# Gaze-Assisted 3D Pointing: A Multimodal Framework for Precise Subcomponent Selection in Dense 3D Volumes

EMMANOUIL PLATAKIS, Aarhus University, Denmark

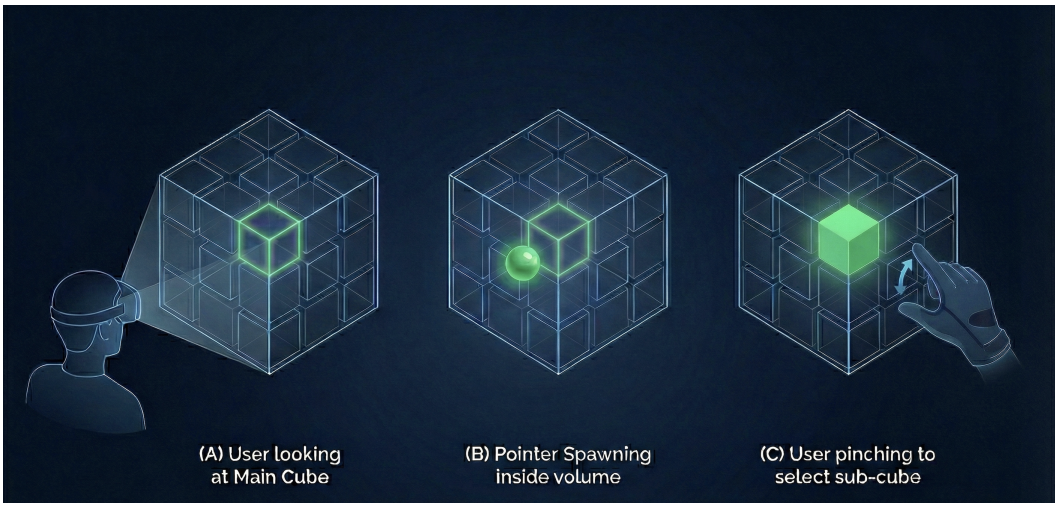


Fig. 1. Overview of the Gaze-Assisted 3D Pointing technique. The user gaze defines the active volume (Main Cube), spawning a 3D pointer that is manipulated via hand tracking to select a specific subcomponent.

As Virtual Reality (VR) environments increase in complexity, users frequently encounter "Subcomponent Selection Tasks," where specific targets are nested within dense volumetric clusters. Traditional ray-casting techniques struggle in these scenarios due to occlusion and depth ambiguity. This study investigates **Gaze-Assisted 3D Pointing** with **Localized Selection**, a multimodal interaction paradigm combining eye tracking for coarse acquisition and hand tracking for fine manipulation. I introduce a "Tool State" architecture allowing for discrete operations (Coloring, Spawning, Deleting, Duplicating) within these dense volumes. I demonstrate the efficacy of this system through three distinct application scenarios: Automotive Assembly, Interior Design, and Abstract Volumetric Manipulation.

Additional Key Words and Phrases: Virtual Reality, 3D Interaction, Gaze-Assisted Pointing, Eye Tracking, Hand Tracking, Multimodal Interaction

## 1 INTRODUCTION

Object selection is the fundamental prerequisite for interaction in spatial computing. While ray-casting remains the industry standard for general selection tasks [2], its efficacy degrades significantly in dense, volumetric environments. This challenge is defined as the *Subcomponent Selection Task*: the requirement to select a specific, small target (e.g., a screw) nested within the volume of a larger parent object (e.g., an engine block). In such scenarios, standard techniques suffer from occlusion and depth ambiguity [1], often forcing users to perform complex navigational maneuvers to obtain a clear line of sight.

Multimodal interaction offers a solution by combining the complementary strengths of eye and hand tracking. Eye gaze is naturally faster than hand pointing for initial target acquisition [6] but suffers from the "Midas Touch" problem—the difficulty of distinguishing between viewing and

Author's address: Emmanouil Platakis, 202402638@post.au.dk, Aarhus University, Aarhus, Denmark.

interacting [3]. Conversely, direct hand input offers high precision but is prone to physical fatigue ("Gorilla Arm") and the "Heisenberg Effect" [10], where the motor action of confirmation (e.g., a button press) inadvertently displaces the pointer.

In this work, I propose a **Gaze-Assisted 3D Pointer**. By constraining the pointer to the bounding volume of the object identified by the user's gaze, I create a temporary "Work Volume," filtering out environmental noise. Unlike previous studies which focused solely on the mechanics of spawning, this work presents a fully realized application framework including tool management, transparency handling, and hierarchical depth selection.

## 2 BACKGROUND

### 2.1 Gaze + Pinch Interaction

The "Gaze + Pinch" paradigm, formalized by Pfeuffer et al. [5], serves as the foundation for modern multimodal VR. They demonstrated that eye gaze effectively replaces the ballistic phase of hand movement (coarse acquisition), reserving the hand for the final correction phase (fine manipulation). This division of labor exploits the natural speed of saccades while mitigating the jitter inherent in eye tracking. My work extends this paradigm by introducing a **volumetric 3D cursor**. Unlike a 2D selection ray, a volumetric cursor spawned by gaze allows for depth-aware manipulation, addressing the specific requirements of dense 3D clusters where a simple ray might penetrate multiple occluded targets.

### 2.2 Occlusion and the Localized Pointer

High target density inevitably leads to occlusion, a critical limitation for standard ray-casting [1]. While techniques like "Exploded Views" [11] solve visibility by expanding object clusters, they impose a high cognitive load by forcing users to mentally reconstruct the original spatial arrangement.

My approach preserves spatial context by employing a **Localized 3D Pointer**. This concept evolves the "Depth Ray" [7], which required manual depth adjustment, by using eye gaze to instantly set the initial depth vector. This synergy is supported by Wagner et al. [8] and Lystbæk et al. [4], who demonstrated that gaze naturally precedes manual input. However, unlike Lystbæk et al.'s "Gaze-Hand Alignment"—which triggers selection implicitly upon alignment—I retain an **explicit pinch gesture**. In dense volumetric environments, implicit triggers risk inadvertent selection of occluding objects (Midas Touch); an explicit manual clutch is therefore necessary to disambiguate the user's intent.

### 2.3 Comparison with Conventional 3D Techniques

To contextualize the necessity of a Gaze-Assisted 3D Pointer, it is critical to analyze the limitations of similar approaches in dense volumetric environments.

*2.3.1 Ray-Casting and the Heisenberg Effect.* While Ray-Casting is the industry standard for general selection [2], it suffers significantly in high-precision tasks. Wolf et al. defined the "Heisenberg Effect" [10], where the discrete motor action required to confirm a selection (e.g., a button press or pinch) inadvertently displaces the ray cursor, causing errors on small targets. In my system, the localized pointer is clamped to the target volume, effectively filtering out this high-frequency angular jitter that plagues standard ray-casting.

*2.3.2 Virtual Hand and Physical Fatigue.* The "Virtual Hand" or "Go-Go" technique maps the user's hand directly to 3D space. While intuitive, interacting with distant objects via direct mapping induces rapid physical fatigue, known as the "Gorilla Arm" effect [2]. Furthermore, in dense clusters,

the visual representation of the virtual hand itself often occludes the target subcomponent. My approach avoids this by acting as a remote extension; the user keeps their hand in a comfortable resting position while the pointer operates at a distance, ensuring line-of-sight is never blocked by the user's own anatomy.

*2.3.3 2D Plane Pointing vs. Volumetric Needs.* Techniques that project 2D input onto 3D space struggle with the "Subcomponent Selection Task." A 2D cursor cannot reach occluded objects nested centrally within a cluster without complex world-rotation mechanics. By utilizing a volumetric pointer spawned via gaze depth, my system accesses these nested targets directly, treating the environment as a volume rather than a surface.

## 3 CONCEPT

The primary objective of this work is to establish a robust interaction framework for *Subcomponent Selection*—the task of identifying and manipulating small targets nested within dense, occluded volumes and test the usability of the Gaze-Assisted 3D Pointer in different scenarios. The design is driven by the necessity to overcome the limitations of standard ray-casting (jitter, depth ambiguity) and direct manipulation (fatigue, occlusion) in professional 3D workflows.

### 3.1 Localized Volumetric Interaction

In standard 3D interaction, "depth overshoot" is a pervasive issue where a selection cursor inadvertently slips behind the intended target. To mitigate this, I introduce the concept of a **Temporary Work Volume**. By utilizing the user's gaze to identify a parent object (e.g., a car), the system constrains the 3D pointer to the geometric bounds of that specific object or a predefined collider bound from the user. This effectively filters out environmental noise and restricts the Degrees of Freedom (DOF) to the relevant interaction space, stabilizing the user's input.

### 3.2 The "Tool-State" Paradigm

Most multimodal research focuses on atomic selection (clicking). However, real-world tasks require complex operations. My concept extends the Gaze-Assisted pointer into a **State-Based Tool**. The pointer is not just a selector; it is a polymorphic effector that changes behavior based on the active tool state (Coloring, Deleting, Spawning). This shifts the interaction model from simple "pointing" to "spatial editing."

## 4 IMPLEMENTATION

The system runs on the **Meta Quest Pro**, utilizing native Eye and Hand tracking via the Meta XR Core SDK in Unity 6. It relies on a modular architecture driven by three core components: The `Interaction Tool Manager`, the `Gaze Pointer Manager`, and the `Index Pinch Manipulator`.

### 4.1 Interaction State Architecture

To support complex workflows beyond simple selection, I implemented a state-machine architecture managed by the `InteractionToolManager`. This allows the user to switch between distinct modes via a left-hand palm menu:

- **Move/Rotate:** Standard 6-DOF manipulation.
- **Duplicate/Delete:** Logic for object management.
- **Color Picker:** For aesthetic customization.
- **Spawn:** For introducing new objects into the scene.

This state machine informs the `IndexPinchManipulator`, ensuring that a "Pinch" gesture is context-aware—executing a drag operation in "Move" mode, or a discrete ray-cast trigger in "Delete" mode.

### 4.2 The Gaze-Assisted 3D Pointer

The core technique utilizes the **`GazePointerManager`**. When the user gazes at a target (e.g., a car chassis) and performs a middle-finger pinch clutch, a 3D pointer is instantiated.

*4.2.1 Hierarchy Depth Selection.* In scenarios where a small collider (e.g., a door handle) is nested inside a large collider (e.g., a car door), standard ray-casting often fails, selecting the outer shell. My system implements a **Depth-Weighted Selection** algorithm. When the 3D pointer overlaps multiple colliders, the system prioritizes the object that is deepest in the transform hierarchy (the innermost child), enabling precise selection of subcomponents without needing to disassemble the parent object.

*4.2.2 Pointer Mapping.* A critical challenge in distant 3D manipulation is mapping physical hand movement to the virtual pointer's movement. A linear 1:1 mapping often feels disconnected due to perspective distortion at depth. To address this, I used a *Visual Angle Gain* technique, which is very similar to the HOMER algorithm[9].

**Algorithm 1:** Visual Angle Pointer Mapping

**Inputs:**
$P_{eye}$ (Camera Position)
$P_{hand}$ (Current Hand Position)
$P_{obj}$ (Current Object/Pointer Position)
$\Delta_{hand}$ (Hand Movement Delta)
$S$ (Sensitivity Factor)

**Procedure:**
$d_{hand} \leftarrow \|P_{hand} - P_{eye}\|$
$d_{obj} \leftarrow \|P_{obj} - P_{eye}\|$
$Gain \leftarrow \frac{d_{obj}}{\max(d_{hand}, 0.01)}$ // Calculate Gain based on the ratio of distances
$\Delta_{scaled} \leftarrow \Delta_{hand} \times Gain \times S$ // Scale the hand movement delta
$P_{new} \leftarrow P_{obj} + \Delta_{scaled}$

This implementation calculates a dynamic gain factor based on the ratio of the eye-to-object distance versus the eye-to-hand distance:

$$\Delta_{virtual} = \Delta_{hand} \times \left( \frac{D_{eye-object}}{D_{eye-hand}} \right) \times Sensitivity \tag{1}$$

By scaling the hand's movement delta ($\Delta_{hand}$) by this distance ratio, the system approximates a 1:1 visual–angle response.

### 4.3 Visual Feedback

An Outline Script highlights the specific subcomponent currently hovered by the 3D pointer, providing immediate feedback before confirmation.

### 4.4 Code Availability & Demo

I provide a link to the GitHub repository associated with this project: **Gaze-Assisted 3D Pointing: A Multimodal Framework for Precise Subcomponent Selection in Dense 3D Volumes**. This repository serves as a comprehensive resource for readers interested in exploring the source code, accessing additional materials, or furthering their understanding of the concepts discussed in the

project. By providing access to the repository, I aim to promote transparency, reproducibility, and collaboration in the realm of academic research and software development.

In the following link, you will find a video demonstrating the three application scenarios: **Demo**.

## 5 APPLICATION SCENARIOS

To validate the utility of the Gaze-Assisted 3D Pointer, I designed three distinct scenarios representing real-world and abstract use cases.

### 5.1 Scenario 1: Automotive Assembly & Customization

**The Challenge:** A user needs to customize the interior of a car. The seats and steering wheel are occluded by the car's roof and doors. Using a standard ray-cast would require the user to physically walk inside the car geometry or awkwardly maneuver the controller through the window.
**The Solution:**

(1) **Localized Entry:** The user middle-pinches to spawn the 3D pointer inside the car volume.
(2) **Tool Application:** The user selects the Color Picker tool from the left-hand menu, selects "Red," and pinches the steering wheel.
(3) **Result:** The steering wheel changes color instantly. The pointer is then moved to the seats to apply the same color. The user never needed to physically move their body; the gaze-assisted pointer acted as a remote extension of their hand inside the vehicle (2).

### 5.2 Scenario 2: Interior Design (Cluttered Environment)

**The Challenge:** An interior designer needs to place a new chair into a crowded apartment scene. The room already contains tables, lamps, and other furniture. Placing an object using 2D screen space projection often results in the object floating in mid-air or clipping through walls due to depth ambiguity.
**The Solution:**

(1) **Spawn Mode:** The user selects "Spawn: Chair" from the menu.
(2) **Surface Snapping:** The user looks at a specific spot on the floor between two tables. The `GazePointerManager` detects the room's "ColliderBound" tag and snaps the 3D pointer exactly to the floor surface at the gaze point.
(3) **Precise Manipulation:** Once spawned, the user switches to "Move" mode. The containment logic allows them to slide the chair along the floor or move it onto a table surface without it drifting through the walls (3).

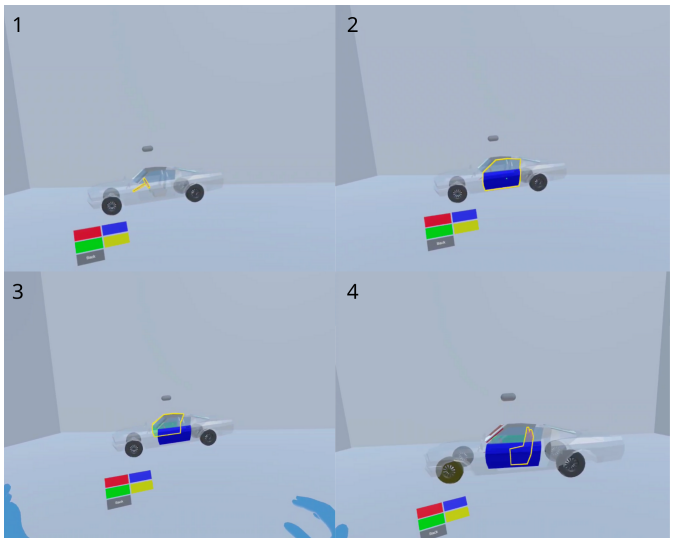


Fig. 2. 1) The user stands outside the vehicle, spawning a 3D pointer inside the cabin to select the steering wheel and paints it in red. 2) The 3D pointer is moved on the left door and it is painted blue. 3) The 3D pointer is moved on the right door to paint it green. 4) The seats are painted red.

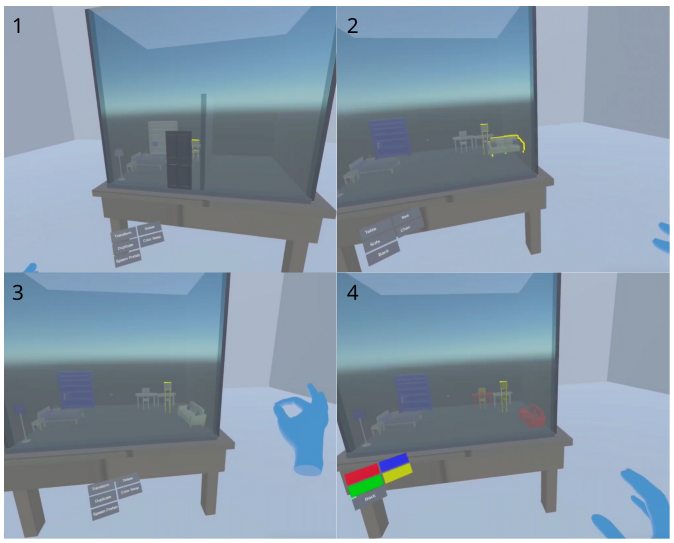


Fig. 3. 1) The user stands outside the doll house in order to do interior design. 2) Spawn mode is selected (Sofa) and the 3D pointer is moved at the point the user wants the sofa to be spawned and spawns it. 3) The Move/Rotate tool is used to place the sofa where exactly the user wants. 4) The sofa is painted red using the Color tool.

### 5.3 Scenario 3: Abstract Volumetric Manipulation

**The Challenge:** To demonstrate the raw precision of the technique, I present a 3x3x3 grid of dense cubes. The user must select and draw specific internal cubes to form a specific pattern (e.g., every

level is a different color). This represents the "worst-case" scenario for occlusion.

**The Solution:**

(1) **Depth Selection:** The user gazes at the cluster and spawns the pointer. Using their hand, the pointer goes deep within the grid.
(2) **Visual Feedback:** As the pointer moves through the dense grid, the Hierarchy Depth Selection logic ensures that only the specific cube enveloping the pointer tip is outlined (Yellow Outline).

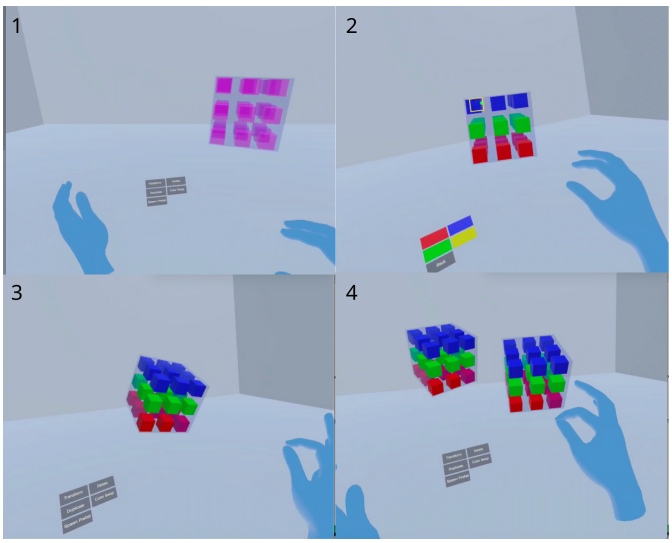


Fig. 4. 1) The user can see the 3x3x3 grid of cubes inside the "Main Cube". 2) Using Color mode the user spawns and moves the 3D pointer inside each subcube to color the lower level subcubes in red, the middle level green and the top level blue. 3) The Move/Rotate tool is used to manipulate the "Main Cube". 4) The Duplicate mode is used to duplicate the "Main Cube".

## 6 DISCUSSION

The results from the application scenarios demonstrate that the transition from simple ray-casting to a **Gaze-Assisted 3D Pointer** fundamentally changes the interaction paradigm for dense volumetric environments.

### 6.1 The Role of Geometric Containment

A persistent issue in 3D manipulation is the "Heisenberg Effect" [10], where the motor act of selection introduces jitter that displaces the cursor. My implementation of **Geometric Containment Logic** (via `Collider.ClosestPoint`) proved to be a robust countermeasure. By effectively clamping the pointer to the active work volume (e.g., the room or the car), the system filters out high-frequency hand noise. This suggests that future 3D interaction frameworks should move away from unconstrained 6-DOF input and instead adopt "context-aware constraints" derived from the user's gaze target.

### 6.2 Resolving Occlusion via Multimodal Feedback

The integration of the **Depth-Weighted Selection** addresses the problem of standard ray-casting in occluded environments. Standard ray-casting forces users to accept the first object hit by the ray.

By using the 3D pointer and algorithmically prioritizing the deepest child in the hierarchy, I enable users to "reach through" obstacles. This decoupling of *visual occlusion* from *physical selection* is essential for tasks such as automotive assembly, where disassembly is not always a viable option for accessing internal components.

### 6.3 The Tool-State Paradigm in Spatial Computing

Most gaze-assisted research focuses on atomic selection tasks. By introducing the `InteractionToolManager`, it is demonstrated that localized pointing is scalable to complex toolchains (Coloring, Duplicating, Deleting, Spawning). The success of Scenario 2 (Interior Design) validates that gaze can serve not just as a selection mechanism, but as a *spatial anchor* for tool application—defining "where" an action happens (the floor) before the hand defines "what" happens (placing a chair).

### 6.4 Reflections on Application Integration

Integrating the technique into functional scenarios (Automotive, Architecture) highlighted that gaze acts effectively as a "Spatial Anchor." In the Interior Design scenario, the user looks at the floor to define *where* the interaction happens, and then uses the hand to define *how* it happens. This separation of concerns reduced the cognitive load typically associated with 6-DOF placement. By locking the pointer to the "ColliderBound" of the room, the system effectively reduces the interaction dimension from 3D (free space) to 2D (surface sliding) without explicit mode switching, streamlining the workflow for CAD-like tasks.

### 6.5 Pain Points and Technical Limitations

Despite the precision benefits, the implementation revealed specific pain points:

- **Dependency on Collider Fidelity:** The "Containment Logic" and "Depth-Weighted Selection" are heavily dependent on the quality of the underlying mesh colliders. In the Automotive scenario, if the car door's collider is not strictly separate from the chassis, the depth algorithm struggles to disambiguate them. This implies that this interaction paradigm requires a rigorous "asset preparation" phase, unlike ray-casting which works on raw geometry.
- **Tracking Occlusion:** While the "Middle Pinch" clutch allows for single-handed operation, the Meta Quest Pro's optical tracking occasionally struggles to detect the middle finger when the hand is rotated at oblique angles.

### 6.6 Revisiting Related Work in Hindsight

Comparing my results back to the "Depth Ray" proposed by Vanacken et al. [7], I found that automating the depth definition via eye-tracking significantly increases fluidity. Where Vanacken's users had to manually adjust depth, my system's use of Gaze-Spawn provides an instantaneous "entry" into the volume. However, this automation introduces a new challenge: the "Midas Touch" risk of looking at the wrong volume. My implementation of the "2-Step Manipulation" (Look → Pinch to Spawn → Pinch to Act) proved essential here. It re-introduces the *intent* confirmation that Lystbæk et al.'s "Gaze-Hand Alignment" [4] attempts to remove. While Lystbæk argues for implicit selection, my experience with dense clusters suggests that explicit clutching is non-negotiable when precision is paramount.

### 6.7 Benefits

The primary benefit observed is the ability to "reach through" geometry. In the Abstract Manipulation scenario, selecting a central cube surrounded by neighbors was trivial, a task that would be nearly impossible with standard ray-casting without dissecting the scene.

## 7 CONCLUSION & FUTURE WORK

In this work, I have presented a robust implementation of a **Gaze-Assisted 3D Pointing** system tailored for dense volumetric environments. By synthesizing eye tracking for coarse acquisition with hand tracking for fine manipulation, a unified framework has been developed that solves the twin challenges of occlusion and depth ambiguity.

Future research will extend this framework by integrating asymmetric bi-manual manipulations to support complex affine transformations, such as scaling. Additionally, I aim to validate the system's utility in high-fidelity professional domains, specifically within Architectural Engineering and Computer-Aided Design (CAD) workflows. Future iterations of this system should also address the "Convex Hull" limitation. Currently, the containment logic uses ‘Collider.ClosestPoint‘, which approximates bounds. For complex, non-convex architectural shapes (e.g., a spiral staircase), this logic needs to be upgraded to support complex mesh traversal, ensuring the pointer follows the exact surface topology rather than a bounding volume. Ultimately, this work establishes a blueprint for the next generation of spatial interaction techniques, facilitating the transition of Virtual Reality from a medium of passive visualization to one of precise, active creation.

## REFERENCES


[1] Ferran Argelaguet and Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers & Graphics* 37, 3 (2013), 121–136.

[2] Doug A Bowman and Larry F Hodges. 1997. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*. 35–ff.

[3] Robert JK Jacob. 1995. Eye tracking in advanced interface design. *Virtual environments and advanced interface design* 258, 288 (1995), 2.

[4] Mathias N Lystbæk, Peter Rosenberg, Ken Pfeuffer, Jens Emil Grønbæk, and Hans Gellersen. 2022. Gaze-hand alignment: Combining eye gaze and mid-air pointing for interacting with menus in augmented reality. *Proceedings of the ACM on Human-Computer Interaction* 6, ETRA (2022), 1–18.

[5] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. 2017. Gaze+ pinch interaction in virtual reality. In *Proceedings of the 5th symposium on spatial user interaction*. 99–108.

[6] Linda E Sibert and Robert JK Jacob. 2000. Evaluation of eye gaze interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 281–288.

[7] Lode Vanacken, Tovi Grossman, and Karin Coninx. 2009. Multimodal selection techniques for dense and occluded 3D virtual environments. *International Journal of Human-Computer Studies* 67, 3 (2009), 237–255.

[8] Uta Wagner, Mathias N Lystbæk, Pavel Manakhov, Jens Emil Sloth Grønbæk, Ken Pfeuffer, and Hans Gellersen. 2023. A fitts' law study of gaze-hand alignment for selection in 3d user interfaces. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15.

[9] Curtis Wilkes and Doug A Bowman. 2008. Advantages of velocity-based scaling for distant 3D manipulation. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*. 23–29.

[10] Dennis Wolf, Jan Gugenheimer, Marco Combosch, and Enrico Rukzio. 2020. Understanding the heisenberg effect of spatial interaction: A selection induced error for spatially tracked input devices. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–10.

[11] Difeng Yu, Qiushi Zhou, Joshua Newn, Tilman Dingler, Eduardo Velloso, and Jorge Goncalves. 2020. Fully-occluded target selection in virtual reality. *IEEE transactions on visualization and computer graphics* 26, 12 (2020), 3402–3413.