH_OT_extrude_region={"use_normal_flip
":False, "use_dissolve_ortho_edges":False, "mirror":False},
TRANSFORM_OT_translate={"value":(0, 0, 1), "orient_type":'GLOBAL',
"orient_matrix":((1, 0, 0), (0, 1, 0), (0, 0, 1)), "orient_matrix_type":'GLOBAL',
"constraint_axis":(False, False, True), "mirror":False, "use_proportional_edit":False,
"proportional_edit_falloff":'SMOOTH', "proportional_size":1,
"use_proportional_connected":False, "use_proportional_projected":False,
"snap":False, "snap_elements":{'INCREMENT'}, "use_snap_project":False,
"snap_target":'CLOSEST', "use_snap_self":True, "use_snap_edit":True,
"use_snap_nonedit":True, "use_snap_selectable":False, "snap_point":(0, 0, 0),
"snap_align":False, "snap_normal":(0, 0, 0), "gpencil_strokes":False,
"cursor_transform":False, "texture_space":False, "remove_on_cancel":False,
"use_duplicated_keyframes":False, "view2d_edge_pan":False,
"release_confirm":False, "use_accurate":False, "use_automerge_and_split":False})
bpy.ops.mesh.select_mode(type='FACE')
bpy.ops.mesh.select_all(action='SELECT')
bpy.ops.mesh.normals_make_consistent(inside=False)
```

```
bpy.ops.object.mode_set(mode='OBJECT')

num = 1
while os.path.exists(os.path.join(f"..\\floorplanapp\\models\\model{num}.glb")):
    num += 1

bpy.ops.export_scene.gltf(filepath=f"..\\floorplanapp\\models\\model{num}.gltf")
bpy.ops.wm.quit_blender()
```

A.4 Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Upload Model</title>
        <script type="module" src="https://ajax.googleapis.com/ajax/libs/model-viewer/4.0.0/model-viewer.min.js"></script>
        <script src="https://cdn.tailwindcss.com"></script></head>
    <body class="flex flex-col bg-[#1A1D2E] text-[#E0E3E8]">
        <div id="loading" class="absolute min-w-full min-h-screen z-20 backdrop-opacity-10 backdrop-invert bg-white/30 flex items-center justify-center">
            <button disabled type="button" class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4 focus:ring-blue-300 font-medium rounded-lg text-sm px-5 py-2.5 text-center me-2 dark:bg-blue-600 dark:hover:bg-blue-700 dark:focus:ring-blue-800 inline-flex items-center">
                <svg aria-hidden="true" role="status" class="inline w-4 h-4 me-3 text-white animate-spin" viewBox="0 0 100 101" fill="none" xmlns="http://www.w3.org/2000/svg">
                <path d="M100 50.5908C100 78.2051 77.6142 100.591 50 100.591C22.3858 100.591 0 78.2051 0 50.5908C0 22.9766 22.3858 0.59082 50 0.59082C77.6142 0.59082 100 22.9766 100 50.5908ZM9.08144 50.5908C9.08144
```

```
73.1895 27.4013 91.5094 50 91.5094C72.5987 91.5094 90.9186 73.1895 90.9186
50.5908C90.9186 27.9921 72.5987 9.67226 50 9.67226C27.4013 9.67226
9.08144 27.9921 9.08144 50.5908Z" fill="#E5E7EB"/>
            <path d="M93.9676 39.0409C96.393 38.4038 97.8624 35.9116 97.0079
33.5539C95.2932 28.8227 92.871 24.3692 89.8167 20.348C85.8452 15.1192
80.8826 10.7238 75.2124 7.41289C69.5422 4.10194 63.2754 1.94025 56.7698
1.05124C51.7666 0.367541 46.6976 0.446843 41.7345 1.27873C39.2613 1.69328
37.813 4.19778 38.4501 6.62326C39.0873 9.04874 41.5694 10.4717 44.0505
10.1071C47.8511 9.54855 51.7191 9.52689 55.5402 10.0491C60.8642 10.7766
65.9928 12.5457 70.6331 15.2552C75.2735 17.9648 79.3347 21.5619 82.5849
25.841C84.9175 28.9121 86.7997 32.2913 88.1811 35.8758C89.083 38.2158
91.5421 39.6781 93.9676 39.0409Z" fill="currentColor"/>
            </svg>
            Loading...
          </button>
        </div>
        <nav class="bg-gray-800">
          <div class="mx-auto max-w-7xl px-2 sm:px-6 lg:px-8">
            <div class="relative flex h-16 items-center justify-between">
              <div class="flex flex-1 items-center justify-center sm:items-stretch
sm:justify-start">
                <h1 class="text-white text-xl md:text-3xl font-bold">3D Scene
Virtualization for Floor Planning</h1>
              </div>
            </div>
          </div>
        </nav>
        <!-- Upload form. Note enctype attribute! -->
        <form class="flex flex-col items-center space-y-4 md:flex-row md:space-y-0
md:space-x-4 md:justify-center p-10" action="{% url "home" %}" method="post"
enctype="multipart/form-data">
          {% csrf_token %}
          {{ message }}
          {{ form.docfile }}
```

```
        <button id="submit" type="submit" class="bg-gradient-to-r from-[#4C5FD5]
to-[#5963E0] text-white px-5 py-2 rounded-lg shadow-md hover:scale-105
hover:shadow-lg transition">Upload plan</button>
    </form>
    <main class="min-h-screen py-6 px-100 bg-gradient-to-b from-gray-800 to-
gray-600 rounded-3xl">
    {% if documents %}
        <p class="text-2xl text-center pb-2 text-white">Recents:</p>
        <ul class="flex flex-wrap gap-6 justify-center">
            {% for document in documents %}
                <li class="hover:shadow-[0px_0px_15px_rgba(100,149,237,0.5)]
transition-all p-4 rounded-lg bg-gray-800">
                    <model-viewer class="model" src="{{ document }}" ar shadow-
intensity="1" camera-controls touch-action="pan-y">
                    <button slot="ar-button" class="px-6 py-3 text-lg font-bold text-white
bg-opacity-20 backdrop-blur-lg bg-gray-800 border border-white rounded-xl
shadow-lg hover:bg-opacity-40 transition-all duration-300">
                        View Model
                    </button>
                </model-viewer></li>
            {% endfor %}
        </ul>
    {% else %}
        <p>No documents.</p>
    {% endif %}
    </main>
    <div id="overlay" class="absolute flex items-center justify-center h-screen w-
screen bg-white/10 backdrop-blur-lg hidden">
        <model-viewer src="{{ documents.0 }}" ar shadow-intensity="1" camera-
controls touch-action="pan-y" class="h-full w-screen">
            <button slot="ar-button">
                OPEN AR
            </button>
```

```
            </model-viewer>
            <button id="close" class="absolute top-10 right-10 bg-red-500 text-white px-
3 py-1 rounded">X</button>
        </div>
        <script>
            const models = document.querySelectorAll(".model");
            let loadingBottom = document.getElementById('submit');
            let loadingScreen = document.getElementById('loading');
            let overlayScreen = document.getElementById('overlay');
            loadingScreen.style.display = 'none';
            loadingBottom.onclick = () => {
                loadingScreen.style.display = 'flex';
            }
            models.forEach(model => {
                model.addEventListener('click', () => {
                    overlayScreen.querySelector("model-viewer").src = model.src;
                    console.log(model.src);
                    overlayScreen.classList.remove('hidden');
                    window.scrollTo({ top: 0, behavior: "smooth" });
                    document.body.style.overflow = "hidden";
                })
            })
            overlayScreen.querySelector("#close").onclick = () => {
                overlayScreen.classList.add('hidden');
                document.body.style.overflow = "auto";
            }
            var openModel = "{{ openModel }}";
            if (openModel == "True") {
                overlayScreen.classList.remove('hidden');
            }
        </script>
    </body>
</html>
```

B. SCREENSHOTS

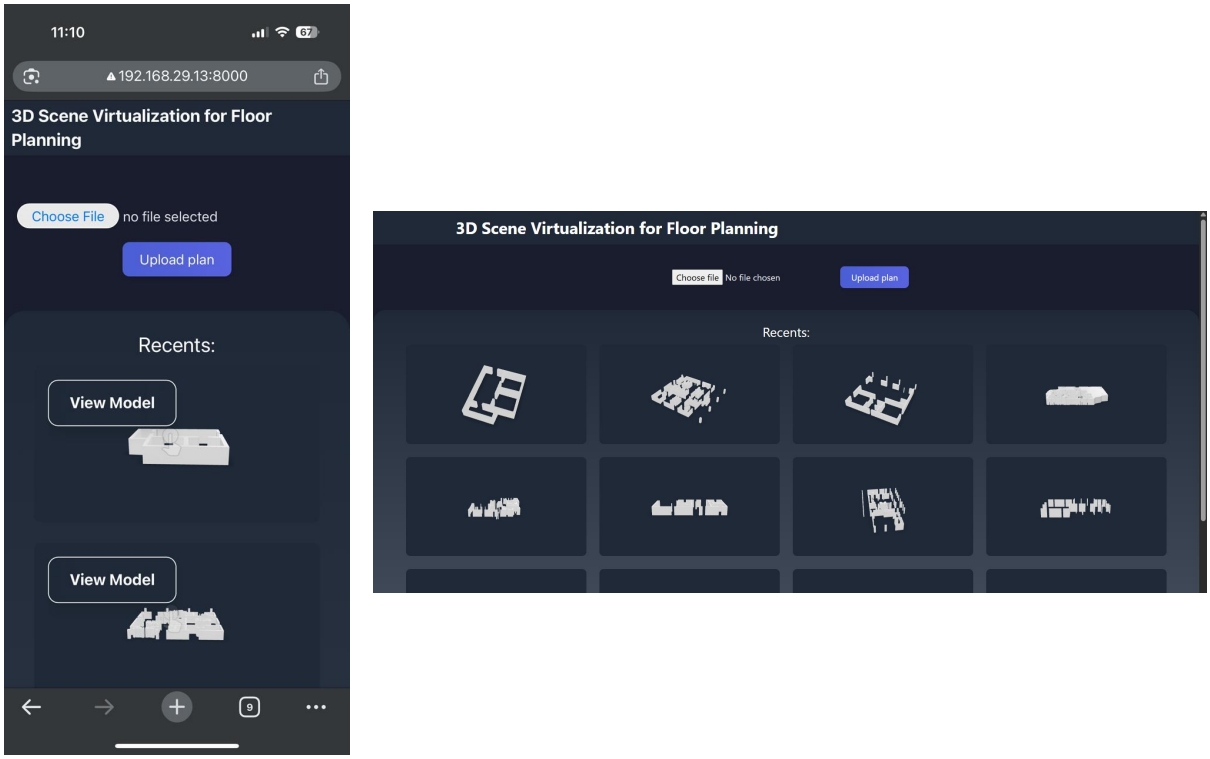

Fig B.1 Django web user interface (mobile and desktop)

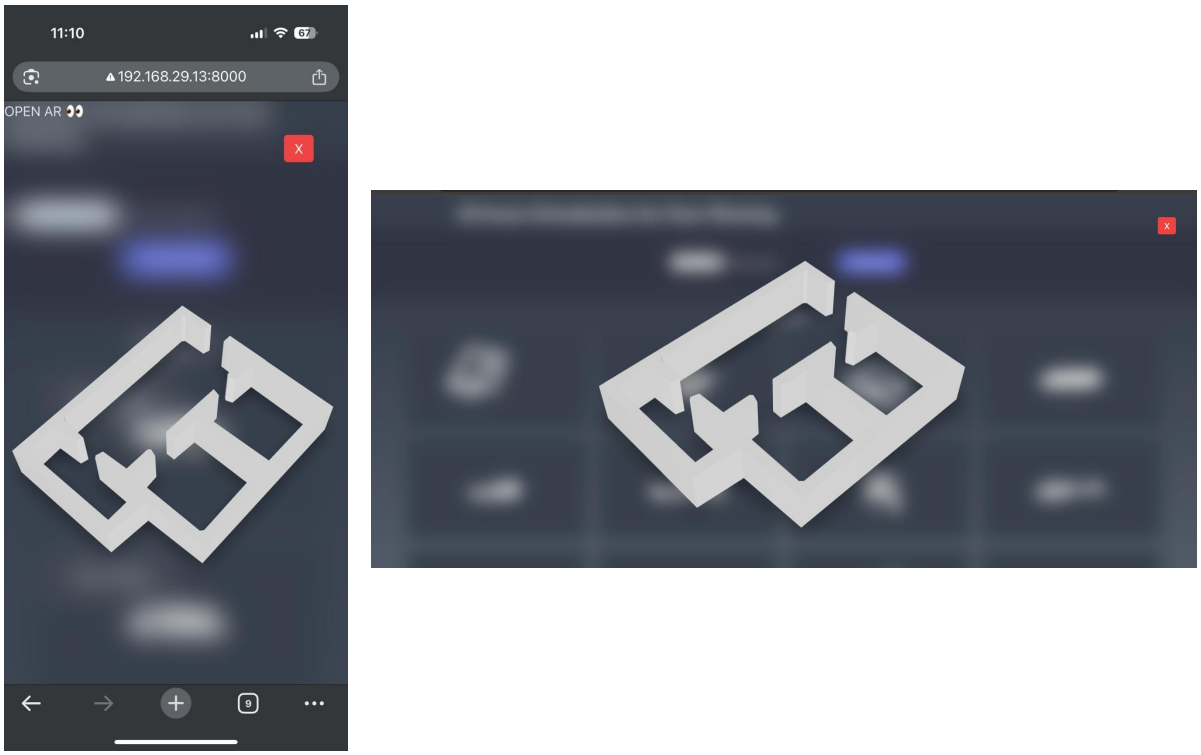

Fig B.2 3D floor plan model viewer

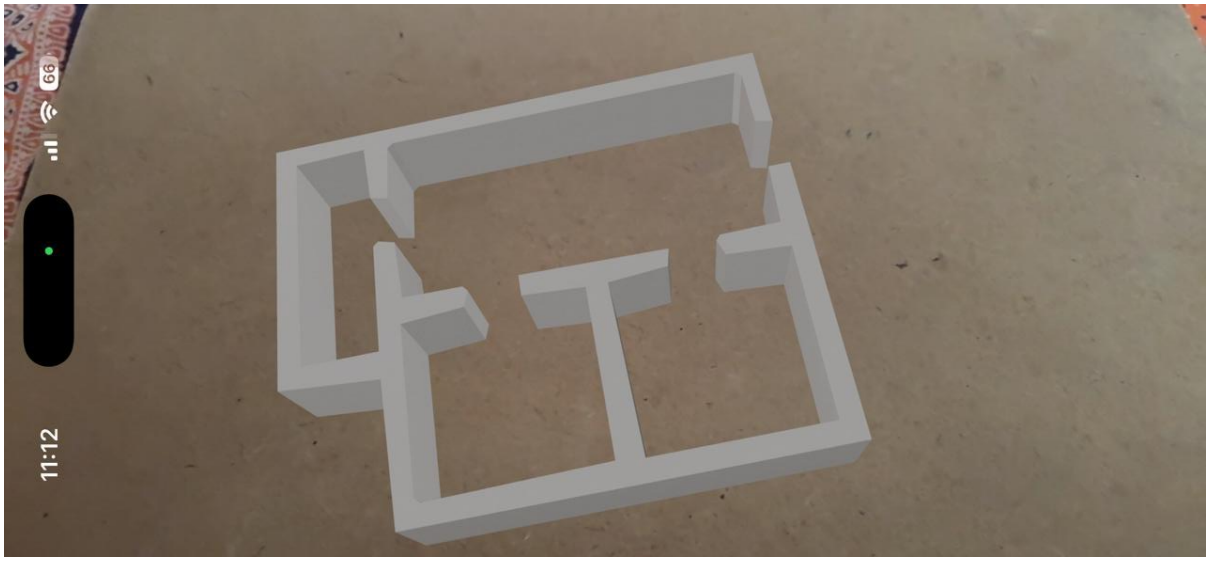

Fig B.3 AR floor plan view

C. CONFERENCE CERTIFICATE

Fig C.1 Conference Certificate

Fig C.2 Conference Payment Receipt

D. RESEARCH PAPER

3D Scene Augmentation for Floor Planning

Allam Sai Sravana Kumara Vignesh
Sathyabama Institute of Science and Technology
Chennai, IN
allam.vignesh@gmail.com

Gangadhara Ravi Teja
Sathyabama Institute of Science and Technology
Chennai, IN
gangadhararaviteja092003@gmail.com

Ms. B Bala Sai Gayathri, M.E.
Sathyabama Institute of Science and Technology
Chennai, IN
balasaigayathri.cse@sathyabama.ac.in

***Abstract*—Using 3D virtual environments in floor planning offers major improvements over traditional 2D blueprints. By transitioning from traditional 2D blueprints to detailed 3D models, designers and architects can gain a more comprehensive and interactive understanding of spatial arrangements. The process involves turning 2D plans into 3D models through data collection, modeling, and rendering. Challenges related to software integration, data management, and user interface design are discussed, with strategies for overcoming these issues. AR then allows users to explore these models in a realistic, interactive way before any physical construction begins. This makes it easier to collaborate and make decisions based on a clear view of how the design will work in real life. This approach helps people see and adjust designs before they are physically built, making the planning process more accurate and collaborative.**

Keywords—Machine Learning, Augmented Reality, Floor plan, Edge detection, Computer Vision

I. INTRODUCTION

Two-dimensional blueprints have long been used in the floor planning industry to represent spatial designs. Even though 2D drawings have been crucial to the fields of architecture and interior design, they usually fall short of capturing the intricate detail of a given space. The use of three-dimensional (3D) scene virtualization is changing how spaces are perceived, created, and experienced because of technological advancements.

By creating detailed three-dimensional models of environments, 3D scene virtualization allows designers and architects to work with their designs more accurately and immersive than they could with only 2D sketches. Experts can produce realistic representations of floor plans using advanced modeling software and augmented reality (AR) technologies, which improves understanding of spatial relationships and design elements.

This approach improves decision-making and design accuracy in addition to visual clarity. Users can virtually navigate and examine spaces, assess the impact of different design choices, and make instantaneous changes. This interactive procedure enhances teamwork and offers a more intuitive and engaging method to present and improve design ideas, fostering better collaboration within the team.

The use of 3D modeling for floor plans is a relatively new method that has grown in popularity recently. Photogrammetry, laser scanning, computer-aided design (CAD), and other methods are some of the methods used in 3D modeling. Even though CAD software is frequently used to create 3D models of buildings, the process can take three to four days. Another method that calls for high- quality images is Photogrammetry. Although it can be costly, laser scanning is also effective. On the other hand, the suggested system offers more features like zoom-in and zoom-out along with quicker results. Additionally, no prior knowledge of CAD software is necessary to operate the suggested system, which is simple to use.

II. LITERATURE SURVEY

[1] The Virtual Experience Toolkit (VET) uses computer vision and deep learning to quickly and accurately create 3D models from real indoor environments. This automated process improves the efficiency and precision of 3D visualization, enhancing VR applications. They integrated a single application to automate virtualization of 3D objects using deep learning methods. It can handle indoor objects and can work with around 200 classes. This framework was used to analyze scenes from the ScanNet dataset and some custom indoor scenes. This approach is set to be used for mental treatment by creating a safe virtual environment created using VET. The user will experience new realities without phobic situations. The VET can be highly scaled and is an automatic framework.

[2] The combination of virtual reality (VR) and 3D animation has revolutionized effects in the media industry, resulting in more expressive and captivating work. VR adds realism to user experience by integrating physical elements and character interactions. Motion capture and 3D or aerial photos enable real-time physical interaction with 3D models. However, advanced workflows and data loads are issues. But, innovations in visual media do offer new levels of realism and creativity that were never experienced before. They gave a new 3D technology method for designing animation scenes that is faster than the binocular vision method for point cloud computation. By using a grid node for depth information, the new method significantly reduces computation time. For example, at 12 experiments, it takes 48.97 minutes compared to 128.97 minutes for the binocular vision method.

[3] This study aims to explore the use of deep learning and 3D visualized scenes in wireless identifying anomalies in a sensor image. Techniques for detecting and classifying anomalies are centered on a number of issues, such as equipment malfunctions, signal interference, and an overall amount of pattern data. CNN-based deep learning algorithms are ideal for these kinds of anomaly identification methods. One only needs to produce a collection of training images that can be used as an adequate representation of common phenomena in order to correctly identify instances of a 3D visualized complex scene that also contain abnormalities. Because it enables a much more dependable system by reacting in real time to damage or interference patterns, this application is very helpful when analyzing sensor data on a larger scale. Furthermore, it facilitates a clearer understanding of how sensor anomalies function, leading to a

more positive method of operating and evaluating sensor systems. In conclusion, the techniques enable the creation of a thorough strategy for tracking sensor network performance and identifying issues, expanding the range of their uses and dependability.

[4] The purpose of this study is to learn how to use three-dimensional modeling technology to create virtual environments for interactive teaching. In addition, the data-driven approach is used to model the equipment's dynamic states, and VRML enables user-equipment interaction in a virtual environment.Additionally, the inclusion of multi-role interaction technology in the virtual environment promotes collaboration between students and teachers as well as between students. Students' practical skills, teamwork abilities, and participation in the learning activities are all developed through this kind of active and participatory learning. Additionally, the virtual operation environment increases learners' motivation in addition to improving training outcomes. It discusses using 3D virtual reality technology to create training environments for electrical equipment in higher education. Using 3ds Max, virtual components are statically modeled, rendered to match standard experimental equipment, and then exported. This approach facilitates comprehensive sensory simulations and realistic training scenarios.

[5] The authors of this paper discuss a surface transformer-based, one-stage 3D object detection method that incorporates a transformer 3D encoder and a new local umbrella surface description. Finely detailed local features are captured by the local umbrella surface description, which encodes first order gradient slices of each object point as feature vectors. In order to better describe the object, the transformer 3D encoder combines local surface features with global long-term features extracted from the input point cloud. The network can focus on the central regions of the point cloud scenes by using an instance- aware down sampling technique. The results show that this technique outperforms the other one-stage detectors among the current methods.

[6] Using image processing and augmented reality (AR), this paper presents a novel approach to reconstructing three-dimensional models from two-dimensional floor plans. Computer vision algorithms are used to determine the arrangement and measurements of a 2D paper plan that was taken with a smartphone camera. Interactive 3D representations of the 2D models are produced using AR tools, allowing for dynamic exploration. Tested on real building floor plans, the approach produced accurate and efficient results. This development provides a new generation of augmented reality tools that help designers or architects collaborate and communicate more effectively. The improved visualization of the designs helps clients comprehend them more clearly.

[7] In this paper, the authors attempt to show how virtual environments can be used in training as well as in the maintenance and repair of manufacturing facilities and industrial plants. The authors describe a software framework designed to improve industrial system assembly, operation, and maintenance tasks in the context of 3D modeling, animation, and simulation. Because the platform provides trainees with realistic and interactive experiences, it lowers training costs and increases training efficiency. By simulating their operation in an interactive setting, it also aids in the maintenance of industrial installations. Two applications illustrate the framework's adaptability and show how it can enhance industrial procedures. Additionally, this represents a significant advancement in the application of virtual environments for industrial maintenance and training. The system's performance improvement during real-world applications is encouraging because it points to a wide range of potential uses for this technology.

[8] Although the Virtual Repository is still under development, this paper outlines its development with the goal of serving as a research and teaching tool for museum collection access. The repository improves the research and education outreach of distant researchers, students, and the general public by filling in the gaps in remote access. For many decades, it builds a permanent online library of artifacts, fossils, locations, and information for use in research and education. The goal is to produce researchers who are focused on museums and conduct analyses that were previously impossible. Additionally, the repository offers researchers, educators, and students new ways to engage with virtual collections, which has the potential to change how science is conducted and taught. It addresses the challenges researchers face with data aggregation in various scientific fields, particularly in natural history and archaeology. It points out that limited access to collections has led to issues with data comparability and reliance on published conclusions. To overcome these challenges, it advocates for the creation of virtual repositories using advanced 3D technologies and image-based database systems. These virtual collections, accessible online, can significantly enhance scientific analysis and make collections available to researchers, students, educators, and the public globally.

[9] This study introduces an improved image matching method that addresses the registration of three-dimensional CT and two-dimensional x-ray images. It is important to register these images so that the area of interest is precisely maintained within the radiation zone when the patient is exposed to radiation. The authors examine the application of various matching criteria to attain proper image registration and observe that partial volume interpolation- based mutual information (MI) appears to be the most effective. Three search strategies have been tested for this purpose, and the best one for in-plane refined search is a modified Powell method. The average mean square error for translation and rotation using the suggested algorithm is as low as 0.26 mm and 0.38 degrees, respectively, according to qualitative and quantitative analysis of the results obtained. The simulations validate the suggested approach, which asserts that it is clinically viable. The correspondence problem solution in medicine is improved by this study.

[10] This work focuses on low-cost methods of producing and viewing 3D objects by developing a novel architecture for 3D acquisition, transmission, and display. It is cloud-based and provides explanations for all 3D tasks, making it an extensible and user-friendly interface. In order to increase the depth of 3D materials, the system has a crucial 2D to 3D video conversion technology that is supplied both in the synthesis stage and in the real-time processor. Image-based rendering techniques with replication structure are used to enable thin penetrability and guarantee the efficient and high-quality representation of 3D content. When anti-aliasing pixel rendering is used, the quality of the

3D presentation is improved. By using integrated 2D/3D windows (i2/3DW) with a micro retarder, the user can interact with the 3D environment by viewing both 2D and 3D images on the same screen. Such a localized 2D/3D display can be used in web services, 3D TVs, movies, and games. The paper also lists a few 3D image formats that may be utilized in the future for 3D technologies. Generally speaking, this strategy should make it easier to create an operational breakthrough that will enable the deployment of more sophisticated 3D algorithms for wider applications.

III. EXISTING SYSTEMS

AR platforms like Microsoft HoloLens, Apple ARKit, and Google ARCore are laying the foundation for 3D space. To capture the spatial context, these systems integrate sensor fusion and computer vision. They can place 3D virtual simulations in real life for viewers to view and interact with, increasing immersion, by comprehending the surfaces, lights, and motions of objects. These platforms can be used in a wide range of fields, including gaming, retail, education, and even industrial design. For instance, ARKit makes use of SLAM technology and depth information to enable extremely accurate 3D object placement. In order to improve body-less movement, Earth also recognizes motion tracking and comprehends the environment. These platforms may, however, require supporting hardware for improved processing power which may be difficult for low- end devices.

Marker-based systems use physical markers, like tags, QR codes, or other preset patterns, to position the 3D content in a scene. These markers were used in the development of platforms like Vuforia and ARToolkit, which allow users to precisely position virtual objects in the real world. In order to attach the augmentation to a marker, the system ascertains the marker's position, orientation, and scale after a camera takes a picture of it. This method works well in controlled environments like museums, product displays, or AR-based training. Its dependability and low processing requirements, which enable it to function on outdated devices, are its most valuable features. Quite the opposite, the use of such markers does not allow the flexibility since an augmentation can only be done in a predetermined place where the marker ID can be seen.

Marker-less AR uses sophisticated features like SLAM to recognize and create surfaces and other features in space because it lacks physical markers. The two most popular tools in this field are ARCore and ARKit, which enable the addition of depth, textures, and geometry to any space. These systems are appropriate for applications such as navigation, virtual interior design, and augmented reality games like Pokemon GO. Working with marker-less AR has the advantage of being more adaptable than marker- based techniques and operating in more unpredictable and dynamic environments. However, such AR necessitates complex hardware and suffers from poor tracking accuracy in feature-poor environments (plain walls or low light).

3D models of actual locations can be created and reconstructed with the use of technologies like Li-DAR and photogrammetry. In order to create accurate representations of real space, including virtual objects, these systems collect and integrate texture and depth data. Apple devices can now precisely record the depth of their surroundings in real time thanks to Li-DAR integration, which enables occlusions and more lifelike interactions with other virtual objects in augmented reality. Overlapping images then produce a photogrammetry model, which is used extensively in computer games, anthropology, and even urban planning. These technologies are demanding in computation and for larger space modeling, but they guarantee a very high degree of realism and accuracy.

Augmented interactive 3D holograms are used by Microsoft HoloLens, Magic Leap, and other comparable devices to interact with real-world applications. These holographic systems use spatial mapping and gesture recognition to enable smooth user interaction with augmented elements. Joint modeling, medical training, and military training are a few possible uses. For example, engineers can view 3D models of machine designs, and medical students can use virtual structures to learn about the anatomy of the human body. Although holographic enhancement offers the highest level of immersion and customization, it is too costly for average users and necessitates precise calibration to maintain the necessary spatial relationship as well.

By projecting three-dimensional images onto asymmetrical shapes like buildings, sculptures, or objects, these projection systems enhance the visual appeal of the physical world. Moving images produced by technologies like 3D projection mapping are frequently used in advertising, show business, and architectural visualization because they are not static with respect to the mapped surfaces. Motion, texture, and artificial lighting are all animated and visualized by them. For instance, projection mapping transforms event landmarks into a storytelling tool. Though some one-time setups, calibrations, and special equipment are required, this method is undoubtedly sufficient for dramatic presentations; however, it detracts from the realism of everyday or small-scale use approaches.

In the meta verse, Meta Quest Pro and Spatial are more than just cooperative systems. Regardless of user distance, these virtual systems allow users to interact with augmented 3D scenes in real-time. Cloud computing is necessary for these systems to perform a number of tasks, including alignment and spatial coordination. These features include virtual meetings, remote assist training, and collaborative design, among others. For example, students may work in groups on AR projects, and architects may work together on building designs. This tool is very useful for teamwork, but it has drawbacks as well. It needs a powerful internet connection and sophisticated equipment, which makes it challenging to use in places with limited bandwidth or on devices with less sophisticated features.

Engines like Unreal Engine and Unity are industry standards for producing augmentation scenes. These engines employ cutting-edge rendering technologies, which drive the application of realistic lighting and physics to produce augmented environments of the right caliber. Developers can create applications that are tailored for various mobile devices and AR headsets by integrating AR/MR plugins like Vuforia and AR Foundation into the application. However, game engines are not just used for gaming; they are also used for AR education and industrial simulations. However, smaller teams or individuals may find it difficult to afford the knowledge and resources needed to create augmented scenes.

Augmented elements have been added by Oculus Quest and HTC Vive, allowing users to work with virtual content while simultaneously being aware of their physical surroundings. With VR devices, for example, users can use real objects, like chairs and other gadgets, in a simulating environment by using displacement cameras. In industries like entertainment, prototyping, and training, this is frequently done. It is true that VR systems with pass- through cameras offer a great deal of immersion, but they are typically more expensive and have a steep learning curve for both the developer and the user to get the most for the devices.

IV. Proposed System

The advances in technology has given rise to fresh viewpoints in the domains of interior design, real estate, and application architecture. It takes a long time and requires a high level of technical expertise to convert 2D plans into 3D models using the traditional method. Furthermore, there is still a problem with showing these models in real time in AR. To address these issues, we present a Django application in this paper that automates the process of converting 2D floor plans into 3D models. It creates GLTF models that are appropriate for augmented reality and that are easily viewed in the environment or as simple 3D models by combining edge detection, vectorization, and Blender scripting.

The proposed system combines various image processing processes with 3D modeling and augmented reality visualization to provide a straightforward operating environment for the architect, designer, and end users. As shown in Figure 2, e-commerce users can upload a 2D floor plan because the web interface is easy to use. To create the outlines of the fundamental structural components, such as walls, doors, and windows, the uploaded image is put through preprocessing steps like edges detection and vectorization. These vectors are then processed by a Blender headless script to create a 3D model in GLTF format. After that, the client receives the model back with features that allow for 3D or augmented reality interaction.

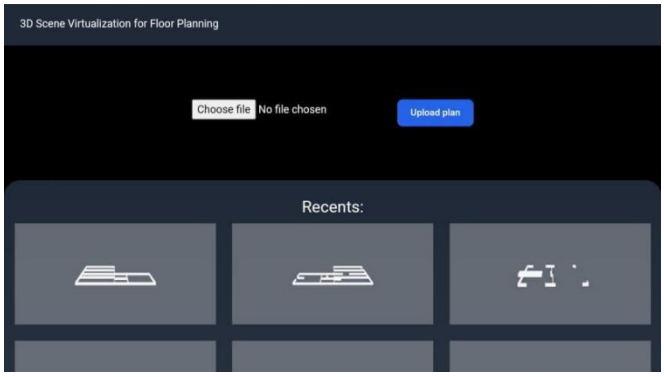

Fig. 1. UI for uploading floorplans

The vector data is processed by a custom Blender script that is running in headless mode on the server. Headless operation improves efficiency and enables remote execution by executing depth 3D model development without a graphical user interface. After parsing the geometric data, this script creates a three-dimensional shape. The structures are given height and dimensions using this script so that the model somewhat resembles the original floor plan. The end product is a GLTF file format that facilitates AR visualization across a wide range of platforms and is simpler to transfer.

The system begins as a Django web application that allows users to submit 2D floor plans in different image formats. The image undergoes a compatibility and quality check upon upload. Before the edges are identified, the original image goes through a phase where noise reduction and binarization are applied to improve image clarity. Since edges have a significant impact on the final product, it is also possible to argue that accurate 3D reconstruction cannot be produced without the right edges.

The image undergoes edge detection using Canny edge detection techniques following preprocessing. This technique identifies the edges and contours of the floor design plans. The chosen edges are then converted into vector data so that the design can be altered more easily. The scaled layout images must be vectorized in order to improve the retrieval of geometric information, which is why it is necessary as an input for the creation of a 3D model.

However, one of the system's main challenges continues to be the effective detection of edges for complex or low-quality floor plans. During the edge detection phase, adaptive threshold and robust preprocessing techniques are employed to increase efficacy. The automatic use of Django in Blender's 3D model generation process is also pertinent. This enables the system to operate remotely, which is crucial for the server implementation of the application, and eliminates the graphical overhead that slows down the process. Optimized scripts are used to prepare Blender models for rendering, improving both rendering speed and model quality.

The proposed system is widely applicable in fields such as architecture, interior design, and real estate. For instance, architects can use it to rapidly create 3D models from blueprints, while real estate brokers can use interactive augmented reality to help prospective buyers see the property. Furthermore, by allowing users without technical expertise to utilize it, the system can expand the application of 3D modeling. AR integration also helps users become more interested in and understand spatial structures, which helps translate a design from a two-dimensional plan into practical settings.

V. System Architecture

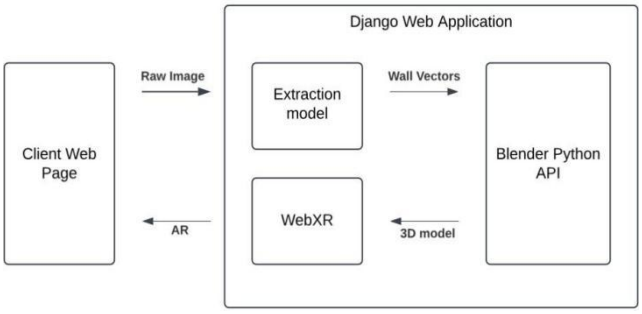

Fig. 2. Higher level Architecture diagram

VI. Methodology

By creating fully interactive 3D models from two-dimensional floor plan images, the "3D Scene Augmentation for Floor Planning" project seeks to give users an augmented reality experience. The process involves a number of steps, including image uploading, image preprocessing, edge

extraction, 3D model synthesis, and embedding AR tools. All of these stages serve as the foundation for giving users a striking and captivating environment that is integrated with 3D and AR floor plans. The following subsections of the methodology elaborates on how each of the components works.

The first step involves the user uploading a 2D floor plan image using a Django framework-based web application. The compatibility of the various image file formats in which the floor plan images can be uploaded is verified. A preprocessing step is applied to the image in order to enhance its quality before more complex operations are carried out on it. The image above is subjected to noise filtering, edge sharpening, and conversion to greyscale. The image is primed for edge detection and vectorization through this preprocessing, which also creates the ideal environment for accurate 3D model generation and rendering.

The primary objective of this project is identifying the floor plan's edges. Walls, windows, and doors are among the structural elements that are identified using the Canny edge detection algorithm. The next step after detecting edges is vectorization, which transforms the pixel-based edges into geometric shapes like curves, straight lines, etc. for use in 3D modeling. This stage guarantees that the 2D floor plan and vector graphics for 3D rendering are included in the clear and expandable structural information.

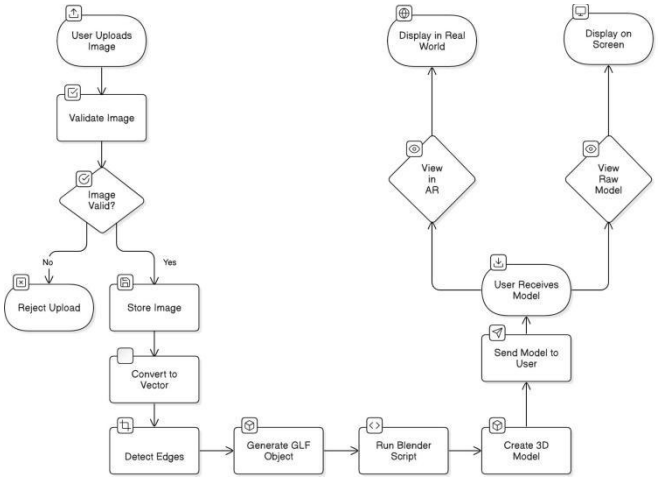

Fig. 3. Flowchart

The next stage involves taking the 3D models of the rooms after vector defining them, which is accomplished by running Blender in headless mode on the server. Blender's robust 3D rendering capabilities are used to create a 3D model from the vectorized image of the floor plan. The Blender script, which interprets vector data, controls an automated process that extrudes the walls, adds height to the structures, and constructs objects (in this case, doors and windows) in three dimensions. This method works faster and produces models more efficiently on a server because it doesn't require human input and doesn't use a graphical user interface. Optimization techniques are used to make the generated 3D model as efficient as possible in AR and to make it easy to load on a variety of devices. This means that in order to improve rendering speed without compromising visual quality, the model's polygon count and complex geometry must be reduced. The user can safely access and retrieve the 3D model after it has been optimized and uploaded to cloud-based systems. Because a large number of users can utilize the model without causing the system's performance to deteriorate, this approach ensures scalability. The result is a GLTF file, which is a lightweight, interactive three-dimensional file.

Once a 3D model has been created and stored in the database, AR can be used to access the model's functionality directly. By using WebXR API or other AR-enabling frameworks, AR can be utilized on smartphones or AR goggles. It gives users an immersive experience by enabling them to view the two-dimensional floor plan as a three-dimensional, movable object. To provide a clear image of the layout and dimensions of the floor plan, users can manipulate the model using augmented reality (AR) by rotating it in real time, zooming in and out, and scaling it.

The system provides two modes of interaction once the 3D model is finished. The viewer has the option to view the 3D model in augmented reality on their mobile device or download it in the GLTF format and view it in other 3D modeling programs. Specialists like architects and designers, as well as clients who wish to view the object arrangement in augmented reality or through a 3D model viewer, require this multitasking approach. Both personal and professional applications can enhance 3D scenes thanks to the interactive features.

Blender operates in headless mode to create the 3D models at the server level. Any plan that a user uploads to the server causes Blender scripts to run automatically on the server side. Backend functionality is provided by Django. It is in charge of all incoming requests that are sent to the Blender instance by the user. The user doesn't have to wait for the model to be processed and finished on a personal computer because Django and Blender work together to complete all the processes. Because fewer processing engines are used in this method, the overall cost of the approach is also decreased.

The user-friendly application uses new technologies to convert 2D floor plans into formats that allow for the creation of intricate 3D models. It makes use of Blender headless 3D modeling, augmented reality, and sophisticated system image processing techniques for edge detection to create an engaging and successful user experience. It is an entirely automated process that is capable, adaptable, and made for situations like communicating with clients and rendering building designs. The method ensures that the process is rendered quickly as a result of high-quality model generation while still allowing users to interact with the 3D design or 3D asinine, making it useful in many modern design processes. Refer to Figure 3 for a simplified flowchart on the working of the proposed model.

VII. Result and Analysis

The project's goals included improving the current models to operate better and reconstructing them in three dimensions from two-dimensional floor plans using unprocessed photos. The 3D models created were expected to improve the system's overall performance. In addition, the program offered 3D modeling for use in virtual or augmented reality settings.

The system's vectorization procedures and cutting-edge Canny edge detection algorithm both demonstrated a consistently high success rate in defining the floor plan's

outer edges. Despite some layering, it was still possible to outline windows, doors, and walls in the floor plans. The system was reasonably successful in preprocessing the floor plan to get it ready for rendering a 3D model, though there were still problems with very dense or low quality imagery. Figure 4 shows an example of converting flow plan images to 3d model.

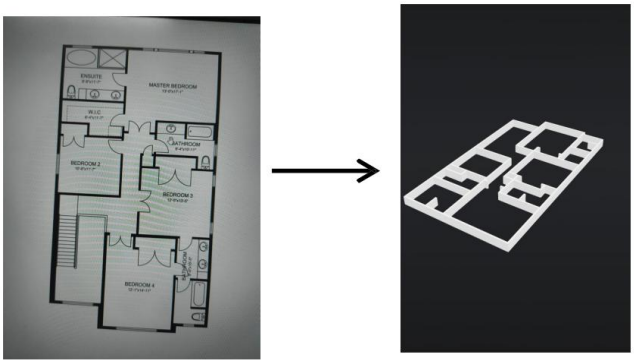

Fig. 4. 2D floorplan to 3D model

When using Blender in headless mode to convert the 2D architectural floor plans to 3D models, satisfactory results were obtained. With precisely extruded elements and meshed walls, the graphical models produced from vectorized data are true to the original. Because it allowed for integration with the viewers of both 3D models, the model's conversion into the GLTF format was suitable for this use. The capabilities of augmented reality integrated with the WebXR API were tested on a variety of devices. Because users could view and interact with the 3D model in real time, the augmented reality experience was consistent. Along with the inclusion of suitable scaling, rotating, and zooming features, the requirement to interact with the model in a real-world communal setting was also given. Additionally, because the system was compatible with mobile devices and consistently maintained a steady frame rate, users were able to fully visualize floor plans within their surroundings.

As shown in Figure 5, the 3d model was successfully augmented on to a coffee table. The overall performance of the system was evaluated using the total time required to complete the tasks as well as the number of resources used for each task. Depending on how complex the floor plans were, it took less than a few minutes to create 3D models on the server side using blender scripts. Users' concerns about scalability and access speed were addressed with the help of the strategy of saving and retrieving models via cloud storage. The speed of model generation and the quality of AR interactions were not significantly impacted by changes in load or the number of concurrent active users.

Fig. 5. 3D floor plan Augmentation

Architects, designers, and end users who tested the "3D Scene Augmentation for Floor Planning" system during its trial period gave very positive feedback, praising the system's usability and AR features, the accuracy of the 3D models, and the user-friendly interface. Customers in particular wanted to be able to view and work with the floor plans in 3D and AR formats because it made it easier for them to visualize the spaces. The users, however, requested additional features, such as the ability to instantly alter 3D objects and modify the features of multi-story buildings.

The system undoubtedly produced excellent results, but there was still opportunity for some enhancements. It was observed that the edge detection method occasionally suffered from more intricate floor plans or edges that were difficult to see, leading to small errors. The accuracy of edge detection and the general capacity to vectorize intricate 2D outlines into 3-dimensional models should both be enhanced by utilizing machine learning algorithms for automated pattern analyses. Additionally, adding augmented reality features like virtual tours or more tools for user customization would greatly enhance the user experience. All things considered, the system performed edge detection, modeling, and AR interface tasks efficiently, laying solid groundwork for future advancements in 3D scene augmentation technology and virtual design applications.

VIII. Conclusion

The "3D Scene Augmentation for Floor Planning" project successfully uses modern methods to turn 2D floor plans into captivating three-dimensional models. Through the use of Blender for 3D model creation, image processing with edge detection, and real-time interaction with the physical world through AR technologies, the framework facilitates interaction of floor plans in headless and augmented reality formats. By simplifying spatial layouts and facilitating interaction with them, the integration of these technologies enhances understanding of design theory and provides an impressive array of functionalities in architectural visualization and representation.

Despite the success, there is still room for improvement. When dealing with intricately designed floor plans or poorly defined borders, the edge detection technique is limited, which results in minor errors. The use of machine learning (ML) for more sophisticated edge detection and vectorization methods on more intricate floor plans can help to mitigate this drawback. In addition, there should be more options for AR features, like walkthroughs and real-time editing for virtual tours. Overall, this project opens up possibilities for 3D scene augmentation, including altering the way floor plans are viewed and the architectural planning and execution design process.

IX. Future work

A variety of modifications can significantly improve the system's functionality, accuracy, and user experience. First, the edge detection and vectorization step needs to be improved. As is, it is anticipated that the system will have errors in interpreting complicated or imprecise floor plans and geometry, leading to a relatively lower level of precision. The system might be strengthened by incorporating machine learning algorithms or deep learning models for edge and boundary detection, especially when working with intricate or distinctive building designs that demand a higher level of accuracy.

AR support is another goal that will be added in the future to address the system's current limitations. Although the system currently allows the user to place three-dimensional models in augmented reality, more advanced features like 3D visual tours, real-time user editing, or 3D models with multiple users would be advantageous. Additionally, enabling users to view various floor levels in augmented reality would be a useful addition to multi-story buildings. This would be advantageous for more experienced architects and designers working on more intricate projects.

Future directions for the project will include the integration of cloud storage and synchronous communication tools, allowing multiple users to collaborate on a floor plan while editing a 3D model or plans. The idea for additional actions would also be to increase the possibilities of AR, incorporate real-time changes, and make the system more adaptable. Additional enhancements would increase the system's efficacy as a spatial visualization tool by, among other things, streamlining the floor plan upload process, enhancing the user interface (UI) and user experience (UX), expanding the 3D customization options, and making it simpler for more users, including architects, interior designers, and real estate agents.

REFERENCES

[1] C. Garcia, P. Mora, M. Ortega, E. Ivorra, G. Valenza and M. L. Alcañiz, "Virtual Experience Toolkit: Enhancing 3D Scene Virtualization From Real Environments Through Computer Vision and Deep Learning Techniques," 2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE), Milano, Italy, 2023, pp. 694-699, doi: 10.1109/MetroXRAINE58569.2023.10405757.

[2] L. Li, "Application of Computer 3D Technology in Graphic Design of Animation Scene," 2022 2nd International Conference on Computer Graphics, Image and Virtualization (ICCGIV), Chongqing, China, 2022, pp. 42- 45, doi: 10.1109/ICCGIV57403.2022.00013.

[3] G. Li, "Abnormal Recognition Technology of 3D Virtual Scene Image Based on Wireless Network Sensor," 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), Ballari, India, 2023, pp. 1-6, doi: 10.1109/AIKIIE60097.2023.10389837.

[4] S. Zhang, "Application of 3D Digital Technology in Virtual Laboratory Training," 2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI), Zhuhai, China, 2022, pp. 39-42, doi: 10.1109/IWECAI55315.2022.00015.

[5] Y. Liu and K. Ning, "Surface Transformer for 3D Object Detection," 2022 2nd International Conference on Computer Graphics, Image and Virtualization (ICCGIV), Chongqing, China, 2022, pp. 159-164, doi: 10.1109/ICCGIV57403.2022.00038.

[6] P. Deshmukh et al., "2D to 3D Floor plan Modeling using Image Processing and Augmented Reality," 2023 International Conference on Recent Advances in Electrical, Electronics & Digital Healthcare Technologies (REEDCON), New Delhi, India, 2023, pp. 682-687, doi: 10.1109/REEDCON57544.2023.10151165.

[7] J. El-Chaar, C. R. Boer, P. Pedrazzoli, S. Mazzola and G. D. Maso, "Interactive 3D virtual environments for industrial operation training and maintenance," The Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety, Guiyang, China, 2011, pp. 1376-1381, doi: 10.1109/ICRMS.2011.5979485.

[8] H. D. G. Maschner, C. D. Schou and J. Holmes, "Virtualization and the democratization of science: 3D technologies revolutionize museum research and access," 2013 Digital Heritage International Congress (DigitalHeritage), Marseille, France, 2013, pp. 265-271, doi: 10.1109/DigitalHeritage.2013.6744763.

[9] Y. Lei and Y. Zhang, "An improved 2D-3D medical image registration algorithm based on modified mutual information and expanded Powell method," 2013 IEEE International Conference on Medical Imaging Physics and Engineering, Shenyang, China, 2013, pp. 24-29, doi: 10.1109/ICMIPE.2013.6864496.

[10] T. -R. Jeng et al., "New 3D Image Technologies Developed in Taiwan," in IEEE Transactions on Magnetics, vol. 47, no. 3, pp. 663-668, March 2011, doi: 10.1109/TMAG.2010.2102344

3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

20 Not Cited or Quoted 3%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
2% Publications
2% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.