Robust Smartphone Screen Integration with Deep Learning for Virtual Reality Pass-through

Lucas Hartman[1], Ethan Pigou[2], Nicholas Strzelczyk[3], Santiago Gomez-Rosero[4], Miriam A. M. Capretz[5]
Department of Electrical and Computer Engineering
Western University, London, Ontario, Canada, N6A 5B9
Email: [1]lhartma8@uwo.ca, [2]epigou@uwo.ca, [3]nstrzelc@uwo.ca, [4]sgomezro@uwo.ca, [5]mcapretz@uwo.ca

***Abstract*—Virtual reality is revolutionizing immersive experiences, yet the seamless integration of everyday devices, such as smartphones, remains a challenging frontier. This paper presents a method for dynamically integrating a live smartphone screen into the virtual reality pass-through view. Our approach leverages a lightweight convolutional neural network (CNN) to detect the smartphone in real time, accurately determining its position, scale, and orientation within the camera feed. By synchronizing live screen capture with inertial sensor data, our system computes precise affine transformations that ensure the overlay remains perfectly aligned with its physical counterpart, even during rapid movements. Experimental evaluations demonstrate that our solution achieves an average of 29.2 detection runs per second, delivering a stable and high-fidelity integration without the need for additional hardware. Evaluations also showed a 0.44 increase in detection F1 score during live comparisons to the baseline alternative, which does not use deep learning. This dynamic overlay enhances visual clarity and interaction in virtual reality and bridges the gap between virtual and real-world interfaces, empowering users to access notifications, messages, and productivity applications seamlessly.**

***Index Terms*—Virtual Reality, Convolutional Neural Networks (CNN), Deep Learning, Pass-through**

I. Introduction

Virtual reality (VR) technology is transforming how users interact with digital content, delivering immersive experiences that blend virtual environments with real-world elements. An emerging component of these experiences is the pass-through mode, which allows users to view their physical surroundings while engaged in VR. Despite its potential, current pass-through implementations often suffer from low visual fidelity, making everyday tasks, such as reading a smartphone screen, difficult to perform without breaking immersion. As a result, users must remove their headsets for these interactions, disrupting the overall VR experience.

Virtual reality devices are important because they provide immersive, interactive environments that enable users to safely experience and manipulate realistic simulations of complex real-world scenarios [1]. Whether checking notifications, reading messages, or accessing productivity applications, the ability to interact with a smartphone without leaving the VR environment enhances both convenience and user engagement. While not particularly useful for VR gaming, in contexts such as remote work and virtual meetings, quick and accurate access to smartphone content while remaining immersed in VR can significantly improve the overall experience. Virtual reality is also finding growing applications in industrial training settings [2], and the constant need to remove headsets to check smartphones can cause significant productivity losses.

Existing attempts to solve this problem exist, both in research and commercial settings. However, these methods often rely on additional hardware or standard image processing techniques that are not robust across diverse environments. They also tend to struggle with misalignment, lag, and poor adaptability to rapid movements. Commercial solutions navigate this challenge by simplifying the interaction space, often displaying only notifications, which fails to capture the dynamic nature of smartphone content.

In contrast, our approach leverages computer vision to overcome these challenges. This paper uses a lightweight object detection model to accurately locate the smartphone within the camera feed and dynamically overlay a live screen capture onto the device. This deep learning method continuously adjusts the overlay in real-time for changes in position, scale, and orientation, ensuring that the smartphone remains properly aligned with its physical counterpart even during swift movements. By avoiding the need for extra hardware and overcoming the limitations of traditional image processing, the approach in this paper offers a robust and adaptable integration of smartphone functionality into the VR pass-through view.

The remainder of this paper is organized as follows. Section II discusses the key concepts for VR pass-through, you only look once, and image projection. Section III reviews related work and the limitations of current methods. Section IV details our method implementation, while Section V presents our experimental results. Finally, Section VI offers concluding remarks and directions for future research.

II. Background

This section introduces the main concepts involved in the implementation of the proposed method. Subsection A provides an overview of pass-through mode, Subsection B details the You Only Look Once (YOLO) model utilized for smartphone detection. Section C discusses the concepts related to image projections.

This research has been funded by the Alliance Missions project. Grant ALLRP 577133-2022.

A. Virtual Reality Pass-through

Pass-through mode is a technique that provides a hybrid between VR and real-world awareness [3]. This mode allows users to see the real world through cameras mounted on a VR headset while still immersed in a virtual environment. The real-world images are processed and displayed inside the VR headset, creating a seamless blend of physical and virtual worlds. This approach enhances user safety and comfort by allowing them to remain aware of their surroundings while enjoying a VR experience. Pass-through technology has various applications, including augmented reality (AR) experiences [4], mixed reality applications [5], and enhanced navigation and interaction in VR environments.

B. You Only Look Once

You Only Look Once (YOLO) [6] is a deep learning-based object detection algorithm that processes an image in a single pass to identify and locate multiple objects. Unlike traditional methods that use sliding windows or region proposals, YOLO treats object detection as a regression problem, predicting bounding boxes and class probabilities directly from the image. Because of this, YOLO is significantly faster than other approaches, such as Fast R-CNN [7], making it well-suited for real-time applications.

The latest YOLO version, YOLOv11 [8], enhances detection accuracy and efficiency, excelling in dynamic settings with rapidly moving objects. It improves object localization and tracking for stable, responsive detection while removing the need for predefined anchor boxes, increasing adaptability and robustness.

The lightweight variant model YOLO11n included in [8], retains YOLOv11's core enhancements while optimizing for resource-limited environments. By reducing parameters and computational load, it prioritizes speed and efficiency with minimal accuracy loss.

C. Image Projection

Image projection is a fundamental concept in computer graphics and image processing, which involves mapping a two-dimensional image onto a three-dimensional surface or another two-dimensional plane [9]. This technique is widely used in various applications, including VR, AR, and computer vision.

In the context of VR and AR, image projection plays a crucial role in creating immersive experiences. It allows virtual objects or images to be superimposed onto the real world or a virtual environment in a way that maintains correct perspective and scale [10]. This is achieved through a series of transformations that consider the position, orientation, and intrinsic parameters of the camera or display device. There are several types of image projection techniques, each with its own set of algorithms and mathematical models [11]. Perspective projections are the most popular as they simulate the way human eyes perceive the world, where objects appear smaller as they are farther away. These projections are used to create a sense of depth in a scene.

In VR pass-through systems, image projection can be used to overlay digital content captured by the headset's cameras onto the real world. An existing example would be projecting a virtual keyboard onto a physical desk. In the context of this paper, it is used to display a smartphone screen within the user's field of view while ensuring it aligns correctly with the physical phone. The accuracy of the projection directly impacts the user's experience, as any misalignment or distortion can break the sense of immersion. To achieve precise image projection, various techniques such as homography transformation, camera calibration, and depth estimation are often employed [11]. These techniques help in determining the correct position, orientation, and scaling of the virtual image to ensure a seamless integration between the virtual and real-world elements.

III. Related Work

Pass-through modes on current VR headsets, such as Meta's Quest 3 and Apple's Vision Pro, represent significant advancements in bridging the physical and virtual worlds [12]. These systems aim to enable users to engage with their physical environment while remaining immersed in VR. Meta's Quest 3 employs external cameras to project the real-world environment into the VR space but struggles with rendering high-detail visuals, particularly on electronic screens such as smartphones. This limitation hinders tasks requiring fine visual resolution, such as reading or interacting with apps. Apple's Vision Pro addresses this issue by overlaying a fixed representation of the smartphone screen within the user's field of view, improving clarity but reducing immersion due to its static and non-contextual implementation. These challenges highlight the need for more dynamic and immersive solutions to seamlessly integrate physical devices like smartphones into the VR experience.

Virtual desktops are a well-established approach to extending real-world functionalities into virtual spaces [13, 14]. These systems project a computer interface into VR, enabling users to interact with traditional desktop applications, browse the internet, and manage files within a virtual workspace. While effective for productivity, virtual desktops are inherently designed for stationary use and are not well-suited for devices like smartphones, which move around continuously while operated. The placement of virtual screens within the VR space also requires user intervention, which disrupts continuity and detracts from the seamless experience that VR aims to provide.

Commercial efforts to integrate smartphones into VR environments have largely focused on notification systems. Platforms such as the Oculus Companion App and HTC Viveport allow users to receive and respond to smartphone notifications, such as calls, messages, and app alerts, directly within the VR interface [15]. While these systems enhance connectivity, they provide only partial access to smartphone functionality. Interactions are typically limited to dismissing or acknowledging notifications, forcing users to physically handle their devices for more complex tasks. This results in a fragmented

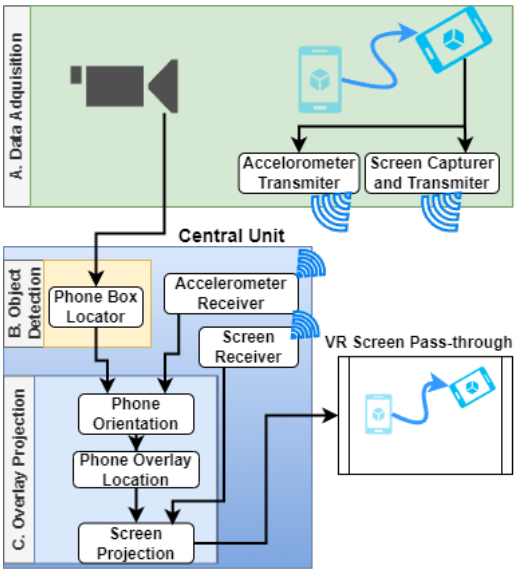

Fig. 1. Overview of the system architecture.

experience that breaks immersion and fails to fully utilize the capabilities of pass-through technology.

Several studies have explored methods for embedding smartphone interfaces into augmented reality environments, such as spatial anchoring [16] and fiducial markers [17]. While these techniques are effective, they have notable limitations. The work of Zhu *et al*. [16] lacks object detection, assuming continuous phone usage and restricting the system's adaptability to dynamic movements and varying environmental conditions. Kyian and Teather's [17] method depends on a constantly visible fiducial marker on the screen, which is a requirement that can disrupt natural interactions and is prone to issues with lighting and occlusions. Bai *et al*. [18] presented a robust approach that integrates smartphones into an AR setting, but it relies on external hardware to provide precise location data to the VR headset. In contrast, our approach leverages computer vision to dynamically detect the phone and its position, offering both a robust and lightweight alternative that overcomes these drawbacks.

IV. Methodology

The proposed approach overlays the smartphone screen onto the camera's live feed by connecting the smartphone to a centralized unit, detecting the smartphone within the camera view using a pre-trained object detection model, and then projecting the screen capture onto the camera image in real-time. Figure 1 shows how these various components interact and provides an overview of the system architecture. The main objectives are to detect the smartphone within the view, correctly orient the overlay, and achieve real-time performance.

Subsection A describes the data acquisition, subsection B explains the object detection process, and subsection C details the overlay projection and subsection D describes the evaluation metrics used.

A. Data Acquisition

To create the intended effect, data need to be acquired from various sources and combined in a centralized area. Specifically, three data streams are required: the smartphone screen capture, the smartphone's inertial sensor data, and the camera feed. A centralized unit gathers and combines these streams to achieve the desired overlay effect. The smartphone screen is captured in real time and stored as image frames, which are sent wirelessly to the centralized unit. Similarly, the accelerometer data is streamed via a lightweight socket connection to the centralized unit, which gives the inertial sensor data. The camera feed is also continuously processed for overlay projection. The centralized unit which gathers and processes the data may be the VR headset's onboard processing unit, an external processing server, or even the smartphone itself.

Both the screen overlay and the orientation of the smartphone are sent directly from the phone to the centralized unit, and then what the camera sees is used as input for the object detection detailed in the next subsection.

B. Object Detection

Detecting the smartphone within the camera feed is achieved using a pre-trained object detection model. To integrate into the approach, the model must be lightweight and perform accurate single-stage detection, which simultaneously predicts bounding boxes and associated class probabilities in a single pass. The model processes input images resized to a fixed resolution to ensure consistency and robustness across frames. Following detection, only the bounding box corresponding to the smartphone is retained, and its coordinates are smoothed over time to stabilize the overlay projection.

C. Overlay Projection

Once the data is collected onto the central unit, the orientation, background screen, smartphone screen, and bounding box are combined to create the intended projection effect. Three main steps are performed to project the smartphone screen capture onto the camera image: resizing, rotation, and placement.

The dimensions of the bounding box and the original smartphone screen capture are used to compute the new width (w') and height (h') of the overlay. The resizing is calculated using:

$$w' = \begin{cases} \text{int}\left(\frac{wB_w}{B_h}\right) & \text{if } \frac{w}{h} > \frac{B_w}{B_h}, \\ w & \text{otherwise} \end{cases}$$
$$h' = \begin{cases} h & \text{if } \frac{w}{h} > \frac{B_w}{B_h}, \\ \text{int}\left(\frac{hB_h}{B_w}\right) & \text{otherwise} \end{cases} \quad (1)$$

where w and h are the original dimensions, and B_w and B_h are the bounding box dimensions.

Using the smartphone's accelerometer data, the target rotation angle θ is calculated (via a standard `atan2` formulation) and used to derive a 2-dimensional affine rotation matrix. The computed matrix is then used to rotate the overlay using:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \end{bmatrix} \right) + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \quad (2)$$

where (x, y) are the original coordinates of a point in the overlay, and (x', y') are the coordinates after applying the rotation by θ about the center (c_x, c_y).

The new center (c_x, c_y) and final placement coordinates are determined using:

$$c_x = x_0 + \frac{w - w'}{2}, \quad c_y = y_0 + \frac{h - h'}{2}, \quad (3)$$

where (x_0, y_0) are the top-left coordinates of the bounding box.

Finally, the overlay is placed onto the camera feed using appropriate clipping to ensure it remains within the frame. Only the pixels within the detected bounding box are used in the final output to counteract boundary effects introduced by the affine transformation. The result is a dynamically resized, rotated, and precisely positioned overlay that integrates the smartphone screen seamlessly into the camera's live view.

D. Metrics

The model performance is evaluated using two metrics: frames per second (FPS), which quantifies the processing speed of the model on the central unit, and F1 score, which assesses the detection accuracy.

Frames per second (FPS) determines the speed at which a given model can compute phone detection and is calculated using Eq 4.

$$FPS = \frac{total\ frames}{total\ time} \quad (4)$$

$$F1\ score = \frac{TP}{TP + 0.5 * (FP + FN)} \quad (5)$$

The F1 score is used to assess the model's consistency in detecting the phone's screen. It is computed from a confusion matrix, where true positives (TP) represent frames in which the model correctly detects exactly one phone, false positives (FP) correspond to frames where more than one phone is detected, and false negatives (FN) indicate frames where no phone is detected. False positives are considered zero since the phone remains in-frame throughout all tests. F1 score is calculated using Eq. 5.

V. Results

This section describes the experiments and results of the proposed approach. The experimental setup is first outlined, including hardware specifications, software tools, and key parameters. Next, results from two experiments are presented, evaluating the system's performance on both pre-recorded and live footage. Finally, the results are analyzed, and their implications are discussed. For reproducibility purposes, the implementation details can be found on our repository[1].

[1]GitHub: https://github.com/LucasHartmanWestern/Screen-Sight/

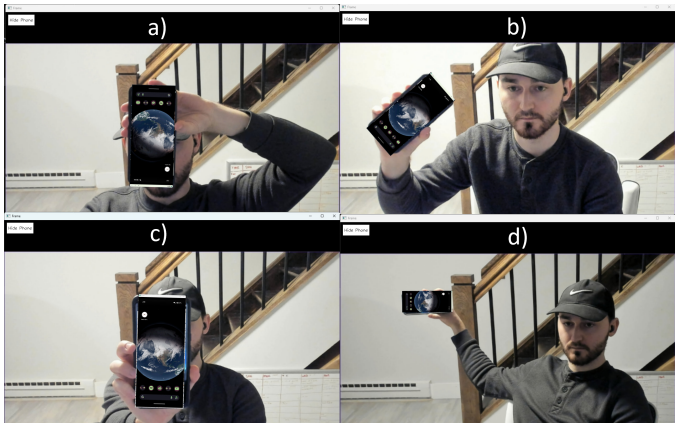

Fig. 2. A sample frame showing the smartphone detected by YOLO with its screen capture overlaid while the phone is (a) upside down & near, (b) diagonal & near, (c) upright & near, and (d) horizontal & far.

Figure 2 shows how the proposed approach detects the smartphone and how its screen capture is overlaid onto the live camera feed with the correct orientation and scaling. The alignment is maintained precisely even during rapid smartphone movements, ensuring a seamless pass-through experience within the VR environment.

A. Experimental Setup

This section outlines the conditions used to obtain the results presented in this paper. It details the hardware, software, package, and test environment configurations used to collect the analyzed data.

The proposed approach is compared against Kyian and Teather's [17] fiducial marker-based method using 6x6 ArUco markers generated and detected with the OpenCV library in Python. This was chosen as the baseline comparison due to its identical use case and lightweight implementation. ArUco markers are binary-coded patterns surrounded by a black border, allowing for robust identification and pose estimation. In our comparison, these markers are displayed on a smartphone screen at varying opacities (25%, 50%, 75%, and 100%) to evaluate the trade-off between detection accuracy and visibility of the fiducial marker.

1) Hardware & Software: The evaluation was conducted using three different centralized units, as detailed in Table I. The smartphone used to collect all results was a Pixel 8 Pro running Android 15.

TABLE I
Central Units Used During Experiment Runs

Central Unit	Proccesor	GPU	RAM	OS
1	AMD Ryzen 9 5900X	RTX 3080	64 GB	Windows 11
2	Apple M1 Max	-	32 GB	MacOs Sonoma 14
3	Intel i7 9th Gen	GTX1650	16 GB	Windows 11

The object detection pipeline was implemented in Python using PyTorch (v2.5.1), OpenCV-Python (v4.11), Ultralytics (v8.6), NumPy (v1.26), PyGetWindow (v0.0.9), Pillow (v11.1), WebSocket-Client (v1.8), and TorchVision (v0.20), openpyxl (v3.1). For capturing the smartphone's screen scrcpy [19] was used, which is a tool that mirrors the Android smartphone's screen. For sensor data, SensorServer [20] was used, which is a native Android application that streams inertial sensor data.

2) Package Configuration: The YOLOv11n [8] model was selected for these results due to its low parameter count and real-time performance. Table II shows the parameters used for the object detection module.

3) Background Environment: The experimental data for the performance comparison was collected across several environments and then aggregated based on whether or not the background was noisy and the lighting conditions. Clean backgrounds were ones where the smartphone was being held in front of a plain wall, while noisy backgrounds had both stationary and moving objects aside from the phone itself. The objects in the background were carefully chosen such that they were represented in the training dataset of the pre-trained YOLO model. The lighting conditions of the environments were either bright or dark, with dark environments relying solely on the smartphone as the light source and bright environments having external light sources. Each combination of environment and model was evaluated for 30 seconds with three runs. During evaluation, the smartphone would display the fiducial marker and be moved, rotated, and scaled relative to the camera.

B. Experiment 1: Performance Comparison

The first experiment used pre-recorded videos to ensure a more accurate performance comparison by eliminating potential variations present in live testing. This experiment was conducted under the same environmental conditions as the live evaluation, but instead of processing the data in real-time, the videos were recorded beforehand and later analyzed using the ArUco and YOLO-based approaches across all three central units. In this setup, videos featured a smartphone displaying fiducial markers at different opacities. FPS was used as the only performance metric in this experiment because model evaluation of pre-recorded videos results in the same F1 score regardless of hardware.

TABLE II
YOLO PARAMETERS CONFIGURATION

Parameter	Value
Input Resolution	1280 × 960 pixels
Confidence Threshold	25%
Non-Max Suppression Threshold	45%

TABLE III
PERFORMANCE COMPARISON BETWEEN MODELS USING DIFFERENT CENTRAL UNITS.

Model	Central Unit	FPS
ArUco	1	63.36 ± 6.02
	2	**82.34** ± 9.78
	3	35.77 ± 4.89
YOLO	1	**58.26** ± 3.78
	2	33.44 ± 1.22
	3	41.85 ± 2.01

Results are reported as mean ± standard deviation over 5 runs

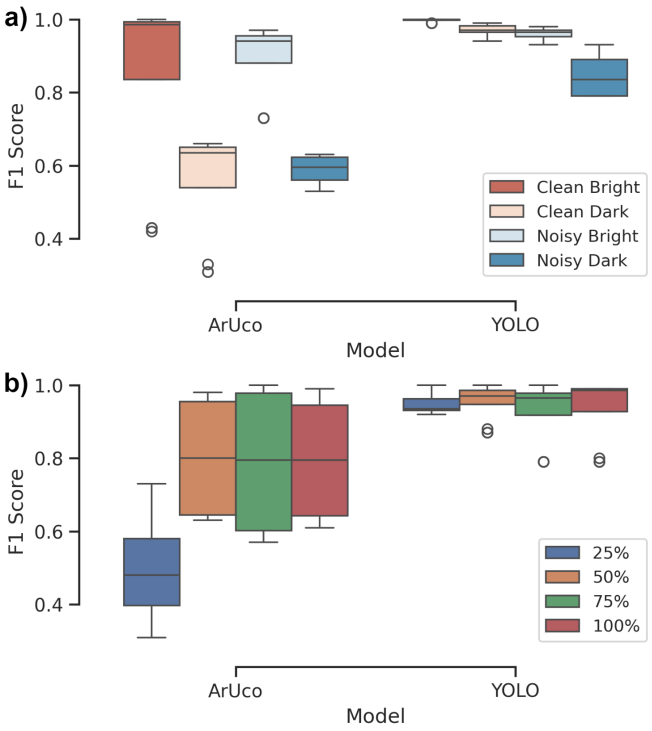

Fig. 3. F1 score comparison of YOLO and ArUco-based approaches on pre-recorded video for: a) the various background environments, b) for the various marker opacities.

Figure 3 shows the results of the tests on pre-recorded videos with variation over two scenarios: a) varying background and lighting conditions and b) opacity on the markers. These tests demonstrated that when given the same input, the YOLO-based approach had consistently improved the F1 score, showing it is better suited for more diverse use-case environments. Table III shows the number of frames per second (FPS) that were processed when using both approaches across the different central unit hardware configurations.

C. Experiment 2: Live Results

The second experiment evaluated the system in a live environment, processing video in real-time. This test assessed the system's ability to operate under dynamic conditions and provided insights into its performance in practical applications.

Table IV compares the computational efficiency and accuracy of the two examined models. The mean of FPS measures the computational efficiency that the system ran on while the

TABLE IV
LIVE COMPARISON BETWEEN THE YOLO AND ARUCO-BASED APPROACH WITH A CLEAN AND NOISY BACKGROUND.

Model Type	Clean Background		Noisy Background	
	FPS	F1 Score	FPS	F1 Score
YOLO Model	31.22 ± 5.83	**0.82 ± 0.16**	28.26 ± 11.47	**0.67 ± 0.24**
ArUco 25%	53.68 ± 16.24	0.25 ± 0.22	**51.07 ± 2.93**	0.23 ± 0.20
ArUco 50%	**59.18 ± 1.06**	0.42 ± 0.18	50.26 ± 4.68	0.41 ± 0.10
ArUco 75%	58.82 ± 1.65	0.45 ± 0.14	50.00 ± 4.93	0.42 ± 0.16
ArUco 100%	57.96 ± 2.40	0.41 ± 0.19	49.63 ± 3.66	0.45 ± 0.11

Results are reported as mean ± standard deviation over 5 runs

TABLE V
LIVE COMPARISON BETWEEN THE YOLO AND ARUCO-BASED APPROACH WITH DARK AND BRIGHT LIGHTING CONDITIONS.

Model Type	Bright Lighting		Dark Lighting	
	FPS	F1 Score	FPS	F1 Score
YOLO Model	26.39 ± 4.73	**0.80 ± 0.16**	33.09 ± 11.09	**0.69 ± 0.25**
ArUco 25%	**53.87 ± 4.95**	0.15 ± 0.13	50.89 ± 15.71	0.32 ± 0.23
ArUco 50%	52.72 ± 6.96	0.47 ± 0.10	56.73 ± 3.15	0.36 ± 0.16
ArUco 75%	51.99 ± 6.72	0.44 ± 0.14	**56.82 ± 3.48**	0.43 ± 0.17
ArUco 100%	52.11 ± 5.75	0.43 ± 0.16	55.48 ± 4.29	0.42 ± 0.15

Results are reported as mean ± standard deviation over 5 runs

approach was operational. The F1 score is used to represent how consistently accurate the model was in detecting the phone. While the average FPS was lower using the YOLO model, the F1 score was significantly better in environments with both a clean and noisy background. Table V compares the F1 score and computational efficiency in a light and dark environment. The YOLO model again shows a clear increase in F1 score across both environment settings. Across all environments, the average FPS using YOLO was 29.20 FPS with an average F1 score of 0.87.

Figure 4 shows the results when the proposed approach is inactive and active. The smartphone primarily displays text in this example, showcasing the increased spatial resolution achieved by the projected overlay.

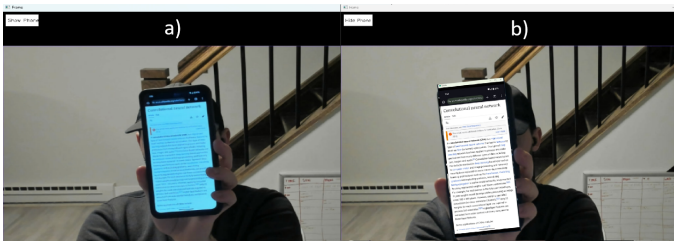

Fig. 4. A sample frame showing a comparison between when the tool is (a) inactive and (b) active.

D. Discussion

Background noise had an impact on both models, with the approach using the YOLO model having significantly higher F1 score but a slower performance. Many of the dropped frames using the fiducial marker-based approach occurred while the smartphone was moving, showing how motion blur can negatively impact the sharpness of the markers, which is required to detect them accurately. The lighting conditions also impacted performance. The impact was more significant when using the fiducial marker-based approach, as glare and over-exposure can decrease the spatial resolution of the screen. The approach using the YOLO model was less affected by the lighting conditions, but performance was slightly impacted in darker settings, with the outline of the phone being less noticeable.

The YOLO model achieved an average of 29.20 FPS during live processing across the various environments, reliably detecting the smartphone with an average F1 score of 0.87. While not as computationally efficient, this is a 0.44 increase compared to the average 0.43 F1 score when using a marker-based approach. Modern VR headsets such as the Apple Vision Pro and Meta Quest 3 typically run games in 30-60 FPS. If the location, scale, and orientation of the projection are calculated once every three frames, it will reliably surpass the 60 FPS target. If the user prefers less interpolation at the sacrifice of frame rate, they can run the detection once every frame while still achieving near the commonly used 30 FPS target.

VI. Conclusion and Future Work

This paper has presented a method to bring smartphone functionality into the VR pass-through experience by directly overlaying a live screen capture onto the camera feed. The approach uses a lightweight object detection model to locate the smartphone and continuously adjust the overlay in real-time. The system maintains accurate alignment and smooth performance even when the smartphone is moving quickly in a variety of environments, ensuring that users can interact with their devices without leaving the immersive VR experience.

The experimental results confirm that this solution meets the real-time performance requirements typical of modern VR devices while providing a stable and clear display of smartphone content. This integration makes it easier for users to stay connected and perform essential tasks, such as reading messages or checking notifications, without breaking their VR immersion.

Future work might extend support to full 3-dimensional orientation and incorporate hand masks to ensure the overlay does not obstruct the user's hands during smartphone interactions. Future work may also include developing a native application using this approach that can run on a Meta Quest 3 headset. Overall, this work takes an important step toward merging everyday technology with VR, offering users a more connected and fluid experience.

VII. Acknowledgments

This research/project/work is supported by the Vector Scholarship in Artificial Intelligence, provided through the Vector Institute. Additionally, this work is supported by the Canadian Graduate Scholarship Masters, awarded by the Government of Canada. The authors would like to thank Dhruv Sagre for their contributions during the initial stages of this research project. S.G.-R. thanks the SENESCYT from Ecuador for the support received during his doctoral studies in Canada.

References

1. Wang, P. & Bailenson, J. N. Virtual Reality as a Research Tool. *SSRN Electronic Journal.* ISSN: 1556-5068 (2024).
2. Radhakrishnan, U., Koumaditis, K. & Chinello, F. A systematic review of immersive virtual reality for industrial skills training. *Behaviour & Information Technology* **40,** 1310–1339. ISSN: 0144-929X (Sept. 2021).
3. Larroque, S. Digital Pass-Through Head-Mounted Displays for Mixed Reality. *Information Display* **37,** 17–21. ISSN: 0362-0972 (July 2021).
4. Carmigniani, J. & Furht, B. in *Handbook of Augmented Reality* 1, 3–46 (Springer New York, New York, NY, Jan. 2011).
5. Speicher, M., Hall, B. D. & Nebeling, M. What is Mixed Reality? in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* **15** (ACM, New York, NY, USA, May 2019), 1–15. ISBN: 9781450359702.
6. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *cv-foundation.org* (June 2015).
7. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39,** 1137–1149. ISSN: 0162-8828 (June 2017).
8. Jocher, G. & Qiu, J. *Ultralytics YOLO11* 2024. https://github.com/ultralytics/ultralytics.
9. Foley, J. D., Van Dam, A., Feiner, S. K., Hughes, J. F. & Phillips, R. L. *Introduction to computer graphics* (Addison-Wesley Reading, 1994).
10. Mahvash, M. & Besharati Tabrizi, L. A novel augmented reality system of image projection for image-guided neurosurgery. *Acta Neurochirurgica* **155,** 943–947. ISSN: 0001-6268 (May 2013).
11. Salomon, D. *Transformations and Projections in Computer Graphics* ISBN: 978-1-84628-392-5. http://link.springer.com/10.1007/978-1-84628-620-9 (Springer London, London, 2006).
12. Bailenson, J. N. *et al.* Seeing the world through digital prisms: Psychological implications of passthrough video usage in mixed reality. *Technology, Mind, and Behavior* **5,** 1–16. ISSN: 2689-0208 (June 2024).
13. Liu, K., Zhang, W., Li, W., Wang, T. & Zheng, Y. Effectiveness of virtual reality in nursing education: a systematic review and meta-analysis. *BMC Medical Education* **23,** 710. ISSN: 1472-6920 (Sept. 2023).
14. Oyelere, S. S. *et al.* Exploring the trends of educational virtual reality games: a systematic review of empirical studies. *Smart Learning Environments* **7,** 31. ISSN: 2196-7091 (Dec. 2020).
15. Xie, B. *et al.* A Review on Virtual Reality Skill Training Applications. *Frontiers in Virtual Reality* **2.** ISSN: 2673-4192 (Apr. 2021).
16. Zhu, F., Sousa, M., Sidenmark, L. & Grossman, T. PhoneInVR: An Evaluation of Spatial Anchoring and Interaction Techniques for Smartphone Usage in Virtual Reality. in *Proceedings of the CHI Conference on Human Factors in Computing Systems* **16** (ACM, New York, NY, USA, May 2024), 1–16.
17. Kyian, S. & Teather, R. Selection Performance Using a Smartphone in VR with Redirected Input. in *Symposium on Spatial User Interaction* (ACM, New York, NY, USA, Nov. 2021), 1–12. ISBN: 9781450390910.
18. Bai, H., Zhang, L., Yang, J. & Billinghurst, M. Bringing full-featured mobile phone interaction into virtual reality. *Computers & Graphics* **97,** 42–53. ISSN: 00978493 (June 2021).
19. Genymobile. *Scrcpy v3.1* Feb. 2025. https://github.com/Genymobile/scrcpy.
20. Umer Farooq. *SensorServer v6.4.0* 2025. https://github.com/umer0586/SensorServer.