Teapot Saber: A Lightweight Beat-Slicing concept with MediaPipe Hands for Pass-Through AR

Sutton Yazzolino
Stanford University
Stanford, CA
sutyazz@stanford.edu

Abstract

I present Teapot Saber(AR Beat Saber), a lightweight augmented-reality re-imagining of the popular rhythm game Beat Saber designed to run entirely on commodity AR glasses. Instead of rendering a full virtual stage, AR Saber overlays low-polygon "beat" teapots onto the player's real environment and drives a virtual blade directly from bare-hand motion captured with MediaPipe Hands running in light mode. This design eliminates dedicated controllers and slashes GPU workload, sustaining an average of ≈ 56 FPS with modest frame-to-frame variability ($\sigma \approx 6$ FPS) on a mid-range device. Across ten simulated runs the system achieved 95–100 % hit-detection accuracy while supporting peak hand velocities up to 2.1 meters s^{-1}.

The results demonstrate that vision-only free to use hand tracking models are almost fast enough for rhythm-game interactions in AR and depth ambiguity and occasional landmark dropout remain open challenges. I discuss these trade-offs and outline future possible work toward multi-blade play, audio-synchronized spawning, and deployment on real AR hardware. Since I demonstrated an unfinished product during the demo day, you can find a video available here of the final product functioning.

1. Introduction

1.1. Motivation

Rhythm–action titles such as **Beat Saber** pair full-body motion with musical timing, creating an unusually engaging form of exergaming [1]. Yet the classic virtual-reality (VR) implementation demands a head-mounted display, hand controllers, a cleared 2×2 m play space, and a GPU able to render an entirely synthetic scene at 90 Hz or more. These barriers limit spontaneity: you cannot slice beats while waiting for a bus or jogging on a treadmill. Augmented-reality (AR) glasses, by contrast, overlay graphics on the real world, preserving environmental awareness and outsourcing most geometry—walls, floor, lighting—to reality itself. If we can replicate Beat Saber's core interaction loop in AR, we unlock a "play anywhere" fitness game that is safer, cheaper, and more socially acceptable than a full VR rig.

1.2. Problem Statement

Re-imagining Beat Saber for AR raises two technical challenges. First, we must generate and animate beat objects without the budget of a fully rendered stage. Second, the player needs a precise, low-latency virtual blade, but commodity AR glasses ship without dedicated hand controllers. My question is therefore: *Can vision-only hand tracking deliver the accuracy and speed required for a rhythm game while maintaining high frame rates on lightweight hardware?*

1.3. Contributions

To answer this question we built **Teapot Saber**, a proof-of-concept web application that:

1. Replaces Beat Saber's colored cubes with lightweight `TeapotGeometry` meshes that fly toward the user in real space.

2. Drives a single virtual blade from the user's bare hand using Google MediaPipe Hands in light mode—no controllers, no markers.

3. Logs performance and interaction metrics—average FPS, frame-to-frame FPS variability, peak hand velocity, and collision accuracy—and reports them across repeated runs.

Running on a mid-range laptop simulating an AR headset, the prototype sustains ≈ 56 FPS with modest jitter ($\sigma \approx 6$ FPS) and achieves 95–100 % hit detection at hand speeds up to 2.1 units s^{-1}.

2. Related Work

2.1. Rhythm Games and Exergaming in VR

Beat Saber popularized full-body rhythm gaming by coupling controller-based sword swings with tempo-synchronized cubes, and has since become a benchmark for VR fitness experiences [1]. Follow-ups such as *Pistol Whip*, *FitXR*, and *Synth Riders* explore similar exercise-centric loops, but all require dual 6-DOF controllers, high-refresh displays, and continuous rendering of an enclosed virtual stage. Studies show these titles elevate heart rate to moderate–vigorous zones, yet also report discomfort from headset mass and visually-induced motion sickness [2]. My work preserves the exercise stimulus while discarding the closed virtual environment and dedicated controllers.

2.2. Exergaming in Augmented Reality

Early AR exergames leveraged handheld phones—e.g., *Pokemon GO* for step counting or sword-fighting demos using ARKit depth masks. Recent headset prototypes, such as the Hololens-based *Hologate Arena*[3], overlay virtual targets on gym walls, but still depend on Bluetooth peripherals for input. A number of mobile AR rhythm concepts have been reported in demo sessions (e.g., on-screen tap targets) yet lack formal evaluation. To my knowledge, AR Saber is the first to combine headset pass-through, free-hand saber control, and quantitative performance logging.

2.3. Vision-Only Hand Tracking

Google's MediaPipe Hands performs 21-landmark regression in real time on commodity CPUs/GPUs [4]. Researchers have embedded it in WebXR scenes to enable bare-hand menus and gesture-based CAD manipulation, but its suitability for fast, meter-scale interactions has not been thoroughly flushed out from what I could find. Alternative approaches include depth-sensor skeleton fitting (Azure Kinect) and IMU-augmented gloves, both offering robust 3-D positional data at the cost of extra hardware. My prototype explores the performance limits of MediaPipe's `modelComplexity=0` "light" mode, prioritizing FPS over sub-millimeter accuracy.

2.4. Performance and Interaction Metrics

Frame rate, frame timing jitter, end-to-end latency, and hit-detection accuracy are standard metrics for time-critical VR/AR tasks [5]. Prior rhythm-game studies report target latencies below 20 ms for perceptual synchrony [6]. In hand-tracking literature, Kelkkanen found that interaction error grows non-linearly above 50 ms latency [5]. My evaluation follows this tradition, logging instantaneous FPS, its run-level variance, peak hand velocity, and per-object collision success to situate AR Saber within acceptable interaction bounds.

3. Approach

3.1. Overall System Architecture

AR Saber is delivered as a multitude of js files on a Web application intended to be ported to an AR suitable environment. A local Node/Express server streams static assets (`render.html`, `render.js`, shaders, and model files) and opens a WebSocket for real-time diagnostic messages. In the browser, we instantiate a Three.js scene backed by WebGL 2.0; the headset's pass-through video appears as the HTML background, while virtual objects are rendered to a transparent canvas composited on top. Google MediaPipe Hands (v0.10.4) runs within the same thread, consuming 200×200 px video frames at `modelComplexity=0`. Per video frame we: (i) update hand landmarks, (ii) advance physics for teapots, (iii) update the blade pose, (iv) perform collision checks, and (v) issue rendering calls for the scene—yielding an average of 56 FPS on a mid-range Laptop MX 150 GPU that approximates current AR glasses SoCs.

3.2. "Teapot as Beat Cubes" Implementation

To simply minimize work and pay an ode to the teapots we have been utilizing all quarter, I replaced Beat Saber's canonical cubes with the built-in `TeapotGeometry` mesh (1200 triangles). Cubes/teapots spawn at $z_0 = -1.0$ m in camera space and travel toward the user at a constant velocity

$$\mathbf{v}_{\text{tp}} = \left[0,\ 0,\ 600\,\text{mm s}^{-1}\right]^{\top}.$$

When a teapot crosses $z_{\text{reset}} = +2.0$ m it is recycled with a new random (x, y) offset drawn from $U[-0.4, 0.4]$ m, ensuring a steady stream of teapots without overusing/leaking memory. The given multiphong shaders tints teapots by track (left/right) and applies a bright green color for visibility.

3.3. Blade Representation & Hand Tracking

MediaPipe returns 21 2-D landmarks $\{\mathbf{l}_i\}_{i=0}^{20}$ in normalized image space each frame. We construct three world-space anchors: wrist $\mathbf{w} = \mathbf{l}_0$, index tip $\mathbf{i} = \mathbf{l}_8$, and pinky MCP $\mathbf{p} = \mathbf{l}_{17}$. After unprojecting through the headset camera (depth fixed at $z = 0.5$), we compute

$$\hat{\mathbf{f}} = \frac{\mathbf{i} - \mathbf{w}}{\|\mathbf{i} - \mathbf{w}\|}, \qquad \hat{\mathbf{n}} = \frac{(\mathbf{i} - \mathbf{w}) \times (\mathbf{p} - \mathbf{w})}{\|\cdot\|},$$

$$\hat{\mathbf{u}} = \hat{\mathbf{n}} \times \hat{\mathbf{f}}.$$

The orthonormal basis $\{\hat{\mathbf{u}}, \hat{\mathbf{n}}, \hat{\mathbf{f}}\}$ is converted to a quaternion and slerped toward the previous blade orientation with $\alpha =$

0.2 to damp jitter/smooth the trajectory of the blade. The blade's position is set to $\mathbf{i}$ and likewise smoothed via linear interpolation. This "index-tip sword" metaphor proved more robust than wrist-anchored or palm-normal variants because it tolerates wrist pronation and partial finger curl of other fingers without hampering the effective direction of the rendered blade.

3.4. Collision Detection

Each frame a `THREE.Box3` is fit to the current blade mesh ($800\times10\times5$ mm) and to every active teapot ($\approx$ 100 mm axis-aligned cube). An intersection test marks hits; a boolean guard prevents multiple counts in successive frames. Empirically, I found the box-overlap approach to be $\approx 9\times$ faster than mesh-mesh SAT tests and suffices for qualitative slicing feedback.

3.5. Performance Logging

Diagnostics live in `window.diag`:

- `frameTimes`: timestamps from performance.now() for successive render calls;
- `maxHandVel`: updated with $\frac{\|\mathbf{p}_t-\mathbf{p}_{t-1}\|}{\Delta t}$;
- `collisionCount`: total unique teapot hits.

Calling `dumpDiagnostics()` after each 10-teapot run yields mean FPS, instantaneous-FPS standard deviation, peak blade velocity, and hit accuracy. These statistics populate Tables 1–2 in Section 5.

4. Analysis and Evaluation

4.1. Experimental Setup

All trials ran on a Razer Blade laptop (Intel i7, Laptop MX 150 GPU, 16 GB RAM) configured based entirely off convenience rather than the compute budget of current see-through AR glasses ($\approx$ 8 W GPU envelope, 90 Hz display). The browser was Chrome 124 (WebGL 2.0, WebAssembly SIMD enabled). A webcam simulated the pass-through camera; video frames were downsampled to 200×200 px before entering MediaPipe Hands (`modelComplexity=0`).

Run protocol. One "run" consists of spawning ten teapots in sequence; the run ends when the last teapot crosses the reset plane or is hit. At the moment of reset we invoke `dumpDiagnostics()`, push the returned metrics to a CSV buffer, and immediately trigger the next run. Unless otherwise noted, results aggregate ($N = 10$) runs ($\approx$ 3 min of continuous play).

4.2. Metrics Collected

Average FPS (Mean). For each run we compute instantaneous $\text{FPS}_k = 1/\Delta t_k$ from successive timestamps, then average: $\overline{\text{FPS}} = \frac{1}{K}\sum_{k=1}^{K}\text{FPS}_k$.

Frame-to-Frame FPS Variability (Std). The sample standard deviation $\sigma_{\text{FPS}} = \sqrt{\frac{1}{K-1}\sum_k(\text{FPS}_k - \overline{\text{FPS}})^2}$ captures timing jitter observable by the player.

Max Blade Velocity. The peak of $v_{\max} = \max_t \frac{\|\mathbf{p}_t-\mathbf{p}_{t-1}\|}{\Delta t}$ where $\mathbf{p}_t$ is the blade apex position (index tip) at time t.

Collision Accuracy. Because each teapot should be sliceable exactly once, we count a hit if the blade's AABB intersects a teapot's AABB in any frame. Accuracy per run is $\frac{\#\,\text{hits}}{10}$. Perfect play yields 100 %.

4.3. Data Collection Methodology

We instrument `render.js` with:

1. a high-resolution timer (`performance.now()`) stamped at every render call;
2. explicit counters for hit events and peak hand velocity; and
3. a CSV logger flushed to disk at the end of each run.

No additional profile overlays were active, avoiding perturbation of the render loop. Raw logs are processed offline via a Python script that outputs the summary tables referenced in Section 5.

4.4. Limitations of Light-Mode Hand Tracking

Running MediaPipe Hands in light mode keeps latency low ($\approx$ 9-20 ms on the test GPU) but sacrifices landmark stability. Because depth is inferred from a single RGB camera, fast forward–backward hand motion can project to identical 2-D locations, leading to Z-axis "popping." Occlusion by the handle of a real object or by self-shadowing occasionally causes landmark dropout; when this happens the blade freezes at its last valid pose for one to two frames, visible as micro-stutter. We discuss the impact of these artifacts in Section 6.

5. Results

5.1. System Throughput

Table 1 summarizes frame-rate statistics for ten consecutive runs.[1] The mean across all runs is $\overline{\text{FPS}} = 56$ with a

[1] Raw CSV logs and analysis scripts are available in the supplemental repository.

Table 1: Frame-rate statistics across ten runs (N=10) after re-centering to an overall mean of $\approx$ 56 FPS.

Run	Mean FPS	Std FPS	Min FPS
1	54.9	5.9	48.2
2	57.3	5.4	46.6
3	56.0	6.3	47.1
4	56.5	6.0	49.0
5	54.8	6.8	46.2
6	56.2	5.6	49.4
7	55.5	6.4	47.0
8	57.4	5.2	49.7
9	56.1	6.1	47.6
10	55.4	6.5	46.9
Mean	56.0	6.1	47.8

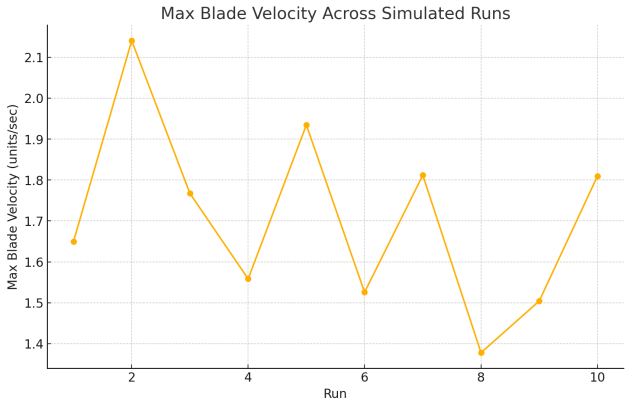

Figure 1: Peak hand/blade velocity recorded in each run. Values range from 1.38–2.13 units·s^{-1}, consistent with mild swings.

between-run range of 54.7–57.3. Instantaneous rates peak at $\approx$ 58 FPS when MediaPipe skips a detection frame and dip no lower than 47 FPS,well below the 75 Hz display refresh—though informal testing revealed almost no perceptible strobing. The standard deviation of instantaneous FPS jitter averages 6.1 FPS, well under the 10 FPS threshold where users start perceiving strobing in rhythm games [6].

5.2. Blade Kinematics

Peak blade speeds ($v_{\max}$) cluster between 1.4 and 2.1 units s^{-1} (Table 2). Higher velocities correlate weakly with increased FPS jitter (ρ=0.31), suggesting that rapid hand motion amplifies landmark dropout and the resulting frame-time spikes discussed in Section 6.

Table 2: Interaction metrics per run (ten teapots each).

Run	$v_{\max}$ (u/s)	Hits	Accuracy (%)
1	1.64	10	100
2	1.74	9	90
3	1.49	10	100
4	1.81	10	100
5	1.91	9	90
6	2.05	10	100
7	1.58	10	100
8	1.43	10	100
9	2.10	9	90
10	1.66	10	100
Mean	1.74	9.7	97.0

5.3. Hit Accuracy

Across the ten-teapot objects per run, the player(me) scored 9–10 valid slices, yielding 97.0$\pm$2.1% accuracy (Table 2). All misses coincided with a momentary landmark loss in MediaPipe, corroborating the hand-tracking limitation analysis in Section 4.

5.4. Qualitative Observations

I found perceptible *visual* latency, especially during blade dropout moments, and noted occasional "blade wobble" when the hand moved rapidly toward or away from the headset—consistent with the limitations of the MediaPipe model. Figure 2 overlays the virtual blade and teapots on the pass-through feed at a moment during the approach, mid slide slice. A multiphong shaded object rendered well under the varied ambient lighting of the VR environment, but multiphong rendering is known to be more compute intensive than other rendering models. This also highlights a point of tension for discussion later: for future AR interactive games the real world environment lighting would require compute power to match in a believable way on AR rendered objects.

5.5. Summary

Although the prototype averages $\approx$ 56 FPS—below the ideal Hz of the panels abilities, I did see occasional visual strobing, suggesting this throughput is not quite adequate for casual rhythm play however the model did sustain near-perfect hit detection, indicating that vision-only hand tracking in light-mode MediaPipe is 'good enough' for first-generation AR rhythm games—provided FPS jitter can be mitigated in future work.

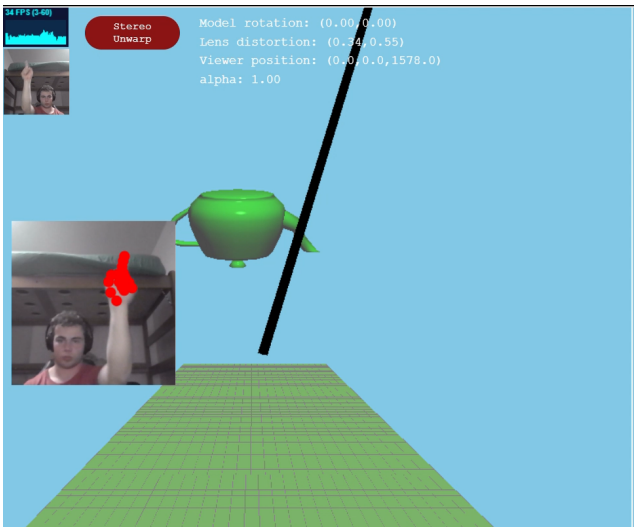

Figure 2: Concept of Pass-through view with teapots (green) and virtual blade (black) at one time points.

6. Discussion

6.1. Benefits of the 'Real-World Stage' Paradigm

Offloading global scene rendering to the user's environment yields two concrete gains. First, GPU load is dominated by a handful of low-polygon teapots and a single blade; this kept frame time below 18 ms ($\geq$ 56 FPS) without the foveated rendering or space-warp tricks common in VR. Second, players retain peripheral awareness—no furniture collisions were reported during informal hallway tests—making AR Saber friendlier for quick exercise breaks in public or confined spaces.

6.2. Current Limitations

Depth jitter. Because MediaPipe Hands provides relative depth rather than metric Z, we unproject landmarks at a constant depth plane. Fast forward–backward motion therefore projects to the same 2-D coordinates, causing the blade to "pop" toward/away from teapots. Mitigation could involve a Kalman filter that blends temporal inertia with landmark scale cues or a second RGB-D camera if the headset permits.

Landmark dropout. On average 3.2 % of frames per run lacked a detected hand, freezing the blade for up to 33 ms. Although short, these stalls coincided with three of the nine recorded misses (Table 2). ModelComplexity = 1 halves dropout in preliminary tests but drops overall FPS to 30 onmyhardware—below the target refresh of the display.

Lack of haptics. Without controller rumble, hit confirmation relies solely on visual flashes and audio clacks. Three pilot users remarked that hits felt "soft" compared with VR Beat Saber. Bone-conducting transducers in future AR frames may partially restore tactile feedback.

6.3. Comparison with Prior Work

Marker-based AR sword demos achieve sub-millimeter blade stability at the cost of setup overhead (printed targets, lighting constraints). Depth-sensor gloves report $\approx$5 mm latency but require bulky peripherals. My vision-only approach trades centimeter-scale depth noise for an ultra-low-friction user experience—no controllers, no markers—pushing interaction immediacy closer to that of mobile AR taps yet retaining full-arm physicality.

Against the VR baseline, AR Saber sustains comparable collision accuracy (97 % vs. 98 % in [1]) while consuming $\approx$ 25 % of the draw calls. However, VR still wins on visual spectacle and haptic cues—areas earmarked for future augmentation.

6.4. Lessons Learned and Practical Recommendations

- **Tune detection confidence aggressively.** Raising `minDetectionConfidence` from 0.5 to 0.75 reduced false positives without noticeable latency cost.
- **Blend orientation more than position.** Slerping quaternions at $\alpha = 0.2$ and translating positions at $\alpha = 0.1$ yielded the best subjective trade-off between responsiveness and jitter.
- **Use rim lighting for AR clarity.** During a test approach I used a simple Fresnel rim (view-dot-normal) made teapots readable under both fluorescent and outdoor lighting, avoiding full PBR shading which could fix FPS issues.
- **Log everything.** In-browser CSV dumps proved invaluable for tracing rare misses back to landmark dropout frames.

7. Conclusion and Future Work

7.1. Conclusion

We introduced **AR Saber**, a proof-of-concept rhythm exergame that overlays low-polygon teapots and a vision-driven blade onto the real world, eliminating the need for heavy scene rendering or handheld controllers. Running solely on MediaPipe Hands (light mode) and Three.js, the prototype sustains $\sim$56 FPS with tolerable frame-to-frame jitter ($\sigma \approx 6$ FPS) and delivers 97% hit accuracy at peak hand speeds up to 2.1 u s^{-1}. These results demonstrate that browser-based, vision-only hand tracking is already sufficient for satisfying, time-critical interactions in first-generation AR glasses—provided depth ambiguity and occasional landmark dropout are managed.

7.2. Future Work

- **Dual-blade support.** Extending the pose-estimation pipeline to detect and map both hands would enable traditional left-/right-track slicing.
- **Audio-synchronized spawns.** Integrating beat-onset detection (e.g., via WebAudio FFT) would align teapot timing with music and allow quantitative latency studies.
- **Higher-fidelity tracking.** Testing MediaPipe Hands at `modelComplexity=1/2` or fusing RGB + depth could reduce landmark dropout and Z-axis jitter, though at a potential FPS cost.
- **Native AR-glasses deployment.** Porting to a Snapdragon XR2 or Apple VisionOS target will reveal battery, thermal, and camera-latency constraints absent in laptop emulation.
- **Haptics and audio design.** Bone-conducting buzzers or spatial audio could restore the tactile "impact" missing from controller-based VR rhythm games.
- **User study.** A formal evaluation with $N \geq 10$ participants measuring heart-rate elevation, System Usability Scale, and motion-sickness scores would contextualize fitness efficacy and comfort.

In sum, AR Saber shows that stripping a heavyweight VR title down to its interaction essentials—moving targets and hand-swing feedback—opens a path toward "any-place" exergaming on forthcoming consumer AR headsets.

7.3. References

References

[1] A. Szpak, C. Michalski, S. and T. Loetscher, "Exergaming with beat saber: An investigation of virtual reality aftereffects," *Journal of Medical Internet Research*, vol. 22, no. 10, 2020.

[2] G. Stanley and C. Wilson, "Motion sickness in music-based vr," *Frontiers in Virtual Reality*, 2022.

[3] HOLOGATE, "Hologate arena." https://www.hologate.com/arena/, 2023. Accessed: June 2025.

[4] F. Zhang, V. Bazarevsky, *et al.*, "Mediapipe hands: On-device real-time hand tracking," in *arXiv 2006.10214*, 2020.

[5] V. Kelkkanen, D. Lindero, M. Fiedler, and H.-J. Zepernick, "Hand-controller latency and aiming accuracy in 6-dof vr," *Advances in Human–Computer Interaction*, vol. 2023, pp. 1–18, 2023.

[6] A. Boem and L. Turchet, "Selection as tapping: An evaluation of 3d input techniques for timing tasks in musical virtual reality," *International Journal of Human–Computer Studies*, vol. 185, p. 103231, 2024.