BACHELORARBEIT

REMIRE: A MIXED REALITY PLATFORM FOR BCI-BASED MOTOR REHABILITATION

Verfasser

Vladyslav Tsalko

angestrebter akademischer Grad

Bachelor of Science (BSc)

Wien, 2025

Studienkennzahl lt. Studienblatt:	A 12042414
Fachrichtung:	Informatik - Allgemein
Betreuerin / Betreuer:	Univ.-Prof. Dipl.-Ing. Dr. -Ing. Moritz Grosse-Wentrup

Contents

Abstract

I present REMIRE (REhabilitation in MIxed REality), an enhanced mixed reality platform for upper limb rehabilitation, building upon the original REVIRE system. REMIRE transitions rehabilitation tasks from fully virtual environments into a mixed reality setting, allowing users to interact with both virtual and real-world objects in a natural and intuitive manner. The system retains core features such as task performance tracking, while introducing improved user interfaces, refined interaction design, and greater functional flexibility. By anchoring exercises to real-world surfaces like tables and enabling direct hand-based manipulation, REMIRE increases accessibility of rehabilitation training. These improvements aim to provide a more engaging, user-friendly, and realistic experience for patients, paving the way for future studies in clinical and home-based neurorehabilitation scenarios.

1 Motivation

1.1 Why is mixed reality beneficial for stroke rehabilitation?

Mixed reality allows users to stay grounded in their real physical environment while interacting with digital content. Unlike virtual reality, where users are fully immersed in a simulated space and disconnected from their surroundings, mixed reality overlays virtual objects onto the user's actual environment.

In mixed reality, users can interact with virtual objects placed on real tables in their own space. This makes tasks feel more natural and realistic, reducing cognitive load and improving spatial orientation. Because users remain aware of their real environment, they are less likely to feel disoriented or detached, which can often happen in fully virtual environments where the brain perceives the scene as a game or simulation.

By anchoring tasks in the user's real-world context, mixed reality increases engagement and focus. Users are more likely to perform tasks correctly and with intention, as the interactions feel meaningful and directly connected to their physical space. This realism can be especially important in stroke rehabilitation, where motor learning and spatial awareness are critical.

1.2 Limitations of REVIRE that needed to be improved

The original REVIRE platform had technical and design limitations that impacted performance, usability, and extensibility:

1. **Lack of Configurability**: Tasks could not be easily adjusted individually in terms of difficulty or time constraints using UI.
2. **Performance Issues from Inefficient Models**: Key 3D models, such as bottles and glasses, were overly complex. For example, the bottle model contained nested geometry—essentially a bottle inside a bottle—resulting in unnecessarily high polygon counts (up to ~20,000 polygons per object instead of ~1,000). This significantly degraded performance and led to frame rate drops.
3. **No Visual Boundaries for Task Area**: The system lacked a way to visually indicate the maximum allowed interaction area. When users placed their hands on the virtual table to define the interaction zone, there was no feedback or confirmation that their input was registered or saved correctly.
4. **Rigid and Duplicated Codebase**: Adding new tasks was unnecessarily difficult due to poor architectural design. Each task was fully implemented in a separate class, often duplicating logic and increasing maintenance complexity. Task-specific objects were also duplicated unnecessarily, increasing the number of objects in the scene.
5. **Unnatural Interaction Mechanics**: Object interaction was unintuitive and uncomfortable even for healthy users, let alone individuals recovering from a stroke. The lack of realistic, responsive interaction significantly limited the platform's effectiveness for rehabilitation.
6. **Poor Expandability**: Due to the monolithic and rigid structure of the project, integrating new tasks or features was difficult and error prone. This severely limited the platform's scalability and adaptability to new rehabilitation scenarios.

1.3 Goals of REMIRE

The primary goal of REMIRE is to create an easy-to-use, mixed reality platform for individuals undergoing upper limb rehabilitation after a stroke. The system aims to provide realistic, intuitive interactions with virtual objects anchored in the user's real environment, reducing cognitive load and increasing the relevance of each task. By prioritizing usability, realism, and adaptability, REMIRE seeks to support effective rehabilitation at home or in clinical settings.

2 Related Work

2.1 Foundational Work

The application developed in this project builds upon the existing virtual reality platform REVIRE, originally developed by the Research Group Neuroinformatics at the University of Vienna. The foundational work for this project is the paper "REVIRE: A Virtual Reality Platform for BCI-Based Motor Rehabilitation" by L. Mihic Zidar et al. [1]

This paper presents a VR-based rehabilitation system designed to support upper limb motor recovery after stroke through immersive, task-specific exercises combined with EEG recording. The platform incorporates full hand tracking and provides a realistic home-like environment where users complete everyday motor tasks such as pouring, drinking, and cube moving. Data on hand trajectories and EEG were collected to study motor learning, with results showing improvements in task performance and EEG patterns over time. These findings demonstrated the platform’s feasibility for BCI-based rehabilitation both inside and outside clinical settings.

3 Algorithm / Design / Major Idea

3.1 Justification of Methodology / Design

The design of REMIRE was guided by both user needs in post-stroke rehabilitation and technical limitations observed in the original REVIRE system (see Section 1). The key design goals were:

- Grounding interactions in reality through MR by placing virtual objects directly on the user’s physical table.
- Reducing cognitive load by keeping users oriented in their real environment, improving focus and safety.
- Increasing flexibility and scalability with an improved architecture that allows easier task creation and customization.
- Increasing the realism of interactions with objects by reworking the interaction logic so that it seems intuitive and natural.

- These goals were implemented using Meta Quest 3 hardware, Meta XR SDK, and Unity as the development engine.

3.2 Alternative Approaches Considered

3.2.1 Hardware

During the planning phase, we considered developing REMIRE for the **Apple Vision Pro** due to its advanced spatial computing capabilities, high-resolution displays, and industry-leading hand tracking. However, several critical limitations led us to exclude it from this project. At the time of development, the Vision Pro was not officially supported in Austria, raising concerns about regional availability, support reliability, and potential compatibility issues.

Instead, we chose the **Meta Quest 3**, which—despite its lower resolution and less precise hand tracking—proved to be a more practical platform. It is significantly more affordable, widely available, and backed by an open ecosystem with a large and active developer community.

3.2.2 Software

Unity XR Interaction Toolkit was initially considered as the interaction layer for REMIRE due to its cross-platform compatibility and standardized XR input handling.

The original REVIRE was built specifically for the Meta Quest 2 using the older Oculus Integration SDK, which limited flexibility and extensibility. With REMIRE, our focus shifted to the Meta Quest 3, taking advantage of its improved hardware and mixed reality capabilities. To achieve the highest possible performance and feature depth, I chose to use the **Meta XR SDK** directly. This SDK is tightly integrated with the Meta hardware and provides full access to its native capabilities, ensuring better rendering performance, lower latency in hand tracking, and a more seamless passthrough and scene understanding experience.

3.3 Functionality and Key Design Concepts

3.3.1 Scene understanding & real-world surface detection

REMIRE uses the Mixed Reality Utility Kit (**MRUK**) to enable spatial awareness and anchor virtual objects within the user's real-world environment. This functionality is central to creating context-aware rehabilitation tasks that adapt to the user's physical space.

Before spatial interaction begins, the user must scan their surroundings. This can be done in one of two ways:

- By creating a Space Setup manually through the Meta Quest device settings.
- Or, upon launching the REMIRE app, the user is prompted to either:
    - Create a new space,
    - Edit an existing space,
    - Or continue using a previously saved room scan.

3.3.2 Object Classification and Filtering

- Semantic Label Filtering

After room scanning, MRUK loads all detected objects with semantic labels (e.g., "floor", "wall", "door", "table", etc.). REMIRE filters these results and retains only objects labeled as **tables**, as only they are suitable for upper-limb task placement.

- Surface Thresholding

Not all detected tables are suitable for use. The system filters out tables whose **surface dimensions** fall below a defined threshold. The threshold is derived from the spatial requirements of the largest interactable object used in training (e.g. stairs from Move Cube task). This prevents placement on overly small surfaces.

3.3.3 Hand tracking:

REMIRE uses the **Meta XR Hands API**, which provides access to full skeletal hand data, including finger joints, palm position, and tracking confidence. This data is used to detect user intent and enable natural interactions with virtual objects and UI without the need for controllers.

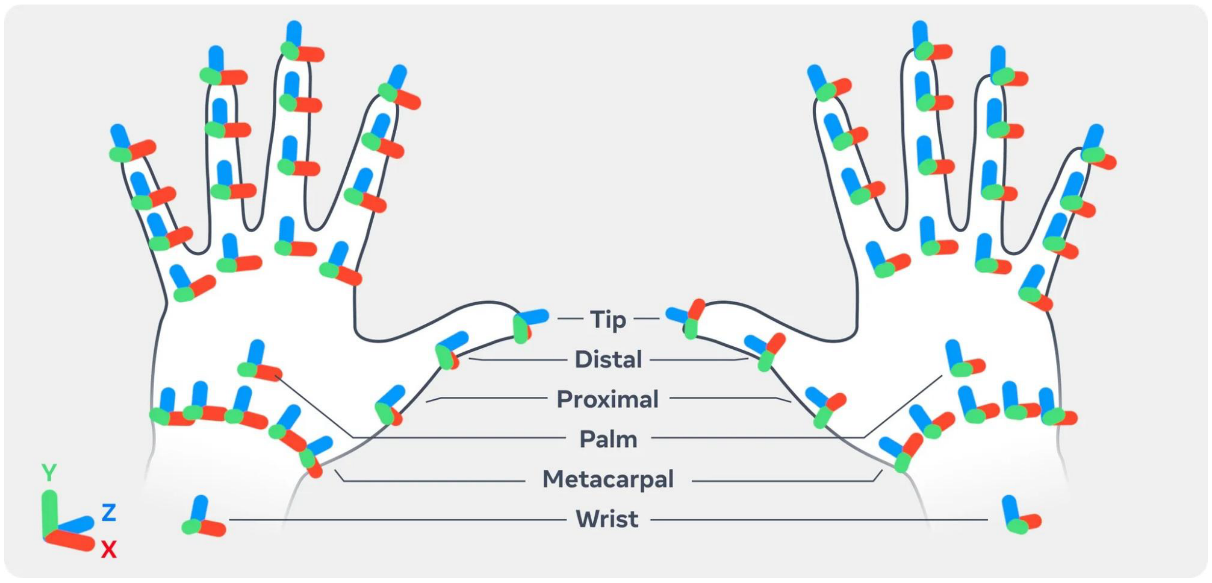

Abbildung 1 OpenXR Hand Skeleton

3.3.4 User-Specific Hand-Based Calibration

Once a valid table is selected, the user is asked to place both hands on the surface. These hand positions define the horizontal task boundary—the maximum area the user can comfortably reach. The system records:

- **Primary boundary**: the dominant-hand side (right hand for right-handed users, or left for left-handed users),
- **Secondary boundary**: the opposite hand,
- And calculates the **center point** between the two hands.

This calibration ensures that all objects are spawned within a space that matches the user's natural reach and the physical constraints of the real-world surface, improving comfort, task realism, and accessibility.

3.3.5 Object placement

To adapt to varying table sizes and user reach, the space between the center between hands and each hand is **divided into three segments**. These segments are used to determine the spawn position of task-related objects based on the selected difficulty level:

- **Easy**: Objects appear in the first third between the center and the corresponding hand.
- **Medium**: Objects appear in the second third of the same span.
- **Hard**: Objects appear exactly at the hand's position — the outermost reachable area.

This segmentation ensures that the task layout is always **personalized to the user's reach** and **proportional to the physical table**. It eliminates the need for fixed coordinates or hardcoded distances and enables scalable difficulty that adapts dynamically to each user's setup.

The placement logic guarantees that all spawned objects are both reachable and challenging in a controlled way, regardless of table size or user arm span, which is especially critical in rehabilitation scenarios where physical capabilities may vary widely.

Before any object is spawned, two additional safety checks are performed:

1. **Table Edge Constraint**
   The system calculates whether the selected spawn point would cause the object—or any part of it—to hang off the table (especially for larger objects like stairs or cubes). If this would happen, the object's position is automatically adjusted by the necessary offset to ensure that it is fully placed on the table surface.
2. **Hand Collision Check**
   To avoid spawning objects directly inside the user's hand (which can happen if the hand remains within the spawn area), the system checks whether the hand overlaps with the intended spawn zone. If it does, the object will not appear until the hand is moved out of the way. This prevents discomfort and ensures a smooth, realistic spawning experience.

These additional adjustments contribute to a safer, cleaner, and more consistent interaction space—crucial for users who may have limited precision or slower reaction times during post-stroke rehabilitation.

3.3.6 User Interaction Design

3.3.6.1 UI Interaction

The system allows users to interact with on-screen UI elements using natural hand gestures:

- **Pointing**: Users can navigate the interface by simply pointing at UI panels with their hands.
- **Pinching**: Selection is confirmed by pinching the thumb and index finger together, mimicking a click gesture. This gesture is detected using the Meta XR Hands API, providing a familiar and low-effort interaction mechanism.

3.3.6.2 UI Positioning and Camera Following

To maintain accessibility and prevent the UI from being lost when the user moves or rotates:

- The UI **follows the camera at a fixed distance** based on the canvas.

- It **updates its position only when the user rotates their head beyond a certain threshold angle**. This prevents constant UI repositioning, which could otherwise be distracting or irritating during rehabilitation exercises.
- This approach ensures that critical information (e.g., session progress, instructions, settings) remains easily visible without dominating the user's view or disrupting immersion

3.3.6.3 Object interaction

In the original REVIRE implementation, object interaction relied on scripts attached to each hand. A combination of thumb contact and any other finger triggered object pickup. The release occurred when the distance between the fingers exceeded a predefined threshold. For two-handed interactions, a separate script managed dual-hand logic. This setup led to a fragmented architecture with multiple classes—some responsible for single-hand grabbing and others for two-hand grabbing—with a shared variable determining which one should be active at runtime. This approach was rigid and hard to scale or customize for new objects or rules.

In REMIRE, this logic was reworked completely. Instead of each **hand deciding** whether it is grabbing an object, now each **object decides** whether it is being grabbed. This inversion of responsibility leads to a cleaner and more scalable architecture.

A dedicated GrabRules class was introduced to define how each object in REMIRE can be grabbed. This class encapsulates the rules an object uses to determine whether it should enter or exit the "held" state. Instead of attaching separate grabbing logic to each hand, all interaction logic is now object-driven and centralized through GrabRules.

Each interactable object contains its own instance of GrabRules, which defines:

- **Required Fingers**
  Required fingers are defined using an enum with bitwise flags, allowing for efficient and flexible combination checks. Each finger (e.g., Thumb, Index, Middle, etc.) is represented as a unique bit, enabling fast bitwise comparisons between the fingers currently touching the object and the rule's required configuration. This approach results in compact rule definitions and rapid evaluation of grab conditions, while also providing precise control over which finger combinations are valid for interaction.
- **Grab Type**
  The GrabType property defines the logic used to evaluate whether the current set of touching fingers satisfies the rule. The following grab types are supported:
  - **Contains**:
    The user must be touching the object with **at least all required fingers**, but may also be using additional fingers.
    Example: Grab with thumb and index — allowed even if middle finger is also touching.
  - **AnyWithMain**:
    Requires one **main finger** (typically the thumb) from a defined list plus **any one**

    additional finger.
    Example: Grab with thumb + any of other fingers
  - **Any**:
    Any **one or more** fingers from a given list is enough to satisfy the rule.
    Example: Grab is allowed if **any** of the [index, middle, ring] fingers touch.
  - **ExactMatch**:
    The object is only grabbed if **exactly** the required finger combination is used — no more, no less.
    Example: Only thumb + index; adding another finger will break the grab.

This rule-based approach allows objects to be picked up, held, or released based on logical conditions rather than fixed input classes per hand. For example:

- A glass can be picked up with the thumb and index finger and continue to be held if the user switches to thumb and middle finger—**as long as a valid combination is maintained**.
- If the object requires two hands, it checks whether its defined rules are satisfied simultaneously for both hands.
- If no valid rule is satisfied, the object is automatically released.

This decoupled interaction model offers several advantages:

- **Unified grabbing logic** regardless of hand count.
- **Easier customization**: each object can define its own interaction behavior.
- **Dynamic finger switching** without dropping the object.
- **Improved realism**: For instance, if an object like a glass is squeezed too hard (i.e., finger positions move too close to the object's center), the object can be **destroyed**, simulating a shattering event.

3.3.7 Performance and architectural improvements

3.3.7.1 Polygon Optimization

During testing, it was observed that scenes containing the bottle model experienced noticeable performance drops, while those without it ran smoothly. Upon inspection, the bottle model was found to contain over 20,000 polygons due to deeply nested geometry—essentially a "**matryoshka**" structure of bottles within bottles. This unnecessary complexity placed a heavy load on the rendering pipeline.

To address this, the redundant geometry was removed, and only the outermost mesh was preserved. The same optimization process was applied to the glass model, reducing the polygon count from approximately 20,000 to around 1,000. These changes resulted in a measurable increase in frame rate and rendering stability.

3.3.7.2 Task System Refactoring

In the original REVIRE platform, each task had its own dedicated class, often duplicating logic for state management, scoring, object spawning, and task evaluation. This led to redundant code, reduced maintainability, and made it difficult to add new tasks efficiently.

In REMIRE, this was replaced with a unified architecture centered around an abstract base class called **Task**. This class defines shared functionality common to all tasks, such as:

- Task start, pause, and restart behavior,
- General lifecycle management,
- Default scoring system hooks,
- and scene reset or cleanup logic.

Concrete task classes now inherit from Task and override only the necessary methods specific to their functionality. For example:

- **SpawnObjects**() – defines how and where objects spawn,
- **EvaluateTask**() – calculates performance (e.g., amount of water poured, or podest sequence),
- **AreAllObjectsSatisfyConditions**() – checks task-specific success criteria.

This structure ensures all tasks share a consistent lifecycle and logic flow, while still allowing customization where needed. It dramatically improves code clarity, enables easy expansion to new tasks, and eliminates duplication—making REMIRE more scalable and maintainable.

4 Implementation

4.1 Project Setup and Tooling

The REMIRE application was developed using the Unity game engine and deployed on the Meta Quest 3 headset. The setup prioritized native support for mixed reality features such as passthrough, hand tracking, and scene understanding.

- **Engine**: Unity 6000.0.41f1 LTS
  Chosen for its long-term support and stability with XR plugins.
- **Primary SDK**: Meta XR SDK for Unity (v74)
  Used for passthrough rendering, XR Hands API, and headset integration. Provided full access to Meta Quest 3's mixed reality capabilities, including hand skeleton tracking and environmental awareness.
- **Scene Understanding**: Mixed Reality Utility Kit (MRUK)
  Enabled real-world surface detection and semantic labeling of room elements (e.g., tables, floors, walls). Used to filter for valid table surfaces and define the task space.

- **Target Platform**: Meta Quest 3
  The app was developed and optimized specifically for Meta Quest 3, making use of its improved performance and passthrough quality.
- **Build Platform**: Android
  REMIRE was built using Unity's Android build target with XR Plugin Management enabled for Meta Quest.
- **Development Tools**:
  - **Unity Hub** (for managing Unity projects and versions)
  - **JetBrains Rider** (as the primary IDE for writing and debugging C# scripts)
  - **Meta Quest Developer Hub** (for installing .apk on Meta Quest 3 and managing device settings)
  - **Git** (for version control)

This setup ensured that the application could take full advantage of Meta Quest 3's mixed reality capabilities, while maintaining compatibility with Unity's XR development pipeline.

4.2 Code Architecture

The project is structured into logical modules:

- **Scene Setup Module**
  Handles room loading using Meta's scene understanding (via the Mixed Reality Utility Kit). It identifies labeled surfaces (tables) and filters them based on size.
- **Calibration Module**
  Once a valid table is selected, the user places both hands on the surface. Their hand positions are recorded to define task boundaries. These positions are further divided into zones for object spawning based on difficulty.
- **Grab Interaction System**
  Uses the Meta XR Hands API and a custom-built GrabRules system that defines how each object can be grabbed based on finger combinations and hand presence. This allows for dynamic rule validation without relying on hand-specific logic.
- **Task System**
  A new abstract Task base class provides the common functionality for all tasks, including start, pause, reset, and score tracking. New tasks are implemented by inheriting from this base class and overriding only task-specific behavior like object creation or completion conditions.
- **UI Module**

  Provides a streamlined interface to guide the user through setup and session progression. This includes:

  - Real-time display of task instructions and progress
  - Settings menus for selecting difficulty levels and task types
  - Feedback on calibration, table detection, and interaction readiness
  - Summary screen at the end of each task showing session statistics

4.3 How to Use / Reproduce

To run or further develop REMIRE:

1. **Hardware Requirements**
    - Meta Quest 3 with hand tracking enabled
    - A real-world room with at least one table of sufficient size
2. **Room Scanning**
    - Either pre-scan a room using the Quest's Space Setup in settings
    - Or allow REMIRE to prompt room scanning at launch
3. **Build Setup**
    - Unity version compatible with Meta XR SDK – 6000.0.41f1
    - Meta XR SDK + Mixed Reality Utility Kit installed via Unity Package Manager
    - Android platform build settings with Quest 3 configuration
4. **Workflow**
    - On launch, the app guides the user through scene scanning and table selection.
    - The user performs hand calibration by placing both hands on the selected table.
    - Based on difficulty, virtual rehabilitation tasks are spawned within the user's reachable area.
    - Object interaction is managed via hand presence and GrabRules.
5. **Source Code**
    - The full source code and project files are available here.

5 Conclusion and Future Work

5.1 Summary of Findings

REMIRE demonstrates the feasibility and potential of using mixed reality for upper-limb rehabilitation after stroke. By grounding interactions in the user's real environment—specifically, by placing virtual rehabilitation tasks directly on physical tables—the system significantly improves realism, spatial awareness, and user engagement compared to traditional virtual reality approaches.

I re-engineered the architecture of the original REVIRE platform to allow greater flexibility, scalability, and natural interaction. Key contributions include:

- A hand-calibrated interaction zone tailored to the user's reach.
- A custom GrabRules system enabling realistic and modular hand-object interaction.
- A modular task system supporting easy extension of new rehabilitation activities.
- An intuitive user interface for configuration, guidance, and session feedback.

5.2 Limitations of the Approach

Despite improvements, the current REMIRE implementation has several limitations:

- It is restricted to Meta Quest 3 hardware, limiting compatibility with other platforms.
- Hand tracking, while functional, still suffers from occasional instability and misdetections, particularly when fingers are occluded or in poor lighting conditions.
- The tasks remain limited in number and complexity; they may not yet fully cover the variety of motions required for comprehensive rehabilitation.

5.3 Outline of Future Work

5.3.1 Clinical Evaluation

To validate REMIRE's therapeutic value, future work will include studies with real stroke patients. These will help assess usability, comfort, recovery progress, and clinical outcomes in both supervised and home-use settings. Since the predecessor platform, REVIRE, was already evaluated with participants and showed positive trends in performance and neural activity, a future evaluation of REMIRE will allow for direct comparison of outcomes, highlighting the impact of mixed reality and improved interaction design on rehabilitation effectiveness.

5.3.2 Expanded Task Library

To better target a variety of motor functions, new tasks simulating daily activities—like flipping a card or playing a simple version of Jenga—could be introduced. Some tasks could involve hybrid MR + real-object interactions for added realism.

5.3.3 Therapist Dashboard & Remote Monitoring

Developing a secure remote interface for therapists could enable remote supervision, replay of task performances, and real-time feedback. Therapists could adjust settings per session or patient remotely, enabling home-based telerehabilitation.

5.3.4 Enhanced Hand Interaction

To further increase the realism and precision of object manipulation, future versions of REMIRE will integrate the **Meta Interaction toolkit** alongside the existing custom GrabRules system. This hybrid approach will introduce **pose-conditioned grabbing**, adding an additional layer of control over how objects can be picked up and held.

Currently, grabbing is determined solely by touch input from specific fingers, evaluated through bitwise-defined finger rules. While this system provides flexibility, it does not account for the actual shape or pose of the user's hand. By incorporating pose recognition from the Meta XR Interaction SDK, we can define **allowed grab poses** for each object, such as a pinch, cylindrical grip, or flat palm.

Example: A user touching the glass with thumb and index might currently trigger a grab. With pose gating, only a correctly shaped cylindrical grip would allow the glass to be lifted.

6 Bibliography

[1] Mihic Zidar, L., Raggam, P., Mohammadian, F., Barłoga, A., Grosse-Wentrup, M. (2024). REVIRE: A Virtual Reality Platform for BCI-Based Motor Rehabilitation. In Proceedings of the 9th Graz Brain-Computer Interface Conference. Graz, Austria.

7 Appendix

7.1 References

- Meta Developers, "Unity Development Guide."
- Unity: Core SDK v74
- Unity: MR Utility Kit v74

7.2 Use of GPT Model

During the writing phase of this thesis, the GPT language model was utilized for spell checking and rephrasing content to enhance readability and coherence of the text.