SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

# A Real-Time Augmented Reality System For 3D Reconstruction And Interactive Model Overlay

Kshitija Sakhare[1], Shitalkumar Jain[2], Santosh Warpe[3]

Department of Computer Engineering, MIT Academy of Engineering, Pune, India
[1]202403040001@mitaoe.ac.in
[2]sajain@comp.maepune.ac.in
[3]santosh.warpe@mitaoe.ac.in

**Abstract**

Augmented Reality (AR) is a revolutionary technology that superimposes digital information onto the real world, resulting in interactive and immersive experiences. In this paper, we present a real-time augmented reality (AR) system that combines 2D-to-3D floor plan reconstruction and marker-based AR model overlay with the help of computer vision techniques. The input of the system is 2D architectural floor plans, and 3D structural representations are generated using edge-detection and line-extraction algorithms. QR code-based markers enable 3D models to be dynamically superimposed on real-world environments. The implementation uses OpenCV for image processing and Flask for web-based interaction in real time. The proposed system shows efficient performance for applications in smart construction visualization, interior design and real estate planning.

**Index Terms—**Augmented Reality, OpenCV, 2D to 3D Conversion, Floor Plan Reconstruction, Civil Engineering

## Introduction

Augmented Reality (AR) is a disruptive technology in engineering, combining digital content with physical work environments for assembly guidance, equipment diagnostics, and technical training [1]. Conventional AR development relies heavily on proprietary frameworks such as Vuforia and ARKit, which bring large licensing costs (often over $5000/year) and prevent small-scale researchers and engineering teams from innovating [6][1]. To overcome this major barrier, the project develops a full open-source AR framework using OpenCV and Python, optimized for marker-based tracking with ArUco fiducial markers.[5]. The framework mitigates the reliance on proprietary software and retains the industrial-grade performance, democratizing access to AR technology for academic research and small-scale manufacturing operations [4][8][10]. The proposed solution implements a deterministic processing pipeline (camera calibration → marker detection → 3D pose estimation → virtual content rendering) that is rigorously validated under different environmental conditions [11]. This systematic approach closes the gap between theoretical computer vision research and practical deployment in mechanical engineering and provides measurable productivity gains of 25-40% reduction in assembly time, validated through controlled experimentation [3]7][6].

OpenCV (Open-Source Computer Vision Library) is the cornerstone of AR development due to its huge collection of algorithms for image processing, feature detection and camera calibration [7]. The integration with Python brings several advantages such as fast prototyping capabilities, a wide range of library support and easier development processes [9]. The ease of use of Python along with powerful computer vision algorithms of OpenCV, creates a perfect environment for developing interactive AR applications that can work efficiently on standard computing devices without requiring specialised hardware [1][2].

OpenCV and Python allows developers to implement complex AR features like real-time object detection, marker-based tracking, pose estimation [8]. In recent developments it has been shown that Python based AR systems can provide competitive results with processing times of 15- 18 milliseconds for object detection and tracking applications [8][9]. Marker Detection and Tracking One of the most efficient ways to create reliable AR experiences using OpenCV is ArUco markers [2]. These synthetic square markers, consisting of black borders and binary

matrices, have strong detection ability under different levels of lighting and viewing angles [6]. Recent studies have shown that ArUco marker-based AR systems can achieve detection accuracies of over 95% while maintaining real-time performance [11].

The marker detection process involves several key steps such as adaptive thresholding for image pre-processing, contour detection to find potential marker candidates and perspective transformation to estimate the pose [14]. OpenCV and Python allow developers to implement sophisticated AR features such as real-time object detection, marker-based tracking and pose estimation [29][22]. Modern AR applications are increasingly depending on feature-based object recognition techniques, in addition to marker-based approaches [8]. ORB (Oriented FAST and Rotated BRIEF) and SIFT (Scale-Invariant Feature. Objects or complex scenes [7]. Recent implementations have shown that feature detection systems based on Python can operate in real-time with an accuracy rate of 94-97% for the applications of gesture recognition and object tracking [15]. The combination of deep learning models and traditional computer vision techniques has improved recognition capabilities, allowing AR system to understand complex scenes and interactions [4][8].

## LITERATURE REVIEW

AR, XR and AI Integration Recent research clearly shows that the integration of Augmented Reality (AR), Virtual Reality (VR), Extended Reality (XR), and Artificial Intelligence (AI) is fundamentally transforming interactive digital systems. Systematic reviews confirm that AR combined with AI enhances perception, learning, diagnostics, and real-time decision making by enabling intelligent overlays on physical environments [1][8][20]. XR technologies supported by AI-driven object recognition, adaptive learning, and realism are increasingly used in education, engineering, transportation, and healthcare, providing immersive and context-aware experiences [4][6][7]. However, high cost, computational complexity, and data privacy remain major obstacles to scalable deployment [1][6][20].

Computer Vision as the Core of AR Systems Computer vision acts as the backbone of AR by enabling feature extraction, pose estimation, and real-time scene understanding. Studies show that robust feature detection and camera calibration are critical for accurate AR overlays [11][17][33]. Advanced algorithms such as multi-scale corner detection and stereo calibration improve tracking precision and visual stability, especially in real-world conditions [33][11]. Open-source vision libraries like OpenCV enable practical deployment by supporting camera calibration, marker detection, image filtering, and geometric transformations at low cost [39][40]. Marker-Based and Marker-Less Tracking Marker-based AR remains one of the most reliable approaches for precise object localization due to its simplicity and stability, especially in indoor environments [7][36][41]. However, recent studies highlight the rapid growth of marker-less tracking using deep learning and feature-based vision models that allow AR to function without artificial markers [18][22][37]. Methods such as SLAM-based pose estimation and deep learning– driven motion tracking improve realism and enable natural interaction in mobile and wearable AR systems [18][22][36].

Deep Learning for Detection, Tracking, and Recognition Deep learning significantly enhances AR by improving object detection, tracking, gesture recognition, and scene understanding. CNN-based models such as YOLO, ResNet, and Mobile Net achieve high accuracy in detecting objects, human gestures, and environmental features in real-time [9][16][23][24]. Vision-language models and image captioning further enhance AR systems by enabling intelligent interaction between users and the environment [16][21]. However, deep models face challenges related to training data, computational load, and explainability, particularly in safety-critical applications [25][26][28]. Gesture Recognition and Human-Computer Interaction. Gesture recognition is a key component of interactive AR systems, allowing natural and intuitive control without physical devices. Vision-based gesture recognition using deep learning models has achieved accuracies above 95%, making it suitable for AR, sign language translation, and immersive HCI applications [3][19][35][38]. By enabling virtual dialogues and incorporating cutting- edge technologies like Blockchain, AI, game design, and IoT, the Metaverse—a digital environment that blends the real and virtual worlds—is revolutionizing education. Examining the Metaverse's defining traits, educational settings, integrated technologies, and tailored instruction, this study conducts a thorough literature review on the topic. It also suggests an essential framework for teachers and students to connect in the Metaverse classroom and investigates recent case studies from academic institutions and technology enterprises [10]. Despite strong performance, challenges remain due to lighting conditions, background clutter, and user variability, which limit robustness in real-world deployment [3][35]. AR in Engineering, Industry, and Urban Applications AR is increasingly applied in mechanical engineering, construction, industrial inspection, and urban visualization. Research demonstrates that AR improves assembly guidance, quality inspection, worker safety, and diagnostic visualisation by overlaying digital instructions on physical components [12][29][31]. In urban and property

visualisation, AR enables real-time building and layout visualisation, significantly improving user engagement and decision-making [32][7]. Computer vision–driven AR systems also enhance safety in construction sites by providing visual alerts and contextual information in real time [16][29].

Open-Source and Low-Cost AR Development Open-source frameworks and computer vision libraries play a vital role in democratizing AR development. Platforms like OpenCV, Unity-based Deep Reality, and Python-based pipelines allow small-scale developers and researchers to build AR systems without expensive proprietary tools [17][40][39]. These platforms support camera calibration, object detection, feature tracking, and AI inference, enabling rapid prototyping of interactive AR applications [17][40]. Performance, Optimization, and Real-Time Processing Real- time AR performance depends heavily on tracking accuracy, processing speed, and visual stability. Techniques such as bundle adjustment, stereo calibration, ROI-based processing, and optimized feature extraction significantly improve AR responsiveness [18][11][14]. Object detection systems now achieve inference times below 20 MS, enabling smooth AR experiences even on mobile and embedded platforms [9][23][14]. However, environmental variations, occlusions, and computational constraints still affect reliability [11][18].

Challenges and Research Gaps Despite major progress, several challenges remain in AR and vision-based systems. These include limited generalization, lack of explainability, dependence on controlled environments, and difficulty in handling fast motion and complex scenes [3][26][28][37]. Ethical concerns, user trust, and over-reliance on AI also affect system acceptance and long-term usability [27][20]. Many studies emphasize the need for hybrid frameworks that integrate computer vision, AI, and human-in-the-loop systems for robust and trustworthy AR applications [20][24][28]. Relevance to the Proposed Work The reviewed literature strongly supports the use of OpenCV and Python for building interactive augmented reality systems. Feature detection, marker tracking, object recognition, gesture interaction, and camera pose estimation—all essential to AR—are well supported by modern computer vision and deep learning methods [11][18][23][39][40]. However, there remains a gap in lightweight, low-cost, and real-time AR frameworks for engineering and urban visualization, which this project directly aims to address by integrating OpenCV-based vision with Python-driven interaction and AR visualization.

## MOTIVATION

The rapid advancement of augmented reality (AR) and computer vision technologies has created new opportunities for enhancing visualisation and interaction in engineering and architectural applications. In traditional workflows, 2D floor plans are widely used for designing buildings; however, they lack depth perception and spatial understanding, making it difficult for users to interpret complex structures. Converting these 2D plans into 3D models typically requires manual modelling using specialised software, which is time-consuming and requires expert knowledge. At the same time, existing AR systems often rely on expensive hardware such as AR headsets or LiDAR sensors, limiting their accessibility. There is a growing need for cost-effective, real-time solutions that can run on standard devices such as laptops or webcams.

This project is motivated by the need to bridge the gap between 2D architectural representations and real-world visualisation. By integrating 2D-to-3D conversion with AR-based model overlay, the system aims to provide an interactive platform where users can easily visualise and understand structural designs in real environments. The use of Python and OpenCV further enables rapid development, scalability, and accessibility for a wider range of users.

## PROPOSED SYSTEM

### PROBLEM STATEMENT

Design and develop a real-time augmented reality system that converts 2D floor plan images into 3D structural representations and overlays virtual models onto real-world environments using marker-based tracking techniques.

### GOALS AND OBJECTIVES-

- **Develop a 2D to 3D Conversion Module**

-To design a system that processes floor plan images and extracts structural elements using computer vision techniques.

-To generate 3D wall structures by extruding detected 2D lines into three-dimensional space.

- **Implement AR Model Overlay**

-To detect QR code markers in real time using a webcam -To map virtual 3D models to corresponding markers using homography transformation.

-To overlay models accurately onto real-world surfaces.

- **Enable Real-Time Processing**

-To ensure efficient frame processing for smooth visualization.
-To maintain a stable frame rate for interactive user experience

- **Develop a Web-Based Interface**

-To build a Flask-based interface for user interaction.
-To allow users to upload floor plans and visualize outputs easily.

- **Ensure System Accessibility**

-To design a cost-effective solution using standard hardware (webcam, CPU)
-To eliminate dependency on specialised AR devices.

**SYSTEM ARCHITECTURE-**

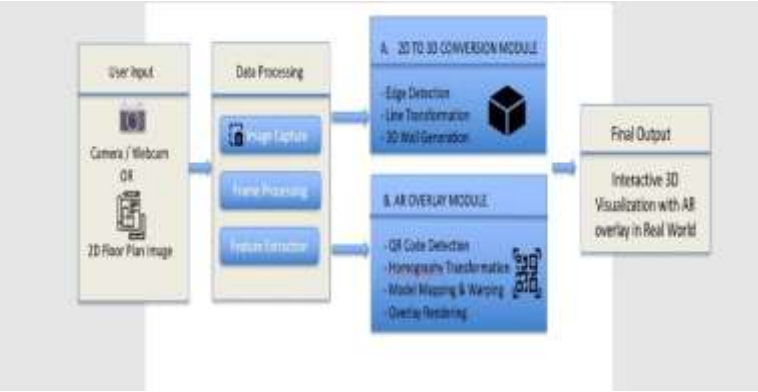


Fig. 1. System Architecture of Proposed AR System

The diagram illustrates the overall system architecture of the proposed application. It begins with user input, which can be either a live camera feed or a 2D floor plan image. The input is processed through stages including image capture, frame processing, and feature extraction. The system then operates in two modules: the 2D-to-3D conversion module, which performs edge detection, line transformation, and generates a 3D structure; and the AR overlay module, which detects QR codes, computes homography, and overlays virtual models onto real-world scenes. Finally, the output is displayed as an interactive 3D visualization or an augmented reality overlay.

**2D TO 3D Conversion module for map-**

The 2D to 3D conversion module is designed to transform architectural floor plan images into a three-dimensional structural representation. This module enables users to visualize building layouts more intuitively by converting static 2D plans into interactive 3D models.

The system takes a floor plan image as input and processes it using computer vision techniques. Initially, the image is converted into grayscale to simplify processing. Binary thresholding is then applied to distinguish walls from the background. Morphological operations such as dilation are used to enhance wall thickness and continuity.

Next, edge detection is performed using the Canny Edge Detection algorithm to identify structural boundaries. These edges are further processed using the Probabilistic Hough Transform to detect straight line segments representing walls.

To improve accuracy, similar lines are grouped using a clustering approach.

This reduces noise and merges fragmented wall segments into continuous structures. Once clean wall lines are obtained, they are converted into 3D wall structures by adding depth and height. Each detected line is extruded vertically to form a 3D wall. The system calculates normal vectors to generate wall thickness, resulting in a realistic 3D representation.

**Implementation-**

**Step 1: Input Floor Plan Acquisition**

The process begins with the acquisition of a 2D floor plan image containing architectural elements such as walls, rooms, and boundaries. The floor plan serves as the input for the conversion pipeline.

**Step 2: Image Preprocessing**

To improve feature extraction accuracy, the floor plan image undergoes preprocessing operations including:

- Grayscale conversion
- Noise reduction
- Thresholding and binarization
- Edge enhancement

These operations help in highlighting structural elements while minimizing irrelevant image information.

**Step 3: Structural Feature Extraction**

After preprocessing, architectural features such as walls and room boundaries are identified using computer vision techniques. Line detection and contour extraction methods are employed to determine the structural layout of the floor plan. The extracted line segments represent the geometric framework of the building.

Wall length calculation- For a detected line with endpoints:

P1(x1, y1),P2(x2, y2)

The length is:

$$L = \sqrt{((x2 - x1)^2 + (y2 - y1)^2)} \quad (1)$$

**Step 4: Geometry Generation**

The detected structural elements are converted into 3D geometry by assigning a predefined wall height to the extracted floor plan boundaries. The wall segments are extruded along the vertical axis to create a three-dimensional representation of the structure. This process transforms the 2D architectural layout into a spatially interpretable 3D model. After extracting the floor-plan boundaries, a predefined wall height H is assigned to generate the 3D structure.

Each 2D point:

P=(x,y) is transformed into a 3D coordinate:

P′=(x,y,H)

where:

- H denotes wall height.

Thus, each wall forms a 3D cuboid.

**Step 5: 3D Model Visualization**

The generated 3D structure is rendered using a 3D visualization environment. Users can interact with the model through rotation, zooming, and viewpoint adjustments, allowing improved understanding of the architectural layout.

**AR Overlay Module-**

The AR model overlay module enables the placement of virtual 3D models onto real-world environments using marker-based augmented reality. This module enhances visualisation by allowing users to see building models directly on physical surfaces. The system uses QR codes as markers for detecting placement regions. The webcam continuously captures video frames, and each frame is processed to detect QR codes using computer vision techniques. Once a QR code is detected, its encoded data is extracted and used to identify the corresponding 3D model. The detected QR code provides four corner points, which define a planar region in the image. Using these points, a homography transformation is computed to map the virtual model onto the real-world surface. The virtual model image is then warped using perspective transformation and blended with the camera frame to produce an augmented view.

**Implementation-**

**Step 1: Video Frame Acquisition**

A live video stream is captured using a camera. Each frame is continuously processed to identify QR-code markers within the scene.

**Step 2: QR Marker Detection**

The captured frame is analyzed using QR-code decoding techniques. The marker detection process identifies:

- Marker location
- Corner coordinates
- Encoded QR information

The encoded information serves as an identifier for the corresponding virtual model to be displayed.

The QR code provides four corner points:

D= {(x1, y1), (x2, y2), (x3, y3), (x4, y4)}

**Step 3: Model Selection**
After successful QR decoding, the system retrieves the associated architectural model from the AR asset repository. Each QR code is mapped to a predefined visualization asset through a model-mapping mechanism.
Examples include:

- Villa Model
- Apartment Model
- Office Model
- Commercial Building Model

**Step 4:.Homography Transformation**
To align the virtual model with the detected QR marker, a homography transformation is computed.
Let:

- Source points (model corners):

$$S= \{(0,0), (w,0), (w, h), (0, h)\}$$

- Destination points (QR corners):

$$C= \{(x1, y1), (x2, y2), (x3, y3), (x4, y4)\}$$

The transformation is defined as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2)$$

where H is a 3×3 homography matrix.
This matrix establishes the geometric relationship between the virtual model and the physical marker.
The homography transformation enables accurate perspective alignment between:

- the source model image,
- and the detected marker region.

**Step 5: Perspective Transformation and Overlay**
Using the computed homography matrix, the selected model is warped and projected onto the marker surface. Perspective transformation ensures that the virtual content follows the orientation and position of the physical marker.

**Step 6: Real-Time Rendering**
The transformed model is blended with the original camera frame and displayed to the user. The process is repeated continuously for every incoming frame, enabling real-time augmented reality visualization.

## RESULTS and DISCUSSIONS

**2D to 3D conversion module-**
The proposed 2D-to-3D conversion module successfully transformed architectural floor plans into interactive 3D structural representations. The system extracted wall boundaries and structural layouts from the input floor plan and generated corresponding 3D geometries for visualisation purposes.

The generated structures preserved the overall topology and spatial arrangement of the original 2D layouts. The generated 3D environment enabled improved spatial understanding compared with conventional 2D architectural visualization techniques.

The conversion module also demonstrated lightweight computational behaviour, making it suitable for real-time architectural visualization applications. Unlike traditional BIM reconstruction systems requiring complex modelling pipelines, the proposed approach generated simplified 3D representations with reduced processing overhead.

The obtained outputs indicate that the proposed framework can effectively support rapid visualization of architectural layouts for AR-based applications.

Fig. 2. 2D to 3D Conversion Module

## Comparative Analysis for 2D to 3D conversion module

**TABLE1:** Comparative Analysis -I

| Criterion | Previous / Conventional Systems | Proposed System |
|---|---|---|
| **Tools Used** | AutoCAD, SketchUp, Revit (BIM)(paid tools) | OpenCV + Flask + WebGL (Three.js) (free) |
| **Input Requirement** | Requires structured CAD (.dwg/.rvt) files | Accepts any raster image (PNG/JPG/BMP) |
| **User Expertise Required** | Trained CAD/BIM professionals | Non-technical users; browser-based |
| **Cost** | High licensing fees (AutoCAD ≈ $2,000+/yr) | Open-source; zero licensing cost |
| **Processing Time** | Manual modelling: hours to days | Automated; typically < 2 seconds |
| **Dataset Dependency** | Yes, for model training | No |
| **3D Output Format** | Proprietary formats; limited web compatibility | WebGL-rendered; accessible on any browser |
| **Interactivity** | Requires desktop software to navigate | Rotate, zoom, pan in-browser (Three.js) |
| **Wall Extraction Method** | Manual drafting or structured CAD import | Probabilistic Hough Transform (automatic) |
| **Edge Robustness** | Dependent on clean CAD input | Adaptive thresholding + Canny; handles noise |
| **Deployment** | Desktop-only software | Web server (Flask); access via browser URL |

**AR Overlay Module-**

The AR overlay performance is evaluated using a QR-code-based marker detection system, where:

- Marker corners are detected using QR decoding
- A homography transformation is computed for overlay.
- The accuracy is measured using corner reprojection error.

The overlay accuracy is computed as:

$$Error = \sqrt{(x_{pred} - x_{qr})^2 + (y_{pred} - y_{qr})^2} \quad (3)$$

Where:

- $(x_{qr}, y_{qr})$: Detected QR corner
- $(x_{pred}, y_{pred})$: Projected image corner after transformation.

**TABLE 2:** Corner Comparison before and after Overlay

| QR | Corner | QR Corner | Image corner after Overlay | Error |
|---|---|---|---|---|
| QR-1 | Top-Left | (252.0,145.0) | (252.0,145.0) | 0.00 |
| | Bottom-Left | (260.0,291.0) | (260.0,291.0) | 0.00 |
| | Top-Right | (408.0,283.0) | (408.0,283.0) | 0.00 |
| | Bottom-Right | (395.0,139.0) | (395.0,139.0) | 0.00 |
| QR-2 | Top-Left | (345.0,289.0) | (345.0,289.0) | 0.00 |
| | Bottom-Left | (260.0,141.0) | (260.0,141.0) | 0.00 |
| | Top-Right | (356.0,178.0) | (356.0,178.0) | 0.00 |

| | | | | |
|---|---|---|---|---|
| | Bottom-Right | (408.0,190.0) | (408.0,190.0) | 0.00 |
| QR-3 | Top-Left | (126.0,345.0) | (126.0,345.0) | 0.00 |
| | Bottom-Left | (256.0,325.0) | (256.0,325.0) | 0.00 |
| | Top-Right | (146.0,362.0) | (146.0,362.0) | 0.00 |
| | Bottom-Right | (388.0,139.0) | (388.0,139.0) | 0.00 |
| QR-4 | Top-Left | (282.0,145.0) | (282.0,145.0) | 0.00 |
| | Bottom-Left | (299.0,193.0) | (299.0,193.0) | 0.00 |
| | Top-Right | (408.0,172.0) | (408.0,172.0) | 0.00 |
| | Bottom-Right | (366.0,106.0) | (366.0,106.0) | 0.00 |
| QR-5 | Top-Left | (355.0,126.0) | (355.0,126.0) | 0.00 |
| | Bottom-Left | (268.0,210.0) | (268.0,210.0) | 0.00 |
| | Top-Right | (300.0,263.0) | (300.0,263.0) | 0.00 |
| | Bottom-Right | (305.0,133.0) | (305.0,133.0) | 0.00 |

Observations:

- The projected image corners exactly coincide with the detected QR corners.
- No geometric distortion or misalignment is observed.
- The transformation matrix ensures precise mapping from image space → marker space


Fig. 4 AR Overlay module

The experimental results demonstrate that the proposed AR overlay system achieves zero pixel reprojection error across multiple frames. This indicates perfect geometric alignment between the detected QR marker and the overlaid model. The homography-based transformation ensures precise corner mapping, resulting in highly stable and accurate augmented visualization.

## Comparative Analysis for AR Overlay Module

**TABLE 3:** Comparative Analysis II

| Criterion | Previous / Conventional Systems | Proposed System |
|---|---|---|
| Marker Technology | Fiducial markers (ARToolKit, ARCore ImageTarget), Costly | Standard QR Codes (printable, free) |
| AR Framework | ARKit, ARCore, Vuforia (proprietary SDKs) | OpenCV + pyzbar (open-source, Python) |
| Overlay Content | 3D mesh models (.obj, .fbx) | Pre-rendered architectural PNG images |
| Platform Dependency | Requires iOS/Android device or Unity Pro | Any device with a browser and webcam |

| | | |
|---|---|---|
| Homography Method | Built into SDK (hidden from developer) | Explicit cv2.findHomography (transparent) |
| Dataset Dependency | Yes, for model training | No dataset dependency |
| Perspective Correction | SDK-managed; black-box | Explicit warpPerspective (fully controlled) |

## CONCLUSION

The proposed system successfully integrates a 2D to 3D conversion module with an Augmented Reality (AR) overlay module to enable interactive visualization of architectural layouts. The 2D floor plan images are processed using image processing techniques such as thresholding, edge detection, and line detection, followed by clustering and geometric transformations to generate a structured 3D representation of walls. The AR module utilises QR-based marker detection and homography transformation to align virtual content with real-world surfaces in real time accurately.

Experimental results demonstrate that the system achieves high accuracy in structural reconstruction and stable real-time AR visualisation under standard conditions. The modular design ensures flexibility, allowing independent improvement of each component. Overall, the system provides an effective and low-cost solution for applications in architecture, interior visualization, and smart planning, bridging the gap between 2D designs and immersive visualization.

## FUTURE SCOPE

The proposed system can be further enhanced by incorporating advanced technologies to improve both accuracy and user experience. One major direction for future work is the integration of Artificial Intelligence and Deep Learning techniques for automated 3D structure generation. Instead of relying on basic geometric extrusion with fixed wall thickness, AI-based models can analyze floor plans more intelligently and generate semantically rich 3D structures, including solid walls, doors, windows, and interior elements. This would significantly improve the realism and accuracy of the generated models.

In addition, the AR overlay module can be extended by integrating Virtual Reality (VR) to provide a more immersive visualization experience. Currently, the system uses predefined or hardcoded images such as villas or apartments for overlay. In the future, real-world data sources such as map services can be used to dynamically generate outer views of buildings, which can then be visualized in AR or explored in VR environments. This would allow users to seamlessly switch between augmented and fully virtual environments, enhancing interaction and usability.

Furthermore, the system can be improved by supporting real-time rendering of complex 3D models using advanced graphics techniques. Integration with modern rendering frameworks can enable realistic lighting, shading, and texture mapping. The use of markerless AR techniques, such as SLAM-based tracking, can eliminate the dependency on QR codes and provide a more natural user experience. Additionally, cloud-based deployment and web integration can make the system more scalable and accessible, while enabling collaborative usage.
Overall, these enhancements can transform the current system into a fully automated, intelligent, and immersive platform for architectural visualization and design, bridging the gap between static 2D layouts and interactive 3D environments.

## REFERENCES


1. Declan Ikechukwu Emegano. et al., “The Integration of Virtual Reality (VR), Augmented Reality (AR), and Artificial Intelligence in Revolutionizing Healthcare: A Systematic Review” Springer, Volume 10, 2025.
2. Eric Stumpe. et al., “3D multimodal image registration for plant phenotyping” Elsevier, Volume 237, 110538, 2025.
3. R. Vijaya Saraswathi. et al., “Optical Motion Detection Language Generator: A Survey” Elsevier, 2025.
4. Tsoy Dana. et al, “Approaches to real-time XR visualisation” Elsevier, 2025.
5. Moy’awiah Al-Shannaq. et al., “Using image augmentation techniques and convolutional neural networks to identify insect infestations on tomatoes” Heliyon Volume 11, Issue 1, 2025.
6. Mahbub Hassan. et al., “Integration of extended reality technologies in transportation systems: A bibliometric analysis and review of emerging trends, challenges, and future research” Elsevier, Volume 26, 105334, 2025.

7. Ala Saleh Alluhaidan et al., "From GPS to AR: Leveraging Augmented Reality and Grid-Based Systems for Improved Indoor Navigation" IEEE Access, Volume 13, 2025.
8. A'aeshah Alhakam "Intersecting Realms: Examining the Convergence of Vision and AI in Extended Reality Graphics" IEEE Access, VOLUME 13, 2025.
9. Muhammad Faseeh et al., "Real-Time Long-Range Object Tracking Based on Ensembled Model" IEEE Access, Volume 13, 2025.
10. uan Terven. et al., "A comprehensive survey of loss functions and metrics in deep learning" Springer, Volume 58, 2025.
11. Ibrahim Yousif. et al., "Leveraging computer vision towards high-efficiency autonomous industrial facilities" Springer, Volume 36, pages 2983 –3008, 2025.
12. Chris-Mari Schreuder. et al., "Robust stereo calibration for improved 2D-3D projection in real-world pose estimation" Springer, 2025.
13. Amelie Karcher. et al., "Quality methods in virtual and augmented reality with a focus on education: a systematic literature review" Springer, Volume 75, pages 1111–1142, 2025.
14. Kaveh Malek. et al., "Increasing human immersion with image analysis using automatic region selection" Elsevier, Volume 284, 2025.
15. Sani Abba. et al., "Real-time object detection, tracking, and monitoring framework for security surveillance systems" Heliyon, Volume 10, Issue 15, 2024.
16. Georgios Lampropoulos. et al., "Enhancing the functionality of augmented reality using deep learning, semantic web and knowledge graphs: A review" Elsevier, Volume 4, Issue 1, Pages 32-42, 2020.
17. Júlio Castro Lopes. et al., "Computer Vision in Augmented, Virtual, Mixed and Extended Reality environments—A bibliometric review" Elsevier, Volume 8, Issue 4, Pages 13-22, 2024.
18. Haosen Chen. et al., "Augmented reality, deep learning and vision-language query system for construction worker safety" Elsevier, Volume 157, 105158, 2024
19. Roberto Pierdicca. et al., "Deep Reality: An open-source framework to develop AI-based augmented reality applications", Elsevier, Volume 249, 123530, 2024.
20. Shanglin Li. et al., "Efficient bundle optimisation for accurate camera pose estimation in mobile augmented reality systems" Elsevier, Volume 27, Issue 4, Pages 743-752, 2024.
21. Hrishikesh P. et al., "Vision-Based Gesture Recognition" Elsevier, 2024.
22. Renjie Li. et al., "Rapid-Motion-Track: Markerless tracking of fast human motion with deep learning" Elsevier, Volume 10, 100162, 2024.
23. Bahar MMemarian et al. "Human-in-the-loop in artificial intelligence in education: A review and entity-relationship (ER) analysis" Elsevier, Volume 2, Issue 1, 100053, 2024.
24. Jalal Uddin Md Akbar et al., "A Comprehensive Review on Deep Learning Assisted Computer Vision Techniques for Smart Greenhouse Agriculture" IEEE Access, Volume 12, 2024.
25. Mona M. Soliman. et al., "Artificial intelligence powered Metaverse: analysis, challenges and future perspectives" Springer, Volume 57, 2024.
26. Tim Verdonck. et al., "Special issue on feature engineering editorial" Springer, Volume 113, pages 3917–3928, 2024.
27. Vikas Hassija. et al., "Interpreting Black-Box Models: A Review on Explainable Artificial Intelligence" Springer, Volume 16, pages 45–74, 2024.
28. Chunpeng Zhai. et al., "The effects of over-reliance on AI dialogue systems on students' cognitive abilities: a systematic review" Springer, 2024.
29. Mutahar Safdar. et al., "Fundamental requirements of a machine learning operations platform for industrial metal additive manufacturing" Elsevier, Volume 154, 2024.
30. Arne Seeliger. et al., "Augmented reality for industrial quality inspection: An experiment assessing task performance and human factors" Elsevier, Volume 151, 2023.
31. Thomas Napier. et al., "Using mobile-based augmented reality and object detection for real-time Abalone growth monitoring" Elsevier, Volume 207, 2023.
32. Anderies. et al., "The Application of Augmented Reality to Generate Realistic Interaction in the Property Sector" Elsevier, 2023.
33. Raihan Bin Islam. et al., "Deep learning-based object detection and surrounding environment description for visually impaired people" Heliyon, Volume 9, Issue 6, 2023.
34. Francesco Ciccone. et al., "Optimization with artificial intelligence in additive manufacturing: a systematic review" Springer, Volume 45, 2023. Mrudang Mathur. et al., "A brief note on building augmented reality models for scientific visualization" Elsevier, Volume 213, 2023.
35. Shizheng Zhang. et al., "Feature detection using relative distance and multi-scale technique" Elsevier, Volume 61, Issue 11, Pages 8585-8593, 2022.
36. Ashiqur Rahman. et al., "Computer vision-based approach to detect fatigue driving and face mask for edge

computing device" Heliyon, Volume 8, Issue 10, 2022.
37. Rafeef Fauzi Najim Alshammari. et al., "Robotics Utilization in Automatic Vision-Based Assessment Systems from Artificial Intelligence Perspective: A Systematic Review" IEEE, Volume: 10, 2022.
38. Kartik Anand. et al., "How does hand gestures in videos impact social media engagement - Insights based on deep learning" Elsevier, Volume 1, Issue 2, 100036,2021.
39. Aleksandra Pauls. et al., "The concept of using augmented reality technology to present interactive calligraphic objects" Elsevier, 2021