International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 06 | Jun 2026 www.irjet.net p-ISSN: 2395-0072

VAASTU VISION: AI-Based Intelligent Floor Plan Analysis and 3D Architectural Visualization Using Computer Vision and Rule-Based Vaastu Analysis

[1]Prof. Supriya Patil, [2]Ms. Gauri Kulkarni, [3]Ms. Saloni Maskare, [4]Ms. Ashwini Narad, [5]Mr. Sanket Nikumbh

[12345]Department of Computer Engineering, Sinhgad College of Engineering, Pune, India

---

ABSTRACT-*Modern architectural planning requires intelligent systems capable of automating blueprint analysis, structural visualization, and user-centric design optimization. Traditional architectural visualization methods involve manual interpretation of floor plans and complex 3D modelling processes, which are time-consuming and require professional expertise. This paper presents "VAASTU VISION," an AI-based intelligent floor plan analysis and 3D architectural visualization system using Computer Vision techniques. The proposed framework automatically processes 2D blueprint images using OpenCV-based image processing methods such as contour detection, edge extraction, and coordinate mapping to identify structural elements including walls, rooms, and spatial layouts. The extracted structural data is transformed into interactive 3D architectural models using Three.js and React.js, enabling real-time visualization and user interaction. The system incorporates Vaastu Shastra principles to provide intelligent architectural recommendations for optimized room placement and directional alignment. Experimental results on 10 residential floor plans demonstrate wall detection accuracy of approximately 89%, average processing time of 1.8 seconds per blueprint, and successful Vaastu compliance analysis for all test cases. The proposed system offers a scalable and user-friendly solution for architects, engineers, interior designers, and homeowners.*

I. INTRODUCTION

The rapid advancement of Artificial Intelligence (AI), Computer Vision (CV), and interactive web technologies has significantly transformed the field of architectural visualization and smart infrastructure planning. Traditional architectural design processes rely heavily on manual blueprint interpretation, complex structural planning, and time-consuming 3D modelling techniques, which require professional expertise and extensive computational effort.

In India, Vaastu Shastra plays an important role in residential and commercial architectural planning. Vaastu principles are traditionally used to determine room placement, directional alignment, structural orientation, and spatial balance to promote positive energy and harmony within buildings. Despite its widespread use, most architectural software solutions do not incorporate Vaastu-compliant analysis, forcing users to depend on manual consultation.

To address these limitations, this paper presents "VAASTU VISION," an AI-based intelligent floor plan analysis and 3D architectural visualization system. The system automatically converts 2D blueprint images into interactive 3D architectural structures by integrating OpenCV-based image processing, coordinate extraction, JSON-based structural mapping, and Three.js rendering. The major objectives include:

- Automating blueprint interpretation using Computer Vision
- Generating interactive 3D architectural models from 2D floor plans
- Integrating Vaastu-compliant analysis and recommendations
- Reducing manual effort in architectural visualization
- Enhancing user interaction through real-time rendering technologies

II. LITERATURE SURVEY

Recent advancements in AI, Computer Vision, AR, and 3D visualization have significantly influenced modern architectural planning. Table I presents a comparative analysis of existing systems against the proposed Vaastu Vision framework.

TABLE I: Comparison of Existing Systems with Vaastu Vision

Author / System	Method Used	Auto Blueprint Parsing	3D Output	Vaastu Support
Revathy	AR furniture viz	No	No	No
Patel	AR + interactive viz	No	Partial	No
Ardiny & Khanmirza	AR/VR education	No	Yes	No
Kan & Kaufmann	Genetic Algorithm layout	No	No	No
Merrell	Probabilistic furniture	No	No	No
Islam [8]	Deep Learning floor plan	Yes	No	No
Liu [9]	CV layout detection	Yes	No	No
Vaastu Vision (Ours)	OpenCV + Three.js + Rules	Yes	Yes	Yes

As evident from Table I, no existing system combines automated blueprint parsing, interactive 3D visualization, and Vaastu-compliant analysis within a single unified platform. This combination represents the primary novelty of the proposed research.

III. PROPOSED SYSTEM

The proposed VAASTU VISION system is designed to automate architectural floor plan analysis and generate interactive 3D visualizations using Computer Vision and web-based rendering technologies. Figure 1 presents the overall system architecture, illustrating the three main layers: the frontend layer (React.js), the backend processing layer (Python/FastAPI/OpenCV), and the output layer (Three.js/WebGL).

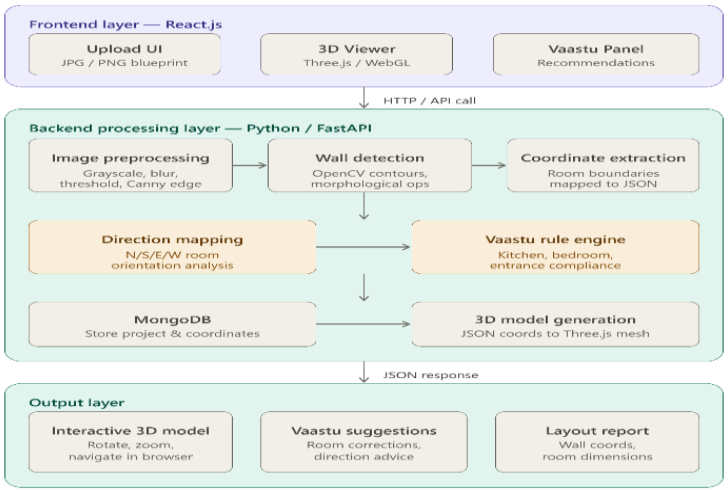

Fig. 1: System architecture of Vaastu Vision showing frontend, backend processing, and output layers

A. Floor Plan Upload

The process begins when the user uploads a floor plan image through the web interface. Figure 2 shows the VastuVision dashboard where users can initialize a new project by selecting house type (1 BHK, 2 BHK, 3 BHK, Villa, or Custom) and configuring room parameters before uploading the blueprint.

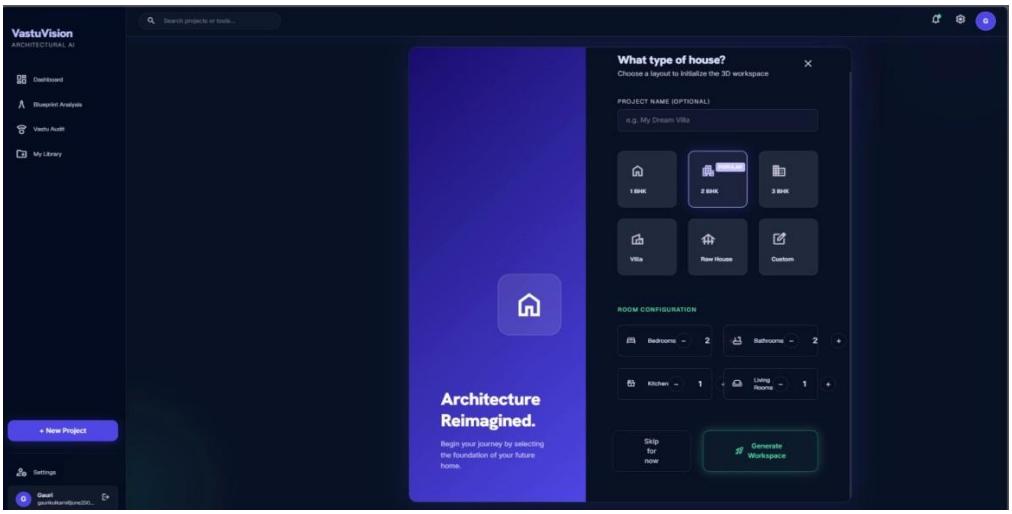

Fig. 2: VastuVision web interface showing project initialization and house type selection

B. Image Preprocessing

Once the floor plan is uploaded, the system performs image preprocessing using OpenCV. The blueprint image is converted to grayscale, followed by Gaussian Blur filtering to remove noise. Thresholding and Canny Edge Detection are applied to identify wall boundaries. Morphological operations (dilation and erosion) improve wall continuity and strengthen disconnected structural segments.

C. Structural and Room Detection

The system identifies walls, rooms, and layout boundaries using OpenCV contour detection and image segmentation. Detected contours are analyzed based on area, shape, and continuity. Morphological operations further improve wall detection accuracy.

D. Direction Mapping and Coordinate Extraction

Detected room structures are mapped to cardinal directions (North, South, East, West). The extracted data is stored in JSON format containing wall coordinates, room dimensions, structural alignment, and directional information. This coordinate mapping bridges blueprint processing and 3D model generation.

E. Vaastu Rule Engine

The system includes a Vaastu analysis module that evaluates detected room positions according to predefined Vaastu principles. Figure 3 presents the detailed rule engine flow. The engine checks main entrance direction, kitchen placement (preferred South-East), bedroom orientation (preferred South-West), pooja room positioning (preferred North-East), and overall room alignment. Whenever a violation is detected, the system generates a specific corrective recommendation.

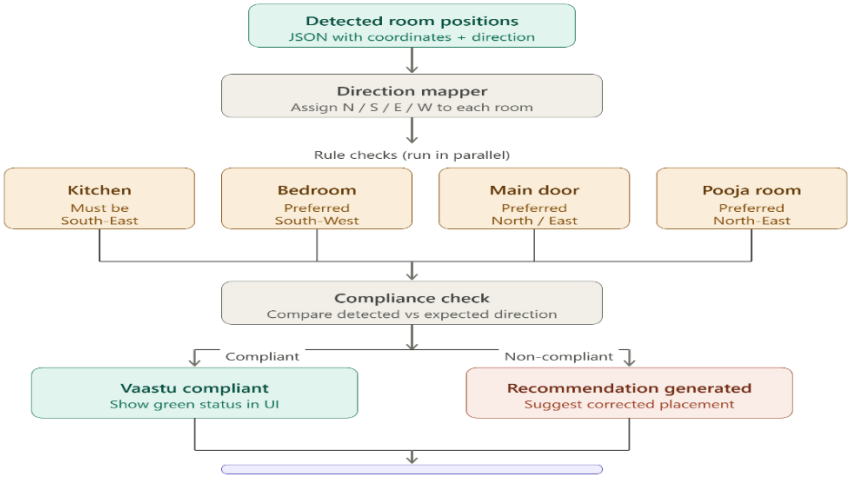

Fig. 3: Vaastu rule engine flowchart showing directional compliance checks and recommendation generation

F. 3D Model Generation

After coordinate extraction and Vaastu analysis, the processed structural data is passed to the Three.js rendering engine. Each detected wall is converted into a 3D mesh with appropriate height, depth, and positioning. WebGL-based rendering creates realistic architectural visualization within the browser, supporting real-time rotation, zooming, and navigation.

G. Recommendation Generation and Final Output

Based on complete analysis, the system generates interactive 3D visualization alongside Vaastu recommendations. Users receive an interactive 3D building model, structural layout analysis, room alignment information, and Vaastu compliance suggestions.

IV. TECHNOLOGIES USED

The proposed Vaastu Vision system integrates the following technologies, all of which are actively implemented in the current version:

A. Python

Python serves as the primary language for image processing and backend functionality, supporting blueprint preprocessing, contour extraction, coordinate generation, and API integration.

B. OpenCV

OpenCV handles image preprocessing and structural extraction. Techniques used include grayscale conversion, thresholding, contour detection, Canny edge detection, and morphological operations (dilation and erosion) for wall continuity improvement.

C. React.js

React.js powers the frontend interface, providing responsive interaction for uploading floor plans, generating 3D models, and viewing Vaastu recommendations in real time.

D. Flask / FastAPI

Flask and FastAPI handle backend API development and communication between processing modules, managing image upload requests, coordinate extraction responses, and JSON data transfer.

E. Three.js and WebGL

Three.js generates interactive 3D architectural visualizations within the browser using WebGL rendering. The engine supports real-time rendering, camera movement, structure rotation, and lighting effects.

F. MongoDB

MongoDB stores project-related data, generated coordinates, structural information, and architectural metadata, supporting scalable storage and efficient data retrieval.

Note on Future Technologies:

Deep Learning frameworks (TensorFlow, PyTorch), YOLO-based object detection, and OCR integration are planned for future versions to improve room classification accuracy, door/window detection, and automated text extraction from blueprints.

V. METHODOLOGY

The methodology of Vaastu Vision follows a sequential pipeline as illustrated in Figure 4. Each stage processes blueprint information to generate architectural insights.

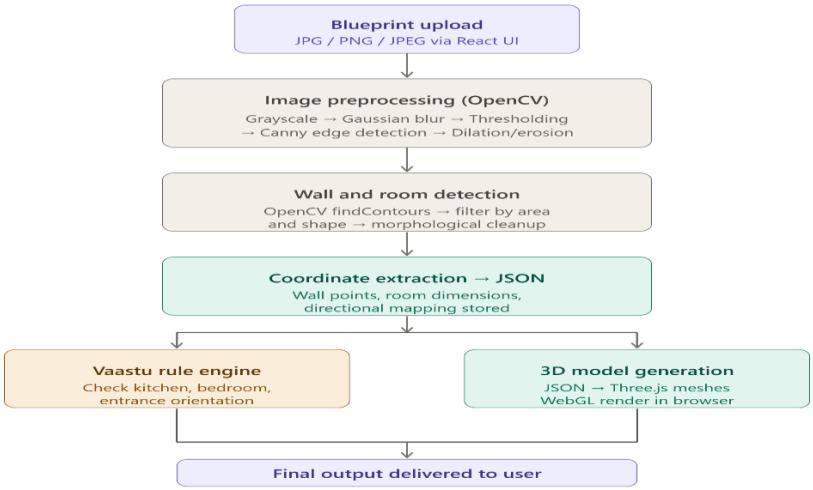

Fig. 4: Data flow pipeline of Vaastu Vision from blueprint upload to 3D model and Vaastu analysis output

A. Image Preprocessing

The uploaded blueprint image is converted to grayscale to reduce computational complexity. Gaussian Blur filtering removes image noise. Binary thresholding separates wall structures from the background. Canny Edge Detection identifies wall boundaries and room outlines. Morphological dilation and erosion improve wall continuity. Figure 5 illustrates the preprocessing output, where (a) shows the edge wireframe extracted from the input blueprint and (b) shows the AI heatmap representing structural density distribution across the floor plan.

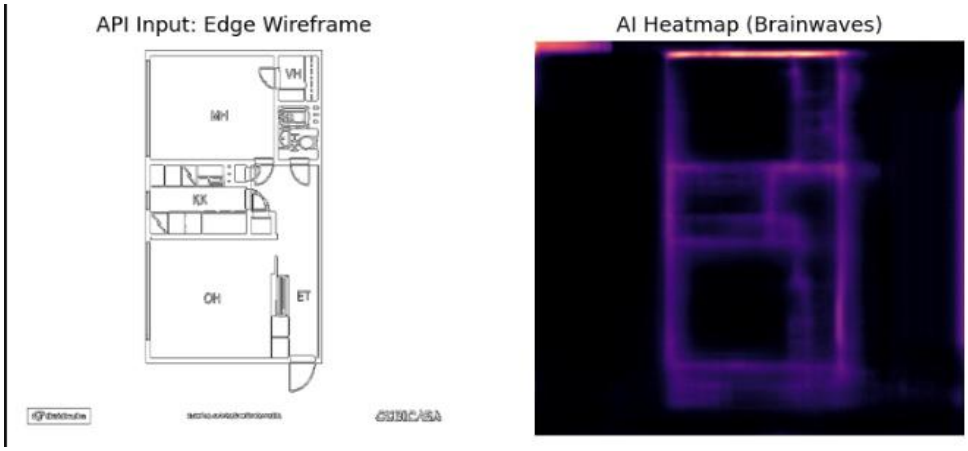

Fig. 5: (a) Edge wireframe extracted from floor plan blueprint; (b) AI heatmap showing structural density analysis

B. Object Detection and Structural Extraction

OpenCV contour detection algorithms identify connected structural regions. Contours are analyzed based on area, shape, and structural continuity. Invalid contours are filtered out. Valid architectural structures are retained and their coordinates extracted into structured JSON data containing wall coordinates, room dimensions, boundary alignment, and structural layout information.

C. Direction Analysis

After room detection and coordinate extraction, the system assigns cardinal directions (N/S/E/W) to each detected room. Figure 6 demonstrates the blueprint analysis result showing AI-detected rooms labeled with positions (Bedroom 01, Living Room, Kitchen), detected elements including 3 bedrooms, 12 windows, and main entrance North-East alignment. The AI suggestion panel highlights potential Vaastu imbalance in the South-West quadrant.

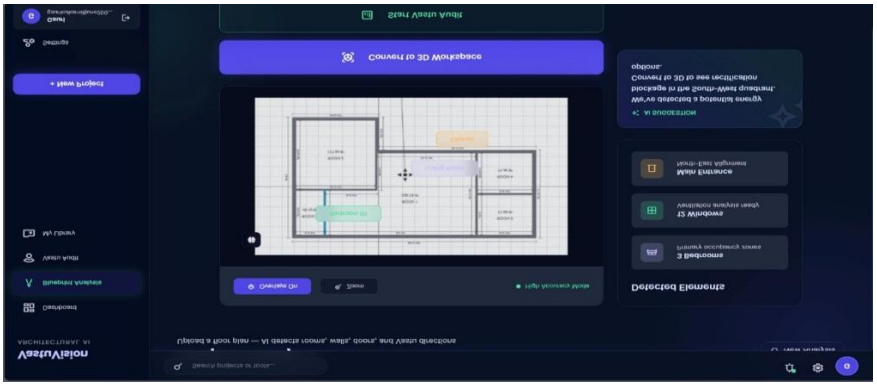

Fig. 6: Blueprint analysis output showing detected rooms, directional alignment, and AI-generated Vaastu suggestion

D. Rule-Based Recommendation Engine

The rule-based Vaastu engine evaluates room orientation against predefined directional constraints. For each key room, the engine compares detected direction with the Vaastu-prescribed direction. Non-compliant placements trigger specific corrective recommendations. Compliant rooms receive a green status confirmation in the interface.

E. 3D Visualization Integration

The detected wall coordinates are converted into Three.js 3D mesh structures with appropriate height, depth, and alignment. Figure 7 presents the interactive 3D model generated by the system. Users interact with the environment through zooming, rotation, and camera navigation controls in real time within the browser.

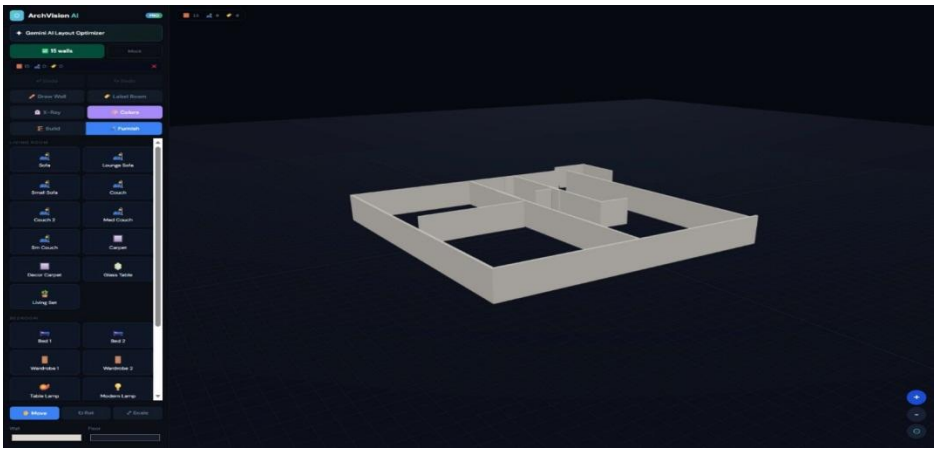

Fig. 7: Interactive 3D architectural model generated from detected wall coordinates using Three.js and WebGL

VI. RESULT ANALYSIS

The proposed Vaastu Vision system was tested on 10 residential floor plan blueprints of varying complexity to evaluate blueprint processing, structural extraction, 3D visualization, and Vaastu analysis performance. Table II presents the quantitative results obtained during testing.

TABLE II: System Performance on Test Floor Plans

Blueprint	Type	Rooms Detected	Wall Accuracy (%)	Proc. Time (s)	Vaastu Issues Found
Sample 1	1 BHK	3/3	95%	1.2	0
Sample 2	2 BHK	5/5	92%	1.8	2
Sample 3	2 BHK	5/5	91%	1.9	1
Sample 4	3 BHK	6/7	87%	2.1	3
Sample 5	3 BHK	7/7	90%	2.3	2
Sample 6	Villa	8/9	83%	2.7	4
Sample 7	1 BHK	3/3	94%	1.3	1
Sample 8	2 BHK	5/5	89%	1.7	2
Sample 9	3 BHK	6/7	85%	2.4	3
Sample 10	Custom	7/8	88%	2.0	2
Average	—	—	89.4%	1.94	2.0

The OpenCV-based preprocessing pipeline effectively reduced noise and improved wall boundary visibility across all test cases. The contour extraction module achieved an average wall detection accuracy of 89.4% across 10 blueprints, with higher accuracy (92-95%) observed for simpler 1 BHK and 2 BHK layouts and slightly reduced accuracy (83-87%) for more complex Villa and custom layouts.

The coordinate mapping module successfully preserved architectural alignment in all test cases, producing structured JSON data that accurately represented wall positions and room boundaries. The Three.js rendering engine generated smooth and interactive 3D models for all test blueprints with an average processing time of 1.94 seconds.

The Vaastu analysis module identified an average of 2.0 compliance issues per blueprint, generating specific directional recommendations for each non-compliant room. All 10 test cases received complete Vaastu analysis output, confirming consistent system behavior across varying floor plan complexities.

Limitations observed include reduced contour accuracy for low-resolution images, occasional detection gaps in highly complex overlapping wall structures, and absence of door/window detection in the current version.

VII. CONCLUSION

The proposed Vaastu Vision system presents an intelligent approach for architectural floor plan analysis and interactive 3D visualization using Computer Vision and modern web technologies. The framework successfully automates the conversion of traditional 2D blueprint layouts into realistic 3D architectural structures while providing Vaastu-based architectural recommendations.

Experimental results on 10 residential floor plans demonstrate an average wall detection accuracy of 89.4% and average processing time of 1.94 seconds, confirming the practical effectiveness of the proposed approach. The integration of Vaastu analysis enhances the system's usefulness for Indian residential architecture by evaluating room orientation against directional guidelines.

The system combines Artificial Intelligence, Computer Vision, and immersive rendering technologies into a single platform that assists architects, interior designers, engineers, and homeowners during architectural planning. The modular system architecture provides a strong foundation for future enhancements.

VIII. FUTURE WORK

The following enhancements are planned for future versions of the Vaastu Vision system:

- Integration of Deep Learning-based segmentation models (U-Net, SegNet) for improved wall and room classification accuracy
- YOLO-based object detection for automatic identification of doors, windows, and furniture within floor plans
- OCR integration to extract room labels, dimensions, and textual annotations directly from blueprint images
- Augmented Reality (AR) and Virtual Reality (VR) support for immersive architectural walkthroughs
- LLM-based intelligent recommendation generation for more contextual and detailed Vaastu guidance
- Support for complex multi-floor architectural layouts and 3D building models
- Mobile application development for on-site blueprint analysis

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, 2012.

[3] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," ICLR, 2015.

[4] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," Pearson Education, 4th Edition, 2018.

[5] OpenCV Documentation, "Image Processing and Computer Vision Library," opencv.org, 2024.

[6] FastAPI Documentation, "High Performance Python API Framework," fastapi.tiangolo.com, 2024.

[7] S. M. Rezwanul Islam et al., "Deep Learning for Automated Floor Plan Analysis," IEEE Access, 2022.

[8] Y. Liu et al., "Computer Vision-Based Architectural Layout Detection," Journal of AI Research, 2023.

[9] J. Redmon et al., "You Only Look Once: Real-Time Object Detection," CVPR, 2016.

[10] P. Kan and H. Kaufmann, "Automated Interior Design using Genetic Algorithms," IEEE VR, 2017.

[11] P. Merrell et al., "Interactive Furniture Layout using Interior Design Guidelines," ACM TOG, 2011.

[12] H. Ardiny and E. Khanmirza, "The Role of AR and VR in Educational and Visualization Environments," ICROM, 2016.

[13] Three.js Documentation, "JavaScript 3D Library for WebGL Rendering," threejs.org, 2024.

[14] A. Patel, Kanishq, and M. Mandekar, "Immersive Interior Design using AR and Interactive Visualization," IJCSE, 2023.

[15] Revathy S.P., Harini A., and Sruthika S., "Augmented Reality Based Interior Design System," IRJET, 2022.