AR-X Flow: A Mobile-Based Augmented Reality Workspace

Anup H. Muralidhar[1]; Adrian Richard Benjamin[2]; Aishwarya S Biradar[3]; B. Sumangala[4]

[1,2,3,4] Department of Computer Science and Engineering (BE CSE), Sir M Visvesvaraya Institute of Technology, Bangalore, India

Publication Date: 2025/12/05

Abstract: Multi-screen setups improve productivity but require expensive hardware and desk space that many users cannot afford. AR glasses offer virtual displays but cost over $3000. This paper presents AR-X Flow, a mobile AR system that creates virtual monitors using smartphones people already own. The system uses pre-trained YOLOv8 to detect laptops and monitors, positioning virtual screens contextually. Planned features include MediaPipe gesture recognition and WebRTC streaming. This work investigates whether mobile AR can democratize multi-screen workflows without specialized hardware. Our implementation demonstrates successful object detection and mobile-backend communication, though substantial development remains for rendering, gestures, and streaming.

Keywords*: Augmented Reality, Virtual Monitors, Gesture Recognition, Mobile AR, Workspace Productivity, Object Detection, Accessibility.*

How to Cite: Anup H. Muralidhar; Adrian Richard Benjamin; Aishwarya S Biradar; B. Sumangala (2025) AR-X Flow: A Mobile-Based Augmented Reality Workspace. *International Journal of Innovative* Science and Research Technology, 10(11), 2516-2521. https://doi.org/10.38124/ijisrt/25nov1378

I. INTRODUCTION

Remote and hybrid work increased demand for productive digital workspaces. Multi-monitor setups help when coding with documentation visible or taking notes on video calls. But monitors cost money, consume desk space, and need power outlets. Students in shared dorms, professionals in coworking spaces, or people working from temporary locations find extra monitors impractical.

AR glasses like HoloLens and Magic Leap promise virtual displays anywhere. The price tag exceeds $3000 with limited availability. Meanwhile, most people own smartphones with cameras, processors, and built-in AR support through ARCore or ARKit.

Other workspace solutions exist but have limitations. VR desktop apps like Immersed offer multiple screens in fully immersive environments but disconnect you from your physical surroundings. You cannot see your keyboard or desk. Wireless display solutions like Apple's Sidecar extend screens to secondary devices, but you still need that physical device. These approaches either cost too much, lack mixed reality context, or do not use hardware people already have.

This creates a question: what if a smartphone app could deliver core virtual monitor functionality using devices people already own? Could mobile AR make multi-screen productivity accessible to everyone?

A. Research Gap and Core Problem

Despite various virtual display technologies, a significant gap exists for accessible, cost-effective solutions that work with hardware people already own. Current mobile AR applications focus on gaming, information overlay, or novelty features. Few explore democratizing multi-screen productivity while addressing three interconnected barriers:

➢ *Cost Barrier*

AR glasses remain prohibitively expensive. Budget options like Nreal or Rokid cost hundreds of dollars. Physical monitors require upfront investment that excludes students on tight budgets and precarious professionals.

➢ *Space Barrier*

Physical monitors need dedicated desk space. Students in shared dorms have no room. Professionals in coworking spaces face shared workspace limitations. Remote workers in small apartments or traveling cannot accommodate multiple large displays. Space requirements fundamentally exclude significant populations.

➢ *Ecosystem Complexity*

Current solutions lock users into specific hardware ecosystems. HoloLens requires Windows. Sidecar requires Apple devices. Wireless display apps demand network configuration and device compatibility. Each solution works only within its ecosystem.

These barriers mean expensive AR systems remain luxury items rather than practical tools for students and budget-conscious professionals.

B. Our Contribution

AR-X Flow addresses this gap as a novel mobile-based AR system designed to work with devices people already own. The system prioritizes accessibility over perfection, investigating whether "good enough" mobile AR can serve users excluded by hardware costs. Our contributions include:

- System Design: A mobile-first AR workspace architecture integrating real-time object detection (pre-trained YOLOv8) and planned gesture control (MediaPipe) within standard smartphone AR frameworks.
- Technical Architecture: A practical architecture splitting responsibilities between mobile AR client (Unity + AR Foundation) and backend server (Python Flask) to address phone computational constraints.
- Challenge Identification: Analysis of anticipated technical obstacles in mobile-first AR workspaces: spatial anchoring stability, gesture interaction reliability, mobile performance constraints, and ergonomic limitations.
- Accessibility Focus: Direct emphasis on zero cost, device accessibility (any mid-range smartphone), and practical usability for real users rather than hypothetical scenarios.

C. Research Question

This paper investigates: Can an accessible, mobile-first AR system effectively replicate core multi-screen productivity functionalities, thereby democratizing virtual workspace technology for everyday users?

Our approach accepts "good enough" mobile AR as valid if it reaches users who cannot access premium hardware. We prioritize accessibility and practical utility over perfect technical performance.

II. RELATED WORK

➢ *AR Workspace Systems:*

Most AR productivity research assumes head-mounted display hardware. Microsoft's Mesh and Meta's Horizon Workrooms enable virtual collaboration but require expensive dedicated devices. Some projects explored tablets or smartphones for AR annotation in physical workspaces, though these add digital notes rather than creating actual virtual displays you can work with. Our mobile-first approach prioritizes virtual display creation on ubiquitous devices.

➢ *Virtual Display Technologies:*

Wireless display solutions like Sidecar and Spacedesk extend screens to secondary physical devices but maintain traditional monitor constraints of cost and space. VR desktop applications like Immersed render multiple screens in fully virtual environments but disconnect users from their physical surroundings. You cannot see your keyboard or notice when someone enters your space. AR-X Flow bridges this gap by creating virtual displays within mixed reality context, keeping you aware of your physical workspace.

➢ *Gesture Recognition:*

MediaPipe and similar hand-tracking libraries enable robust gesture recognition for AR applications. Prior work demonstrates gesture-based interaction works well for selecting and manipulating virtual objects in stable AR environments. However, stability proves critical. When virtual objects lack stable spatial anchoring, gesture precision degrades significantly. The user reaches for something, but the target has drifted slightly, and the system misses the interaction. This presents a practical challenge for mobile implementation where anchoring tends to be less stable than dedicated hardware.

➢ *Mobile AR Limitations:*

Published studies highlight challenges with mobile AR: tracking instability, performance constraints, battery drain, thermal throttling, and ergonomic difficulty of holding phones for extended periods. These barriers limit mobile AR productivity applications. Our work differs by acknowledging these constraints directly rather than treating them as obstacles to overcome. Instead, we investigate whether acceptable functionality within these constraints can serve underserved populations with no better alternatives.

Table 1 Comparative Analysis

Feature/System	**AR-X Flow**	**HoloLens**	**Sidecar**	**Immersed**
Hardware Cost	$0 (existing phone)	$3000+	$300+ (iPad)	$300+ (headset)
Mixed Reality	Yes	Yes	N/A	No
Accessibility	High	Low	Medium	Medium
Target Users	Students, budget professionals	Enterprise	Apple users	VR users

II. SYSTEM ARCHITECTURE

AR-X Flow consists of three primary components communicating over local WiFi: a mobile AR client on the smartphone, a backend server on the laptop, and network connections between them.

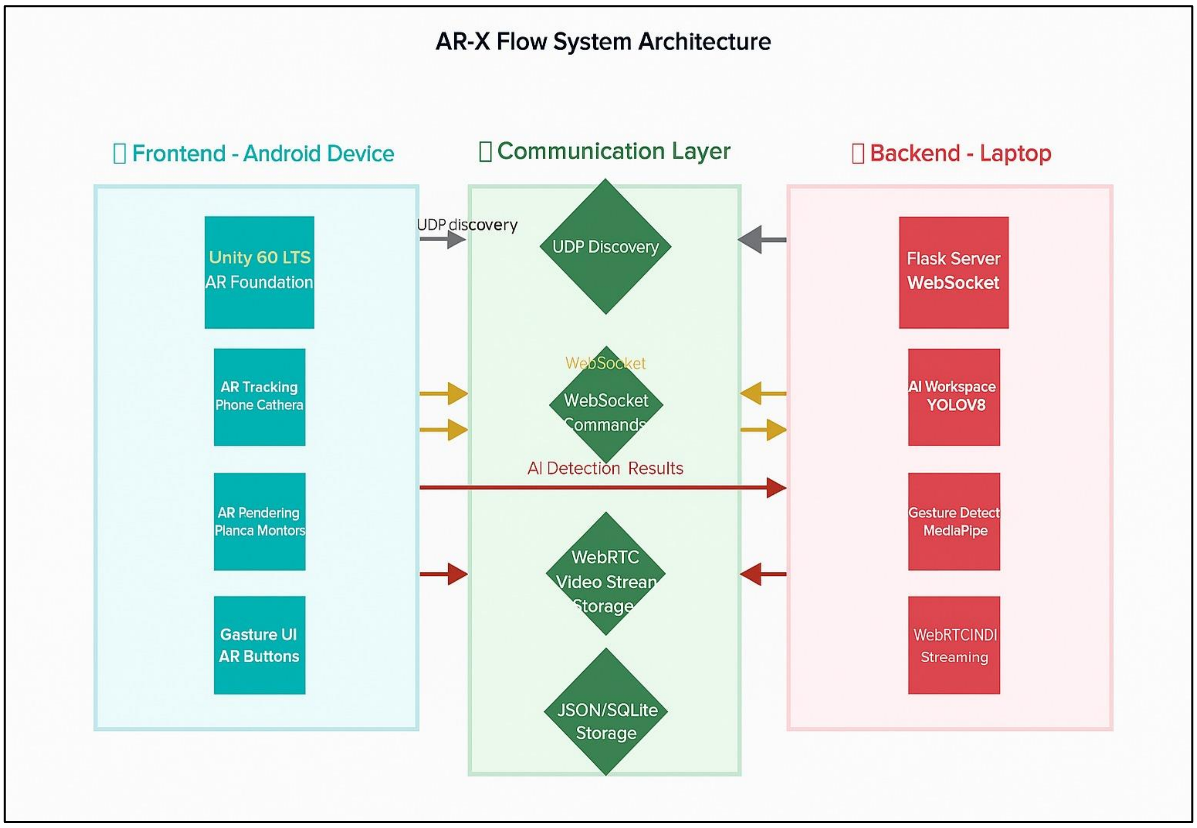

Fig 1: The System Architecture Showing Communication Between the Mobile AR Client on the Smartphone and the Backend Server on the Laptop. Camera Frames are Streamed from Phone to Backend, Where Pre-Trained YOLOv8 Runs Detection. Detection Results are Sent Back to the Phone. Planned WebRTC Streaming Sends desktop Content to Virtual Monitors.

➤ *Mobile AR Client (Unity 6.0 + AR Foundation):*

The smartphone runs the AR application built in Unity 6.0 using AR Foundation. We chose Unity because AR Foundation abstracts platform differences between ARCore and ARKit, letting us write code once and deploy to both platforms. Native development would have required maintaining entirely separate codebases, roughly doubling engineering effort.

The phone camera captures frames continuously and sends them to the backend server via WebSocket. A pre-trained YOLOv8 Nano model running on the backend identifies laptops, monitors, TVs, and screens. When detection occurs, the backend notifies the phone, which positions virtual monitors appropriately in the AR scene. MediaPipe Hands is designed to provide 21 hand tracking points for future gesture recognition.

➤ *Backend Server (Python Flask)*

A Python Flask server runs on the user's laptop, handling object detection. The backend receives camera frames from the phone, processes them with pre-trained YOLOv8 Nano to detect workspace objects, and sends results back to the phone. We chose to run detection on the backend because object detection is computationally expensive and would drain the phone battery quickly while generating heat. Offloading this to the laptop keeps the phone cooler and preserves battery.

A. Technology Selection Rationale

Unity and AR Foundation were chosen for cross-platform compatibility. The alternative (native Android and iOS development) would double engineering effort. We'd implement every feature twice, maintain two codebases, and test on twice as many devices.

Pre-trained YOLOv8 Nano was chosen for reliable object detection without training overhead. Training would require collecting thousands of images, manually annotating them, and iterating until achieving decent accuracy. YOLOv8 Nano from Ultralytics provides pretrained detection for common objects including laptops and monitors, lightweight enough for backend processing while maintaining reasonable accuracy.

WebRTC was chosen for planned content streaming due to widespread browser support, effective Unity plugin integration, and graceful handling of network fluctuations. NDI is a professional-grade alternative offering better performance but adds significant complexity we want to avoid in an MVP.

III. IMPLEMENTATION STATUS

A. *Completed Components:*

- Backend object detection successfully receives camera frames and runs pre-trained YOLOv8 Nano detection. When laptops or monitors are detected, the system logs detection with object type and bounding box coordinates.
- The Unity mobile app connects to the backend server over local WiFi using WebSocket protocol. Camera frames are captured and transmitted to the backend. The round-trip communication works.
- AR Foundation plane detection is integrated into the mobile client. The system can identify horizontal surfaces (tables, desks) and vertical surfaces (walls) where virtual monitors could be placed.

B. *In-Progress Components*

- Virtual monitor rendering exists but needs significant refinement. Basic AR object rendering works, though stable positioning and visual quality need improvement. Making text readable on semi-transparent AR panels while blending with the physical environment is challenging.
- MediaPipe Hands integration is partially complete. The system tracks hand landmarks but gesture pattern recognition (detecting pinches, measuring distances) requires additional development and testing.
- WebRTC integration is in early stages. The infrastructure for capturing desktop content exists, but establishing reliable peer connections and streaming needs completion.

C. *Planned Features*

- Multi-monitor support requires additional UI for management and logic to prevent overlapping virtual displays.
- Persistent spatial anchoring through cloud anchors, hybrid image target anchoring, or predictive drift compensation.
- Collaborative multi-user shared virtual workspaces once single-user functionality is stable.
- AI-powered enhancements like smart sticky notes with NLP features are deferred until core functionality proves viable.

IV. ANTICIPATED TECHNICAL CHALLENGES

➢ *Spatial Anchoring Instability*

AR Foundation's plane detection accumulates tracking errors over time. Virtual objects placed in space tend to drift, sometimes several centimeters over minutes. This breaks the illusion that virtual objects exist stably in physical space. Solutions like cloud anchors, hybrid image target anchoring, or predictive drift compensation will need investigation.

➢ *Gesture Detection Reliability*

Gesture recognition accuracy depends heavily on spatial anchor stability. If virtual monitors drift while hand tracking remains accurate, the system struggles to determine when your hand is "over" a virtual object. You reach for the button, and nothing happens. This is extremely frustrating.

➢ *Mobile Performance Constraints*

Running AR plane detection, receiving/transmitting camera frames, gesture recognition, and video decoding simultaneously will stress phone processors. Mid-range phones will likely experience thermal throttling after 15-20 minutes. The battery drains fast. Frame rates drop from 30 FPS to 10-15 FPS, making the AR experience choppy and unusable.

➢ *Health and Ergonomic Limitations*

Users must hold phones to view AR content. This is fine for brief interactions but extended use causes neck strain, arm fatigue, and eye strain. AR glasses solve this by being hands-free. Mobile AR inherits all the phone's ergonomic problems. This may be an inherent limitation of the mobile form factor.

➢ *Content Streaming Latency*

WebRTC streaming will introduce end-to-end latency from laptop screen capture to phone display rendering. Based on related work with WebRTC on mobile devices, we anticipate 150-300ms latency under good conditions. This is acceptable for static content like reading documentation but may feel sluggish for interactive applications.

V. DISCUSSION

A. *Current State*

AR-X Flow demonstrates feasibility of the core concept. Object detection successfully identifies workspace components. Basic mobile-backend architecture enables communication between phone and laptop over local WiFi. However, substantial development remains. Virtual monitor rendering needs completion. Gesture interaction requires finishing MediaPipe integration. Content streaming needs WebRTC peer connections working reliably.

B. *Comparison with Existing Solutions*

AR-X Flow trades the superior stability and ergonomics of premium solutions for accessibility. We target users explicitly excluded by hardware costs. Compared to HMD-based systems like HoloLens, we offer zero hardware cost but lower performance. Compared to wireless display solutions like Sidecar, we eliminate the need for secondary physical devices but may have higher latency. Compared to VR applications like Immersed, we maintain mixed reality context so you remain aware of your physical surroundings. Compared to generic mobile AR, we apply AR specifically to productivity rather than gaming or novelty features.

C. Broader Implications

If successful, accessible mobile AR workspaces could serve populations currently excluded from virtual workspace technologies entirely. Students and budget-conscious professionals cannot access $3000 AR glasses but already own capable smartphones. This work addresses a socioeconomic barrier in an era of rising educational costs and increased remote work demands. Whether "good enough" mobile AR can serve this underserved population remains to be validated through completing implementation and user testing.

Anticipated challenges suggest mobile AR may not suit all-day workspace use. More realistic applications might be temporary workspaces, supplementary displays for specific tasks, or short-duration work sessions. Understanding these boundaries will help set realistic expectations.

D. Limitations

The system is in the early MVP stage with limited functionality. Virtual monitor rendering, gesture interaction, and content streaming require substantial additional work before user testing becomes possible. We have proven communication architecture and object detection work; everything else remains to be built. No user testing has been conducted. No performance measurements have been taken. Using pre-trained YOLOv8 limits customization for specific workspace scenarios. Development has primarily used Android devices. iOS functionality through AR Foundation should work but has not been thoroughly validated.

VI. FUTURE WORK

A. Priority One

Complete Core Functionality: Virtual monitor rendering quality and stability need significant improvement. Gesture recognition patterns require full implementation and refinement. Content streaming via WebRTC must be completed and optimized.

➢ *Spatial Anchoring*

Cloud anchors can persist anchor points across sessions. Hybrid anchoring can dynamically switch between plane detection and image target tracking. Predictive drift compensation can track drift patterns and apply corrective transformations.

➢ *User Testing*

Usability testing with target users is essential. Performance measurement including latency, gesture success rates, thermal behavior, and battery drain during typical usage. Comparative evaluation against alternatives. Testing with demographic diversity beyond computer science students.

➢ *Multi-User Collaboration*

Once single-user functionality stabilizes, study groups could share virtual whiteboards and remote teams could collaborate on shared virtual screens.

➢ *Performance Optimization*

Adaptive quality scaling when battery or temperature constraints occur. GPU acceleration for video decoding. Selective feature disabling when not needed.

VII. CONCLUSION

AR-X Flow investigates whether smartphones can serve as platforms for accessible virtual workspace technology. The current implementation demonstrates basic feasibility: object detection identifies workspace components using pre-trained YOLOv8, and mobile-backend architecture enables communication over local WiFi.

However, significant development remains. Virtual monitor rendering, gesture interaction, and content streaming all require completion. Anticipated challenges may fundamentally constrain what mobile AR workspaces can achieve compared to dedicated hardware.

The central research question cannot yet be answered definitively. Our MVP demonstrates that core technical components can work together, but whether the resulting system provides genuine utility requires completion of implementation and user testing.

The work addresses a meaningful need. Students and budget-conscious professionals are increasingly expected to maintain productivity in digital environments but often lack resources for proper equipment. If accessible mobile AR can provide sufficient utility despite inherent limitations, it serves a population currently excluded from these technologies entirely.

Going forward, focus must remain on completing foundational functionality before expanding scope. If basic virtual monitor creation, stable positioning, intuitive gesture interaction, and practical content streaming can be achieved, user testing will reveal whether "good enough" mobile AR can serve people with no better alternatives.

The question is not whether mobile AR can become perfect. It won't. Phones will never match the ergonomics, stability, or performance of dedicated AR hardware. Rather, the question is whether an accessible, imperfect mobile AR solution can provide enough utility to users with no better alternatives. That question remains open and worth investigating.

ACKNOWLEDGMENT

Parts of the document organization, grammar review, and reference formatting were assisted by AI-based editing tools. All content and technical descriptions were authored and validated by the project team.

REFERENCES

[1]. Vasarainen, M., Paavola, S., & Vetoshkina, L. (2021). A Systematic Literature Review on Extended Reality: Virtual, Augmented and Mixed Reality in Collaborative Working Life Settings. International Journal of Virtual Reality, 2021.

[2]. Upadhyay, B., Brady, C., Madathil, K. C., Bertrand, J., & Gramopadhye, A. (2024). Collaborative Augmented Reality in Higher Education: A Systematic Review of Effectiveness, Outcomes, and Challenges. Applied Ergonomics, 2024.

[3]. Mehta, P., Narayanan, R., et al. (2024). Multi-User Mobile AR for Cardiovascular Surgical Planning. IEEE VIS, 2024.

[4]. Unity Technologies. (2023). AR Foundation Documentation. Unity 6.0 LTS.https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html

[5]. Google. (2024). MediaPipe Hands: On-device Real-time Hand Tracking. https://google.github.io/mediapipe/solutions/hands.html

[6]. Ultralytics. (2024). YOLOv8: State-of-the-art Object Detection. https://docs.ultralytics.com/

[7]. WebRTC Project. (2024). WebRTC: Real-Time Communication for the Web. https://webrtc.org