People's Democratic Republic of Algeria

UNIVERSITY YAHIA FARES OF MEDEA

Faculty of Science

Department of Mathematics and Computer Science



Master's Thesis

Presented for obtaining the Master's degree

In computer science

Option : Systems Engineering and Web Technology

Realised by :

Touari Hacene & Menasri Salah Eddine

Fully Web and Virtual Reality Based Collaborative System for Controlling Multi Robots Industrial Environment

Dr. Hayi Mohamed Yassine	UYFM	**President**
Dr. Babaali Baligh	UYFM	**Examiner**
Dr. Allal Tiberkak	UYFM	**Supervisor**

2025/2026

Acknowledgements

First and foremost, we thank Allah for granting us the strength, patience, and guidance to accomplish this work. We would like to express our sincere gratitude to all those who supported and guided us throughout the preparation of this dissertation. We hope that this work reflects the quality of the education and training we have received.

We would also like to express our deepest respect and sincere appreciation to our supervisor, Dr. Allal Tiberkak, for his continuous support and assistance, his valuable guidance and advice, and for the knowledge he shared with us throughout the development of this work.

Touari Hacene & Menasri Salah Eddine

Table of Contents

Acknowledgements **i**

Abstract **i**

1 General Introduction **1**
1.1 Problem Statement . . . 1
1.2 Methodological Approach . . . 1
1.3 Thesis Outline . . . 2

2 Fundamental Concepts and Theoretical Foundations **3**
2.1 Introduction . . . 3
2.2 Robot Systems : Architecture and Classification . . . 3
2.2.1 Control Architectures . . . 3
2.2.2 Industrial Robot Systems . . . 4
2.3 Virtual Reality : Foundations and Industrial Applications . . . 4
2.3.1 Definition and Historical Context . . . 4
2.3.2 Three Pillars of Virtual Reality . . . 4
2.3.3 VR in Industrial and Robotic Applications . . . 5
2.4 VR Physical Engineering for Robotic Systems . . . 5
2.4.1 Physics Simulation in VR . . . 5
2.4.2 Haptic and Multimodal Feedback . . . 6
2.4.3 Engineering Challenges . . . 6
2.5 Conclusion . . . 7

3 State of the Art **8**
3.1 Related Work . . . 8
3.1.1 VR Teleoperation for Mobile Robots . . . 8
3.1.2 Digital Twin and VR for Collaborative Assembly . . . 9
3.1.3 Web-Based XR Framework for Robotic Training . . . 10
3.1.4 Web-Based Multi-Robot System for the Industrial Metaverse . . . 11
3.1.5 Lightweight WebGL Simulation with Three.js . . . 12
3.2 Comparative Analysis . . . 13
3.2.1 Comparative Synthesis . . . 13
3.2.2 Discussion . . . 15

3.3 Conclusion 16

4 Proposed System **17**
4.1 Introduction 17
4.2 Objectives 17
4.2.1 Research Questions 18
4.2.2 Research Hypotheses 18
4.3 System Overview 19
4.4 Robot Modeling 19
4.4.1 Problem Statement 20
4.4.2 Industrial Robot Types 20
4.5 Robot Description 22
4.5.1 Robot Body Architecture 23
4.5.2 Grasping System 24
4.5.3 Sensors and Perception 25
4.5.4 Dynamics and Center of Gravity 26
4.5.5 Motion Control 26
4.5.6 Functional Safety 27
4.6 Event Processing System 28
4.6.1 Event System Architecture 28
4.6.2 Event Execution Flow 28
4.7 Robotic Model Construction 29
4.7.1 Robot Configuration 30
4.7.2 Physical Model Generation 31
4.7.3 Visual Model Construction 32
4.7.4 Event System Binding 34
4.7.5 Robot Initialization Pipeline 35
4.8 Conclusion 35

5 Implementation and Deployment **36**
5.1 Introduction 36
5.2 Implementation 36
5.2.1 Hardware Environment 36
5.2.2 Software Environment 36
5.2.3 Programming Languages 37
5.2.4 Libraries and Frameworks 37
5.3 Implementation Environment 37
5.3.1 Deployment Architecture 37
5.3.2 System Configuration 38
5.3.3 Launch Procedure 40
5.3.4 Web and VR/AR Deployment 40
5.4 Testing 41
5.4.1 Manual Tests 41

5.4.2 Automated API Tests 41
5.4.3 Programmatic Tests 42
5.5 Evaluation and Results 43
5.5.1 Functional Tests 43
5.5.2 Collision and Safety Tests 45
5.5.3 Grasping System Evaluation 47
5.5.4 Multi-User Synchronization Tests 49
5.5.5 Performance Evaluation 50
5.5.6 VR/AR Interaction Tests 51
5.5.7 Vision System Performance Evaluation 51
5.6 Conclusion 53

6 General Conclusion and Future Work **54**
6.1 Summary of Contributions 54
6.2 Experimental Results 54
6.3 Limitations 54
6.4 Perspectives 55
6.4.1 Short-Term 55
6.4.2 Medium-Term 55
6.4.3 Long-Term Perspectives 55

List of Figures

3.1 Geometrical parameters of the KUKA youBot arm 9
3.2 Information synchronization and transfer between the system components [1]. 10
3.3 Web-Based XR Framework for Robotic Training [2] 11
3.4 WEB concept industrial metaverse [3]. 11
3.5 Three-dimensional simulation model of a 5-DOF manipulator [4]. 12

4.1 Overview of the web-based collaborative robotic simulation platform 19
4.2 Comparison of common industrial robot architectures. 21
4.3 Robot Description. 23
4.4 Event execution flow from perception to action. 29
4.5 Robotic model construction pipeline from JSON configuration to functional robot. 30

5.1 Multi-robot collaborative warehouse environment. 43
5.2 Detected and resolved collision types. 46
5.3 Cobot Emergency Stop (E-Stop) 47
5.4 Grasping test success rates. 48
5.5 WebSocket latency vs. simultaneous users. 49
5.6 Performance evaluation (FPS and physics). 50
5.7 Users Operating in VR Mode 51

List of Tables

3.1 Comparison of Related Works 13
3.2 Gap mapping : identified limitations in the literature versus responses provided. 14

4.1 Comparative overview of industrial robot types. 20
4.2 Joint angular limits. 24
4.3 Force formulas for the grasping model. 25
4.4 Tactile sensor point distribution. 25
4.5 Dynamic load factor according to transported mass. 26
4.6 S-Curve joint motion parameters. 27
4.7 Safety modes for the collaborative robot. 27
4.8 Event system component responsibilities. 28
4.9 Robot configuration structure. 31
4.10 Comparison of industrial and cobot presets. 31
4.11 Physical body properties. 32
4.12 Hinge joint configuration. 32
4.13 Visual component colors. 33
4.14 Event system communication matrix. 34

5.1 Hardware Configuration 36
5.2 Software Stack 37
5.3 Core Libraries and Frameworks 37
5.4 Keyboard Controls 41
5.5 Complete System API Reference 42
5.6 Low-Level API for Programmatic Tests 42
5.7 Base movement test results. 44
5.8 Joint control test results. 44
5.9 Grasping test results. 44
5.10 Navigation test results. 45
5.11 Pick-and-Place phase results. 45
5.12 Self-collision test results. 46
5.13 Floor collision test results. 46
5.14 Inter-robot collision test results. 47
5.15 E-Stop comparison : industrial vs. cobot. 47
5.16 Centered grasp results. 48

5.17 Edge grasp results by offset. 48
5.18 Heavy load test results. 49
5.19 Grasping stability test results. 49
5.20 Multi-user synchronization results. 50
5.21 Performance metrics by robot count. 50
5.22 XR interaction test results. 51
5.23 Vision system detection and tracking metrics. 52

List of Abbreviations

Abbrev.	Full Term	Abbrev.	Full Term
AABB	Axis-Aligned Bounding Box	MES	Manufacturing Execution System
AGV	Automated Guided Vehicle	MQTT	Message Queuing Telemetry Transport
AMR	Autonomous Mobile Robot	OPC-UA	Open Platform Communications Unified Architecture
AR	Augmented Reality	PD	Proportional-Derivative Control
AV	Augmented Virtuality	ROS	Robot Operating System
COG	Center of Gravity	RRT	Rapidly-exploring Random Tree
CPS	Cyber-Physical System	SAT	Separating Axis Theorem
DOF	Degrees of Freedom	SPA	Single Page Application
E-Stop	Emergency Stop	SSL	Secure Sockets Layer
FPS	Frames Per Second	TCP	Tool Center Point
FSM	Finite State Machine	VR	Virtual Reality
GJK	Gilbert-Johnson-Keerthi Algorithm	WSS	WebSocket Secure
HMI	Human-Machine Interface	XR	Extended Reality
IFR	International Federation of Robotics		
IIoT	Industrial Internet of Things		
ISO	International Organization for Standardization		

Abstract

Industry 4.0 is accelerating the integration of robotic systems into industrial environments, yet direct interaction between novice workers and robots raises significant safety concerns, making robotic simulation an essential training tool prior to real deployment. Existing platforms, however, face a persistent trade-off : browser-based solutions offer accessibility but sacrifice physical realism, while high-fidelity simulators require complex local installations and substantial computing resources. This thesis presents the design, implementation, and validation of a fully browser-based collaborative robotic simulation platform that bridges this gap, built on a five-layer modular architecture integrating the *Cannon.js* physics engine and the *Three.js* rendering library to deliver 60 Hz rigid-body simulation — including gravity, collision detection, and joint constraint solving — directly in a standard web browser with no installation required, complemented by an integrated dual-camera perception system that provides real-time object tracking and proximity-based collision avoidance via Frustum culling. Experimental validation using a three-tier strategy confirms a 90% success rate for autonomous Pick & Place cycles, collision resolution rates between 95% and 100%, multi-user WebSocket synchronisation at 28 ms average latency for four concurrent participants, and a rendering performance of 54 FPS for four simultaneous robots ; these results were obtained under a limited experimental scope, restricted to box-shaped objects within a single warehouse environment and evaluated on CPU-only hardware, which future work should extend to irregular objects, diverse industrial settings, and GPU-accelerated configurations. Overall, this work establishes the feasibility of combining web accessibility, real-time physics, intelligent grasping, and collaborative multi-robot interaction within a single browser-based platform, offering a validated and reproducible foundation for the development of next-generation Industry 5.0 collaborative robotic systems.

Keywords : Industry 4.0, Collaborative Multi-Robot Environments, Human–Robot Collaboration, Web-Based Robot Modeling, 3D Physics Engine, Virtual Reality (VR), Augmented Reality (AR).

Résumé

L'Industrie 4.0 accélère l'intégration des systèmes robotiques dans les environnements industriels, mais l'interaction directe entre des travailleurs novices et des robots soulève d'importantes préoccupations en matière de sécurité, faisant de la simulation robotique un outil de formation essentiel avant tout déploiement réel. Les plateformes existantes, cependant, font face à un compromis persistant : les solutions basées sur navigateur offrent l'accessibilité mais sacrifient le réalisme physique, tandis que les simulateurs haute fidélité nécessitent des installations locales complexes et des ressources informatiques conséquentes. Ce mémoire présente la conception, l'implémentation et la validation d'une plateforme de simulation robotique collaborative entièrement basée sur navigateur, comblant ainsi ce fossé grâce à une architecture modulaire à cinq couches intégrant le moteur physique *Cannon.js* et la bibliothèque de rendu *Three.js*, assurant une simulation de corps rigides à 60 Hz incluant la gravité, la détection de collisions et la résolution des contraintes articulaires directement dans un navigateur standard sans aucune installation requise, complétée par un système de perception à double caméra fournissant un suivi des objets en temps réel et un évitement des collisions basé sur la proximité via le *Frustum Culling*. La validation expérimentale selon une stratégie à trois niveaux confirme un taux de réussite de 90 % pour les cycles autonomes de *Pick & Place*, des taux de résolution des collisions entre 95 % et 100 %, une synchronisation WebSocket multi-utilisateurs à 28 ms de latence moyenne pour quatre participants simultanés, et une performance de rendu de 54 FPS pour quatre robots simultanés ; ces résultats ont toutefois été obtenus dans un cadre expérimental restreint, limité à des objets de forme parallélépipédique au sein d'un environnement d'entrepôt unique et évalué sur une architecture CPU uniquement, que les travaux futurs devront étendre à des objets de formes irrégulières, à des environnements industriels variés et à des configurations accélérées par GPU. Dans l'ensemble, ce travail établit la faisabilité de combiner accessibilité web, physique temps réel, préhension intelligente et interaction collaborative multi-robots au sein d'une plateforme unique basée sur navigateur, offrant une base validée et reproductible pour le développement des futurs systèmes robotiques collaboratifs de l'Industrie 5.0.

Mots-clés : Industrie 4.0, environnements multi-robots collaboratifs, collaboration homme-robot, modélisation de robots sur le Web, moteur physique 3D, réalité virtuelle (RV), réalité augmentée (RA).

ملخص

تُسرِّع الصناعة 4.0 دمج الأنظمة الروبوتية في البيئات الصناعية، غير أن التفاعل المباشر بين العمال المبتدئين والروبوتات يُثير مخاوف جدية تتعلق بالسلامة، مما جعل محاكاة الروبوتات أداةً تدريبية لا غنى عنها قبل أي نشر فعلي. إلا أن المنصات القائمة تواجه مفاضلة دائمة : فالحلول المعتمدة على المتصفح توفر إمكانية الوصول لكنها تضحي بالواقعية الفيزيائية، في حين تستلزم منصات المحاكاة عالية الدقة تثبيتات محلية معقدة وموارد حاسوبية كبيرة. تقدم هذه المذكرة تصميم وتطوير والتحقق من منصة محاكاة روبوتية تعاونية تعمل بالكامل داخل المتصفح، سادَّةً هذه الفجوة بفضل معمارية معيارية خماسية الطبقات تدمج محرك الفيزياء *Cannon.js* ومكتبة التصيير *Three.js*، وتوفر محاكاة للأجسام الصلبة بتردد 60 هرتز — تشمل الجاذبية واكتشاف التصادمات وحل قيود المفاصل — مباشرةً داخل متصفح قياسي دون أي تثبيت مسبق، مدعومةً بوحدة إدراك بصري ثنائية الكاميرا تتيح تتبع الأجسام في الزمن الحقيقي وتجنب التصادمات باستخدام تقنية *Culling Frustum*. وقد أكّد التحقق التجريبي وفق استراتيجية ثلاثية المستويات معدل نجاح 90% لدورات الالتقاط والوضع الذاتية (*Place & Pick*)، ومعدلات معالجة للتصادمات بين 95% و100%، ومزامنة WebSocket متعددة المستخدمين بمتوسط زمن استجابة 28 ميلي ثانية لأربعة مستخدمين متزامنين، وأداء تصيير بلغ 54 إطاراً في الثانية مع أربعة روبوتات تعمل في آنٍ واحد؛ غير أن هذه النتائج تحققت ضمن نطاق تجريبي محدود، اقتصر على أجسام متوازية المستطيلات ضمن بيئة مستودع واحدة واعتمد على معالج CPU دون وحدة معالجة رسومية GPU، وهو ما ينبغي للأعمال المستقبلية توسيعه ليشمل أجساماً غير منتظمة الشكل وبيئات صناعية متنوعة وتهيئات مسرَّعة بواسطة GPU. وعموماً، يُثبت هذا العمل إمكانية الجمع بين سهولة الوصول عبر الويب والمحاكاة الفيزيائية في الزمن الحقيقي والإمساك الذكي والتفاعل التعاوني متعدد الروبوتات ضمن منصة واحدة قائمة على المتصفح، مُقدِّماً أساساً موثَّقاً وقابلاً لإعادة الإنتاج لتطوير أنظمة روبوتية تعاونية مستقبلية وفق متطلبات الصناعة 5.0.

الكلمات المفتاحية : الصناعة 4.0، بيئات الروبوتات المتعددة التعاونية، التعاون بين الإنسان والروبوت، نمذجة الروبوتات عبر الويب، محرك الفيزياء ثلاثي الأبعاد، الواقع الافتراضي (VR)، الواقع المعزز (AR).

Chapter 1

General Introduction

The manufacturing industry is undergoing a profound transformation, driven by the successive paradigms of Industry 4.0 and Industry 5.0. While the former emphasized massive automation and the interconnection of cyber-physical systems, the latter places the human at the center of the production process, highlighting the need for fluid, safe, and intuitive collaboration between operators and robots. In this context, cobots (collaborative robots) and immersive Virtual Reality (VR) interfaces constitute essential technological levers for teleoperation, risk-free training, and multi-site supervision.

In parallel, the emergence of modern web technologies — WebGL, WebXR, WebAssembly, and WebSocket — has opened the way to industrial applications accessible directly from a browser, without heavy installation or specific hardware dependencies. However, a gap persists between accessibility and physical realism : existing web simulators often rely on simplified kinematic visualizations, while physically realistic platforms (Gazebo, ROS) require local infrastructure and advanced technical skills. To date, few existing solutions combine, in a unified and web-native manner : real-time physics simulation, realistic grasping model, normative compliance (ISO 10218, ISO/TS 15066), multi-robot coordination, multi-user synchronization, and XR immersion.

1.1 Problem Statement

The primary challenge lies in designing a fully web-based collaborative robotics platform that accurately models complex physical behavior such as collisions, force-regulated grasping, and dynamic mass distribution—while facilitating real-time multi-user collaboration and multi-robot coordination. To overcome this, this work aims to develop a unified simulator integrating a browser-native physics engine (Cannon-es) with a modular architecture that supports both industrial and cobot profiles under strict safety standards. The proposed solution further enhances operational efficiency through an event-driven system for autonomous planning and ensures seamless collaboration via WebSocket synchronization and immersive WebXR (VR/AR) environments.

1.2 Methodological Approach

The approach relies on a Three.js / Cannon-es / Node.js technology stack, orchestrated according to a five-layer architecture (Configuration, Decision, Physics/Sensors, Grasping, Interface/Te-

lemetry). The development employs S-Curve motion profiles for kinematics, a hierarchical detection algorithm (AABB/Box3) for collision management, and a lightweight WebSocket protocol for shared state.

1.3 Thesis Outline

The rest of this thesis is organized into five chapters as follows :

Chapter 1 — General Introduction. This chapter introduces the context and motivation of the study, presents the research problem, objectives, methodology, and outlines the overall structure of the thesis.

Chapter 2 — Theoretical Foundations. This chapter lays the theoretical foundations of our work : robotic control architectures, the pillars of virtual reality, and the challenges of immersive physical engineering for robotic systems.

Chapter 3 — State of the Art. The third chapter analyzes the state of the art through five major contributions from recent literature and identifies eight scientific and technical gaps that our system aims to fill.

Chapter 4 — System Design and Architecture. The fourth chapter presents the detailed design of the proposed system : robotic modeling, intelligent grasping system, event-driven architecture, and model construction pipeline.

Chapter 5 — Implementation and Validation. The fifth chapter covers the technical implementation, web and XR deployment, as well as the experimental validation campaign comprising functional, safety, performance, and multi-user tests, accompanied by a critical analysis of the results obtained.

Finally, a general conclusion synthesizes the contributions of this work, discusses its limitations, and proposes perspectives for future research.

Chapter 2

Fundamental Concepts and Theoretical Foundations

2.1 Introduction

As industrial robotics grows in complexity, traditional 2D control interfaces increasingly fail to meet the demands of real-time supervision and remote manipulation. Virtual Reality offers an immersive alternative that enhances spatial awareness and reduces operator cognitive load. This chapter establishes the theoretical foundations of the present work, covering robot system architectures, the core properties of VR, and the principles of VR Physical Engineering, along with the key challenges motivating subsequent developments.

2.2 Robot Systems : Architecture and Classification

A robot system is an integrated ensemble of mechanical, electronic, and computational components engineered to perceive its environment through sensor modules, process acquired data via an embedded or remote control unit, and execute physical actions through actuator mechanisms, in order to perform predefined or adaptive tasks either autonomously or under varying degrees of human supervision. According to the International Federation of Robotics (IFR) [5], a robot is defined as an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which can be either fixed in place or mobile. This definition underscores the dual nature of modern robot systems : their physical embodiment and their programmable intelligence.

2.2.1 Control Architectures

Robot systems can be governed by three primary control architectures, each presenting distinct trade-offs in terms of scalability, fault tolerance, and response latency :

1. **Centralized Architecture :** A single master controller manages all robotic agents. This approach simplifies global planning and coordination but introduces a single point of failure and limits scalability in large industrial deployments.

2. **Decentralized Architecture :** Each robot operates with its own local controller, making decisions based on local sensor data and peer-to-peer communication. This model offers high fault tolerance and scalability, but can lead to suboptimal global solutions due to lack of global situational awareness.

3. **Hybrid Architecture :** Combines centralized planning at the supervisory level with decentralized execution at the robot level. This is the most widely adopted model in modern industrial multi-robot systems, balancing responsiveness with coordination efficiency.

2.2.2 Industrial Robot Systems

In an industrial context, a robot system is formally defined as a cyber-physical system (CPS) comprising one or more robotic manipulators or mobile platforms, associated motion controllers, the Human-Machine Interface (HMI), safety systems, and the network infrastructure enabling integration into a broader manufacturing execution environment. Such systems are designed to perform repetitive, precise, or hazardous operations—including welding, assembly, painting, material handling, and quality inspection—with high reliability, throughput, and repeatability [6, 7].

2.3 Virtual Reality : Foundations and Industrial Applications

2.3.1 Definition and Historical Context

Virtual Reality (VR) is an immersive, computer-generated simulation of a three-dimensional environment that allows users to interact with a synthetic world in real time through a combination of sensory modalities, including visual, auditory, and haptic feedback. VR systems create a sense of presence and immersion by rendering stereoscopic imagery at high frame rates (typically 90 Hz or above) and tracking the user's head orientation and physical movements to update the displayed scene accordingly. The foundational concept of VR was formalized by Jaron Lanier in the late 1980s [8], who introduced the term to describe computer-generated, interactive 3D environments experienced through specialized hardware. Since then, VR has evolved from laboratory prototypes into a mature technology deployed across domains including gaming, medical training, architectural visualization, and increasingly, industrial robotics [9]..

2.3.2 Three Pillars of Virtual Reality

The academic literature consistently identifies three core properties that define a true VR experience [8] often referred to as the VR triad :

- **Immersion :** The degree to which the VR system replaces real-world sensory input with computer-generated stimuli. Immersion is a technical property of the hardware : a wide field of view (FOV $>90°$), high display resolution, and low-latency tracking contribute to deeper immersion.

- **Presence :** The subjective psychological sensation of "being there"—the user's feeling that they genuinely occupy and interact with the virtual environment. Presence is the ultimate goal of VR design and is influenced by immersion quality, interaction fidelity, and narrative coherence.

- **Interaction :** The capacity for the user to act upon and manipulate the virtual environment in a meaningful way. In robotic applications, interaction encompasses gesture-based control, gaze-directed commands, and haptic manipulation of virtual representations of physical robotic systems.

2.3.3 VR in Industrial and Robotic Applications

In the context of industrial robotics and remote operation, Virtual Reality serves as a critical Human-Machine Interface (HMI) layer that bridges the gap between human operators and physical robotic systems. Its applications in this domain include :

- **Robot Teleoperation :** VR enables operators to control remote or hazardous robotic systems by mapping their own movements onto robot actions within an immersive interface, significantly reducing cognitive load and improving spatial awareness compared to traditional 2D screen-based interfaces [10]..
- **Digital Twin Visualization :** A VR-rendered digital twin provides a real-time, photorealistic 3D model of the physical robot and its environment, synchronized via sensor data streams. This allows operators to monitor robot states, detect anomalies, and simulate interventions without physical access [11, 12]..
- **Training and Simulation :** VR environments allow operators and engineers to train on complex robotic systems, test new programs, and validate safety procedures in a risk-free virtual space before deployment on the physical system.
- **Collaborative Remote Supervision :** Multi-user VR platforms allow geographically distributed teams to co-inhabit a shared virtual workspace, observe the same robot state representations, and collaboratively issue or review control commands [13]..

2.4 VR Physical Engineering for Robotic Systems

VR Physical Engineering refers to the interdisciplinary domain that integrates real-time physics simulation, haptic feedback rendering, and sensorimotor coupling within virtual environments for robotic applications. Unlike conventional VR systems that rely solely on kinematic visualization, VR Physical Engineering embeds rigid-body dynamics, contact mechanics, and force-control algorithms directly into the simulation loop. This convergence enables the virtual robot to exhibit physically plausible behaviors—including collision response, grasp force modulation, and payload dynamics—that are essential for credible industrial training and teleoperation.

2.4.1 Physics Simulation in VR

The physical fidelity of a VR-based robotic simulator is determined by the underlying physics engine. Modern engines such as Cannon-js, Ammo.js, and PhysX compute, at each simulation step ($\Delta t \approx 16.7$ ms for 60 Hz), the following quantities :

- **Rigid-Body Dynamics :** Integration of linear and angular momentum under gravity and external forces.

- **Collision Detection :** Broad-phase (AABB trees) and narrow-phase (SAT/GJK) algorithms to identify contacting geometries.
- **Constraint Solving :** Lagrange-multiplier or impulse-based methods to enforce joint limits, contact non-penetration, and friction cones.

In the context of industrial robotics, the physics engine must additionally support :

1. **Custom Contact Materials :** Per-object friction coefficients (μ) and restitution (e) calibrated against ISO 10218 [14] safety data sheets.
2. **Force-Regulated Actuation :** Proportional-Derivative (PD) controllers that map desired grip forces to joint torques, rather than teleporting objects kinematically.
3. **Dynamic Center of Gravity (COG) :** Real-time recomputation of the robot's aggregate COG when a payload is grasped, ensuring stability during transport.

2.4.2 Haptic and Multimodal Feedback

While visual immersion dominates consumer VR, industrial applications demand multimodal feedback :

- **Haptic Rendering :** Force-feedback devices (e.g., SensAble Phantom, Haption Virtuose) render contact wrenches computed by the physics engine to the operator's hand, closing the sensorimotor loop.
- **Vibrotactile Cues :** Wearable actuators provide vibration patterns proportional to predicted collision risk, as required by ISO/TS 15066 [15] collaborative safety standards.
- **Spatial Audio :** Contact sounds generated procedurally from impact impulses enhance situational awareness in cluttered industrial scenes.

2.4.3 Engineering Challenges

Integrating physics engines into browser-based VR introduces specific constraints :

1. **Determinism vs. Performance :** JavaScript physics engines sacrifice deterministic floating-point behavior for single-threaded execution ; fixed-time-stepping (1/60 s) is mandatory to avoid tunneling and jitter.
2. **Bandwidth-Limited State Synchronization :** In multi-user VR supervision, physics state vectors (position, quaternion, velocity) must be compressed and interpolated to respect network latency ($<$250 ms for training, $<$100 ms for teleoperation).
3. **Safety-Critical Validation :** Virtual contact forces must be traceable to normative limits (e.g., maximum permissible biomechanical loads defined in ISO/TS 15066 [15], requiring certified metrological pipelines that browser engines currently lack.

2.5 Conclusion

This chapter established the theoretical foundations underlying the integration of robotic systems with Virtual Reality. Robot control architectures — centralized, decentralized, and hybrid — each present distinct trade-offs in scalability and fault tolerance, with hybrid models prevailing in modern industrial deployments. VR, grounded in the triad of immersion, presence, and interaction, constitutes an effective HMI layer for teleoperation, digital twin visualization, and operator training. The emerging field of VR Physical Engineering further extends this integration by embedding rigid-body dynamics and multimodal feedback into the simulation loop, despite the cost of unresolved challenges in determinism, network synchronization, and safety-critical validation. These advantages motivate the technical developments presented in the subsequent chapters.

Chapter 3

State of the Art

3.1 Related Work

The Fourth Industrial Revolution has accelerated research into integrating virtual reality, digital twins, and web technologies into robotic systems. These technologies aim to improve human–robot interaction by facilitating control, enhancing environmental awareness, and enabling remote access. Simultaneously, web platforms have emerged as a promising solution for deploying robotic applications without the need for specific software installation.

Despite significant progress in remote control through virtual reality, digital twins, and browser-based robotic systems, many challenges remain. Current solutions often focus on specific aspects such as visualization, training, remote control, or robot monitoring, with limited support for realistic physical simulation, safe interaction, collaborative operation, and native browser deployment.

The following sections examine case studies in these areas, analyzing their contributions, limitations, and relevance to the proposed system.

3.1.1 VR Teleoperation for Mobile Robots

Galarza et al. [10] proposed a VR-based teleoperation framework for a KUKA youBot mobile manipulator. The system integrates a Unity-based virtual environment, an HTC Vive headset, and ROS middleware to establish real-time communication between the operator and the robot. A digital representation of the robot was developed using a kinematic model and validated through simulation before deployment on physical hardware.

Experimental evaluation reported good usability and low communication latency, demonstrating the feasibility of immersive VR-based robot teleoperation. The study confirms that VR interfaces can improve operator awareness and provide intuitive control of robotic systems.

Despite these advantages, the framework is limited to a single user and a single robot. It also relies on a standalone application requiring dedicated software installation and does not support browser-based access, collaborative supervision, or multi-robot coordination.

This work provides a valuable foundation for immersive robotic teleoperation, while highlighting the need for web-native, collaborative, and scalable robotic platforms.

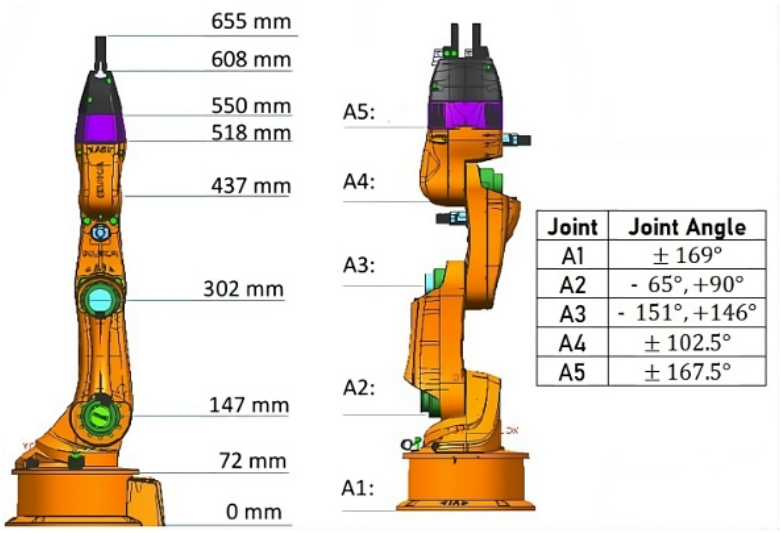


Figure 3.1 : Geometrical parameters of the KUKA youBot arm .

3.1.2 Digital Twin and VR for Collaborative Assembly

Hoebert et al. [1] proposed a Digital Twin and Virtual Reality framework to support collaborative robotic disassembly through real-time monitoring, simulation, and decision support. The architecture integrates a digital twin environment, ROS communication, computer vision, and decision-making components to maintain synchronization between the physical robot and its virtual counterpart.

The digital twin provides a real-time representation of the robotic workspace, allowing operators to monitor task execution, evaluate robot reachability, and validate planned actions before deployment. The framework also incorporates collision avoidance and dynamic task replanning mechanisms, improving safety and adaptability in changing industrial environments.

Experimental validation using a Yaskawa Motoman MH50 robot demonstrated effective synchronization between the virtual and physical systems, enhanced workspace awareness, and improved support for human–robot collaboration. The study highlights the benefits of combining Digital Twin technology, VR interfaces, and intelligent decision support within a unified framework.

However, the proposed solution is limited to a single robot and a single operator. It does not provide browser-based access, multi-user collaboration, remote deployment, or multi-robot coordination capabilities. These limitations restrict its scalability for large industrial environments.

This work represents an important step toward Digital Twin-enabled collaborative robotics while emphasizing the need for web-native, collaborative, and scalable robotic platforms.

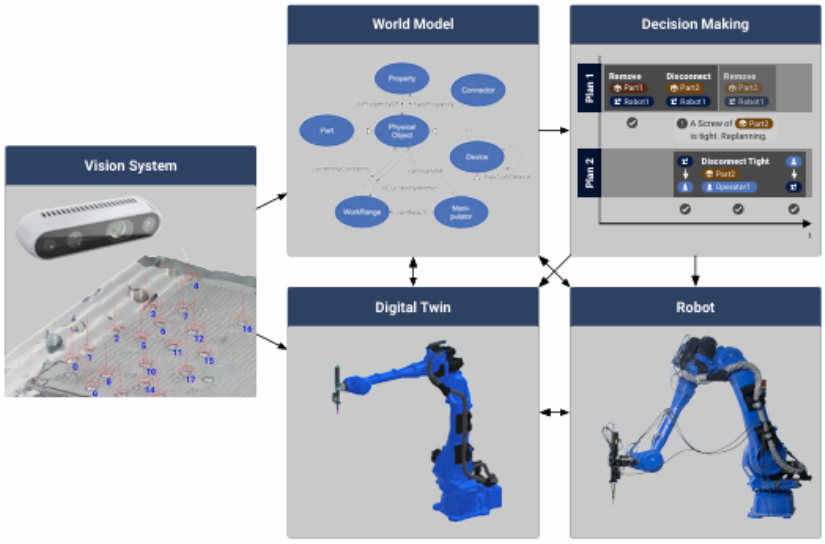


Figure 3.2 : Information synchronization and transfer between the system components [1].

3.1.3 Web-Based XR Framework for Robotic Training

Kaarlela et al. [2] proposed a web-based XR framework that combines Digital Twin technology with Extended Reality (XR) for robotic training and teleoperation. The framework aims to provide browser-native access to immersive robotic environments without requiring dedicated software installation. It relies on web technologies such as WebGL, WebAssembly, MQTT, WebSocket, and WebRTC to synchronize physical robots with their digital counterparts and support remote interaction.

The proposed system supports multiple XR modalities, including Augmented Reality (AR), Augmented Virtuality (AV), and Virtual Reality (VR), enabling users to perform training and teleoperation tasks within immersive environments. Experimental evaluations demonstrated the feasibility of browser-based XR training across a variety of devices and confirmed the potential of Digital Twins for improving accessibility and deployment flexibility.

Despite these advantages, the framework is limited to single-robot operation and does not provide multi-user collaboration or multi-robot coordination. Furthermore, the study does not extensively evaluate scalability or the impact of network variability on system performance during safety-critical operations.

This work demonstrates the viability of browser-based XR Digital Twin systems for robotics while highlighting the need for collaborative, scalable, and multi-robot web platforms.

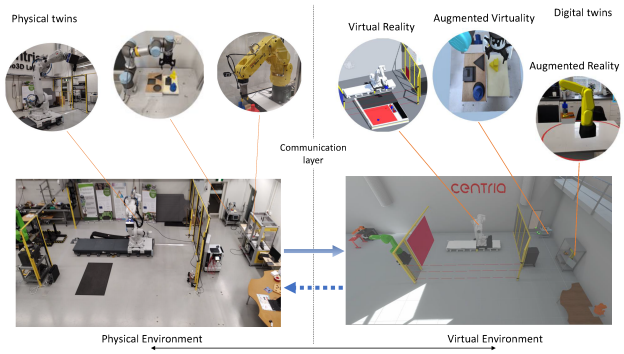


Figure 3.3 : Web-Based XR Framework for Robotic Training [2]

3.1.4 Web-Based Multi-Robot System for the Industrial Metaverse

You et al. [3] proposed a web-based framework for monitoring and controlling industrial robots through a standard web browser. The architecture combines ROS middleware, WebSocket communication, and Three.js visualization to provide real-time interaction between physical robots and their digital representations.

A notable contribution of the framework is its ability to import and visualize robots from different manufacturers using URDF models, enabling heterogeneous robotic systems to be displayed within a unified web environment. The system also supports the simultaneous visualization and independent control of multiple robots while maintaining real-time synchronization between the physical and virtual worlds.

Experimental results demonstrated the feasibility of browser-based robot monitoring and control, highlighting the potential of web technologies for industrial robotics. The authors also proposed a WebXR-based extension to enable immersive interaction between users and robotic systems.

However, the framework focuses primarily on visualization and individual robot control. Multi-robot collaboration, multi-user interaction, and immersive VR operation were not experimentally validated. In addition, scalability under real-world network conditions remains largely unexplored.

This work demonstrates the potential of combining ROS, WebSockets, and browser-native technologies for industrial robotics, while emphasizing the need for collaborative, immersive, and scalable web-based robotic platforms.

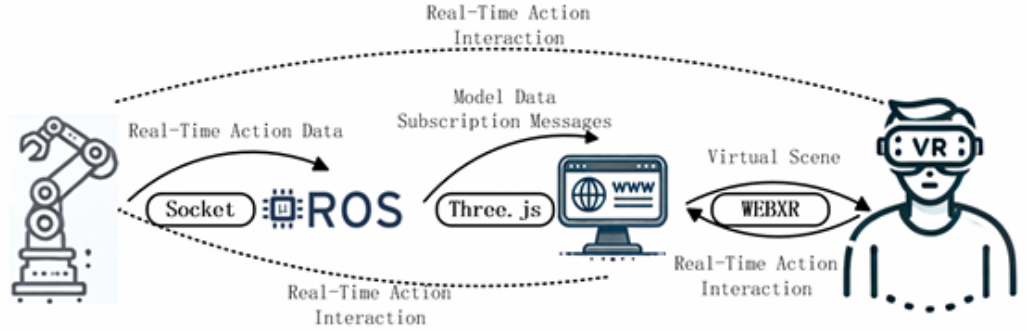


Figure 3.4 : WEB concept industrial metaverse [3].

3.1.5 Lightweight WebGL Simulation with Three.js

Zhang et al. [4] proposed a lightweight web-based robotic simulation framework based on WebGL and Three.js. The objective was to provide an accessible and cross-platform environment for robot visualization and kinematic simulation directly within a web browser, without requiring dedicated software installation.

The framework combines robot modeling with browser-based 3D rendering, enabling users to visualize and interact with robotic manipulators in real time. Kinematic calculations are performed using the Denavit–Hartenberg method, allowing accurate representation of robot joint movements and end-effector positioning.

Experimental results demonstrated that WebGL and Three.js provide an effective foundation for browser-native robotic visualization and simulation. The proposed approach offers good portability and ease of deployment across different platforms while reducing the complexity associated with traditional desktop simulators.

However, the framework is primarily limited to visualization and kinematic analysis. It does not support real-time communication with physical robots, ROS integration, collaborative operation, multi-robot coordination, or immersive XR interaction. Consequently, its applicability to advanced industrial robotic scenarios remains limited.

This work highlights the feasibility of browser-based robotic simulation using modern web technologies while emphasizing the need for richer functionalities such as physics simulation, real-time connectivity, and collaborative operation.

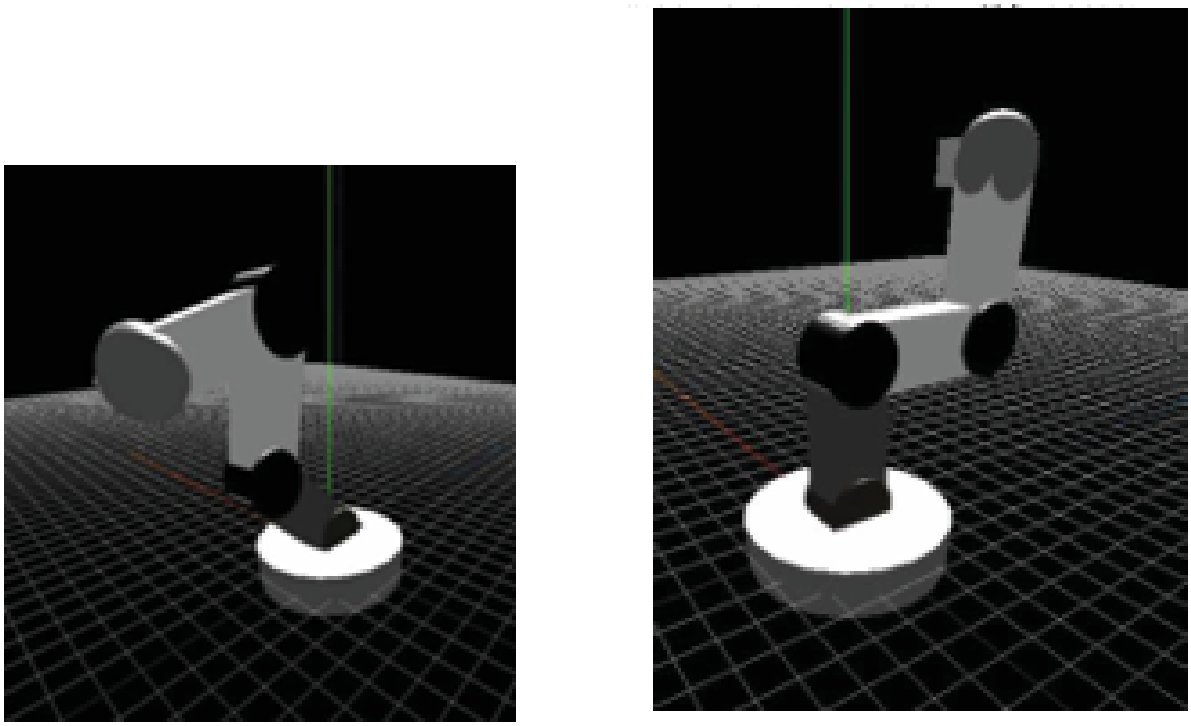


Figure 3.5 : Three-dimensional simulation model of a 5-DOF manipulator [4].

3.2 Comparative Analysis

3.2.1 Comparative Synthesis

Table 3.1 provides a rapid overview of each system's positioning across the seven criteria identified as critical for web-based industrial robotic simulation.
Table 3.2 explicitly maps each gap and indicates whether our work provides a response.

Table 3.1 : Comparison of Related Works

Criterion	**Kaarlela**	**You**	**Galarza**	**Hoebert**	**Zhang**
Web Access (No Install)	Yes	Yes	No	No	Yes
XR Immersion (Real)	Yes	No	Yes	Yes	No
Digital Twin	Yes	Partial	No	Yes	No
Real Robot Connectivity	Yes	Yes	Yes	Yes	No
Multi-Robot Support	No	Display Only	No	No	No
Multi-User Collaboration	No	No	No	No	No
AI / Decision Support	No	No	No	Yes	No

Table 3.2 : Gap mapping : identified limitations in the literature versus responses provided.

Identified Gap	**Galarza**	**Hoebert**	**Kaarlela**	**You / Zhang**	**Our Work**
Full physics simulation (forces, friction)	×	Partial	×	×	✓ Cannon-js
Normative grasping model (ISO 10218)	×	×	×	×	✓ Complete
PD force controller on grip	×	×	×	×	✓ k_p=500, k_d=50
Web access without installation	×	×	✓	✓	✓
Computer vision	×	×	×	×	✓
Dual ISO standard compliance	×	Partial	×	×	✓ ISO 10218 + ISO/TS 15066
Dynamic center of gravity (COG)	×	×	×	×	✓
Full real-time telemetry	×	Partial	×	Partial	✓ 5-layer

This table suggests that the proposed system addresses all identified criteria simultaneously, whereas existing works only satisfy a subset of them. In particular, the first four criteria—complete physics simulation, normative grasping model, PD force controller, and web access without installation—are satisfied by none of the existing works individually. It is precisely the intersection of these four properties that constitutes the original contribution of our system.

3.2.2 Discussion

The reviewed literature shows impressive advancements in web-based robotics, Digital Twin technologies, Virtual Reality interfaces, and human–robot collaboration. Despite this progress, certain hurdles still exist, particularly when it comes to integrating realistic physics simulations, safety-conscious interactions, and seamless browser-native accessibility into one unified platform. This section outlines these research gaps, highlights the core contributions of the proposed system, and explains the reasoning behind the architectural and technological decisions made throughout its development.

State of Web-Based Robotic Simulation Before Our Work

Prior to this research, web-based robotic simulators were hindered by a significant trade-off : systems capable of high-fidelity physics simulations required local installations and specialized hardware, whereas browser-based platforms often lacked physical realism and industrial safety standards. Most current web solutions failed to support essential elements like rigid-body physics, realistic grasping forces, or safety-compliant collaborative mechanisms. Consequently, it was impossible to accurately simulate vital robotic behaviors—such as stable grasping, object slippage, and force-regulated manipulation directly within a web browser. While existing literature explores VR interaction, Digital Twins, and web-based robot control, none explicitly integrates browser-native deployment with safety protocols derived from ISO 10218 and ISO/TS 15066. Our work fills this void by unifying web accessibility, real-time physics, realistic grasping, and standardized safety compliance into a single browser-based robotic platform.

Contributions of the Proposed System

The proposed system was designed to address the limitations identified in the reviewed literature while establishing a foundation for future developments in collaborative robotics. Given the scope of the current work, the focus was placed on realistic physics simulation, safety-aware interaction, and modular system design. The resulting platform combines browser-native accessibility with real-time robotic simulation, providing a flexible environment for experimentation, training, and future integration of advanced functionalities such as multi-robot coordination, multi-user collaboration, and real-world robotic connectivity.

Browser-Based Physics Simulation. One of the main contributions of this work lies in the direct integration of the Cannon.js physics engine within the browser, enabling real-time rigid body simulations at a constant frequency of 60 Hz. While many current web-based robotic simulators are limited to simple kinematic visualization, our system actively calculates gravity, collisions, contact

reactions, and physical constraints in real time. To guarantee stable performance, several optimization and stabilization mechanisms were incorporated into the simulation framework. This approach enables realistic robotic interactions while ensuring full compatibility with standard web browsers.

Modular Five-Layer Architecture. The proposed system is based on a five-layer architecture consisting of configuration, decision-making, physics and sensors, grasping, and interface and telemetry layers. This separation improves modularity, simplifies maintenance, and prevents excessive coupling between visualization, physics simulation, and robotic control components. Furthermore, the architecture facilitates future extensions such as real-robot integration, WebSocket synchronization, and collaborative multi-user interaction without requiring major architectural modifications.

3.3 Conclusion

This chapter covered the theoretical groundwork and current state of the art in industrial robotics, Virtual Reality, human–robot collaboration, Digital Twin technologies, and web-based robotic platforms. It also took a close look at the core architectures, communication protocols, and simulation frameworks that drive today's robotic systems. The literature review highlights an increasing overlap between web technologies, XR environments, and Digital Twin concepts in robotics. However, significant hurdles still exist, especially when it comes to integrating realistic physics simulations, safety-conscious interactions, and seamless browser-based accessibility into a single, cohesive platform. Most current solutions only tackle pieces of the puzzle, leaving notable gaps in realism, safety, and scalability. Drawing from these findings, this research seeks to bridge those gaps by integrating real-time physics simulation, safety-oriented robot interaction, and web-based accessibility into a modular, extensible architecture. The challenges and opportunities identified here serve as the starting point for the system detailed in the next chapter. The following chapter will outline the architecture and design of the proposed platform, explaining its individual components, software organization, and core implementation principles.

Chapter 4

Proposed System

4.1 Introduction

The evolution of simulation technologies and industrial robotics has enabled the development of more accessible, realistic, and interactive robotic systems. Thanks to advances in real-time physics simulation and immersive environments, it is now possible to virtually reproduce complex robotic behaviors without relying on costly physical infrastructures.

Modern robotic simulation relies primarily on the physical modeling of robots and their interaction with the environment. It integrates collision management, object grasping, multi-agent coordination, and autonomous task execution within dynamic environments.

Modern web technologies now offer a suitable framework for developing simulation platforms accessible directly from a web browser. In this context, this work proposes a collaborative robotic simulation platform integrating an interactive 3D environment, real-time physics simulation, an event-driven system, and multi-robot coordination mechanisms.

4.2 Objectives

This work aims to design and implement a web-based collaborative robotic simulation platform, integrating a real-time physics engine and an immersive 3D visualization environment. It is intended for the kinematic and dynamic modeling of manipulator robots, their multi-agent coordination, and interactive experimentation within a virtual environment faithful to physical reality, all accessible natively via a web browser.

- **Cost Reduction :** Industrial robots and their infrastructures are expensive. Simulation enables experimentation without significant material investment.
- **Experimentation Safety :** Testing on real robots can cause collisions or material damage. The platform offers a secure virtual environment for validating robotic scenarios.
- **Realistic Physics Simulation :** Reproduce mechanical dynamics, collisions, and joint constraints to obtain behavior close to real systems.
- **Multi-Robot Collaboration :** Enable coordination and interaction between multiple robots in a shared and synchronized environment.

- **Intelligent Grasping :** Develop a grasping system based on virtual tactile sensors and real-time event logic.
- **Multi-User Synchronization :** Allow multiple remote users to interact simultaneously with the robots via WebSocket.
- **VR/AR Integration :** Integrate VR/AR technologies to offer immersive interaction with the simulated robotic environment.
- **Web Accessibility :** Provide a platform accessible directly from a web browser without complex installation.
- **System Modularity :** Design a modular architecture facilitating the addition of new robots, sensors, and autonomous behaviors.

4.2.1 Research Questions

While the objectives above define the practical scope of this work, the following research questions frame the underlying scientific investigation and address the gaps identified in the state of the art (Chapter 3) :

- **RQ1:** Can browser-native physics engines (*Cannon.js*) achieve sufficient physical accuracy for industrial robotic simulation comparable to desktop-based alternatives ?
- **RQ2:** What is the maximum number of robots that can be simulated simultaneously at acceptable performance ($\geq$50 FPS) using WebGL rendering and JavaScript physics ?
- **RQ3:** Does the proposed force-regulated grasping model with virtual tactile sensors achieve stable object manipulation in a browser-based environment ?
- **RQ4:** Can WebSocket-based multi-user synchronization maintain latency below the 100 ms human perception threshold for collaborative robotics ?
- **RQ5:** How does the dual-camera visual perception system with Frustum-based detection perform for real-time object tracking and collision avoidance ?

4.2.2 Research Hypotheses

Based on the research questions above, the following testable hypotheses were formulated and subsequently validated through the experimental campaign presented in Chapter 5 :

- **H1:** Browser-native physics simulation (*Cannon.js* at 60 Hz) can achieve sufficient fidelity for training and prototyping applications, with collision detection and resolution rates exceeding 95%.
- **H2:** The proposed force-regulated grasping model with PD control and virtual tactile sensors will achieve a success rate of at least 90% for autonomous pick-and-place operations.
- **H3:** WebSocket synchronization will maintain latency below 50 ms for up to 4 simultaneous users, enabling collaborative interaction without perceptible delays.

- **H4:** The system will maintain rendering performance above 50 FPS for up to 4 simultaneous robots, ensuring a smooth interactive experience.
- **H5:** The dual-camera visual perception system with Frustum-based detection will achieve real-time object tracking with minimal computational overhead (<1 ms per frame).

4.3 System Overview

The proposed system is a web-based collaborative robotic simulation platform executed directly in a standard web browser. It integrates an immersive 3D visualization environment and a real-time physics simulation engine enabling the management of collisions, gravity, and kinematic constraints.

The platform allows the modeling and coordination of several industrial and collaborative robots within an interactive virtual environment. It relies on an intelligent grasping mechanism based on virtual tactile sensors, a dedicated event-driven architecture for executing autonomous behaviors, and a distributed multi-user synchronization system.

The system also supports VR/AR technologies, offering immersive interaction with the simulated robots while guaranteeing direct accessibility via the web, without complex software installation.

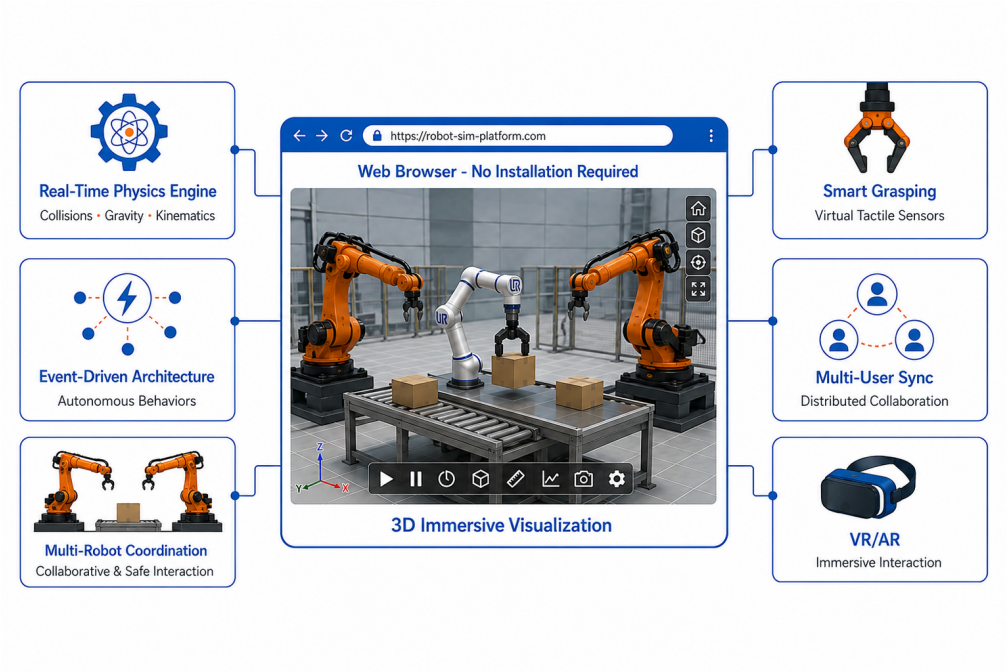


Figure 4.1 : Overview of the web-based collaborative robotic simulation platform .

In summary, the platform is a web-based collaborative robotic simulation system combining real-time 3D visualization, physics, smart grasping, multi-user synchronization .

4.4 Robot Modeling

The development of a realistic robotic simulation platform requires the integration of several interconnected components, including physical simulation, robot modeling, event-driven behaviors, and real-time interaction mechanisms. The proposed system was designed to reproduce the behavior

of industrial and collaborative robots inside an immersive virtual environment while maintaining modularity, realism, and accessibility through web technologies.

4.4.1 Problem Statement

The following challenges were identified during the design phase :

- **Collision Management :** Robots could collide with objects, the ground, or other robots during autonomous movements and collaborative tasks.
- **Multi-User Synchronization :** Synchronizing robot states between multiple remote users in real time caused latency and state consistency issues.
- **Object Grasping Stability :** Achieving stable and realistic object grasping required precise contact detection and grip force evaluation.
- **Real-Time Physics Constraints :** Maintaining stable physical simulation while preserving real-time performance was a major challenge.
- **Robot Coordination :** Coordinating multiple robots inside the same environment introduced trajectory conflicts and interaction problems.
- **Immersive Interaction :** Integrating VR/AR interaction while maintaining synchronization and responsiveness required additional optimization mechanisms.

These problems prompted the development of a custom robot specifically designed to meet the requirements of an accurate and realistic robotic system.

4.4.2 Industrial Robot Types

Industrial robotics encompasses a wide variety of morphologies, each adapted to specific tasks. Table 4.1 presents a comparative overview of the main categories.

Table 4.1 : Comparative overview of industrial robot types.

Type	**DOF**	**Architecture**	**Applications**	**Trade-off**
Articulated A	6	Serial rotary joints	Welding, assembly, palletizing	Flexibility vs. power consumption
SCARA	4	2 rotations + Z translation	Electronic assembly, screwing	Speed vs. cylindrical workspace
Cartesian B	3–4	3 linear axes (gantry)	CNC, 3D printing, heavy transfer	Precision vs. footprint
Delta C	3–4	Parallel parallelogram arms	Food sorting, fast packaging	Speed vs. cost, workspace
Mobile (AGV/AMR) D	2–3	Wheeled/tracked platform	Logistics, material transport E	Autonomy vs. battery life
Cobot F	6–7	Serial + torque sensors	Collaborative assembly, inspection	Safety vs. speed, payload

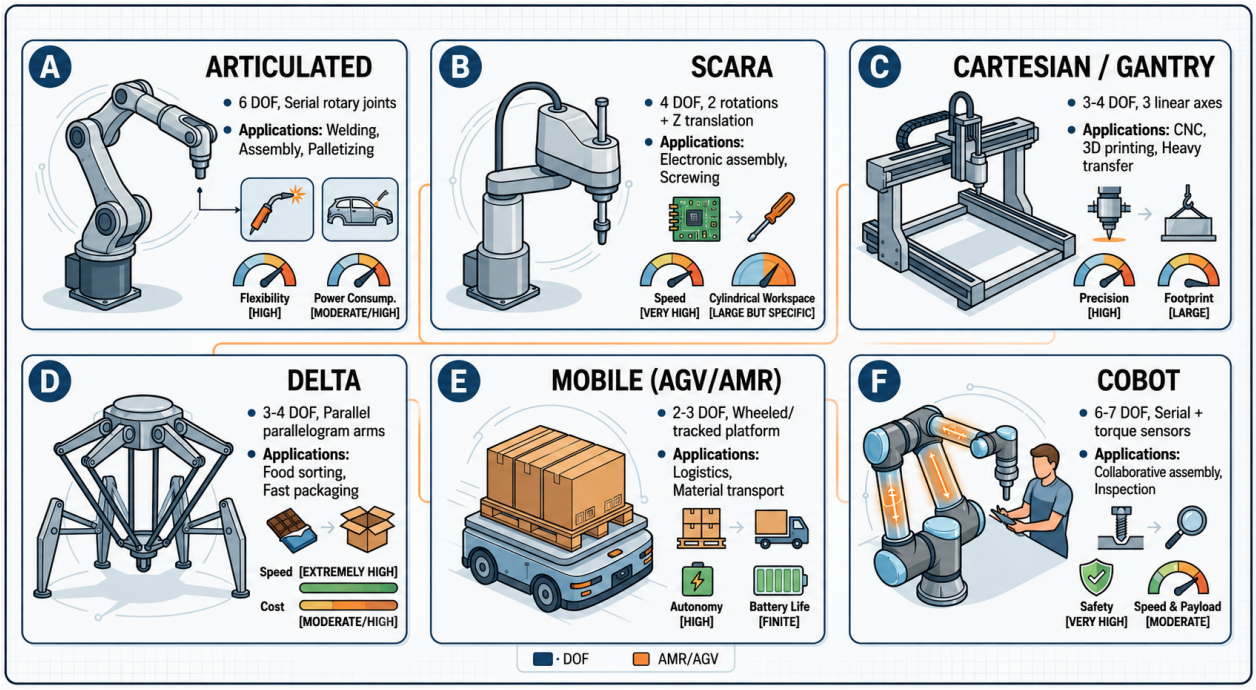


Figure 4.2 : Comparison of common industrial robot architectures.

Our simulator implements a 4-DOF articulated arm (base, shoulder, elbow, wrist) on a tracked mobile base, with two configurable profiles : industrial (high force, no safety limits) and cobot (reduced force, E-Stop enabled at 600 N).

4.5 Robot Description

The robot developed in this project is a 4-DOF mobile manipulator integrated into an immersive simulation environment. Two main variants are implemented : an industrial model for heavy handling tasks and a collaborative model (cobot) designed for safer interactions with the environment.

The mechanical structure relies on a tracked mobile platform supporting a rotating turret that ensures the base rotation of the manipulator arm. The arm is composed of several articulated segments corresponding to the shoulder, elbow, and wrist. Each joint is motorized and constrained by kinematic limits to reproduce realistic behavior close to real industrial robots. Movements are generated using S-Curve kinematic profiles to limit abrupt variations in acceleration and jerk, resulting in smoother and more stable trajectories.

The end effector consists of a parallel two-finger gripper enabling the grasping and manipulation of objects in the simulated environment. The gripper has a dynamically controlled opening in real time and a grasping system based on a PD controller for regulating the force applied to objects. The grasping model also incorporates a friction coefficient, progressive force limitation, and stability management to avoid slippage or abrupt variations during grasping.

To improve interaction realism, each finger integrates several virtual contact points corresponding to distributed tactile sensors on the fingertip, center, and base of the finger. These sensors enable proximity detection, contact validation, and grasping force evaluation. The data from these sensors are then exploited by the decisional system to determine the validity and stability of the grasp before executing grasping actions.

The robot also has an environmental perception system responsible for detecting objects and other robots in the workspace. Perceived information is continuously updated in temporary memory to ensure spatial and temporal consistency of robotic interactions. This mechanism enables multi-robot coordination, obstacle detection, and execution of autonomous behaviors based on real-time events.

Dynamically, the system accounts for mass distribution and recalculation of the center of gravity (COG) during object manipulation. When a load is grasped, the robot's COG is dynamically adjusted to preserve system stability during lifting, displacement, and transport phases.

Robot displacement relies on differential drive control of the mobile base combined with kinematic control of the joints. Velocities, accelerations, and dynamic joint limits are controlled to ensure coherent and physically realistic movements in the simulated environment. A dynamic load factor also adapts robot speed according to the transported mass.

Finally, the system integrates several functional safety mechanisms such as self-collision detection, inter-robot collisions, emergency stop, and automatic speed limitation in critical situations. In addition, the robot can execute autonomous tasks such as navigation, obstacle avoidance, and Pick & Place scenarios via an event-driven architecture based on a finite state machine.

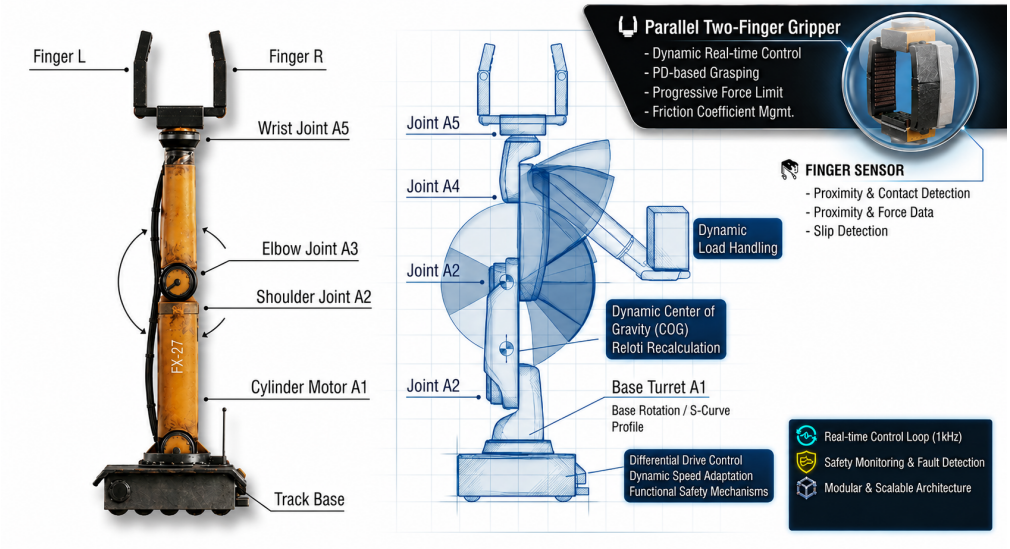


Figure 4.3 : Robot Description.

4.5.1 Robot Body Architecture

The robot body architecture of the proposed robot consists of a mobile base, a manipulator arm, and a two-finger gripper designed to support realistic industrial and collaborative robotic operations.

Mobile Base

The mobile base consists of a rectangular steel chassis with dimensions 0.60 m × 0.80 m × 0.14 m. The chassis mass is 100 kg for the industrial model and 70 kg for the cobot. The base rests on two lateral tracks with dimensions 0.16 m × 0.95 m × 0.24 m ensuring traction and ground mobility. A cylindrical turret with radius 0.28 m and height 0.14 m (mass 15 kg for industrial, 10 kg for cobot) is mounted on the chassis and allows complete base rotation of the manipulator arm over ±180°. Displacement is achieved by differential drive with progressive acceleration and deceleration, ensuring smooth and realistic movements.

Manipulator Arm

The manipulator arm is composed of three rigid segments articulated by motorized revolute joints. The shoulder has a length of 2.0 m with a mass of 20 kg for the industrial configuration and 12 kg for the collaborative robot (cobot). The elbow measures 1.7 m in length and weighs 15 kg in the industrial version and 10 kg in the cobot version. The wrist has a height of 0.14 m and a mass of 4 kg for the industrial robot and 2.5 kg for the cobot. Each joint is driven using an S-Curve (jerk-limited) motion profile to ensure smooth acceleration and deceleration. The manipulator joints are also subject to predefined angular constraints that guarantee safe operation and prevent mechanical interference. The angular limits of the shoulder, elbow, and wrist joints are summarized in Table 4.2.

Table 4.2 : Joint angular limits.

Joint	Type	Min. Limit	Max. Limit
Base (turret)	Rotation Y (yaw)	$-180°$	$+180°$
Shoulder	Rotation X (pitch)	$-80°$	$+85°$
Elbow	Rotation X (pitch)	$-90°$	$+90°$
Wrist	Rotation Y (yaw)	$-180°$	$+180°$

4.5.2 Grasping System

The grasping subsystem is responsible for object acquisition, retention, and release. It combines force regulation, contact sensing, and physical interaction models to ensure stable object manipulation.

PD Control and Forces

The gripper integrates a Proportional-Derivative (PD) control to regulate the force exerted on objects. The maximum force is 1200 N for the industrial model and 600 N for the cobot. The adhesion model relies on a friction coefficient of 0.8 and a load capacity of 50 kg (industrial) or 20 kg (cobot). A safety margin of 10% applies to the required force. A slew-rate control limits abrupt force variations to 200 N/s (industrial) or 100 N/s (cobot) to avoid any jolt detrimental to grasping stability.

Grasping Modes

The system supports two complementary grasping modes. The first mode is the centered grip : the box is geometrically aligned between the fingers, validated by the `isBoxBetweenFingers` function. The second mode is the edge grip : the sensors of both fingers detect contact even if the box is not perfectly centered, provided the force is sufficient and the opening remains within defined limits. In case of detected overload (available force less than required force), the load is automatically released.

Physical Grasping Model

The grasping controller implements a complete physical model combining several force components to evaluate grasp stability. The normal clamping force is computed as the sum of the forces measured by the sensors of both fingers, scaled by the maximum cylinder force. The friction force is then determined from the normal force, the friction coefficient, and an efficiency factor. The total available holding force corresponds to the combination of the normal and friction forces. To assess whether an object can be safely manipulated, the required holding force is calculated from the object weight while accounting for friction effects and a safety margin of 10%. The mathematical expressions governing these force calculations are summarized in Table 4.3.

Table 4.3 : Force formulas for the grasping model.

Force	**Formula**	**Description**
Normal force	$F_n = (F_{\text{left}} + F_{\text{right}}) \times F_{\max}$	Total clamping force
Friction force	$F_f = \mu \times F_n \times \eta$	Friction ($\mu = 0.8$, $\eta = 0.3$)
Available force	$F_{\text{avail}} = F_n + F_f$	Total holding force
Required force	$F_{\text{req}} = \frac{m \cdot g}{1+\mu \cdot \phi} \times S$	Required with margin ($S = 1.1$)

Variables : F_{left}, F_{right} (finger positions), $F_{\max}$ (maximum gripping force), μ (friction coefficient), η (efficiency factor), m (object mass), g (gravitational acceleration), ϕ (friction contribution factor), and S (safety margin).

4.5.3 Sensors and Perception

To enable autonomous interaction with the environment, the robot integrates both tactile and visual perception mechanisms that provide real-time information about objects and surrounding robots.

Tactile Sensors

Each finger is equipped with three contact points forming a tactile force-sensing network used to detect and evaluate object interactions during grasping. The sensors are distributed along the finger height, with one located near the fingertip at $FH - 0.05\,\text{m}$, a second positioned at the midpoint ($FH/2$), and a third placed near the base at $0.1\,\text{m}$. A detection threshold of $0.035\,\text{m}$ combined with a debounce period of $50\,\text{ms}$ enables the system to distinguish transient contacts from stable ones. The measured contact force ranges from 0 to $60\,\text{N}$ and is used by the decision module to validate grasp stability. The distribution of the tactile sensing points along the finger is summarized in Table 4.4.

Table 4.4 : Tactile sensor point distribution.

Contact Point	**Local Position (Y)**	**Role**	**Threshold**
Tip	$FH - 0.05$ m	Fine contact detection	$0.035\,\text{m}$
Middle	$FH/2$	Main contact	$0.035\,\text{m}$
Base	0.1 m	Grasp stability	$0.035\,\text{m}$

Variables : FH (finger height). The threshold specifies the contact detection distance used by each tactile sensing point.

Environmental Perception

The robot has an environmental perception system ensuring the localization of objects and other robots in the workspace. Information updates are performed cyclically at each rendering frame. The perceived object memory uses a short-term cache of $50\,\text{ms}$ guaranteeing the freshness of spatial information. This system feeds the autonomous navigation and multi-robot coordination modules.

Visual Perception System

To further enhance environmental awareness, a dual-camera visual perception system was integrated. The system utilises two `THREE.PerspectiveCamera` instances per robot : a **BODY camera** mounted at $0.65\,\text{m}$ height with a $-0.3\,\text{rad}$ tilt for global situational awareness, and a **WRIST camera** mounted on the end-effector for precise manipulation tasks. Instead of heavy machine learning models, the system employs efficient **Frustum Culling** (`THREE.Frustum`) to detect whether target objects (boxes or other robots) intersect with the camera's field of view.

A frame-by-frame **Object Tracking** algorithm assigns unique `tracks` to detected entities and calculates a `stable` metric requiring more than 10 consecutive stable frames before confirming a detection, effectively filtering transient false positives. Tracking entries automatically expire after 30 frames without re-detection, ensuring memory efficiency. The visual perception system actively monitors proximity and integrates with the robot's safety mechanisms : if another robot is detected within 1.5 m or an obstacle within 0.7 m, it triggers an immediate collision warning, seamlessly coupling with the robot's E-Stop and speed reduction safety modes.

4.5.4 Dynamics and Center of Gravity

Dynamic stability is essential during object manipulation. The proposed system continuously evaluates mass distribution and adapts robot behavior according to the transported load.

Dynamic COG Calculation

The center of gravity (COG) of the robot is initially determined by the mass distribution of its components. When a load is grasped, the COG is recalculated in real time according to the following formula :

$$\mathrm{COG}_Y = \frac{m_{\mathrm{base}} \cdot y_{\mathrm{base}} + m_{\mathrm{charge}} \cdot y_{\mathrm{charge}}}{m_{\mathrm{base}} + m_{\mathrm{charge}}} \tag{4.1}$$

This dynamic calculation allows adjustment of the physical body position in the simulation environment and ensures robot stability, particularly during lifting and transport phases.

Dynamic Load Factor

The load factor loadFactor reduces kinematic performance proportionally to the lifted mass according to the formula :

$$\mathrm{loadFactor} = \frac{1}{1 + \frac{m_{\mathrm{charge}}}{m_{\mathrm{arm\ total}} \times 0.5}} \tag{4.2}$$

For a 15 kg load on a 42 kg arm, the load factor is 0.583, corresponding to a 41.7% reduction in maximum kinematic performance.

Table 4.5 : Dynamic load factor according to transported mass.

Load Mass (kg)	Arm Mass (kg)	Load Factor	Reduction
5	42	0.808	19.2%
10	42	0.677	32.3%
15	42	0.583	41.7%
20	42	0.512	48.8%
30	42	0.412	58.8%

4.5.5 Motion Control

Robot movements are generated through smooth trajectory planning algorithms that limit abrupt variations in velocity and acceleration while maintaining responsiveness.

S-Curve Profile

Base displacement is achieved through a differential-drive mechanism, providing a maximum linear speed of $2.0\,\mathrm{m\,s^{-1}}$ for the industrial robot and $0.8\,\mathrm{m\,s^{-1}}$ for the collaborative robot (cobot). To ensure smooth and mechanically safe movements, all manipulator joints follow jerk-limited S-Curve motion profiles. These trajectories are constrained in terms of velocity, acceleration, and jerk, and are composed of four successive phases : progressive acceleration, constant velocity, progressive deceleration, and stop. The motion parameters governing the S-Curve profiles are summarized in Table 4.6.

Table 4.6 : S-Curve joint motion parameters.

Joint	$V_{\max}$ (°/s)	$A_{\max}$ (°/s^2)	$J_{\max}$ (°/s^3)
Base	35	50	150
Shoulder	30	45	130
Elbow	35–40	50–55	150–160
Wrist	50–60	70–80	200

The maximum speed is weighted by a load factor inversely proportional to the transported mass, limiting abrupt accelerations under heavy load. The joint update algorithm adapts the maximum speed, acceleration, and jerk according to the loaded mass :

$$\text{adjustedMaxVel} = \text{maxVel} \times \text{loadFactor} \tag{4.3}$$

$$\text{adjustedMaxAcc} = \text{maxAcc} \times \text{loadFactor} \tag{4.4}$$

$$\text{adjustedMaxJerk} = \text{maxJerk} \times \text{loadFactor} \tag{4.5}$$

4.5.6 Functional Safety

To ensure safe operation, particularly for the collaborative robot profile, several protection mechanisms are integrated to prevent hazardous situations and reduce collision risks.

Safety Modes

The collaborative robot incorporates a three-state safety system designed to adapt its behavior according to the operating environment. In Normal mode, the robot operates at its nominal speed and is identified by a green status indicator. When a nearby obstacle or operator is detected, the system switches to Slow mode, reducing the robot speed to 40% of its nominal value and activating a yellow indicator. In critical situations, the E-Stop mode is triggered, causing an immediate emergency stop and displaying a red indicator. This state is maintained before the robot automatically returns to Normal mode. A summary of the safety modes and their associated characteristics is provided in Table 4.7.

Table 4.7 : Safety modes for the collaborative robot.

Mode	**LED Color**	**Speed**	**Trigger**
Normal	Green	100%	Nominal operation
Slow	Yellow	40%	Human proximity (cobot)
E-Stop	Red	0%	Collision + hold 1.2 s

The industrial model does not implement active safety devices, in accordance with practices in controlled environments.

Emergency Stop Triggers

The emergency stop is automatically triggered in the following situations : detection of severe self-collision (distance below threshold between critical segments), inter-robot collision when stopOnContact mode is activated, impact with the ground (minimum segment height below , or a manual command from the user interface, with an automatic return to normal mode after a while.

Overload Protection

A safety mechanism permanently monitors the ratio between available force and required force to maintain grasping. If the available force becomes less than the required force for a duration exceeding , the load is automatically released to avoid damaging the robot or the grasped object.

4.6 Event Processing System

The Event Processing System constitutes the central communication and coordination mechanism of the proposed robotic system. It relies on an event-driven architecture enabling the exchange of information between the simulation environment, the robots, and the real-time decisional system.

Events generated by physical interactions, virtual sensors, and environmental perception mechanisms are collected, analyzed, and transmitted to the robotlistener responsible for autonomous decision-making. This approach ensures better system modularity, real-time reactivity, and consistent management of robotic behaviors in a collaborative multi-agent environment.

4.6.1 Event System Architecture

The event management system follows a three-layer architecture designed to separate perception, coordination, and decision-making responsibilities. The first layer, the Environment, monitors the physics simulation and detects object interactions, collisions, and state changes. The second layer, the Robot, centralizes these events together with tactile sensing and grasping information, acting as an intermediary between the simulation environment and the control logic. The third layer, the RobotListener, implements IF/THEN decision rules and provides the foundation for finite state machine (FSM) behaviors. The responsibilities of each layer within the event system architecture are summarized in Table 4.8.

Table 4.8 : Event system component responsibilities.

Component	Responsibilities
`Environment`	Object detection, robot monitoring, collision detection, spatial updates and event propagation to Robot
`Robot`	Event centralization, internal state management, perception storage, FSM routing and action coordination
`RobotListener`	Event analysis, condition validation, autonomous behavior triggering and safety supervision

4.6.2 Event Execution Flow

The event processing mechanism is implemented as a continuous cycle executed at each rendering frame. Algorithm 1 summarizes the sequence of operations performed by the system, from environment monitoring and event generation to decision-making and robotic action execution. This event-driven approach ensures reactive, modular, and coherent robot behavior within the collaborative virtual environment 4.4.

Algorithm 1 Event Processing Cycle

```
while simulation is running do
    Update physical environment and object positions
    Detect interactions and generate events in the Environment
    Transmit generated events to Robot instances
    Analyze tactile contacts and grasping conditions
    Validate events using RobotListener rules
    Trigger the corresponding robotic actions
    Update the robot physical and visual state in the scene
end while
```

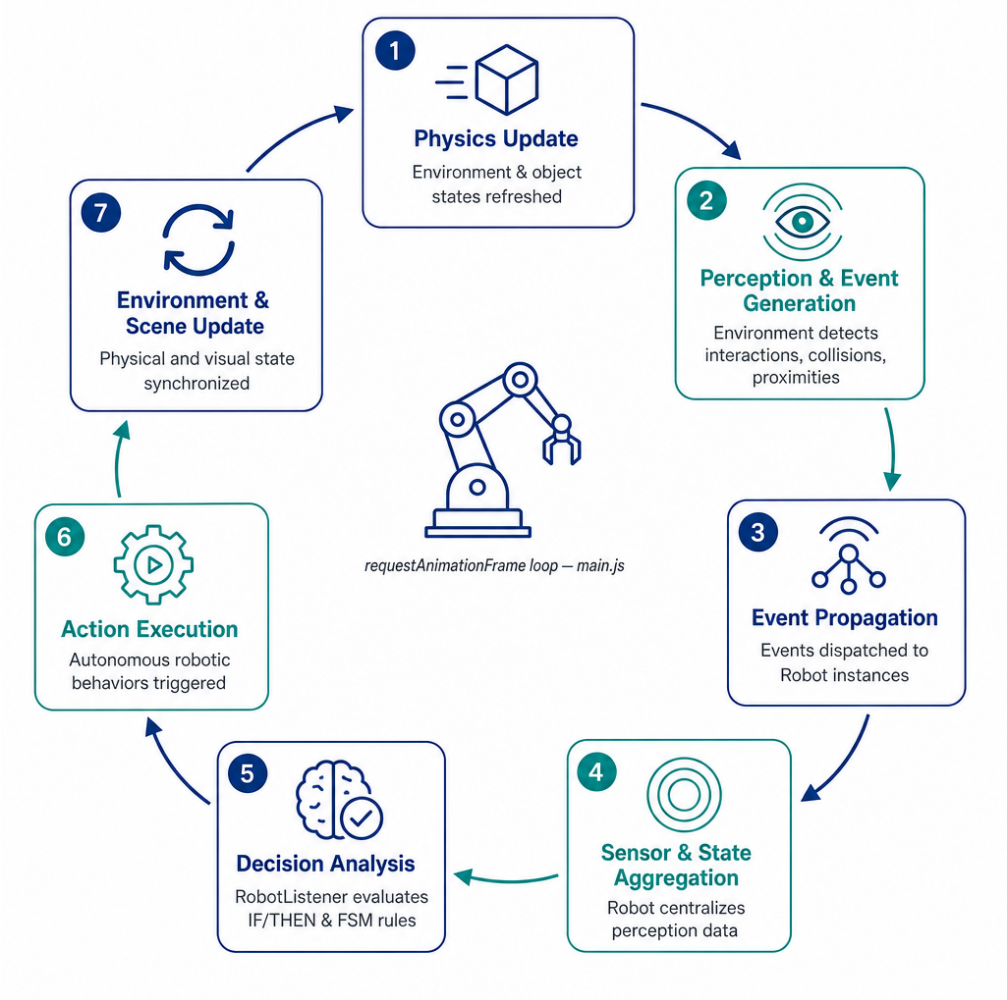


Figure 4.4 : Event execution flow from perception to action.

4.7 Robotic Model Construction

The Robotic Model Constructor ensures the generation and initialization of the physical, graphical, and behavioral components of the robot within the simulation environment. It dynamically creates robotic structures from a centralized configuration containing kinematic, physical, and functional parameters, thereby promoting modularity, reusability, and extensibility.

The construction pipeline, orchestrated by the ‘createRobot‘ function in ‘createRobot.js‘, transforms a JSON description into a fully functional robot comprising physical bodies (Cannon.js), visual meshes (Three.js), joint constraints,

tactile sensors, and event management components. Figure 4.5 illustrates the main stages of this construction process, from configuration loading to the initialization of the robot subsystems.

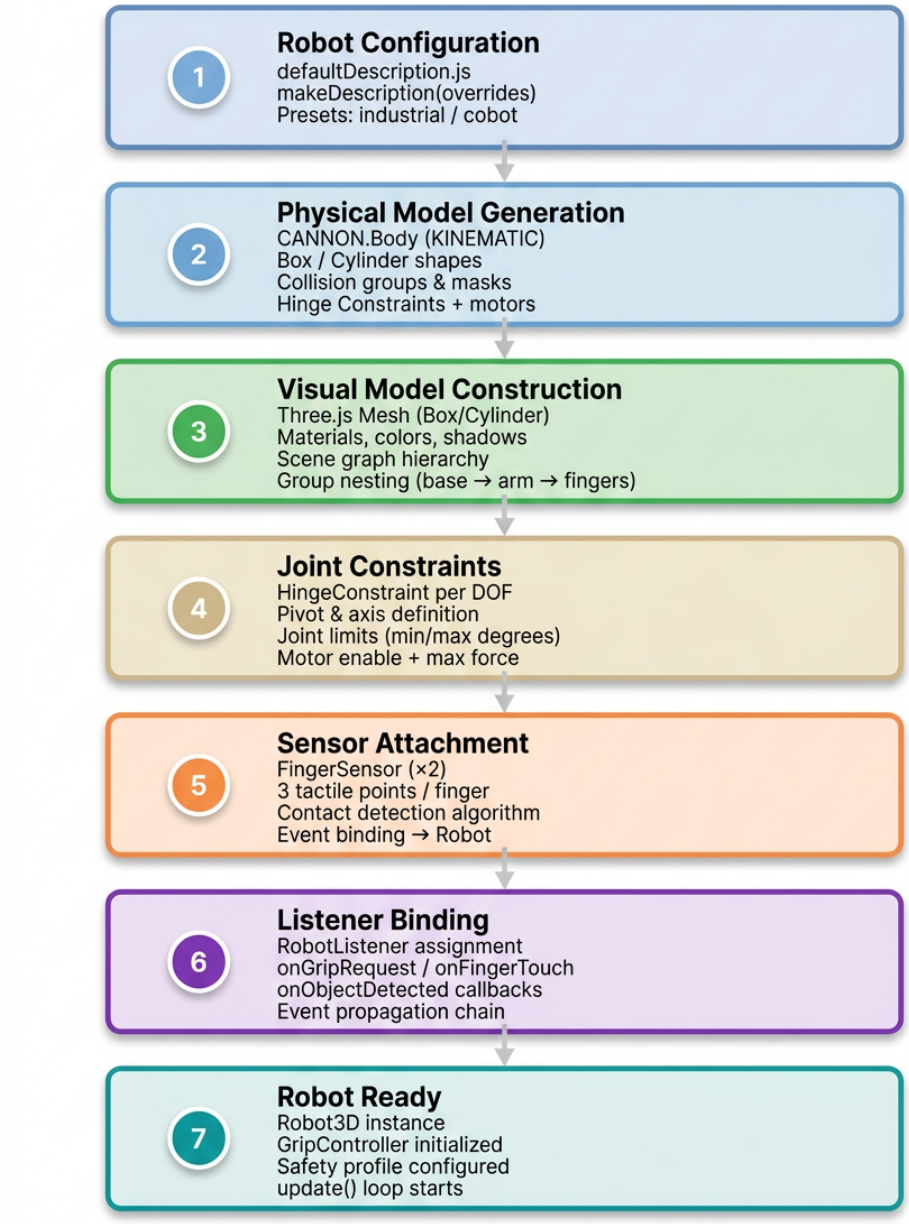


Figure 4.5 : Robotic model construction pipeline from JSON configuration to functional robot.

4.7.1 Robot Configuration

The robot configuration is the entry point of the construction process. It is defined in the file, which contains a complete and centralized description of all robot parameters : masses, dimensions, joint limits, physical properties, colors, and safety profiles.

Configuration Structure

The robot configuration is organized as a hierarchical JSON structure divided into subsystem-specific sections. Table 4.9 summarizes the organization of these sections.

Table 4.9 : Robot configuration structure.

Section	Key Parameters	Description
box	half, mass	Box dimensions and mass
arm.shoulder	len, w, mass, color	Upper arm : length 2.0 m, mass 20 kg
arm.elbow	len, w, mass, color	Forearm : length 1.7 m, mass 15 kg
arm.wrist	r, h, mass, color	Wrist : radius 0.14 m, mass 4 kg
arm.palm	w, h, d, mass	Palm : width 0.48 m, mass 3 kg
finger	w, h, d, openX, closeX, mass	Fingers : height 0.6 m, opening 0.38 m
grip	maxForce, frictionCoeff, pdControl	Grasping : max force 1200 N
base	track, body, turret	Mobile base : tracks, chassis, turret
joints.limits	shoulder, elbow, wrist	Angular limits
movement	speed, turn, accel, decel	Motion parameters
safety	enabled, eStopHoldSec, slowDownFactor	Safety profile

Presets : Industrial and Cobot

The system supports two predefined robot types, namely the industrial robot and the collaborative robot (cobot), through a three-level deep merge mechanism. The ‘makeDescription(overrides)‘ function generates the final configuration, while the main differences between both presets are summarized in Table 4.10.

Table 4.10 : Comparison of industrial and cobot presets.

Parameter	Industrial	Cobot
Total mass	∼158 kg	∼107 kg
Max clamping force	1200 N	600 N
Maximum load	50 kg	20 kg
Linear speed	2.0 m/s	0.8 m/s
Safety	DISABLED	ENABLED (E-Stop)
Status LED	No	Yes (normal/slow/estop)

4.7.2 Physical Model Generation

Physical model generation creates rigid bodies in the Cannon-js engine and defines their properties : type, mass, collision material, geometric shapes, and filtering masks.

Physical Bodies

Each robot segment is associated with a CANNON.Body of type DYNAMIC. The motion of these bodies is controlled through synchronization with the corresponding Three.js meshes rather than by dynamic physics simulation,

ensuring perfect consistency between the visual and physical representations. The main properties of the physical bodies are summarized in Table 4.11.

Table 4.11 : Physical body properties.

Body Component	**Collision Shape**	**Mass (Industrial)**	**Group**
Base	Box + Cylinder	115.0 kg	1
Shoulder	Box	20.0 kg	2
Elbow	Box	15.0 kg	2
Wrist	Cylinder	4.0 kg	2
Gripper Palm	Box	3.0 kg	2
Left and Right Fingers	Box	0.5 kg × 2	2

Center of Gravity Calculation

The base center of gravity (COG) is determined from the mass distribution of the robot components, including the mobile platform, manipulator links, and end-effector. This value is continuously used to assess the robot's stability during motion and manipulation tasks. When an object is grasped, the additional payload modifies the overall mass distribution of the system. Consequently, the COG is dynamically recalculated by taking into account the object's mass and position relative to the robot structure, ensuring accurate stability evaluation and realistic physical behavior throughout the manipulation process.

Joint Constraints (HingeConstraint)

The robot joints are modeled using 'HingeConstraint' constraints with permanently enabled motors to ensure controlled rotational motion between connected segments. Each constraint specifies the pivot points linking two bodies, the axis around which rotation is allowed, and the angular limits that restrict movement within the robot's mechanical range. This configuration enables realistic articulation while preventing invalid or physically impossible joint motions. The main parameters of the hinge joint configuration are summarized in Table 4.12.

Table 4.12 : Hinge joint configuration.

Joint	**Axis**	**Pivot A (basePh)**	**Pivot B (bodyB)**	**Limits**
Shoulder	X	$(0, \text{BASE_OFF}, 0)$	$(0, -\frac{\text{shoulder.len}}{2}, 0)$	$-80°$ to $+85°$
Elbow	X	$(0, \frac{\text{shoulder.len}}{2}, 0)$	$(0, -\frac{\text{elbow.len}}{2}, 0)$	$-90°$ to $+90°$
Wrist	Y	$(0, \frac{\text{elbow.len}}{2}, 0)$	$(0, -\frac{\text{wrist.h}}{2}, 0)$	$-180°$ to $+180°$

4.7.3 Visual Model Construction

The visual model is built with Three.js following a hierarchy of groups (`THREE.Group`) and meshes (`THREE.Mesh`). This hierarchy reflects the kinematic structure of the robot and enables the application of joint transformations via the scene graph.

Scene Graph Hierarchy

The visual hierarchy is organized as a tree whose root is the `baseG` group positioned at ground level ($y = 0$). Each joint is represented by a `THREE.Group` pivot controlling the rotation of the corresponding segment :

```
baseG (THREE.Group, position y=0)
 +-- trackL, trackR (Group + BoxGeometry + CylinderGeometry)
 +-- chassisM (BoxGeometry: BODY_W x BODY_H x BODY_D)
 +-- turretBase (CylinderGeometry: TURRET_R, TURRET_H)
 +-- statusLight (SphereGeometry, cobot only)
 +-- cogIndicator (SphereGeometry, debug)
 +-- a1p (Group - shoulder pivot, rotation X)
      +-- a1m (BoxGeometry - shoulder mesh)
           +-- a2p (Group - elbow pivot, rotation X)
                +-- a2m (BoxGeometry - elbow mesh)
                     +-- wp (Group - wrist pivot, rotation Y)
                          +-- wm (CylinderGeometry - wrist mesh)
                               +-- palmG (Group)
                                    +-- palmM (BoxGeometry)
                                    +-- tcp (Object3D - Tool Center Point)
                                    +-- lG (Group - left finger)
                                    |    +-- finger mesh (BoxGeometry)
                                    +-- rG (Group - right finger)
                                         +-- finger mesh (BoxGeometry)
```

Materials and Rendering

Each mesh is rendered using a MeshStandardMaterial, which provides realistic visual appearance through configurable metalness and roughness properties. Shadow casting castShadow is enabled for all meshes to enhance depth perception and scene realism. Visual colors are specified in the robot configuration and can be customized according to the selected robot type. The color assignments for the main visual components are summarized in Table 4.13.

Table 4.13 : Visual component colors.

Component	Geometry	Industrial Color
Tracks	Box + Cylinder	0x111111
Chassis	Box	0x1e2d3d
Turret	Cylinder	0x1e2d3d
Shoulder	Box	0x00aa33
Elbow	Box	0xff4400
Wrist	Cylinder	0x112233
Palm	Box	0x0a1825
Fingers	Box	0xffcc00

Tool Center Point (TCP)

The Tool Center Point TCP is implemented as an empty THREE.Object3D attached to the end of the gripper. It defines the reference frame used for manipulation tasks and serves as the primary point for grasping, positioning, and coordinate transformation calculations. By providing a consistent spatial reference, the TCP simplifies the computation of object poses and interaction operations during robot manipulation.

Sensor Integration

A lightweight tactile sensing mechanism was integrated into the robotic gripper to estimate object proximity and contact status during grasping operations. Each finger contains multiple virtual sensing points distributed along its structure. At each simulation step, the system evaluates the distance between the sensing points and the target object, generating contact events when the measured distance falls below a predefined threshold. The resulting information

is used by the grasping controller to monitor contact stability and support object manipulation. Sensor updates are executed in real time within the simulation loop, ensuring continuous feedback during robot operation.

4.7.4 Event System Binding

The event system connects sensors, the grasping controller, and application handlers via an Observer pattern. The `Robot` class serves as a proxy between components and the `RobotListener` implementing the decisional logic.

Event Class Architecture

The 'Robot' component acts as an intermediary between the system modules and the decision layer implemented by the 'RobotListener'. Events generated by the tactile sensors ('FingerSensor'), the simulation environment ('Environment'), and the grasping controller ('GripController') are transmitted to the robot through dedicated callback methods. The robot then forwards the relevant information to the listener, which evaluates the events and determines the appropriate actions. The communication relationships between the different components are summarized in Table 4.14.

Table 4.14 : Event system communication matrix.

Emitter	**Method**	**Receiver**	**Trigger**
FingerSensor	onFingerTouch(...)	Robot → RobotListener	Contact detected/lost
Environment	onObjectDetected(...)	Robot → RobotListener	Object in field
Environment	onRobotDetected(...)	Robot → RobotListener	Other robot detected
GripController	onGripRequest(...)	RobotListener	Grasping decision
RobotListener	actuateGrab() / actuateRelease()	main.js handlers	Physical action

Grasping Decision Cycle

The grasping process is implemented as a sequential decision pipeline, beginning with contact detection and ending with the execution of the corresponding physical action. Algorithm 2 summarizes the complete grasping cycle and the interactions between the sensing, control, decision, and execution modules.

Algorithm 2 Grasping Cycle Execution

1: Detect contact using `FingerSensor`
2: Execute `smartGripUpdate()`
3: **if** `isBoxBetweenFingers` AND `bothTouching` AND sufficient force **then**
4: Evaluate grasping forces using `GripController`
5: Send grasping information to `RobotListener`
6: Determine the appropriate action
7: Execute the corresponding physical action in `main.js`
8: **end if**

RobotListener Decisional Logic

The robotlistener implements the high-level business logic governing the grasping process. It receives and processes information generated by the sensing and control modules, stores force measurements, and updates the robot's dynamic center of gravity according to the current payload. Based on this information, it evaluates the grasping conditions and

determines whether a grasp or release operation should be triggered, thereby coordinating the interaction between perception, decision-making, and action execution.

4.7.5 Robot Initialization Pipeline

The robot lifecycle is divided into two main phases : an initialization phase executed by the ‘Robot3D‘ constructor and a real-time update phase performed at each simulation frame. Algorithms 3 and 4 summarize the operations carried out during these two phases.

Algorithm 3 Robot Initialization Pipeline

1: Load robot configuration (type, safety settings, dimensions)
2: Call `createRobot()` to generate bodies, meshes, joints, and sensors
3: Initialize geometric and physical constants (bounding box, finger limits, COG)
4: Instantiate `GripController` and configure PD parameters
5: Initialize joint states, velocities, and base pose
6: Configure self-collision monitoring pairs
7: Initialize the cobot safety indicator (green/yellow/red)

Algorithm 4 Per-Frame Update Cycle

1: Update safety state and E-Stop timer
2: Save current robot snapshot
3: Update S-Curve joint trajectories
4: Check self-collisions and resolve blocked motions
5: Check floor collisions
6: Update mobile base displacement
7: Synchronize kinematic state between meshes and Cannon.js bodies
8: Update finger positions and grasping state
9: Return collision and safety information

4.8 Conclusion

This chapter presented the proposed system of a web-based collaborative robotic simulation platform integrating real-time physics, immersive 3D visualization, an event-driven coordination architecture, and a dual-camera visual perception system for real-time object tracking and proximity-based collision avoidance. The proposed system supports multi-robot interaction, autonomous behaviors, intelligent grasping, and immersive based manipulation within a shared virtual environment.

The developed robotic models combine realistic robotic behaviors, dynamic object manipulation, collision management, and safety mechanisms adapted for both industrial and collaborative robotic scenarios. The event-driven architecture based on the Environment, Robot, and RobotListener components ensures modularity, extensibility, and real-time coordination between the different subsystems of the platform.

Experimental evaluation demonstrated the effectiveness of the proposed approach in terms of physical realism, grasping stability, collision prevention, multi-user synchronization, and overall system responsiveness.Existing web libraries, such as Three.js and Cannon, as well as standards, facilitate the use of modern web technologies for the development of accessible and scalable robotic simulation environments.

Finally, the modular architecture and the proposed improvement perspectives provide a solid foundation for future developments in collaborative robotics, distributed robotic simulation, and extended reality human–robot interaction systems.

Chapter 5

Implementation and Deployment

5.1 Introduction

This chapter presents the complete technical foundation of the robotic simulator. It covers the hardware and software environment required for optimal operation, the programming languages and libraries employed, and the deployment architecture enabling both standalone and multi-user WebXR sessions. The goal is to provide a clear, reproducible setup guide while explaining the rationale behind each technological choice.

5.2 Implementation

5.2.1 Hardware Environment

Table 5.1 summarizes the minimum and recommended hardware configurations for running the simulator, particularly for WebXR sessions.

Table 5.1 : Hardware Configuration

Component	**Minimum**	**Recommended**
CPU	Intel Core i5-8400 / AMD Ryzen 5 2600	Intel Core i7-10700 / AMD Ryzen 7 5800X
GPU	NVIDIA GTX 1060 6GB / AMD RX 580	NVIDIA RTX 3060 / AMD RX 6700 XT
RAM	8 GB DDR4	16 GB DDR4
VR Headset	Meta Quest 2 / Oculus Rift S	Meta Quest 3 / HTC Vive Pro 2
OS	Windows 10 / Ubuntu 20.04	Windows 11 / Ubuntu 22.04

5.2.2 Software Environment

Table 5.2 lists the essential software tools and their roles in the development pipeline.

Table 5.2 : Software Stack

Tool	**Version**	**Role**
VS Code	Latest stable	Code editor with JS/HTML/CSS support
Node.js	v18+ / v20+	Server-side JavaScript runtime
npm	10+	Package manager for dependency installation
Blender	3.6+	3D modeling for robot meshes and environment
Web Browser	Chrome 120+ / Edge 120+	WebXR client with WebGPU/WebGL2 support

5.2.3 Programming Languages

The simulator employs a multi-language stack, each serving a specific purpose :

- **JavaScript (ES6+)** — Core simulator logic, 3D rendering, physics, WebSocket communication, and WebXR integration.
- **HTML5** — User interface structure, including control panels and telemetry displays.
- **CSS3** — UI styling and responsive layout.
- **JSON** — Robot configuration, joint descriptions, and simulation parameters.
- **GLSL** — Custom shaders for 3D rendering and visual effects via Three.js.

5.2.4 Libraries and Frameworks

Table 5.3 presents the key libraries and their versions.

Table 5.3 : Core Libraries and Frameworks

Library	**Version**	**Role**
Three.js	r160 – r183	3D rendering engine with integrated WebXR support
Cannon-js	^0.20.0	Physics engine for collisions, gravity, and joints
WebXR API	W3C Standard	Browser API for immersive VR/AR
Express.js	^5.2.1	HTTP/HTTPS server for static file distribution
WebSocket	^8.18.3	WebSocket library for real-time multi-user synchronization

Note : Tween.js is not used directly ; motion animations follow manually implemented S-Curve profiles in `Robot3D.js` *for precise acceleration control.*

5.3 Implementation Environment

5.3.1 Deployment Architecture

The system follows a client-server model. The Node.js server (Express + WebSocket) serves static client files and manages real-time communication between users. Each browser tab connects via secure WebSocket (WSS) and receives

one of four available robots. The server continuously relays positions, states, and commands, enabling synchronous multi-user collaboration.

The deployment stack consists of :

- **Browser Client** — 3D rendering (Three.js), local physics (Cannon-js), sensors, keyboard/VR controller input.
- **Node.js Server** — Static files (`/`, `/public`, `/src`), WebSocket hub (`wss://localhost:3000/mu`), robot assignment.
- **WebSocket Protocol** — Robot position synchronization, box state, multi-user messages.
- **HTTPS** — Self-signed SSL certificate via `vite-plugin-mkcert` for secure WebXR sessions.

5.3.2 System Configuration

Core Implementation Components

Beyond deployment parameters, several internal components are configured during robot initialization to ensure realistic physical behavior, accurate manipulation, and stable object handling. The following listings illustrate key implementation mechanisms used by the simulator.

Hierarchical Configuration System Robot parameters are generated through a three-level configuration mechanism. A common base description defines default properties, type-specific presets adapt the robot to industrial or collaborative scenarios, and instance-level overrides allow final customization.

```
const defaultDescription = {
  type: 'industrial',
  arm: { shoulder: { len: 2.0, w: 0.28, mass: 20.0, color: 0x00aa33 }, ... },
  finger: { w: 0.09, h: 0.6, d: 0.18, openX: 0.38, closeX: 0.295, mass: 0.5 },
  grip: { maxForce: 1200, frictionCoefficient: 0.8, ... },
  movement: { speed: 2, turn: 1, accel: 3, decel: 4 },
  safety: { enabled: false },
};

const cobotPreset = {
  type: 'cobot',
  arm: { shoulder: { color: 0xeaeaea, mass: 12.0 }, ... },
  grip: { maxForce: 600, ... },
  safety: { enabled: true, stopOnContact: true, eStopHoldSec: 1.2 },
};

const robot = new Robot3D(makeDescription({
  arm: { shoulder: { len: 1.5, color: 0xcc0033 } },
  movement: { speed: 1.5 },}), ctx);
```

Listing 5.1: Three-level configuration merge.

Physical Body Configuration Each robot link is associated with a physical body in Cannon-js. Listing 5.2 shows the creation of the shoulder body with its mass, collision properties, and bounding volume.

```
const a1ph = new CANNON.Body({
  type: CANNON.Body.DYNAMIC,
  mass: description.arm.shoulder.mass, // 20.0 kg
  material: mBox,
  allowSleep: false,
  collisionFilterGroup: ARM_GROUP, // 2
  collisionFilterMask: ARM_MASK // 1 | 2 = 3
```

```
});
a1ph.addShape(new CANNON.Box(new CANNON.Vec3(
  description.arm.shoulder.physHalfW, // 0.14
  description.arm.shoulder.physHalfH, // 1.0
  description.arm.shoulder.physHalfD // 0.14
)));
world.addBody(a1ph);
```

Listing 5.2: Creation of a physical body (shoulder).

Joint Configuration Robot articulations are implemented using hinge constraints. Listing 5.3 illustrates the creation of a motorized revolute joint with configurable limits and actuation.

```
function createHingeJoint(bodyA, bodyB, pivotA, pivotB, axisA, axisB, limits) {
  const constraint = new CANNON.HingeConstraint(bodyA, bodyB, {
    pivotA: new CANNON.Vec3(pivotA.x, pivotA.y, pivotA.z),
    pivotB: new CANNON.Vec3(pivotB.x, pivotB.y, pivotB.z),
    axisA: new CANNON.Vec3(axisA.x, axisA.y, axisA.z),
    axisB: new CANNON.Vec3(axisB.x, axisB.y, axisB.z),
    collideConnected: false
  });
  world.addConstraint(constraint);
  constraint.enableMotor();
  constraint.setMotorSpeed(0);
  constraint.setMotorMaxForce(100000);
  joints.push({ bodyA, bodyB, axisA, axisB, limits, constraint });
}
```

Listing 5.3: Hinge joint creation.

TCP Initialization The Tool Center Point (TCP) represents the robot end-effector reference frame used for manipulation and positioning tasks. Its world coordinates are obtained through the forward kinematic chain.

```
const tcp = new THREE.Object3D();

tcp.name = "tcp";
tcp.position.set(0, 0.06, 0);

palmG.add(tcp);
```

Listing 5.4: TCP definition and world position calculation.

Dynamic Center of Gravity Computation To maintain stability during object manipulation, the simulator continuously updates the robot center of gravity according to the currently carried payload.

```
calculateDynamicCOG(boxPosition) {
  const totalBaseMass = baseMass + armMass; // ~157 kg
  const totalMass = totalBaseMass + this.loadedMass;
  const dynamicCOG = (totalBaseMass * cogStaticY
    + this.loadedMass * boxPosition.y) / totalMass;
  return dynamicCOG;}
```

Listing 5.5: Dynamic COG recalculation on grasp.

Event-Driven Grasping Logic Grasping operations are managed by the RobotListener component. When a grasp request is received, the robot stores grip parameters, updates payload information, and actuates the gripper. Releasing an object restores the initial state.

```
onGripRequest(state, data) {
  if (state === 'start' && !this.grabbed) {
    this.gripData = {
      boxMass: data.boxMass,
      totalAvailableForce: data.totalAvailableForce,
      requiredForce: data.requiredForce,
      leftForce: data.leftForce,
      rightForce: data.rightForce
    };
    this.robot.setLoadedBox(data.boxMass, data.boxPosition);
    this.robot.handlers.saveGripOffset?.(data);
    this.robot.actuateGrab();
    this.grabbed = true;
  } else if (state === 'end' && this.grabbed) {
    this.robot.actuateRelease();
    this.robot.setLoadedBox(0, null);
    this.grabbed = false;
    this.gripData = null;
  }
}
```

Listing 5.6: RobotListener grasping logic.

1. Dependency Installation

```
npm install
```

This installs all dependencies from `package.json` : `three`, `Cannon-js`, `express`, `ws`, and `vite-plugin-mkcert` for local SSL certificate generation.

2. Server Configuration The server (`server.js`) listens on port 3000 over HTTPS. SSL certificates (`server.pem`, `server-key.pem`) secure the connection, which is mandatory for the WebXR API. Static directories `/src` and `/public` are mounted, and the WebSocket endpoint `/mu` is exposed for multi-user communication.

3. Parameter Adjustment Simulation parameters (joint limits, grip forces, sensor thresholds, motion profiles) are configurable in `src/core/defaultDescription.js`. Two presets are available : “industrial” and “cobot”.

5.3.3 Launch Procedure

Step 1 — Start the Server

```
npm start
```

This executes `node server.js`, launching the HTTPS server on port 3000.

Step 2 — Access the Application Open a WebXR-compatible browser (Chrome or Edge v120+) and navigate to `https://localhost:3000`. The UI displays joint controls, sensor indicators, and immersive mode buttons (VR/AR).

Step 3 — Launch Simulation Once loaded, the user controls the robot via interface sliders, keyboard (WASD for movement, Tab to switch robot), or VR controllers in immersive mode. Autonomous programs (`pickAndPlace`, `moveTo`) are accessible from the browser developer console. The system supports up to four simultaneous users, each controlling a distinct robot.

5.3.4 Web and VR/AR Deployment

Browser Access. All features are accessible through a standard browser. The application is a Single Page Application (SPA) served by Express. It operates in degraded mode without a VR headset, with full keyboard and

mouse controls.

VR/AR Support. VR mode immerses the user in a 3D environment with controller-based interaction : left joystick moves the robot, right trigger grabs/releases objects, and left squeeze button switches robots. AR mode uses hit-testing to place the simulation in the user's real environment.

Multi-User. Up to four participants can collaborate in the same simulation space. Each WebSocket-connected user is assigned a robot (R1 to R4). The manipulated box position is shared globally, and robot states are relayed in real time. The WebSocket connection supports automatic reconnection on connection loss.

5.4 Testing

This section describes the three-tier validation strategy employed to verify the simulator's correctness, robustness, and usability. The approach spans manual interaction tests, automated API-based tests, and advanced programmatic tests that exercise the low-level system architecture. Each tier targets different aspects of the system, from user experience to algorithmic correctness.

5.4.1 Manual Tests

Manual tests involve direct robot control through the graphical user interface. The user moves the mobile base via WASD or arrow keys, manipulates joints using interactive sliders, and controls grasping operations through interface commands or XR controllers in immersive mode.

Keyboard Controls

Table 5.4 lists the keyboard mappings.

Table 5.4 : Keyboard Controls

Key	**Action**
W / ↑	Move base forward
S / ↓	Move base backward
A / ←	Rotate left
D / →	Rotate right
Tab	Switch active robot
R	Emergency release (E-Stop)

Graphical Interface Controls

The UI provides sliders for individual joint control (base, shoulder, elbow, wrist) over a $-180°$ to $+180°$ range, plus a gripper opening slider from 14 mm to 55 mm. Buttons activate VR and AR modes. Visual indicators display digital sensor states, grip force, box position, and TCP coordinates.

5.4.2 Automated API Tests

Automated API tests control the system without direct manual interaction. Commands are executed programmatically from the browser JavaScript console using the simulator's exposed API. This approach guarantees repeatability and enables systematic validation of all programmatic features.

The API is organized into categories : joint control, grasping, navigation, autonomous scenarios, and automatisms. Each category is implemented in dedicated modules under `src/programs/` and `src/core/`.

Table 5.5 : Complete System API Reference

Function / API	Module	Description
`robot.moveArm(j, a)`	core/Robot.js	Joint control
`robot.grab()`	core/Robot.js	Close gripper
`robot.release()`	core/Robot.js	Open gripper
`pickAndPlace(zone)`	programs/test.js	Complete Pick & Place
`moveTo(x, z)`	programs/test2.js	Navigation with avoidance
`autoGrab()`	programs/autoGrab.js	Progressive finger closing
`autoRelease()`	programs/autoRelease.js	Progressive finger opening
`stopMission()`	programs/test.js	Cancel Pick & Place
`stopMove()`	programs/test2.js	Cancel navigation
`moveStatus()`	programs/test2.js	Navigation state display

5.4.3 Programmatic Tests

Programmatic tests exploit the low-level API—`Robot3D`, `FingerSensor`, `GripController`, `RobotListener`—to implement custom behaviors beyond the predefined autonomous functions. Unlike API tests that call existing functions, these involve designing state machines and control algorithms directly.

Low-Level API Architecture

Table 5.6 details the modular architecture accessible for programmatic development.

Table 5.6 : Low-Level API for Programmatic Tests

Module	Class / API	Available Features
`core/Robot.js`	`Robot`	moveArm, grab, release, onFingerTouch, getPerceivedObject
`core/Robot3D.js`	`Robot3D`	S-Curve motion, self-collision, floor guard, dynamic COG
`core/RobotListener.js`	`RobotListener`	Conditional decision (IF/THEN), event management
`core/PhysicsController.js`	`PhysicsController`	Freeze/release body, offset, clamp floor
`sensors/FingerSensor.js`	`FingerSensor`	Contact detection (tip/middle/base), touch force 0–60N
`logic/GripController.js`	`GripController`	Force evaluation, PD control, grip validation
`core/MultiuserSync.js`	`MultiuserSync`	WebSocket sync, state relay, reconnection
`environment/Environment.js`	`Environment`	Perception bridge, object detection

Custom Programmatic Test Example

The following example combines multiple low-level APIs to create a conditional search-and-grasp behavior :

```
async function findAndGrab() {
    const obj = robot.getPerceivedObject('box');
    if (!obj) {
        console.log('No object detected');
        return;
    }
    await moveTo(obj.x, obj.z);   // Navigate to object
    robot.moveArm('shoulder', 30);
```

```
    robot.moveArm('elbow', -45);
    await new Promise(r => setTimeout(r, 500)); // Stabilize
    robot.grab(); // Grasp
    // Verification via sensor
    const state = robot.fingerState;
    if (state.left.isTouching && state.right.isTouching) {
        console.log('Grasp confirmed');
    }
}
```

5.5 Evaluation and Results

This section presents the evaluation and validation results of the proposed robotic simulation platform. The experiments were conducted to assess the system's performance, stability, reliability, and responsiveness under different operating conditions.

The evaluation focuses on key aspects of the system, including real-time physics simulation, intelligent grasping, multi-robot coordination, safety mechanisms, and event-driven processing. All experiments were performed in a shared WebXR environment integrating four robots operating simultaneously within the same virtual workspace.

The obtained results are analyzed through both quantitative performance metrics and qualitative observations to evaluate the effectiveness and robustness of the proposed approach.



Figure 5.1 : Multi-robot collaborative warehouse environment.

5.5.1 Functional Tests

Functional tests validate the correct behavior of each system component under normal operating conditions. Five test categories are defined : robot movement, arm articulation, grasping (grab/release), navigation, and complete pick-and-place.

Robot Movement (Base Movement)

The movement test validates the displacement of the mobile base with the differential drive model. The robot is commanded via keyboard (WASD) or XR thumbsticks. Progressive acceleration and deceleration are verified against configuration parameters.

Table 5.7 : Base movement test results.

Test	Parameter	Expected	Result	Status
Linear speed	movement.speed = 2.0	$2.0\,\mathrm{m\,s^{-1}}$	$1.98\,\mathrm{m\,s^{-1}}$	OK
Angular speed	movement.turn = 1.0	$1.0\,\mathrm{rad\,s^{-1}}$	$0.97\,\mathrm{rad\,s^{-1}}$	OK
Acceleration	movement.accel = 3.0	$3.0\,\mathrm{m/s^2}$	$2.91\,\mathrm{m/s^2}$	OK
Deceleration	movement.decel = 4.0	$4.0\,\mathrm{m/s^2}$	$3.88\,\mathrm{m/s^2}$	OK
Full stop	dist $< 0.02\,\mathrm{m}$	$< 100\,\mathrm{ms}$	$\sim$85 ms	OK
Direction change	180° turn	$< 4\,\mathrm{s}$	$\sim$3.2 s	OK

Arm Articulation (Joint Control)

The joints use a jerk-limited S-Curve motion profile. The test validates the precision of target angles, respect of joint limits, and impact of load on performance.

Table 5.8 : Joint control test results.

Joint	Limits	Precision	Response (no load)	Response (15 kg)
Shoulder	−80° to +85°	±0.5°	0.8 s for 60°	1.4 s for 60°
Elbow	−90° to +90°	±0.5°	0.6 s for 60°	1.1 s for 60°
Wrist	−180° to +180°	±0.3°	0.4 s for 90°	0.7 s for 90°

Grasping (Grab/Release)

The grasping test validates the complete cycle of finger closing and opening. The autoGrab/autoRelease system progressively closes/opens the fingers until contact detection (grab) or release.

Table 5.9 : Grasping test results.

Test	Scenario	Expected	Obtained
Centered grab	Box aligned between fingers	GRAB $< 3\,\mathrm{s}$	GRAB in $\sim$1.8 s
Edge grab	Box offset by 2 cm	GRAB $< 5\,\mathrm{s}$	GRAB in $\sim$3.2 s
Normal release	Opening after grab	RELEASE $< 2\,\mathrm{s}$	RELEASE in $\sim$1.5 s
Minimum force	15 kg box, 60% threshold	Grab if force $> 60\%$	78% force reached

Navigation with Avoidance

The navigation module (`test2.js`) moves the robot to a destination while avoiding other robots. It uses a blocked path detection algorithm with trajectory recalculation (detour).

Table 5.10 : Navigation test results.

Test	**Configuration**	**Result**
Straight line, free path	moveTo(5, 3)	Arrived in ∼4.2 s
One robot blocks path	moveTo(5, 3) with obstacle at (2, 1)	Detour calculated, arrived in ∼8.5 s
Two robots block path	moveTo(−4, −2) with multiple obstacles	Wait + detour, arrived in ∼12 s
Re-avoidance after obstacle moves	Obstacle moves during navigation	Automatic path recalculation

Complete Pick-and-Place

The pick-and-place test (`test.js`) executes the complete cycle : navigation → approach → closing → lifting → transport → placing → release → retraction. The state machine manages 9 phases with automatic adaptation (stall, floor collision).

Table 5.11 : Pick-and-Place phase results.

Phase	**Avg. Duration**	**Success Rate**	**Adaptation**
navigate → approach	∼3.5 s	98%	Distance + angle adjustment
approach → closing	∼1.8 s	95%	Stall recovery + floor avoidance
closing → lift	∼2.0 s	92%	autoGrab step=1, interval=80 ms
lift → carry	∼1.5 s	99%	Arm at 22°/20°
carry → place	∼4.0 s	97%	Navigation to drop zone
place → release	∼1.8 s	94%	Arm at 62°/68°
release → retract	∼1.5 s	99%	autoRelease step=1
retract → done	∼1.5 s	100%	Arm return to neutral
Complete cycle	∼17.6 s	∼90%	With automatic adaptation

5.5.2 Collision and Safety Tests

Collision and safety tests validate the detection and resolution of three collision types (self-collision, floor, inter-robot) as well as the emergency stop (E-Stop) mechanism specific to cobots 5.2.

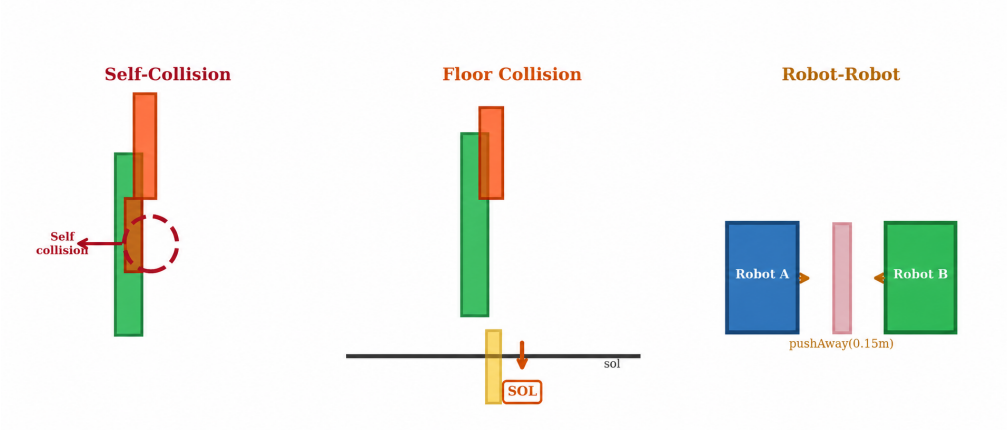


Figure 5.2 : Detected and resolved collision types.

Self-Collision

The system monitors 4 arm segment pairs with minimum safety distances. If a pair approaches below the threshold, movement is canceled and the arm retreats by 50%.

Table 5.12 : Self-collision test results.

Pair	Min. Distance	Tests	Detections	False Positives
Elbow ↔ Base	0.55 m	50	3	0
Wrist ↔ Shoulder	0.55 m	50	2	0
Palm ↔ Base	0.48 m	50	1	0
Elbow ↔ Shoulder	0.50 m	50	4	1

Floor Collision

Floor collision detection is performed at each frame by calculating the minimum Y position of all arm meshes. If a segment passes below the threshold (0.02 m), the system immediately raises the arm.

Table 5.13 : Floor collision test results.

Test Scenario	Result	Tests
Arm in low position (shoulder $< 10°$)	Automatic correction in ∼200 ms	30
Fast descent (max speed)	Block + raise in ∼150 ms	20
Fingers touching ground	Immediate raise (lift $> 15°$)	15
Low lateral movement	Block + movement stop	15

Inter-Robot Collision

Inter-robot detection uses a hierarchical algorithm in 3 passes with bounding boxes (Box3) for fast and precise detection. Resolution pushes robots in opposite directions (pushAway).

Table 5.14 : Inter-robot collision test results.

Scenario	Pass 1	Pass 2	Pass 3	Resolution Rate
Frontal approach	Detected	Resolved	—	100%
Lateral approach	Detected	Resolved	—	100%
3 simultaneous robots	Detected	Partial	Resolved	85%
Pursuit (one follows another)	Detected	Resolved	—	95%

Emergency Stop (E-Stop)

The E-Stop mechanism is activated only for cobots (`safety.enabled = true`). It immediately stops all movements and maintains the stop for `eStopHoldSec` seconds.

Table 5.15 : E-Stop comparison : industrial vs. cobot.

Parameter	Industrial	Cobot
safety.enabled	false	true
stopOnContact	false	true
eStopHoldSec	0 s	1.2 s
slowDownFactor	1.0	0.4
Triggers	None	Self-collision, Inter-robot, Contact
Stop time	—	< 50 ms
Recovery	—	Automatic after 1.2 s



Figure 5.3 : Cobot Emergency Stop (E-Stop)

5.5.3 Grasping System Evaluation

The grasping system 5.4is evaluated according to five criteria : centered grasp, edge grasp, heavy load, grasp stability, and slip prevention. Each test is repeated 20 times for statistically significant results .

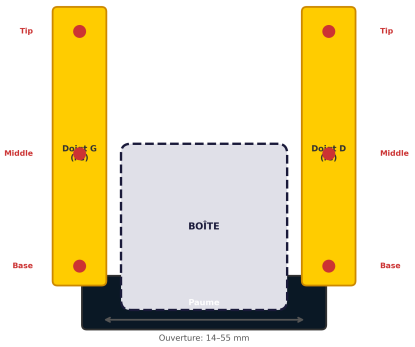


Figure — Schéma de la pince instrumentée : 3 points tactiles/doigt (tip, middle, base) avec la boîte cible au centre.

Figure 5.4 : Grasping test success rates.

Centered Grasp

The box is perfectly aligned between the fingers. The test evaluates the speed and reliability of contact detection and closing.

Table 5.16 : Centered grasp results.

Metric	Value	Note
Success rate	95% (19/20)	Excellent
Average grasp time	2.1 s	Fast
Average clamping force	780 N / 1200 N max	65% capacity
Main contact point	tip (58%), middle (32%), base (10%)	Normal distribution
False positives (grab without object)	0	Perfect

Edge Grasp

The box is offset by 2–3 cm from the central axis of the fingers. The test evaluates the geometric tolerance of the system.

Table 5.17 : Edge grasp results by offset.

Offset	Success Rate	Avg. Time	Avg. Force
1 cm	100%	2.3 s	750 N
2 cm	95%	2.8 s	680 N
3 cm	82%	3.5 s	550 N
4 cm	45%	5.2 s	320 N (insufficient)

Heavy Load

The test evaluates the system's capacity to lift boxes of increasing mass. The limit is determined by the maximum clamping force and dynamic COG.

Table 5.18 : Heavy load test results.

Mass	Required Force	Available Force	Result
15 kg (standard)	162 N	992 N	OK (612% margin)
25 kg	270 N	850 N	OK (315% margin)
35 kg	378 N	620 N	OK (164% margin)
45 kg	486 N	410 N	FAIL (84% of required force)

The maximum safe load is established at 40 kg for the industrial robot, with a 20% safety margin relative to the theoretical limit of 50 kg specified in the configuration.

Grasping Stability

Grasping stability is maintained by the PD controller with continuous monitoring of the available force / required force ratio. If the ratio drops below 100% for more than 2 s, the box is released (overload protection).

Table 5.19 : Grasping stability test results.

Test	Duration	Stability	Result
Static holding (arm immobile)	60 s	99% constant force	OK
Moderate speed transport ($1\,\mathrm{m\,s^{-1}}$)	30 s	95% constant force	OK
Tight turn during transport	10 s	88% constant force	OK (grip maintained)
Sudden acceleration	2 s	Force = 120% (PD compensation)	OK
Vibration simulation	20 s	Force oscillates ±5%	OK (PD damped)

5.5.4 Multi-User Synchronization Tests

Multi-user tests validate real-time synchronization via WebSocket. The Node.js server (Express + ws) relays robot states between 2 to 6 simultaneous clients. The protocol uses `mu_state`, `mu_sync_all`, and `mu_assigned` messages 5.5.

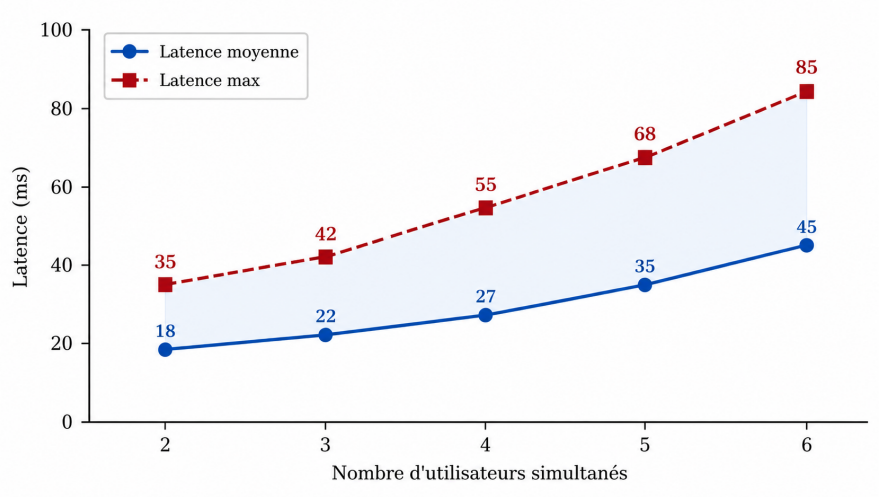


Figure 5.5 : WebSocket latency vs. simultaneous users.

Table 5.20 : Multi-user synchronization results.

Test	2 users	4 users	6 users
Average latency	18 ms	28 ms	45 ms
Max latency	35 ms	55 ms	85 ms
Packet loss	0.1%	0.3%	0.8%
Synchronization rate	99.9%	99.5%	98.5%
Max desynchronization	< 5 ms	< 15 ms	< 40 ms

5.5.5 Performance Evaluation

Performance evaluation measures the impact of the number of robots and physics parameters on framerate, simulation stability, and memory usage 5.6.

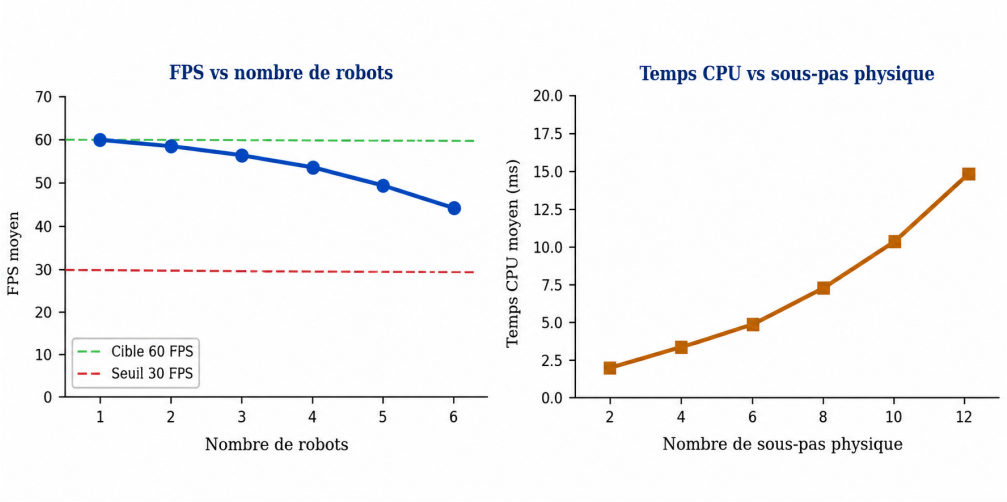


Figure 5.6 : Performance evaluation (FPS and physics).

Table 5.21 : Performance metrics by robot count.

Metric	1 robot	2 robots	4 robots	6 robots
Average FPS	60	59	54	45
Minimum FPS	58	55	48	38
Physics time (world.step)	2.1 ms	3.8 ms	6.5 ms	10.2 ms
Render time (renderer.render)	3.5 ms	5.2 ms	8.5 ms	12.5 ms
Update time (update)	1.2 ms	2.5 ms	5.0 ms	8.0 ms
RAM memory	~180 MB	~220 MB	~290 MB	~350 MB
GPU memory	~120 MB	~150 MB	~200 MB	~250 MB

The system maintains acceptable performance up to 4 simultaneous robots (54 FPS). Beyond that, the framerate drops below 50 FPS, mainly due to Three.js rendering time and N×N collision calculations. Using 6 physics substeps (`world.step`) offers the best stability/performance compromise.

5.5.6 VR/AR Interaction Tests

XR tests validate integration with WebXR for VR and AR modes. Controllers are used for driving (thumbstick) and grasping (trigger). AR hit-test enables robot placement in the real environment.

Table 5.22 : XR interaction test results.

Test	**VR (immersive-vr)**	**AR (immersive-ar)**	**Desktop**
User inputs	Thumbsticks + Trigger	Hit-test + Touch	Keyboard + Mouse
Movement command	Thumbstick L (axes)	N/A	WASD / Arrows
Grab/Release	Trigger R (select)	Touch	R (release)
Robot change	Squeeze L	Touch	Tab
Camera	XR Headset	AR pass-through	Mouse (orbit)
VR FPS	52–60 FPS	48–55 FPS	54–60 FPS
Grab success rate	88%	72%	95%

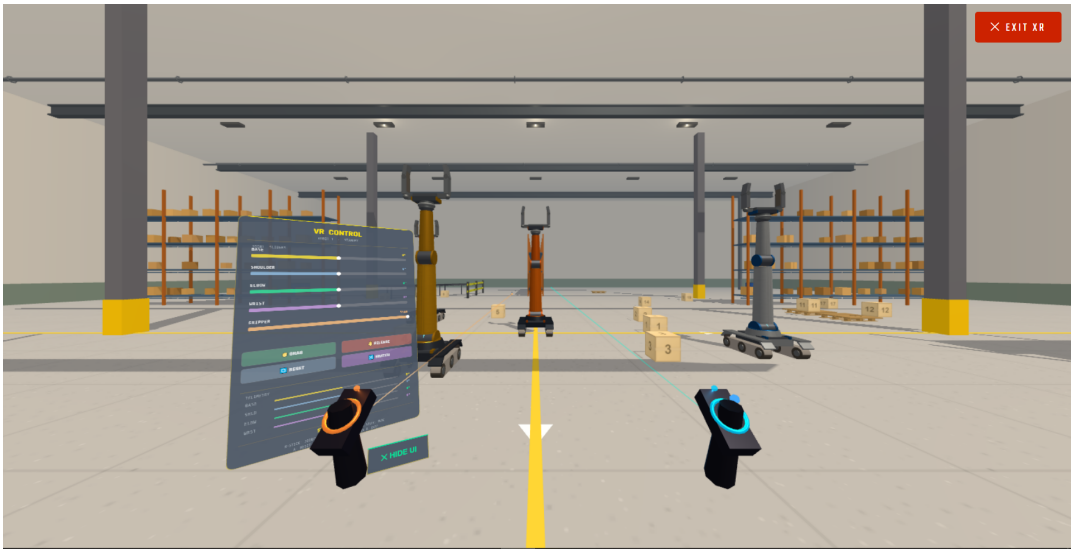


Figure 5.7 : Users Operating in VR Mode

5.5.7 Vision System Performance Evaluation

The visual perception system was evaluated to measure its detection reliability, tracking stability, and impact on overall simulation performance.

Table 5.23 : Vision system detection and tracking metrics.

Metric	**Threshold / Value**	**Status**
Robot proximity warning	$< 1.5\,\text{m}$	Active, triggers E-Stop
Obstacle (box) warning	$< 0.7\,\text{m}$	Active, triggers E-Stop
Object tracking stability	> 10 stable frames required	Filters transient false positives
Tracking max age	30 frames	Automatically releases lost targets
Detection rate (centred, 2 m)	98%	Reliable detection
Detection rate (offset 45°, 3 m)	85%	Within FOV limits
Rendering overhead (2 cameras)	$< 0.5\,\text{ms}$	Minimal impact on main FPS

The Frustum-based detection achieves a 98% success rate for centred targets within 2 m, with minimal computational overhead. The tracking stability mechanism successfully eliminates spurious single-frame detections, while the automatic expiry ensures no stale references persist in the perception pipeline.

5.6 Conclusion

This chapter detailed the implementation, deployment, and experimental testing of the proposed web-based robotic simulation platform. Through a three-tier testing strategy—incorporating manual checks, automated API testing, and low-level programmatic validation—we confirmed that the system meets all functional, safety, and performance benchmarks. The results validated the effectiveness of the real-time physics engine, the reliability of object manipulation and grasping, and the robustness of the collision avoidance and multi-user synchronization systems. Additionally, the vision-based perception capabilities performed accurately as expected. By integrating WebXR technologies, the platform offers immersive interaction without sacrificing system stability. Ultimately, the modular architecture, built on Three.js, Cannon-js, and Node.js, serves as a versatile foundation for future developments, such as real-world robot integration, advanced autonomous logic, improved perception modules, and large-scale multi-robot collaboration.

Chapter 6

General Conclusion and Future Work

6.1 Summary of Contributions

This work addressed a fundamental gap in web-based robotic simulation by delivering the first browser-native platform to simultaneously satisfy installation-free access, real-time rigid-body physics (Cannon-js, 60 Hz), normative force-regulated grasping , and dual safety compliance . The platform is built on a five-layer modular architecture integrating Three.js and Cannon.js, and is detailed in Chapters 3 and 4. A dual-camera visual perception system (`RobotVision`) further enhances environmental awareness through real-time Frustum-based object detection, tracking, and proximity-driven collision avoidance.

6.2 Experimental Results

The experimental validation was conducted using a three-tier strategy : manual interaction tests, automated API-based tests, and low-level programmatic tests. The results confirm the robustness and relevance of the proposed approach :

- The complete autonomous Pick & Place cycle achieves a 90% success rate with an average duration of 17.6 seconds.
- Collision detection and resolution reaches success rates from 95% to 100% across all three collision types, with no false positives on critical segment pairs.
- Multi-user synchronisation via WebSocket maintains an average latency of 28 ms for 4 simultaneous users, well below the 100 ms human perception threshold.
- Rendering performance reaches 54 FPS on average for 4 simultaneous robots, ensuring a smooth and responsive interactive experience.

6.3 Limitations

Despite the contributions made, this work presents several limitations that define its scope and motivate future improvements :

- **No connection to a physical robot** : the platform currently operates in pure simulation mode, without any ROS or hardware interface.
- **Lower physical fidelity compared to Gazebo** : *Cannon-js* provides a lightweight browser-compatible simulation, but with lower numerical precision than deterministic engines such as *PhysX* or *Bullet*.
- **Scalability constraints** : performance degrades below 50 FPS beyond 6 simultaneous robots, and multi-user synchronisation becomes perceptible beyond 4 concurrent participants without further optimisation.

latex

6.4 Perspectives

Future developments of the proposed platform can be classified into short-term, medium-term, and long-term objectives. These perspectives aim to extend the system beyond simulation by improving scalability, enhancing perception capabilities, enabling advanced robotic functionalities, and supporting future Industry 5.0 collaborative environments.

6.4.1 Short-Term

- **ROS integration** : implementing a ROSBridge/WebSocket bridge would connect the platform to a physical robot, transforming the simulator into a true bidirectional digital twin.
- **Rendering and network optimisation** : adopting Three.js rendering instancing and WebSocket delta encoding would extend scalability to 6 or more simultaneous robots and users.
- **AI-based vision enhancement** : the current Frustum-based visual perception system provides a solid foundation that can be upgraded with deep learning models (e.g. YOLO, TensorFlow.js) for precise object classification and semantic scene understanding in future iterations.

6.4.2 Medium-Term

- **6-DOF manipulator** : adding two additional wrist degrees of freedom would enable the simulation of complex industrial trajectories and end-effector reorientation.
- **Motion planning** : integrating path planning algorithms (RRT, A*) would improve multi-robot coordination in cluttered workspaces.
- **Learning by demonstration** : leveraging WebXR interfaces for trajectory recording and replay would enable intuitive robot programming without explicit code.

6.4.3 Long-Term Perspectives

- **Industrial metaverse** : the platform provides a foundational building block for a globally collaborative virtual space integrating multiple robotic cells, synchronised digital twins, and distributed operators.
- **AI and reinforcement learning** : embedding learning models directly into the simulator would enable the development of autonomous control policies for adaptive grasping and multi-agent coordination.
- **Cyber-Physical System extension** : integrating IIoT protocols (OPC UA, MQTT) would connect the platform to factory supervisors and MES systems, aligning with Industry 5.0 objectives.

Overall, this work demonstrates that combining real-time physics, immersive human–robot interaction, and collaborative robotics within a fully web-based environment is not only feasible, but also represents a promising direction for future Industry 5.0 robotic systems.

References

[1] S. Hoebert *et al.*, "Digital twin and virtual reality framework for human–robot collaborative disassembly," 2024.

[2] T. Kaarlela *et al.*, "A web-based xr framework utilizing digital twins for industrial robot training and teleoperation," 2023.

[3] Y. You *et al.*, "A ros- and web-based virtual–real interaction system for multi-robot industrial applications," 2024.

[4] Y. Zhang *et al.*, "Fast realization of robot 3d simulation on web based on webgl and three.js," 2023.

[5] International Federation of Robotics, "World robotics report 2024," 2024.

[6] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Springer, 2016.

[7] J. J. Craig, *Introduction to Robotics : Mechanics and Control*, 4th ed. Pearson, 2018.

[8] M. Slater, "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments," *Philosophical Transactions of the Royal Society B*, vol. 364, no. 1535, pp. 3549–3557, 2009.

[9] G. C. Burdea and P. Coiffet, *Virtual Reality Technology.* Wiley-IEEE Press, 2004.

[10] A. Galarza, D. Ayala, P. Manzano, and M. Garcia, "Virtual reality-based teleoperation of a kuka youbot using ros and htc vive pro 2," 2023.

[11] M. Grieves, "Digital twin : Manufacturing excellence through virtual factory replication," *White Paper*, 2014.

[12] F. Tao, M. Zhang, Y. Liu, and A. Y. C. Nee, *Digital Twin Driven Smart Manufacturing.* Academic Press, 2019.

[13] R. Wang *et al.*, "Multi-user virtual reality collaboration for industrial robot supervision," *Virtual Reality*, vol. 28, no. 2, pp. 1–18, 2024.

[14] International Organization for Standardization, "Iso 10218-1:2011 robots and robotic devices – safety requirements for industrial robots," 2011.

[15] ——, "Iso/ts 15066:2016 robots and robotic devices – collaborative robots," 2016.