

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Intelligent Picking System Assisted by Augmented Reality and Artificial Intelligence with Edge Processing

Daniel Filipe Subtil Mondragão Martins



Master's Degree in Electrical and Computer Engineering

Supervisor: Marcelo Roberto Petry

July 21, 2026

Intelligent Picking System Assisted by Augmented Reality and Artificial Intelligence with Edge Processing

Daniel Filipe Subtil Mondragão Martins

Master's Degree in Electrical and Computer Engineering

Approved in oral examination by the committee:

President: Prof. Daniel Gouveia Costa

Examiner: Prof. Eurico Farinha Pedrosa

Supervisor: Prof. Marcelo Roberto Petry

July 21, 2026

Abstract

Industrial order picking and collaborative manipulation increasingly require robot systems that can adapt to changing objects, layouts, and operator instructions. Vision-language-action policies offer a promising path towards more flexible robot control, but direct speech-to-action execution can be difficult to trust in shared workspaces. Spoken commands may be ambiguous, visually similar objects may be present, and the operator may not be able to inspect what the system understood before the robot starts acting. This dissertation addresses that problem by proposing an extended-reality-assisted intelligent picking workflow in which natural-language intent is first grounded in the visual scene, made visible to the operator, and validated before being passed to a robotic execution branch.

The proposed system connects a Magic Leap 2 extended reality client, an edge-side perception server, and a simulation-based robotic execution environment. The headset captures speech, RGB-D observations, camera metadata, and operator validation actions. The edge server manages interaction sessions, extracts the target object from the command, performs language-conditioned two-dimensional segmentation, estimates a world-space target position, generates object-centric three-dimensional reconstruction artifacts, and serves the resulting hypotheses back to the extended reality interface. The operator can then inspect the transcript, segmentation outline, centroid, reconstructed object, and validation controls before confirming, refining, correcting, or cancelling the result. Once accepted, the validated task state is evaluated in a digital twin using a KUKA/Robotiq pick-and-place environment and a fine-tuned vision-language-action policy.

The validation covered session integration, speech-to-text intent capture, target segmentation, centroid estimation, three-dimensional reconstruction, operator validation, simulated vision-language-action execution, and failure-mode analysis. The results show that the complete perception-to-simulation workflow is functional under controlled conditions. Session integration remained consistent across asynchronous stages, and the operator validation interface was functional in all tested trials. The perception branch was most reliable with short object-focused commands, clearly visible targets, close-range depth observations, and favourable reconstruction viewpoints. In simulation, the dataset bottle achieved 74% success after object-specific fine-tuning and 81% after mixed fine-tuning, while the reconstructed SAM 3D bottle remained more difficult, reaching 44% with the final mixed policy. The expanded-workspace test showed limited spatial generalisation, with success dropping to 20% overall. These results confirm the feasibility of an inspectable extended reality–edge–simulation workflow, while also identifying the main limitations: language ambiguity, segmentation uncertainty, reconstruction scale and orientation errors, reconstructed-asset simulation fidelity, policy overfitting, and the remaining sim-to-real gap.

Resumo

As operações de picking e manipulação colaborativa em contexto industrial exigem cada vez mais sistemas robóticos capazes de se adaptar a alterações nos objetos, no espaço de trabalho e nas instruções dos operadores. As políticas de visão-linguagem-ação representam uma possibilidade promissora para tornar o controlo robótico mais flexível, mas a execução direta de comandos por voz pode ser difícil de validar em espaços partilhados. Uma instrução falada pode ser ambígua, podem existir vários objetos visualmente semelhantes, e o operador pode não conseguir perceber o que o sistema interpretou antes de o robô iniciar a ação. Esta dissertação aborda esse problema através de um fluxo de trabalho de picking assistido por realidade estendida, no qual a intenção em linguagem natural é primeiro ancorada na cena visual, apresentada ao operador e validada antes de ser enviada para a componente de execução robótica.

O sistema desenvolvido integra um cliente de realidade estendida em Magic Leap 2, um servidor de perceção no edge e um ambiente de execução robótica em simulação. Os óculos capturam o comando de voz, observações RGB-D, metadados da câmara e ações de validação do operador. O servidor edge gere as sessões de interação, extrai o objeto-alvo a partir do comando, executa segmentação bidimensional condicionada por linguagem, estima a posição do alvo no espaço, gera artefactos de reconstrução tridimensional centrados no objeto e devolve essas hipóteses à interface de realidade estendida. O operador pode então inspecionar a transcrição, o contorno segmentado, o centroide, o objeto reconstruído e os controlos de validação antes de confirmar, refinar, corrigir ou cancelar o resultado. Depois de aceite, o estado validado da tarefa é avaliado num gémeo digital com um ambiente de pick-and-place KUKA/Robotiq e uma política de visão-linguagem-ação ajustada para a tarefa.

A validação incluiu integração por sessão, captura da intenção por fala, segmentação do alvo, estimação do centroide, reconstrução tridimensional, validação pelo operador, execução simulada com visão-linguagem-ação e análise dos principais modos de falha. Os resultados mostram que o fluxo completo, desde a perceção até à simulação, é funcional em condições controladas. A integração por sessão manteve-se consistente entre módulos assíncronos, e a interface de validação do operador funcionou em todos os ensaios testados. A componente de perceção foi mais fiável com comandos curtos e focados no objeto, alvos bem visíveis, observações de profundidade a curta distância e pontos de vista favoráveis à reconstrução. Na simulação, a garrafa do dataset atingiu 74% de sucesso após fine-tuning específico e 81% após fine-tuning misto, enquanto a garrafa reconstruída por SAM 3D continuou a ser mais difícil, chegando a 44% com a política final. O teste com uma área de trabalho expandida mostrou uma generalização espacial limitada, com a taxa de sucesso global a descer para 20%. Estes resultados demonstram a viabilidade de um fluxo realidade estendida–edge–simulação inspecionável, mas também evidenciam limitações importantes: ambiguidade linguística, incerteza na segmentação, erros de escala e orientação na reconstrução, fidelidade física dos objetos reconstruídos em simulação, sobreajuste da política aos dados de demonstração e a diferença ainda existente entre simulação e execução física.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a globally recognised framework for addressing social, economic, and environmental challenges. The 17 goals cover areas such as poverty reduction, health, education, equality, sustainable production, climate action, innovation, and decent work. Within this framework, the present dissertation is mainly aligned with SDG 8 (Decent Work and Economic Growth), SDG 9 (Industry, Innovation and Infrastructure), and SDG 12 (Responsible Consumption and Production).

This dissertation proposes an Extended Reality assisted, edge-enabled, Digital Twin based workflow for intelligent robotic picking. The system allows an operator to express an object-level task through speech, inspect the perceived target and reconstructed object in an Extended Reality interface, validate or correct the interpretation, and then evaluate the accepted task in a simulated robotic cell. This contributes to industrial innovation by combining human-in-the-loop Extended Reality, perception, object reconstruction, simulation, and vision-language-action policies in a single prototype. It also supports safer and more accessible human-robot interaction by keeping the operator involved before execution, rather than relying on direct speech-to-action robot behaviour. Finally, the use of simulation and digital validation can reduce unnecessary physical trials during development and reconfiguration, contributing indirectly to more efficient use of equipment and resources.

The specific Sustainable Development Goals considered in this dissertation are:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 12 Ensure sustainable consumption and production patterns.

Table 1 summarises how the proposed system relates to selected targets within these goals and indicates example indicators that could be used to assess its contribution in future deployments.

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.2	The proposed workflow supports technological upgrading and productivity in robotic picking by enabling faster task specification, perception-assisted object grounding, and simulation-based validation before execution.	Reduction in task programming or reconfiguration time; number of tasks configured by non-expert operators; simulated task success rate after fine-tuning; reduction in manual intervention during task setup.
	8.8	The human-in-the-loop validation stage allows the operator to inspect and correct the system interpretation before robot execution, supporting safer and more controlled human-robot interaction.	Number of prevented incorrect executions; reduction in unsafe or ambiguous robot commands; operator workload and perceived safety questionnaires; number of near-miss events during physical deployment.
9	9.4	The Digital Twin and simulation branch allow tasks to be tested and adjusted before physical deployment, reducing unnecessary physical trials and supporting more efficient use of robotic infrastructure.	Number of physical trials avoided through simulation; reduction in commissioning iterations; energy or cycle-time per successful task; ratio of successful simulated tasks transferred to the physical cell.
	9.5	The dissertation contributes to research and innovation in intelligent industrial robotics by integrating Extended Reality, edge perception, 3D reconstruction, operator validation, and VLA-based robotic execution.	Number of integrated prototype modules; validation success rates across perception and simulation stages; publications, demonstrations, or workshop participation; reusable software components and datasets.
12	12.2	The use of a Digital Twin for validation can improve resource efficiency during robot task development by reducing trial-and-error directly in the physical cell.	Reduction in failed physical runs during task development; reduction in time and resources spent during commissioning; number of task variants evaluated in simulation before physical testing.
	12.5	Simulation-first testing and operator validation may reduce waste caused by incorrect object handling, failed manipulation attempts, or unnecessary physical prototyping during robotic cell re-configuration.	Number of damaged or discarded objects during deployment tests; reduction in failed physical manipulation attempts; reduction in prototype iterations requiring physical material changes.

Table 1: Contribution of the dissertation to selected UN Sustainable Development Goals.

Acknowledgements

I would like to express my sincere gratitude to Professor Marcelo Roberto Petry for his supervision, guidance, and continuous support throughout this dissertation. His feedback and availability were essential during the development of this work.

I am also deeply grateful to João Graça Martins for all the help provided during the implementation of the system, from technical discussions and debugging to the many practical decisions needed to make the prototype work. His support was fundamental throughout this process.

I would like to thank Guilherme Costa, Guilherme Osório, Luís Leitão, and Rui Martins for their companionship, support, mutual help, and exchange of ideas during this period. This experience would not have been the same without the many conversations, shared difficulties, and laughs along the way.

I also thank the remaining members of iiLab for their kindness, availability, and for creating a welcoming and collaborative environment during the development of this dissertation.

Finally, I would like to thank my family and friends outside the thesis context for their patience, encouragement, and constant support. Their presence was essential throughout this journey.

Daniel Martins

Contents

1	Introduction	1
1.1	Motivation and Context	1
1.2	Objectives and Research Questions	3
1.3	Contributions	5
1.4	System Overview	5
1.5	Dissertation Structure	7
2	Background	8
2.1	Collaborative workcells in industry	8
2.2	Augmented Reality	10
2.3	Edge-assisted perception and scene representation	13
2.4	Foundation models for object segmentation and 3D reconstruction	17
2.5	From LLM and VLM to VLA policies for robot control	20
2.6	Summary	23
3	Literature Review	25
3.1	XR interfaces for HRC and intent inspectability	25
3.2	XR-assisted picking and logistics	26
3.3	Edge-assisted perception and RGB-D scene grounding	27
3.4	Language-conditioned segmentation and 2D target grounding	29
3.5	Object-centric 3D reconstruction and spatial validation	31
3.6	Natural language, mixed reality and grounded robot programming	33
3.7	VLA policies and simulation-based adaptation	34
3.8	Synthesis and gap addressed by this dissertation	36
4	Methodology and System Design	39
4.1	System Design	39
4.2	Design Principles	43
4.3	System Decomposition	45
4.4	Validation Strategy	46
4.5	Architectural and Technological Design Choices	48
4.6	Chapter Summary	50
5	Implementation	51
5.1	Implementation Overview and Deployment Configuration	51
5.2	Operator Intent Acquisition	57
5.3	Target Identification and Grounding	59
5.4	Operator Validation Interface	65

5.5	Simulation Asset and VLA Training Pipeline	69
5.6	Simulated Robotic Execution	72
5.7	Summary	74
6	Validation and Tests	76
6.1	Validation Overview	76
6.2	Session Integration Validation	77
6.3	Speech-to-Text Validation	78
6.4	Target Segmentation Validation	79
6.5	Centroid Validation	81
6.6	Three-Dimensional (3D) Reconstruction Validation	82
6.7	Operator Validation	86
6.8	Simulation and VLA Validation	87
6.9	Failure Mode Analysis	90
6.10	Chapter Summary	91
7	Conclusions and Future Work	93
7.1	Conclusions	93
7.2	Answering the Research Questions	95
7.3	Main Contributions	96
7.4	Limitations	97
7.5	Future Work	98
7.6	Final Remarks	100
	References	101

List of Figures

1.1	High-level XR-assisted picking architecture	6
2.1	Levels of human–robot interaction	9
2.2	Extended reality continuum	11
2.3	Progression from image-space perception to object-level 3D representation. . . .	19
2.4	Conceptual progression from language understanding to robot action.	22
3.1	SAM 3 concept segmentation	30
3.2	SAM 3D object-centric reconstruction	32
4.1	Closed-loop architecture diagram	40
5.1	Implemented information flow	56
5.2	Operator intent acquisition in XR	58
5.3	Segmentation and centroid feedback	63
5.4	Reconstruction artifacts and XR mesh loading	64
5.5	Refinement and manual correction	67
5.6	Final target pose specification	68
5.7	Examples of simulated KUKA pick-and-place execution outcomes.	74
6.1	Qualitative examples of 3D reconstruction behaviour.	85

List of Tables

1	Contribution of the dissertation to selected UN Sustainable Development Goals. .	iv
2.1	From sensor data to scene representation in XR-assisted robotic workcells.	14
2.2	Common visual outputs used to build object-level representations.	18
2.3	Progression from language understanding to robot action.	22
3.1	Systematic comparison of representative works and system classes against the inspectable spoken-command picking workflow.	37
4.1	Main system modules and exchanged data products.	45
4.2	Architectural and technological choices supporting the proposed pipeline.	48
5.1	Main technologies used in the implemented system.	52
5.2	Runtime interface contracts for target identification and grounding	60
5.3	Session states in the XR validation workflow.	66
5.4	Simulation asset and VLA training pipeline.	71
6.1	Summary of the validation groups.	77
6.2	STT validation results.	79
6.3	Segmentation validation conditions.	80
6.4	Segmentation validation results.	80
6.5	Centroid validation results by distance from the headset.	81
6.6	3D reconstruction validation conditions.	83
6.7	3D reconstruction validation results.	83
6.8	Operator validation test.	86
6.9	Fine-tuning stages used for simulated VLA validation.	88
6.10	Simulated VLA evaluation across fine-tuning stages.	89
6.11	Effect of testing outside the trained workspace region.	89
6.12	Main failure modes considered during validation.	91

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
6DoF	six Degrees of Freedom
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
AV	Augmented Virtuality
CAD	Computer-Aided Design
CV	Computer Vision
DT	Digital Twin
GLB	Binary glTF
GPU	Graphics Processing Unit
HMD	Head-Mounted Display
HRC	Human–Robot Collaboration
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
ISO	International Organization for Standardization
LiDAR	Light Detection and Ranging
LLM	Large Language Model
MR	Mixed Reality
PbD	Programming by Demonstration
PLY	Polygon File Format
RGB	Red, Green, and Blue
RGB-D	Red, Green, Blue and Depth
RGBA	Red, Green, Blue, and Alpha
ROS	Robot Operating System
RT-2	Robotics Transformer 2
SAM	Segment Anything Model
SDK	Software Development Kit
STT	Speech-to-Text

ToF	Time-of-Flight
VLA	Vision–Language–Action
VLM	Vision–Language Model
VR	Virtual Reality
XR	Extended Reality

Chapter 1

Introduction

1.1 Motivation and Context

Manufacturing is shifting toward highly customized, fast-changing production, where humans and robots increasingly work side by side [1]. This shift is often framed through the evolution from Industry 4.0 to Industry 5.0, which provides a useful lens to motivate why interaction quality in Human–Robot Collaboration (HRC) matters in practice.

Industry 4.0 is typically associated with the transition to *smart manufacturing*: the integration of cyber-physical systems, pervasive sensing, industrial connectivity, and data-driven automation across the factory [2]. Its core characteristics include vertical and horizontal integration of production systems, real-time visibility of processes, and the ability to reconfigure lines for mass customization through digitized workflows. In this paradigm, robots and automation are leveraged primarily to improve productivity, quality consistency, and flexibility, supported by continuous monitoring and optimization.

Industry 5.0 builds on this digital foundation but shifts the emphasis from technology-driven optimization to broader goals such as human-centricity, resilience, and sustainability [3]. Rather than seeking automation for its own sake, Industry 5.0 highlights the complementary strengths of humans and machines: humans provide context awareness, critical thinking, judgment, and exception handling, while robots contribute precision, repeatability, and sustained throughput. This framing makes human–robot collaboration a central production model, especially in tasks that demand both adaptability and consistent execution under safety and quality constraints [1], [3].

However, effective HRC in shared workspaces is still limited by interaction and communication challenges. Operators often need to understand robot state, intent, and upcoming motion to coordinate safely and efficiently; when this information is implicit or poorly communicated, it can increase cognitive load, reduce trust, and lead to hesitation, errors, or near-misses. At the same time, robots may struggle with ambiguity in human instructions and with handling exceptions in dynamic environments, which can force frequent interruptions and confirmations that slow down work and reduce perceived fluency [1], [4].

A related practical constraint is that setting up and adapting robot behavior in real factories still commonly relies on skilled personnel and established programming interfaces (e.g., teach pendants and waypoint-based configuration). These methods are reliable, but they can be time-consuming when tasks, layouts, or product variants change frequently, and they do not always make the robot's intent and next actions explicit to the operator at the point of work. As a result, there is a strong motivation for interaction layers that reduce reprogramming overhead and make robot state and intent easier to perceive during execution, while remaining compatible with the same safety envelope.

These limitations motivate interfaces that present robot state, intended actions, and task-relevant information directly at the point of work, instead of requiring the operator to infer them from external screens, teach-pendant states, or implicit robot motion. In this context, Augmented Reality (AR) has emerged as a practical interface layer for HRC, helping operators understand robot state and intent and interact with robotic systems more effectively and safely. Recent surveys of AR in industrial HRC report growing adoption and promising impacts on operator performance, task awareness, and perceived safety, while noting integration, calibration, and ergonomics as continuing challenges on the shop floor [4]–[6].

The relevance of AR in this dissertation is therefore twofold. First, it provides a visual interface for presenting robot state, target hypotheses, and validation cues directly in the shared workspace. Second, because it runs on a head-worn device, it can also act as a sensing layer: the headset provides access to the operator's pose, and, depending on the device, hand position or gestures, as well as an egocentric camera view of the workcell. This enables the system to capture interaction-relevant data that would otherwise require additional infrastructure or would be difficult to obtain reliably in frequently reconfigured workcells.

This role can be situated within the mixed reality continuum [7]. In this continuum, AR refers to interactive digital content that is spatially registered to the physical world and presented without replacing the real environment. In other words, AR complements perception and decision-making by overlaying task-relevant information, such as annotations, guidance cues, and intent visualizations, onto the user's view of the workcell [8].

For such overlays to be useful in collaborative workcells, two constraints dominate the design space: spatial accuracy and end-to-end latency. Optical see-through Head-Mounted Displays (HMDs) can provide guidance and awareness, but empirical evaluations commonly report centimetre-level spatial errors under typical conditions, which is acceptable for coarse guidance but limiting for tight alignment near objects, bins, fixtures, or grippers [9]. This motivates hybrid anchoring strategies, such as fiducials or external references, when tasks demand higher repeatability at the working pose [10]. On the latency side, delayed perception or rendering can degrade perceived overlay stability and reduce the fluency of interaction. Edge-assisted architectures address this issue by moving heavier perception models from the wearable device to a nearby server, preserving responsive interaction on the headset while keeping computationally demanding processing close to the workcell [11]. In addition to compute and latency, edge-local processing can reflect industrial constraints around data governance: visual streams, depth maps, and process context

may be considered sensitive, motivating on-premise processing that keeps data within the factory network rather than relying on external cloud services.

Beyond local pose estimation, **AR** combined with Digital Twins (**DTs**) can expose robot state, planned paths, and intent on the shop floor, improving shared situational awareness in multi-user, multi-robot settings. In such setups, a brief alignment step (e.g., fiducial-based registration) can be sufficient to relate the robot and **HMD** coordinate frames before relying on each system's own localization [12], [13]. In intra-logistics and order picking specifically, **AR pick-by-vision** approaches have been explored for years to reduce cognitive load and streamline navigation and selection, and more recent work combines **AR** guidance with learning-based perception for item recognition [14]–[16].

Recent progress in Large Language Models (**LLMs**), Vision–Language Models (**VLMs**), and Vision–Language–Actions (**VLAs**) has opened new possibilities for natural-language robot control. In principle, a robot policy can use visual observations and language commands to generate manipulation actions. However, in collaborative picking scenarios, direct speech-to-action control can be problematic. Natural-language commands may be ambiguous, several visually similar objects may be present, and the operator may not be able to inspect what the model understood before the robot starts moving. This motivates architectures in which target grounding and human validation are explicitly placed before robot execution.

This dissertation focuses on the integration of **AR** interaction, edge-side visual grounding, operator validation, and **VLA**-based robotic execution for intelligent picking. The objective is not to propose a new segmentation model, Three-Dimensional (**3D**) reconstruction model, or robot control architecture, but to design a methodology for connecting these components in a human-centred workflow where the system's interpretation of a natural-language command is made visible and correctable before execution.

1.2 Objectives and Research Questions

The main goal of this dissertation is to develop an **AR**-assisted intelligent picking system that makes language-conditioned robot execution more transparent by grounding the requested target in the visual scene and validating this interpretation before robotic action.

This goal is addressed through a workflow in which an optical see-through **HMD** captures operator intent and egocentric visual observations, while an edge server performs visual grounding and object-centric reconstruction. The resulting Two-Dimensional (**2D**) and **3D** target hypotheses are presented to the operator through spatial **AR** feedback before being forwarded to a **VLA**-based robotic execution branch. The work delivers: (i) a system architecture that connects **AR** interaction, edge-side perception, operator validation, and **VLA**-driven execution; (ii) a working prototype of the **AR**–edge perception and validation workflow; and (iii) a simulation-based evaluation of the **VLA** execution branch in a representative robotic workcell. To achieve this goal, the work is decomposed into smaller, measurable objectives:

- Design a modular **AR**–edge–robot architecture that connects an optical see-through **HMD**, an edge perception server, and a robotic execution branch through clear data flows and session-level synchronization.
- Implement an **AR** interaction workflow for capturing operator speech, visual observations, spatial metadata, and validation decisions.
- Implement an edge-side perception pipeline for language-conditioned object segmentation and object-centric **3D** reconstruction from headset observations.
- Present the resulting **2D** and **3D** object hypotheses through **AR** feedback so that the operator can inspect interpreted target before execution.
- Integrate the validated target state with a **VLA**-based manipulation branch in a robotic **DT**.
- Evaluate the feasibility of the proposed workflow through perception feedback, operator-facing validation mechanisms, and simulation-based **VLA** task execution.
- Identify practical limitations and deployment constraints, including ambiguity, latency, segmentation errors, reconstruction uncertainty, calibration, and transfer from simulation to the physical workcell.

Together, these objectives define the system scope, the prototype implementation, and the simulation-based evaluation setup. Since comparative user experiments and formal safety validation are outside the scope of this dissertation, performance, usability, and safety outcomes are treated as future evaluation dimensions rather than as primary research questions. The study is therefore guided by the following research questions:

1. **Session integration.** How can an **AR**–edge–robot architecture associate speech, Red, Green, Blue and Depth (**RGB-D**) observations, spatial metadata, perception hypotheses, reconstruction artefacts, and validation decisions within the same picking session?
2. **Inspectable perception.** To what extent can language-conditioned **2D** segmentation and object-centric **3D** reconstruction provide inspectable hypotheses for the requested target?
3. **Operator validation.** How can **AR** feedback make the system’s interpretation of a natural-language picking command visible, correctable, and cancellable before robotic execution?
4. **VLA execution in simulation.** How does the **VLA**-based execution branch perform in simulation after receiving a previously grounded and operator-validated target state?
5. **Failure modes and limitations.** What are the main limitations of the proposed workflow, particularly regarding ambiguous language, visually similar objects, segmentation errors, reconstruction uncertainty and transfer from simulation to the physical workcell?

1.3 Contributions

This dissertation contributes to the integration of **AR**, edge-side perception, operator validation, and simulation-based robotic execution for intelligent picking. Rather than proposing a new segmentation model, reconstruction model, or robot policy, the work focuses on how these components can be organised into an inspectable workflow between a human operator and a robotic execution branch.

1. A closed-loop **workflow design** for **AR**-assisted intelligent picking, connecting operator speech, egocentric visual observations, edge-side grounding, operator validation, and a simulated **VLA**-based robotic execution branch.
2. An implemented **session-centric orchestration workflow** that assigns each interaction episode a persistent session identifier and uses it to synchronise operator speech, **RGB-D** observations, camera metadata, timestamps, system states, perception outputs, reconstruction artefacts, and validation events across the **AR** client and the edge server.
3. An implemented **edge-assisted perception workflow** that uses session-linked observations to perform language-conditioned **2D** object segmentation and object-centric **3D** reconstruction, producing visual target hypotheses that can be inspected by the operator instead of being treated as directly executable robot commands.
4. An implemented **AR-based validation interface** that presents the interpreted target through spatial feedback and allows the operator to confirm, refine, manually adjust, or cancel the result before execution.
5. A **simulation-based assessment** of the **VLA** execution branch, using a robotic **DT** to evaluate manipulation after the target has been grounded and operator-validated.

The dissertation also analyses practical limitations that affect deployment beyond the prototype, including latency, calibration, segmentation ambiguity, reconstruction uncertainty, and transfer from simulation to the physical workcell.

1.4 System Overview

This section introduces the proposed system at a high level before the detailed methodological and implementation chapters. The dissertation focuses on an Extended Reality (**XR**)-assisted intelligent picking workflow in which the operator’s spoken intent is grounded in the visual scene, returned as an inspectable target hypothesis, and validated before it is used by the robotic execution branch. The system is therefore not designed as a direct speech-to-action pipeline. Instead, it introduces an explicit perception and validation loop between the human command and the robot action.

The architecture is organised around three main blocks, as shown in Figure 1.1. The first block is the **XR** client running on an optical see-through **HMD**. This client captures the operator’s

free-form instruction, egocentric visual observations, spatial metadata, and interaction events. It also presents the system’s interpretation through spatial feedback, allowing the operator to inspect the target selected by the perception pipeline before execution.

The second block is the edge perception server. It receives session-linked speech and visual observations from the XR client, derives an object-level target prompt from the transcript, performs language-conditioned 2D segmentation, and prepares an object-centric 3D reconstruction of the selected target. The resulting outputs are treated as perception hypotheses rather than final execution commands, since they may still be affected by ambiguity, occlusion, segmentation errors, reconstruction uncertainty, or spatial misalignment.

The third block is the robotic execution branch. After the operator validates the grounded target state in XR, the confirmed object information can be made available to a VLA-based manipulation policy. In the current dissertation, this branch is evaluated in simulation through a robotic DT. This scope allows the work to evaluate the operator-facing grounding and validation workflow, together with the feasibility of simulation-based VLA execution, without claiming a validated physical robot deployment.

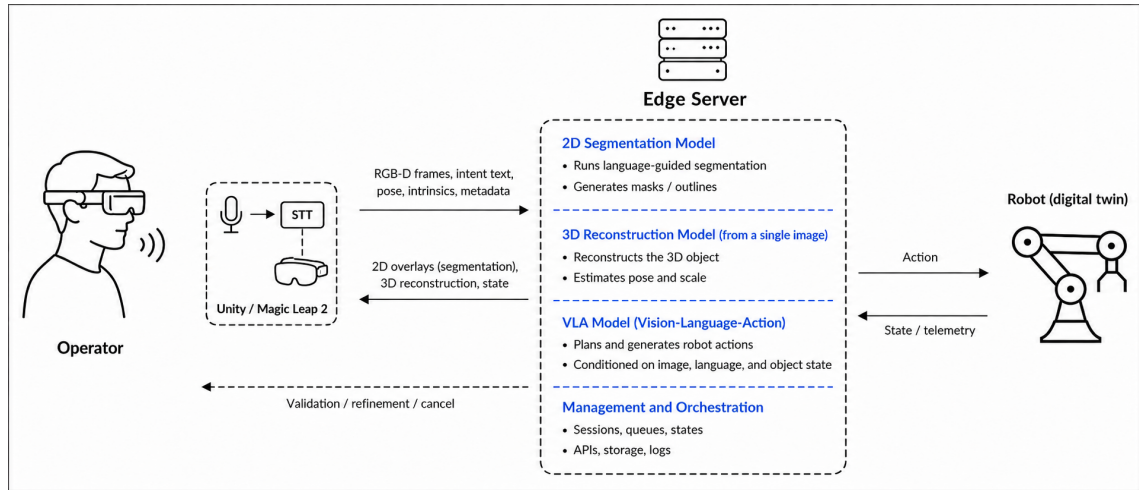


Figure 1.1: High-level XR-assisted picking architecture. The operator uses the HMD to provide a spoken command and capture visual context. The edge server grounds the requested target through 2D segmentation and object-centric 3D reconstruction. The resulting hypothesis is returned to the XR client, where the operator can inspect, refine, manually adjust, confirm, or cancel the result before the validated target state is passed to the simulated robotic execution branch.

The information flow is session-centric. Each interaction episode links the operator’s speech, captured Fvisual observations, camera and pose metadata, perception outputs, spatial estimates, and validation decision under the same session identifier. This preserves traceability between the command, the visual evidence, the generated object hypothesis, and the operator’s final decision.

Human validation is placed between perception and execution. The operator can inspect the transcript, the segmented target, the estimated spatial location, and the reconstructed object representation before accepting the result. If the target hypothesis is ambiguous, incorrectly

segmented, poorly aligned, or unsuitable for execution, the operator can refine, manually adjust, or cancel it instead of allowing the robot branch to act on an unverified interpretation.

The scope of the dissertation is limited to an integrated prototype and validation workflow. The **XR**-edge branch is implemented to support session integration, inspectable perception, and operator validation. The robotic branch is evaluated in simulation using a **DT** and a **VLA**-based policy. Transfer to the physical workcell remains outside the validated scope of this version and would require additional calibration, hardware integration, safety validation, and robustness testing.

1.5 Dissertation Structure

The remainder of this document is organised as follows:

Chapter 2 presents the background concepts required to understand the proposed system. It covers collaborative robotic workcells, **AR** and **AR** interfaces, spatial registration and latency constraints, edge-assisted perception, language-conditioned visual grounding, object-centric **3D** reconstruction, and **VLA** models for robot control.

Chapter 3 reviews the related work most relevant to the dissertation. The review is organised around **AR**-assisted **HRC**, edge-enabled perception, language-based robot interaction, visual grounding, **3D** object reconstruction, and **VLA**-based manipulation. The chapter identifies the gap addressed by this work: making the system's interpretation of a spoken picking command inspectable and correctable before robot execution.

Chapter 4 describes the methodological approach. It presents the overall research strategy, the system architecture, the interaction and data-flow design, the selected tools and technologies, and the planned evaluation methodology. This chapter establishes how the dissertation connects the **AR** client, edge perception server, validation interface, and robotic execution branch.

Chapter 5 details the implementation of the proposed system. It describes the Magic Leap 2 Unity client, speech acquisition, **RGB-D** capture, Robot Operating System (**ROS**) and Hypertext Transfer Protocol (**HTTP**) communication, edge-side **2D** segmentation, **3D** reconstruction, artefact loading, transform handling, **AR**-based validation tools, guided refinement, and the connection to the **VLA**-based execution branch.

Chapter 6 presents the tests and validation activities. It evaluates the system from multiple perspectives, including component-level communication tests, perception and artefact feedback, operator-facing validation mechanisms, guided refinement behaviour, and simulation-based robotic execution in NVIDIA Isaac Sim. The chapter also analyses failure cases such as ambiguous speech, segmentation errors, reconstruction artefacts, latency, and limitations of the simulated execution branch.

Finally, Chapter 7 concludes the dissertation. It summarises the main findings, the limitations of the current work and the future work that can proceed this dissertation.

Chapter 2

Background

This chapter introduces the concepts needed to understand the proposed XR-assisted intelligent picking pipeline. It begins with collaborative robotic workcells, where humans and robots share tasks and where the operator must remain aware of what the robot is about to do. From this need for shared awareness, the chapter moves to Extended Reality as a way to present the system’s interpretation directly in the workspace. It then discusses how the operator’s speech and visual observations can be processed on a nearby edge server to ground the intended target in the scene. This leads to foundation models for object segmentation and 3D reconstruction, which allow the system to transform a natural-language command into inspectable 2D and 3D object hypotheses. Finally, the chapter introduces VLA models as the robotic execution layer that can use this validated target state to support manipulation.

2.1 Collaborative workcells in industry

Industrial robots were traditionally deployed in fenced cells, where humans and robots were separated and did not share the same workspace during automatic operation. Safety was enforced mainly by physical separation and by interlocks that stopped the robot whenever a gate was opened. This paradigm is reflected in standards such as International Organization for Standardization (ISO) 10218-1, which specify safety requirements for industrial robots in terms of design, protective measures, and safety functions [17]. These standards are used here to frame the safety context of industrial robot workcells; however, formal safety validation or certification of the robotic cell is outside the scope of this dissertation.

Collaborative applications represent a different operating context. In collaborative workcells, humans and robots may work closer together and sometimes on the same task, provided that the specific application is risk-assessed and controlled according to the applicable standards, in particular ISO/TS 15066 [18]. In this sense, collaboration is not an intrinsic property of a robot model, but a property of a particular application, tool, task, speed, workspace layout, and safety concept. Surveys such as Villani, Pini, Leali, *et al.* [19] and Li, Hu, Zhou, *et al.* [20] describe practical deployments in assembly, logistics, and machine tending and relate them to these standards.

The way a robot shares space and time with a human can be summarised through three broad configurations:

- **Separated cell:** the robot operates inside a dedicated cell, physically separated from the human. In this configuration, each remains in its own area, and the human does not enter the active robot workspace during automatic motion;
- **Shared hall:** the robot and the human are present in the same broader environment, but their workspaces do not overlap. This corresponds to coexistence: both operate in the same hall or production area, but each remains within a distinct workspace;
- **Shared workspace:** the robot and the human may use the same workspace. This category includes different levels of coupling: in synchronised work, they use the same area at different times; in cooperation, they share the workspace at the same time while working on related parts of the task; and in collaboration, they are closely coupled in space and time, potentially acting on the same object or task.

Figure 2.1 illustrates this progression from cell-based separation, through coexistence, to closer forms of task coupling. In this dissertation, the relevant issue is not only whether the robot and operator share the workspace, but whether the operator can understand the system's interpretation before the robotic execution branch is allowed to act.

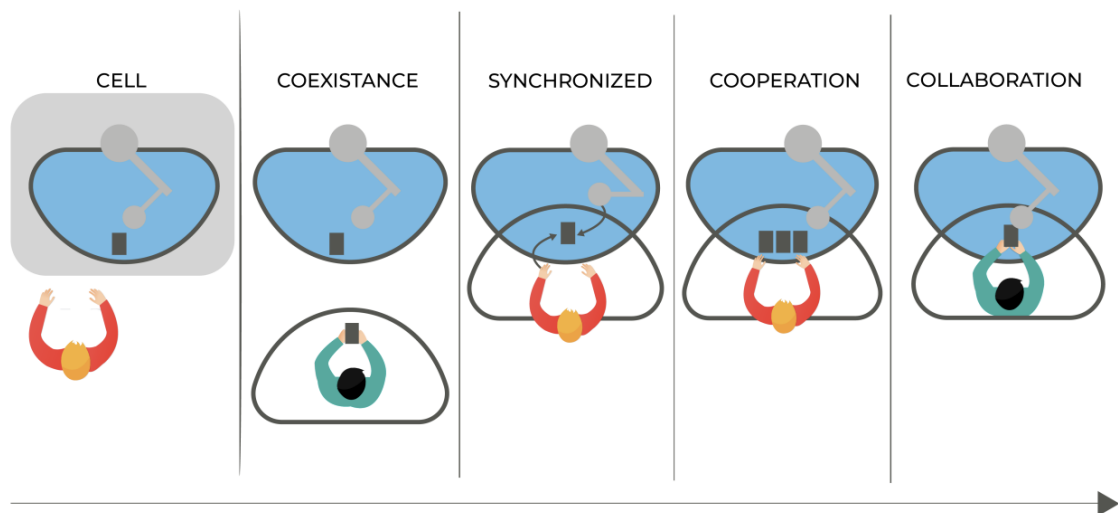


Figure 2.1: Levels of human–robot interaction, from separated robotic cells to collaborative task execution [5].

In this dissertation, the relevant scenario is closer to the shared-workspace end of this spectrum, where the operator and robot remain part of the same task context and the operator must understand the system's interpretation before execution.

Many industrial manipulators are designed as six-axis arms, giving the robot six Degrees of Freedom (6DoF) for positioning and orienting its tool in space [21]. These degrees of freedom can be understood as three translational components, which determine where the tool is placed,

and three rotational components, which determine how the tool is oriented. In practical terms, this allows the robot to move its end-effector to a target region and orient the gripper appropriately for operations such as grasping, placing, or inspection.

Programming robots in collaborative workcells still relies mainly on traditional methods:

- **Teach pendant:** the operator uses a handheld device to jog each joint, record waypoints and configure speeds and Input/Output (I/O);
- **Programming by Demonstration (PbD):** the operator physically guides the robot through a motion and the controller records the trajectory or key poses;
- **Offline programming:** the program is created in a simulation or Computer-Aided Design (CAD) environment and later downloaded to the robot;

These methods are robust and widely used, but they can be time-consuming to adapt when picking tasks, object locations, product variants, or workspace layouts change frequently. They also do not always make the robot’s intended target, next action, or execution context explicit to the operator at the point of work. In this dissertation, the central **HRC** issue is therefore not only safe physical coexistence, but also semantic and spatial transparency: the operator must understand which object the system has selected before the robot execution branch is allowed to act. This motivates the use of **AR** feedback and language-based interaction as an additional layer for XR-assisted picking, where the system’s interpretation of a command can be inspected, corrected, or confirmed before execution.

2.2 Augmented Reality

AR is a way of combining digital information with the physical environment so that both appear in the same view. A widely cited definition by Azuma [8] describes **AR** systems as those that (i) combine real and virtual content, (ii) are interactive in real time and (iii) register virtual objects in three dimensions with the real world. In practice, the user continues to see the real environment, while additional information is drawn at specific positions and orientations in space.

Milgram and Kishino [7] place **AR** within a broader “reality–virtuality” continuum, which is useful as a historical lens to describe experiences ranging from purely real to purely virtual environments. At one end of this continuum are purely real environments; at the opposite end are purely virtual environments, where the user experiences only synthetic content. **AR** lies close to the real end: the physical world remains primary, and virtual content is overlaid on top of it. This contrasts with Virtual Reality (**VR**), where the real environment is largely or completely replaced by a virtual one.

In more recent literature, the discussion is often framed using the umbrella term **XR**, which refers collectively to **AR**, Mixed Reality (**MR**), and **VR** and better reflects the diversity of current devices and experiences [22]. Under this framing, **MR** is commonly used to describe experiences in which virtual content is not only overlaid but also appears more tightly integrated with the physical

environment (e.g., persistent spatial anchoring, interaction with real objects, and context-aware behavior), which is especially relevant for modern **HMDs** used on the shop floor. In the original reality–virtuality continuum, Augmented Virtuality (**AV**) also appears as an intermediate region closer to the virtual end of the spectrum, where elements of the real world are incorporated into an otherwise predominantly virtual environment [7]. Figure 2.2 represents this reality–virtuality continuum by positioning **AR**, **AV**, **MR**, and **VR** between the real and virtual ends of the spectrum. This visual distinction is useful in this dissertation because the proposed interface does not replace the physical workcell, as in **VR**, but augments the operator’s view with spatially registered feedback.

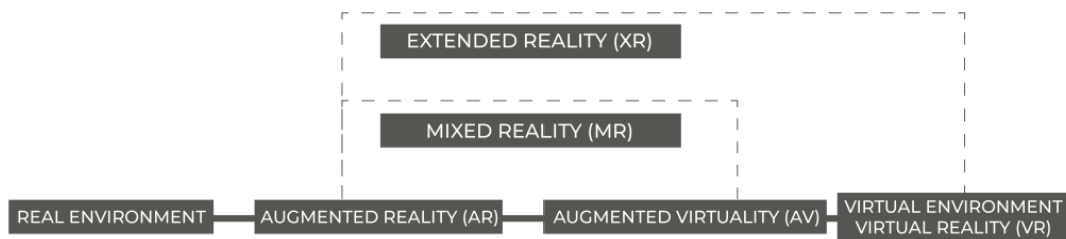


Figure 2.2: Extended reality continuum, positioning **AR**, **AV**, **MR**, and **VR** between the real and virtual ends of the spectrum. [5].

The same basic idea of overlaying information on the real world can be realised with different devices:

- handheld devices such as tablets or smartphones, used as a “window” into an augmented view;
- spatial projectors that cast symbols or instructions directly onto work surfaces and parts;
- **HMDs**, where the worker wears a device that places digital content in the field of view.

Two common **AR HMD** display architectures are optical see-through and video see-through [8]. In an optical see-through **HMD**, the user observes the physical environment directly through transparent or semi-transparent optics, while virtual content is optically combined with the real-world view. This allows the user to maintain direct visual contact with the surrounding workspace, including peripheral cues and unexpected hazards, which is particularly relevant in industrial settings.

In a video see-through **HMD**, the physical environment is first captured by one or more cameras and then displayed on screens together with the virtual content. This gives the system more control over the final composed image, since both real and virtual elements are rendered digitally, but it also means that the user’s perception of the environment depends on camera capture, display latency, resolution, dynamic range, and possible image-processing artefacts.

Both approaches can support spatially registered **AR** feedback. Optical see-through devices tend to favour direct awareness of the real environment, whereas video see-through devices offer greater control over visual composition and occlusion handling. The choice between them therefore

depends on the application requirements, including safety, latency, field of view, visual fidelity, and the degree of integration needed between physical and virtual content.

Although implementations differ, most AR systems follow a similar processing chain. First, sensors such as cameras and inertial units capture images of the scene and motions of the device. Second, a tracking and mapping component estimates the 6DoF pose of the device and builds a map of the environment. Third, virtual content is registered in that map, meaning that each virtual object is assigned a position and orientation in a chosen coordinate frame. Finally, the virtual objects are rendered and shown to the user under low-latency requirements. These steps run continuously so that the overlays stay aligned as the user moves.

Two technical constraints are particularly relevant for industrial applications. The first is spatial accuracy and stability: overlays should line up with the physical world. Small pose errors that are acceptable when showing a distant label can become problematic when a highlight is supposed to coincide with a bin edge, a part or a gripper. Studies of optical see-through HMDs report errors on the order of a few centimetres in typical conditions, which is good enough for guidance but can limit tight alignment tasks near contact [9]. While device-specific performance differs across headsets, these studies illustrate the typical magnitude of the problem and the motivation for additional anchoring when high repeatability is required.

The second constraint is end-to-end latency: when the operator moves or the scene changes, the overlays should update quickly. Higher latency increases the apparent “lag” between real motion and virtual content and can reduce user trust and task fluency.

To improve local accuracy, industrial deployments often complement the internal tracking of the headset with external references such as fiducial markers placed near the areas where precision is most important. These markers allow a quick calibration step and effectively “lock” virtual content to shelves, fixtures or bins, even if the global map drifts slowly over time. Recent evaluations on HoloLens 2 show that combining markerless tracking with fiducial-based anchoring can significantly improve the repeatability of overlays at the working pose [10].

On the application side, systematic reviews report a growing number of AR solutions in manufacturing and maintenance. Egger and Masood [23] review AR in intelligent manufacturing and highlight assembly guidance, quality inspection and training as common use cases. Palmarini, Erkoyuncu, Roy, *et al.* [24] focus on maintenance and show that AR can reduce errors and task time by putting relevant instructions directly where the technician is looking. At the same time, these and other studies report practical issues such as limited field of view, occlusions, lighting variability and device comfort, and recommend that AR be used as a focused support layer that presents only the information the worker needs at the point of work [23]–[25].

Within HRC, AR has been proposed as an interface that makes robot state and intent visible in the shared workspace. Costa, Petry, and Moreira [5] found that spatial overlays can help operators follow assembly steps, identify parts, visualise robot paths and understand safety zones. Chang and Hayes [6] reached similar conclusions, while also noting ergonomic and tracking issues that must be addressed for shop-floor deployment. Other work combines AR with DTs to show robot trajectories, joint states and planned actions in situ, sometimes for multiple users and robots at

once [12], [13]. In all these cases, the value of **AR** comes from reducing split attention: instead of looking back and forth between screens, buttons and workpieces, the operator can see the relevant information directly overlaid on the physical environment.

Finally, **AR** systems in industrial environments raise questions of privacy, data handling and user well-being. Headsets typically capture video, depth and motion data of both the workspace and the people in it. Guidance documents such as the XR4HUMAN Code of Conduct for the Human-Centered and Ethical Development of Immersive Technologies stress principles such as human-centred design, data minimisation and clear communication of data practices and risks [26]. In this dissertation, these principles inform design decisions such as processing **RGB-D** data on a nearby edge server instead of in the cloud, logging only the information needed for evaluation, and making the data flows explicit to participants in the experimental workcell.

2.3 Edge-assisted perception and scene representation

In everyday language, perception means using the senses to understand what is around us. In robotics and **XR**, the idea is similar, but it is expressed through sensors, models, and coordinate frames: raw measurements such as images, depth maps, poses, and robot states are transformed into information that the system can use to reason about the environment and about itself. A common view in robotics is that perception is not simply a direct reading of the world, but an estimation process based on incomplete and noisy sensor data [27]. For a collaborative workcell, this perceived state should answer practical questions such as: Which objects are present? Where are they? What is the robot doing? Where is the operator looking from? Which object is relevant to the current task?

Historically, industrial automation started with simple and reliable sensors such as limit switches, photoelectric barriers, proximity sensors, safety scanners, and light curtains. These devices provide binary or low-dimensional signals indicating, for example, whether an object has reached a certain position or whether a person has entered a protected area. They remain essential in industrial environments, especially for safety and interlocks, but they do not provide a rich understanding of the scene. They can say that something is present, but not what it is, how it looks, where its visible boundaries are, or whether it corresponds to the object requested by the operator.

With the introduction of digital cameras and faster processors, Computer Vision (**CV**) became a practical way to add richer information to industrial systems. Early vision systems relied on two-dimensional images and hand-crafted algorithms to read barcodes, inspect parts, detect defects, or check dimensions. These systems were useful when the environment was well controlled and the visual task was clearly defined. However, collaborative picking and manipulation require more than recognizing a pattern in an image. For a robot to act on a perceived object, the visual system must support object localisation, pose estimation, and grasp-related reasoning; in a collaborative workcell, this spatial interpretation also has to be related to the robot and to the human operator sharing the task environment [19], [28].

This limitation motivated the use of three-dimensional sensing. Depth sensors, stereo cameras, structured-light cameras, Time-of-Flight (ToF) sensors, and Light Detection and Ranging (LiDAR) make it possible to measure distance and recover geometric information about the workspace. When colour and depth are combined in an RGB-D sensor, the system obtains both appearance and geometry. The Red, Green, and Blue (RGB) image provides colour, texture, labels, and semantic cues, while the depth image provides per-pixel distance information. Together, they allow the system to relate what is visible in the image to where it is located in physical space.

This link between image space and physical space is central in robotic perception. A detection or segmentation in a 2D image is useful, but by itself it is not enough for manipulation or spatial feedback. A robot needs a location in a metric frame, and an XR interface needs to place virtual information in the real workspace. Using camera intrinsics and depth values, pixels can be projected into 3D points. In the pinhole camera model, a pixel (u, v) with depth z can be mapped into the camera frame as

$$X = \frac{(u - c_x)z}{f_x}, \quad Y = \frac{(v - c_y)z}{f_y}, \quad Z = z, \quad (2.1)$$

where f_x and f_y are the focal lengths and c_x and c_y are the principal point coordinates. This simple geometric relation is what allows a visual observation to become a spatial observation, provided that calibration, synchronization, and camera pose estimation are sufficiently accurate [29].

As perception systems became richer, the focus also shifted from isolated sensor readings to scene representation. A scene representation is an organized description of the elements that matter for the task. In a robotic workcell, it may include object identities, object locations, segmentation masks, 3D points, estimated poses, support surfaces, free space, robot state, and task status. For picking tasks, the most relevant representation is often object-centred: the system must know which object is being referred to, where it is, and whether it can be inspected or manipulated. Table 2.1 summarises this progression from raw sensor measurements to task-level scene representation.

Table 2.1: From sensor data to scene representation in XR-assisted robotic workcells.

Level	Typical information	Role in the workcell
Raw sensing	RGB images, depth maps, inertial data, robot joint states	Captures the current state of the physical environment and the robotic system.
Image-space perception	Detections, segmentation masks, object outlines	Identifies visually relevant regions or objects in the camera view.
3D grounding	Point clouds, centroids, support surfaces, approximate poses	Relates visual evidence to metric space for manipulation and XR placement.
Scene representation	Object state, robot state, task state, validation state	Organizes perception into information that can be used by the robot and inspected by the operator.

This representation is important not only for the robot, but also for the human interface. In a traditional robot cell, perception may remain hidden inside the control system. In an **XR**-assisted collaborative cell, perception can be externalized to the operator. A highlighted object, a segmentation outline, a centroid marker, or a reconstructed **3D** shape can show what the system currently believes about the scene. This makes perception part of the interaction loop. The operator is no longer only giving commands; the operator can also inspect whether the system has interpreted the scene correctly.

However, the perception needed for this kind of interaction is computationally demanding. Head-mounted displays such as Magic Leap 2 or HoloLens 2 are wearable computers equipped with cameras, sensors, and displays, but they are constrained by battery capacity, thermal limits, memory, and available processing power. At the same time, **XR** interfaces in collaborative workcells must react quickly to changes in the scene and to operator commands. This creates a tension between the complexity of modern perception models and the need for responsive interaction on the headset.

This tension can be understood by comparing three possible processing strategies:

- **Device-only processing:** perception runs directly on the headset or on the robot controller. This reduces dependence on the network, but limits the size and complexity of the models that can be used because wearable devices have strict compute, power, and thermal constraints.
- **Cloud-only processing:** heavy perception and learning models run in distant data centres. This option usually provides the greatest elastic scalability, since computation and storage can be expanded beyond the local device or workcell. However, it can also introduce unpredictable end-to-end delay, network jitter, bandwidth costs, and data-governance concerns, especially when industrial visual data are transmitted outside the local environment.
- **Edge-assisted processing:** an on-premise or nearby server hosts the heavier perception services. The headset remains responsible for sensing, tracking, rendering, and interaction, while the edge server processes computationally demanding tasks and returns compact results over the local network.

Edge-assisted perception therefore sits between the device-only and cloud-only extremes. Shi, Cao, Zhang, *et al.* [30] define edge computing as a paradigm in which computation, storage, and networking resources are placed close to data sources and users, reducing the need to send all data to distant cloud infrastructure. Satyanarayanan [31] frames this shift in terms of mobile and wearable applications, where wide-area cloud round trips may be too slow or too unpredictable for interactive use. This is particularly relevant for **AR** and robotics because both domains depend on timely interaction with the physical world. In industrial workcells, the same idea also supports controlled data handling. Camera images, depth observations, and process context may contain sensitive information about the workspace, products, operators, or production process. Processing these data within the factory network can therefore reduce unnecessary exposure to external cloud services and make data locality an explicit architectural constraint [30]. This does not remove

the need for cybersecurity, access control, or privacy-aware data management, but it explains why edge-assisted perception can be attractive beyond latency and compute alone.

For perception workloads, edge assistance is useful because image and depth streams are data-intensive and because modern visual models may require Graphics Processing Unit (**GPU**) acceleration. The headset can remain responsible for sensing, tracking, rendering, and operator interaction, while the edge server runs heavier perception services and returns compact results such as object masks, outlines, poses, or reconstructed artifacts. This division preserves the interactive role of the **XR** device while allowing the system to use more advanced perception methods than would be practical on-device.

The design of an edge-assisted vision pipeline is not only a question of where computation runs, but also of what data are transmitted and when. Research on edge video analytics identifies several strategies for maintaining responsiveness, including early filtering, adaptive compression, and model partitioning [11]:

- **Early filtering:** reducing uplink load by sending only selected frames, regions of interest, or downsampled data when appropriate.
- **Adaptive compression:** adjusting image resolution, quality, or bitrate to balance visual fidelity against latency and bandwidth constraints.
- **Model partitioning:** splitting a perception model so that lightweight stages run locally, while heavier stages run on the edge server.

These strategies are especially relevant in **XR** systems because the user experience depends not only on average latency, but also on avoiding slow or unstable feedback loops. Surveys of edge video analytics discuss these design choices in the context of real-time video applications [11]. Architectures for head-worn devices illustrate the same trade-off: for example, DeepMix studies hybrid **3D** object detection for headsets, combining lightweight local processing with heavier edge-side inference to support mobile perception under device constraints [32]. The exact division between local and edge computation depends on the hardware, the network, the perception model, and the latency budget of the application.

In this way, edge-assisted perception connects the physical workspace, the **AR** interface, and the robotic system. Sensors provide raw observations, **RGB-D** processing relates image evidence to **3D** space, scene representation organizes the relevant task information, and edge computing makes it possible to use heavier perception models without overloading the headset. In this context, the edge server is not only an inference accelerator: it can also act as a session-level integration point where speech, visual observations, camera metadata, segmentation results, reconstruction artefacts, and validation-related state are associated with the same interaction episode. These concepts provide the background for the following section, where foundation models for object segmentation and **3D** reconstruction are introduced as modern tools for building object-level representations from visual observations.

2.4 Foundation models for object segmentation and 3D reconstruction

The previous section introduced perception as the process of transforming sensor data into a useful representation of the workspace. Traditionally, this representation often depended on task-specific vision pipelines: a detector trained for a fixed set of objects, a segmentation method tuned for a particular bin, or a reconstruction algorithm designed for a controlled capture setup. Such approaches are consistent with classical computer vision practice, where algorithms are typically designed, trained, or configured for a particular visual task and operating condition [33]. They can work well in stable industrial environments, but they become harder to maintain when objects, layouts, lighting conditions, or operator instructions change frequently.

Foundation models have changed this perspective. Instead of training a separate model for each narrow visual task, a foundation model is trained on broad data at scale and can be adapted to a wide range of downstream tasks [34]. In CV, this has led to models that can segment, recognise, describe, or reconstruct objects with less task-specific engineering than earlier systems. For collaborative picking, this is important because the target object may not always be selected from a fixed menu. It may be described by the operator through natural language, by object category, colour, position, or another visual property.

A central task in this context is object segmentation. Object detection usually returns a bounding box around an object, which is useful for locating it approximately in an image. Segmentation goes further by estimating which pixels belong to the object [35]. This distinction matters for manipulation and XR feedback [28]. A bounding box may include background, neighbouring objects, or empty space, while a segmentation mask gives a more precise description of the visible object region. In an XR interface, a segmentation outline can also be shown directly to the operator, making the system's interpretation easier to inspect.

The Segment Anything Model (SAM) [35] introduced a general promptable approach to image segmentation. Rather than being limited to a fixed set of classes, SAM can produce masks from prompts such as points, boxes, or existing masks, and was trained with a large-scale data engine to support broad zero-shot transfer across segmentation tasks [35]. This represented an important shift: segmentation became less tied to manually engineered pipelines and more interactive, allowing a user or another system component to guide the model towards the region of interest.

For language-based robotic interaction, however, geometric prompts alone are not always enough. An operator is more likely to say 'pick the bottle' or 'select the red object' than to provide a point or bounding box. This motivates open-vocabulary and language-conditioned segmentation, where the prompt can be a word or short phrase. SAM 3 [36] extends the Segment Anything line in this direction by introducing promptable concept segmentation. It can detect, segment, and track object instances in images and videos based on concept prompts, defined as short noun phrases, image exemplars, or combinations of both [36]. This type of model is especially relevant for systems where natural-language intent must be grounded in a visual scene.

Segmentation alone, however, still operates mainly in image space. A mask can indicate which pixels correspond to the intended object, but a robotic system also needs a spatial interpretation.

For picking, the object must be related to the physical workspace: its approximate position, extent, orientation, support surface, and relationship to the robot must be estimated. This is where segmentation becomes the first step towards object grounding. By combining a mask with depth, camera intrinsics, camera pose, or other spatial evidence, the system can move from a two-dimensional visual hypothesis to a three-dimensional object hypothesis. Table 2.2 compares common visual outputs used to build object-level representations, from simple image-space localization to richer 3D forms that can be inspected or used by downstream systems.

Table 2.2: Common visual outputs used to build object-level representations.

Output	What it represents	Main limitation
Bounding box	Rectangular image region that approximately contains an object	Includes background and does not describe object shape precisely.
Segmentation mask	Pixel-level visible region of an object	Remains mostly image-space unless combined with depth or pose information.
Point cloud	Set of 3D points measured or inferred from sensor data	Can be noisy, sparse, incomplete, or difficult to render as a solid object.
Mesh	Surface-based 3D representation of an object	May be uncertain in scale, pose, or occluded regions, especially when reconstructed from limited views.
Splat-based representation	Renderable 3D appearance using point-like or Gaussian-like primitives	Useful for visualization, but not always directly suitable for manipulation planning.

Figure 2.3 complements Table 2.2 by showing the progression from image-space object localization to richer spatial representations. This progression is important because each representation provides more information for XR inspection and robotic execution, but also introduces different sources of uncertainty.

The next step is 3D reconstruction. In general terms, 3D reconstruction is the process of estimating the shape, structure, or appearance of an object or scene in three dimensions. Classical reconstruction methods often rely on multiple calibrated images, feature matching, stereo geometry, or depth fusion. These methods are powerful when sufficient views and accurate calibration are available, but they can be difficult to apply in fast interaction scenarios where the system must provide feedback from limited observations.

Recent learning-based reconstruction models use learned priors about object shape and appearance to infer 3D structure from fewer observations. SAM 3D follows this direction by proposing a

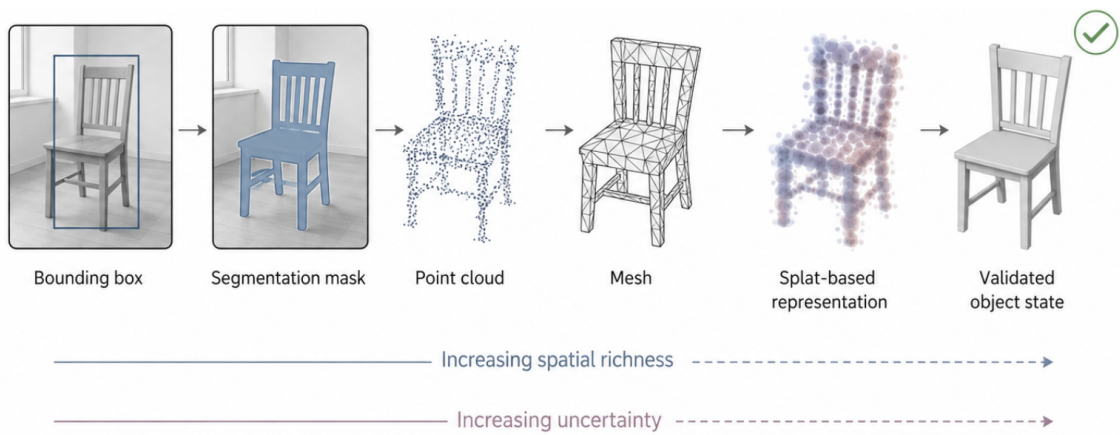


Figure 2.3: Progression from image-space perception to object-level 3D representation.

generative model for visually grounded **3D** object reconstruction from a single image, predicting geometry, texture, and layout while addressing natural images where clutter and occlusion are common [37]. This type of model is useful for **XR**-assisted interaction because it can produce an object-centric **3D** hypothesis that can be visualized, inspected, and refined, even when a complete multi-view scan is not available.

It is important to distinguish between a reconstructed object hypothesis and a metrically verified model of the object. A single-view reconstruction may be visually plausible while still being uncertain in scale, pose, hidden geometry, or fine contact details. Occluded parts of the object must be inferred rather than observed directly. Symmetric objects may have ambiguous orientation. Reflective, transparent, or textureless objects can also create difficulties for both segmentation and reconstruction. Therefore, a generated **3D** artifact should not automatically be treated as ground truth for robot execution.

Different **3D** representations may also serve different purposes. Polygon meshes are common because they are widely supported by simulation engines, game engines, and **XR** applications. Point clouds are useful for directly representing measured geometry from depth sensors. More recent radiance-field and splat-based representations, such as **3D** Gaussian Splatting, represent scenes using collections of Gaussian primitives and can support high-quality real-time novel-view rendering [38]. In the context of **XR** feedback, these representations are relevant because they affect how easily a reconstructed object can be rendered, aligned, and inspected by the operator.

For robot systems in collaborative applications, the value of segmentation and reconstruction is not only computational. These models can also make perception more transparent. A segmentation mask shows what part of the image the system selected. A reconstructed object shows how the system has interpreted that selection in **3D**. When these outputs are presented through **XR**, the operator can compare them with the real workspace and decide whether the system has understood the command correctly. This is particularly important when several similar objects are present, when language is ambiguous, or when the scene is partially occluded.

Foundation models therefore provide powerful tools for building object-level representations

from visual observations, but they also introduce uncertainty that must be managed. Their outputs depend on prompt quality, image quality, viewpoint, occlusion, calibration, and the distribution of objects seen during training. In safety-sensitive or collaborative settings, these outputs should be treated as hypotheses to be validated rather than final decisions. This motivates the use of human-in-the-loop XR feedback before the result is passed to a robot execution layer.

In summary, object segmentation provides a way to identify the visible region corresponding to a target, while 3D reconstruction extends this target hypothesis into a spatial form that can be inspected and used by downstream systems. Foundation models make these steps more flexible and less dependent on fixed object classes, which is valuable for natural-language picking scenarios. At the same time, their limitations justify the validation layer that connects perception to trustworthy robotic action.

2.5 From LLM and VLM to VLA policies for robot control

Natural language is an attractive interface for robotics because it allows humans to describe tasks at a high level, without specifying every joint motion, waypoint, or control parameter. Instead of programming a robot through a teach pendant or manually defining a sequence of actions, an operator may simply express an intention such as “pick the cardboard box”, “move the small package to the container”, or “place the selected part in the assembly tray”. For collaborative environments, this form of interaction is particularly appealing because it is closer to how humans communicate tasks to each other.

LLMs are one of the main reasons why this type of interaction has become more feasible. Modern LLMs are usually based on the Transformer architecture, which uses attention mechanisms to model relationships between elements in a sequence [39]. Trained on large text corpora, these models can interpret instructions, generate structured outputs, answer questions, and support task-level reasoning. In robotics, this makes them useful for interpreting operator commands, decomposing goals into steps, generating plans, or producing intermediate task descriptions.

However, language alone is not enough for robot control. A command only becomes meaningful when it is grounded in the current physical situation. For example, the instruction “pick the cardboard box” depends on whether the box is visible, where it is, whether there are multiple similar boxes, whether the robot can reach it, and whether the grasp is feasible. A language model may understand the sentence, but it does not automatically know the state of the workspace or the physical capabilities of a particular robot. This gap between words and the physical world is known as the grounding problem [40].

Grounding is especially important in collaborative workcells, where ambiguity can lead to incorrect or unsafe behaviour. A human may resolve an ambiguous instruction by looking at the scene, asking for clarification, or relying on context. A robot system needs an equivalent mechanism: the words used by the operator must be connected to perceived objects, spatial relations, task constraints, and robot affordances. Approaches such as SayCan [41] illustrate this principle by combining language-model reasoning with robot-specific feasibility estimates, so that

actions are selected not only because they match the instruction, but also because they are possible in the current environment.

VLMs address part of this limitation by connecting text with visual observations. Instead of processing language and images separately, **VLMs** learn representations that relate visual content to natural-language descriptions. CLIP [42] is a well-known example: it learns visual representations from natural-language supervision and enables images and text to be compared in a shared embedding space. This type of model is important for open-vocabulary perception because it allows visual systems to reason about concepts that are described in language rather than restricted to a fixed set of predefined object classes.

For robotics, **VLMs** provide a bridge between operator instructions and the visual scene. They can support tasks such as referring expression comprehension, open-vocabulary object recognition, visual question answering, and language-conditioned target selection. In a picking scenario, this means that a command can be interpreted not only as text, but also in relation to what is visible in the workspace. Nevertheless, most **VLMs** are still primarily perception or reasoning models. They can help identify what the instruction refers to, but they do not necessarily generate the low-level actions needed to move a manipulator.

VLA models extend this idea by incorporating action into the model. Instead of mapping only between images and text, a **VLA** policy maps visual observations and language instructions to robot actions. In this formulation, the model receives information about the scene and the task, and produces actions that can be executed by a robot, such as end-effector commands, joint commands, or action tokens. This makes **VLA** models an important step towards general robot control from natural-language instructions.

Robotics Transformer 2 (RT-2) [43] is a representative example of this direction. It co-fine-tunes **VLMs** on robotic data and casts robot actions into a format that can be generated by the model, allowing visual-language knowledge to transfer into robotic control. More recent models, such as OpenVLA [44] and the π_0 family [45], [46], continue this trend by treating robot control as a foundation-model problem, where policies are trained across diverse robot embodiments, tasks, and data sources. The goal is not merely to recognize objects or describe scenes, but to generate actions that are appropriate for the perceived environment and the language instruction.

This progression from **LLMs** to **VLMs** and then to **VLAs** can be understood as a gradual reduction of the gap between human intention and robot motion. **LLMs** provide flexible language understanding. **VLMs** connect language to visual evidence. **VLAs** connect language and vision to action. Figure 2.4 visualises this progression by comparing the dominant input–output relation of each model family. The figure should be read as a conceptual background summary rather than as a specific architecture used in this dissertation.

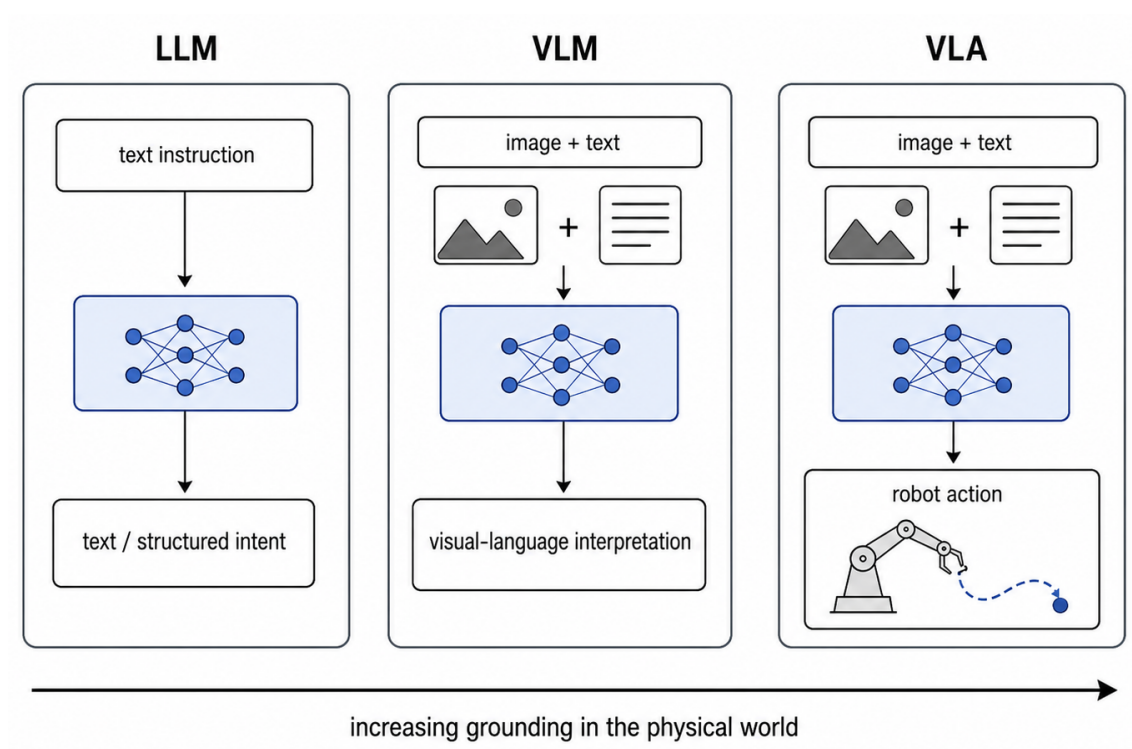


Figure 2.4: Conceptual progression from language understanding to robot action.

In principle, this makes it possible for an operator to give a natural-language command and for the robot to act directly on the visual scene. This is a powerful capability, but it also introduces new challenges. Table 2.3 summarises this progression from language understanding to robot action, highlighting the role and limitation of each model family in robotic systems.

Table 2.3: Progression from language understanding to robot action.

Model type	Main input–output relation	Role in robotics	Main limitation
LLM	Text $\rightarrow$ text or structured intent	Interprets instructions, decomposes tasks, and supports task-level reasoning	Does not directly perceive the physical scene.
VLM	Image + text $\rightarrow$ visual-language interpretation	Grounds language in visual observations and supports open-vocabulary perception	Usually does not generate executable robot actions.
VLA	Image + text $\rightarrow$ robot action	Generates actions conditioned on visual observations and language commands	Can be opaque if target selection and intended action are not inspectable.

The main challenge is transparency. If a **VLA** model receives an image and a command and directly outputs robot actions, the operator may not know what object the model selected or how the command was interpreted. In cluttered or visually ambiguous scenes, this can be problematic. The robot might choose the wrong instance, confuse similar objects, or generate an action that appears reasonable internally but does not match the operator’s intention. In traditional robot programming, the operator can often inspect waypoints, targets, or program logic. In **VLA**-based control, the decision process may be less explicit.

This opacity is particularly relevant in **HRC**. A robot in collaborative applications does not operate in isolation; it acts near people, shared objects, and task-dependent constraints. The operator must therefore remain aware of what the system intends to do before execution begins. For this reason, **VLA** models should not be treated as replacements for grounding, validation, safety checks, or human supervision. Instead, they should be integrated into a larger architecture where their inputs and intended effects are made inspectable.

One way to improve inspectability is to separate target grounding from action generation. Before the robot executes an action, the system can first identify the object or region referred to by the command, present that interpretation to the operator, and allow correction if necessary. In this view, the **VLA** policy does not receive an unverified speech command in isolation. It receives a task context that has already been connected to visual evidence and validated at the interaction level. This reduces ambiguity and provides the operator with an opportunity to intervene before motion is generated.

The need for task-specific adaptation is another important aspect of **VLA** control. Although generalist policies aim to transfer across robots and tasks, performance still depends on the quality, diversity, and relevance of the data used for training or fine-tuning. Robot embodiment, camera viewpoint, gripper type, object appearance, workspace layout, and task distribution all affect execution success. Simulation environments and **DTs** can support data collection, policy testing, and preliminary evaluation, but they provide only partial evidence for physical deployment due to calibration, safety, and sim-to-real transfer constraints.

In summary, **LLMs**, **VLMs**, and **VLA** models provide a conceptual path from natural-language instruction to robot action. **LLMs** help interpret intent, **VLMs** help connect intent to visual observations, and **VLA** policies generate actions from language and perception. For collaborative picking, the value of these models lies in making robot interaction more flexible and natural. Their limitation is that, if used directly, they may hide the system’s interpretation from the operator. This motivates architectures in which language-conditioned perception, **XR**-based inspection, and human validation are placed before **VLA**-driven execution.

2.6 Summary

This chapter introduced the background concepts needed to motivate the proposed AR-assisted intelligent picking architecture. The discussion began with collaborative robotic workcells, where the central challenge is not only physical coexistence between humans and robots, but also the need

for the operator to understand what the system has perceived, inferred, and intends to do next. In this context, transparency becomes a practical requirement for safe and effective collaboration.

AR, within the broader **XR** landscape, provides an interface mechanism for making this information visible in the workspace. Instead of leaving perception and decision-making hidden inside the robotic system, spatial feedback can expose intermediate hypotheses such as the interpreted target, the segmentation outline, the estimated object location, or a reconstructed object representation. However, this feedback depends on accurate registration, stable tracking, and acceptable latency.

The chapter then showed that **RGB-D** perception, scene representation, and edge-assisted processing provide the technical basis for grounding operator instructions in the physical workcell. **RGB** observations provide visual and semantic evidence, depth and pose information support spatial localisation, and edge computing allows heavier perception and reconstruction models to run close to the data source while keeping the wearable client responsive. At the same time, these representations remain uncertain: masks can be incomplete, depth can be noisy, reconstructed geometry can contain artefacts, and coordinate transforms can introduce alignment errors.

Finally, foundation models, **LLMs**, **VLMs**, and **VLA** policies were introduced as a way to increase flexibility in language-based robot interaction and action generation. These models make it possible to move beyond fixed menus and rigid programming interfaces, but they also introduce uncertainty and opacity because their internal interpretation of language, visual evidence, and action feasibility is not automatically visible to the operator. For this reason, the concepts presented in this chapter motivate a pipeline in which a spoken command is not sent directly to robotic execution. Instead, the system first grounds the command in the visual scene, presents the resulting interpretation as an inspectable hypothesis through AR feedback, and requires operator validation before the robotic execution branch is allowed to act.

Chapter 3

Literature Review

This chapter reviews the literature most directly related to the proposed XR-assisted intelligent picking workflow. The dissertation is positioned at the intersection of HRC, spatial XR interfaces, edge-assisted perception, language-conditioned visual grounding, object-centric 3D reconstruction, and VLA-based robot execution. The objective is not to survey each field exhaustively, but to identify the specific results, assumptions, and limitations that motivate the architecture developed in this work: an operator expresses a picking intent through speech; egocentric visual observations are processed by an edge perception server; the target is grounded through 2D segmentation and object-level 3D reconstruction; the interpretation is exposed to the operator through XR; and only then is the validated target state passed to the robotic execution branch.

Compared with conventional robot-programming interfaces, this workflow shifts the critical interaction problem from manual waypoint specification to inspectable target grounding. In language-conditioned picking, the most immediate risk is often not that a robot cannot generate a motion, but that the system has grounded the spoken instruction in the wrong object, instance, or pose. The literature reviewed in this chapter is therefore organised around one central question: how can perception and robot intent be made visible and correctable before robot execution in a collaborative picking task?

3.1 XR interfaces for HRC and intent inspectability

HRC has become increasingly relevant in industrial environments where product variability, small batches, and frequent reconfiguration make fully automated cells difficult to maintain. Surveys of industrial HRC emphasise that collaborative workcells must combine the precision and repeatability of robots with human judgement, contextual awareness, and exception handling [1], [19], [20]. This combination creates two coupled requirements. First, the robot must remain inside an appropriate safety envelope, as framed by industrial robot and standards for collaborative application [17], [18], [47]. Second, the operator must understand what the robot perceives, intends, and is about to do. Safety-rated hardware and protective functions reduce physical risk, but they do not by themselves solve the interaction problem of making robot state and intent legible at the point of work.

AR and, more broadly, XR have been proposed as interface layers for this problem because they can present task-relevant information directly in the shared workspace. The systematic review by Costa, Petry, and Moreira [5] shows that AR has been used in industrial HRC for assembly guidance, inspection, safety-zone visualisation, path preview, and operator assistance. The survey by Chang and Hayes [6] reaches a similar conclusion, classifying AR support according to task, information content, and interaction modality. Across this literature, spatial overlays can reduce split attention and improve awareness by placing information near the workpiece, robot, or relevant workspace region instead of on a separate screen. However, both reviews also identify recurring barriers: limited headset field of view, tracking instability, calibration overhead, occlusion, ergonomic fatigue, and incomplete integration with robot controllers and industrial workflows.

For this dissertation, the most important function of XR is not only instruction display, but making the system’s interpretation inspectable. In traditional industrial programming, an operator may inspect waypoints, program blocks, or teach-pendant states before execution. In learning-based perception and VLA control, intermediate interpretations are less explicit. This makes legibility particularly important. Dragan, Lee, and Srinivasa [48] distinguish between predictable motion, which matches what an observer expects once the goal is known, and legible motion, which helps the observer infer the goal. Although their work focuses on robot motion, the same principle applies earlier in the interaction loop: before motion begins, the operator should be able to infer which object the system selected and what action is being prepared.

Overlay stability and registration accuracy strongly influence whether this inspectability is useful in practice. Soares, Sousa, Petry, *et al.* [9] empirically evaluate optical see-through headset accuracy and repeatability, showing that centimetre-level errors can be expected under typical conditions. Such accuracy can be sufficient for guidance or awareness, but may be limiting when an overlay must coincide tightly with an object, gripper, bin edge, or support surface. Fiducial anchoring can reduce these errors at critical locations. Costa, Petry, Martins, *et al.* [10] evaluate fiducial marker tracking on HoloLens 2 and show that local markers can improve repeatability where precise alignment matters. Multi-user and digital-twin-oriented systems further demonstrate how XR can display robot trajectories, cell state, and simulated robot behaviour in situ [12], [13], [49].

The literature therefore supports the use of XR as a human-centred communication layer in collaborative workcells, but also shows that XR feedback must be selective, stable, and connected to the robot’s actual perception and action pipeline. In the present dissertation, this motivates an XR validation interface where the operator does not simply receive task guidance, but inspects the target object hypothesised by the perception system before the robot execution branch is allowed to proceed.

3.2 XR-assisted picking and logistics

Order picking is a natural domain for AR because it involves repeated identification, selection, and placement actions in visually dense environments. Early pick-by-vision work showed that

head-worn displays could support warehouse picking by presenting item and location information in the worker’s field of view, reducing the need to alternate between paper lists, handheld terminals, and shelves [14]. Later reviews of **AR** in supply-chain and logistics contexts identify warehousing, order picking, inventory management, and training as promising application areas, while also noting barriers such as device comfort, user acceptance, integration with warehouse management systems, and the risk of visual overload [50].

Recent empirical studies reinforce that **AR** picking should be evaluated beyond task time and error rate. Elbert and Sarnow [51] frame **AR** in order picking as both a benefit and a risk: additional information can guide the worker, but excessive or poorly placed information can overload attention. Windhausen, Heller, Hilken, *et al.* [52] study smart glasses in warehouse picking from the perspective of worker well-being, showing that subjective strain, comfort, and user experience must be considered alongside operational performance. Gentile, Musolino, Dastan, *et al.* [53] compare numeric-code, **2D**-map, and **3D**-map visualisations for order picking with **AR** glasses and show that the form of spatial information affects both performance and preference. Together, these studies caution against assuming that richer visual feedback is automatically better. Effective picking overlays should be minimal enough to avoid clutter, but informative enough to support confident selection.

Most **AR** picking systems, however, focus on human-only workflows or on guiding a worker to a target already known by a warehouse management system, barcode, list, or menu. In such cases, the system does not need to infer the target from natural language and live egocentric perception. The present dissertation addresses a different problem: the target is spoken by the operator, grounded in the current visual scene by an edge perception pipeline, displayed as a perception hypothesis, and then used by a robot execution branch. This changes the role of **XR**. Instead of only guiding a human to a known item, **XR** is used to verify whether an Artificial Intelligence (**AI**)-enabled robotic system has correctly grounded a natural-language command in the current scene.

Warehouse **CV** surveys reinforce this distinction. Tadjine, Ouafi, Benlamoudi, *et al.* [15] show that warehouse automation increasingly uses heterogeneous sensing and computation stacks, including cameras, depth sensors, wearable interfaces, and robots. This heterogeneity makes direct comparison across systems difficult and increases the need for clear intermediate representations. Applied prototypes that combine **AR** guidance with learning-based recognition can improve task performance in specific experimental setups, but they often depend on deployment-specific calibration, constrained item sets, or retraining as inventories and environmental conditions change. In the present dissertation, this limitation motivates an architecture that does not rely on a fixed menu or a purely human-facing **AR** list. Instead, perception outputs are made visible and correctable before downstream robot action.

3.3 Edge-assisted perception and RGB-D scene grounding

A language instruction in a picking task only becomes useful when it can be connected to what the system sees. If an operator says “pick the bottle”, the robot-facing system must relate those words

to an object in the current scene, distinguish it from neighbouring objects, and estimate where it is in the workspace. This makes perception a central part of the interaction loop. In an **XR** setting, the headset is a natural sensing point because it observes the scene from the operator's perspective and can display feedback in the same workspace. At the same time, a wearable device is not an ideal place to run every perception stage: battery, heat, memory, and compute limitations restrict what can be processed locally and how consistently results can be returned.

Edge computing is commonly introduced to address this tension. By placing computation, storage, and networking resources close to the data source, edge architectures reduce the need to send every image or perception request to a distant cloud [30], [31]. This is especially relevant for interactive **XR** and robotic systems, where delayed feedback can make an overlay appear disconnected from the physical scene. It is also relevant in industrial environments, where images of the workspace, products, tools, and operators may raise data-locality, privacy, or security concerns.

The literature on edge video analytics shows how this type of architecture can make visual inference more responsive. Xu, Razavi, and Zheng [11] describe techniques such as adaptive compression, frame selection, model partitioning, and edge/cloud scheduling, with particular attention to tail latency rather than only average latency. This matters because an occasional slow response can be more disruptive to an interactive system than a slightly higher but stable processing time. Work on mobile and **AR** edge systems follows the same direction: the wearable or mobile client keeps the interaction lightweight, while nearby infrastructure handles more demanding perception stages [32], [54], [55].

However, in a collaborative picking scenario, edge perception is not only a question of where a model runs. A useful perception result must remain connected to the scene that produced it. A mask, detection, or reconstruction is only meaningful if it can still be related to the instruction, the image frame, the camera metadata, and the spatial state of the workspace. Otherwise, perception becomes a detached output: it may be computationally correct, but difficult to inspect, validate, or use for robot action.

This is where **RGB-D** scene grounding becomes important. A **2D** segmentation mask can indicate which pixels belong to the target object, but pixels alone do not tell the system where the object is in the workcell. Camera intrinsics, depth, and camera pose provide the bridge from image evidence to spatial interpretation. In classical camera geometry, image points can be related to camera coordinates using intrinsics and depth, and then transformed into a world frame using camera pose [29]. For **XR**-assisted robotics, this link is essential: the same object hypothesis must be meaningful both as a visual overlay for the operator and as a spatial estimate that can inform later robotic reasoning.

The literature therefore motivates edge-assisted perception as more than a way to increase compute. It supports a broader interaction pattern in which visual evidence is processed near the workcell, associated with spatial information, and returned quickly enough to remain useful in the operator's view. The remaining challenge is that efficient perception is not automatically transparent perception. For language-based picking, the system still needs to preserve the connection

between the command, the visual evidence, and the spatial hypothesis so that the resulting target interpretation can be inspected before any robotic action is triggered.

3.4 Language-conditioned segmentation and 2D target grounding

Object segmentation is central to the proposed workflow because it transforms a natural-language target into an inspectable image-space hypothesis. Traditional object detection returns bounding boxes or class labels, which can be sufficient for inventory or coarse localisation but are less informative when an operator must verify the exact selected object in a cluttered scene. Pixel-level segmentation provides a more precise visual representation of the selected region and is therefore better suited for XR validation.

Before focusing on the Segment Anything family, it is useful to situate language-conditioned segmentation within a broader set of open-vocabulary visual grounding approaches. One line of work uses language to localise objects with bounding boxes rather than masks. Models such as OWL-ViT and Grounding DINO show that text queries, category names, or referring expressions can be used to detect objects beyond a fixed closed-set vocabulary [56], [57]. These approaches are useful for narrowing the target region, but a bounding box may still include background, neighbouring objects, or empty space. Another line of work moves directly toward text-conditioned masks. CLIPSeg, for example, generates segmentation masks from text or image prompts, while reasoning-segmentation approaches such as LISA explore cases where the target must be inferred from a more implicit or complex instruction [58], [59]. These works show that language can guide both localisation and segmentation, but they also highlight that the output remains a visual hypothesis rather than a complete robot-ready object state.

The SAM introduced a general promptable segmentation framework trained with a large-scale mask data engine, enabling broad zero-shot transfer across segmentation tasks [35]. This work was important because it made segmentation less dependent on fixed object classes and more amenable to interactive prompting through points, boxes, or masks. However, the original SAM formulation is primarily geometrically prompted. In a natural-language picking task, the operator usually does not provide a point or bounding box; the operator says something like ‘pick the bottle’, ‘select the apple’, or ‘move the object on the left’.

Language-conditioned and open-vocabulary segmentation address this limitation by connecting text prompts to visual regions. SAM 3 is particularly relevant because it extends the Segment Anything line toward promptable concept segmentation. Carion, Gustafson, Hu, *et al.* [36] describe SAM 3 as a model that detects, segments, and tracks objects using concept prompts, including short noun phrases and visual exemplars. This aligns directly with a picking scenario where a free-form speech transcript must be reduced to a target phrase and resolved against the current camera view.

Figure 3.1 illustrates this shift from promptable visual segmentation to promptable concept segmentation. While earlier promptable segmentation systems typically rely on geometric prompts such as points, boxes, or masks, SAM 3 supports the use of visual concepts, including noun phrases and positive or negative exemplars, to identify object instances in images and videos.

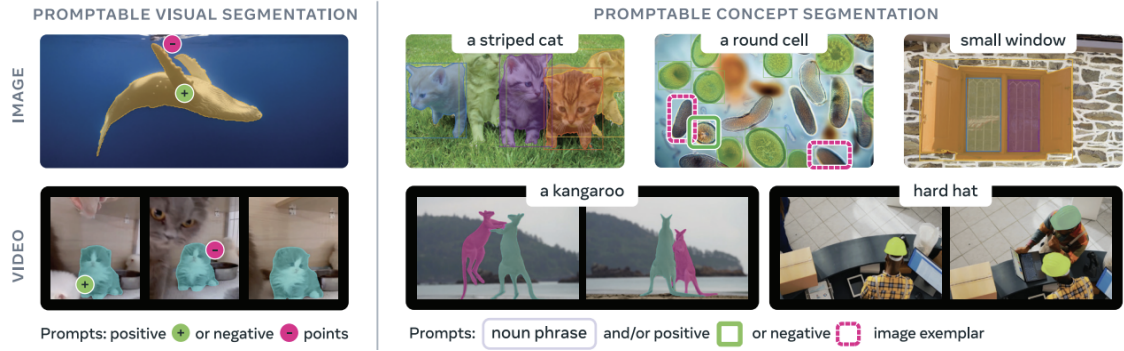


Figure 3.1: SAM 3 promptable concept segmentation [36].

This distinction is important for XR-assisted picking because the operator does not normally provide a geometric prompt. Instead, the target is expressed through speech and must be resolved against the current visual scene before any downstream 3D reconstruction or robot execution is considered.

Nevertheless, language-conditioned segmentation should not be treated as a final decision. Several failure modes are important in HRC. If there are multiple instances of the same object, a command such as ‘pick the bottle’ may be visually grounded to the wrong instance unless additional spatial or contextual information is available. Visually similar objects can also make the mask look plausible while still corresponding to the wrong target. Linguistic expressions such as ‘the left one’, ‘the small box’, or ‘the object near the corner’ depend on viewpoint, scene layout, and shared context, and may remain ambiguous from a single camera observation. Occlusions introduce another source of uncertainty: the mask may cover only the visible part of the object, merge the target with a neighbouring object, or miss parts of the boundary that are hidden behind clutter.

A further limitation appears when the segmentation result must support spatial reasoning. A mask may be correct in 2D image space while still being insufficient for manipulation if the associated depth is noisy, missing, or misaligned with the RGB image. Reflective or transparent objects, poor lighting, motion blur, and partial views can make this problem worse. Even when the outline appears visually correct, the selected region may correspond only to a visible fragment, the estimated centroid may be unstable, and the mask may not provide enough information about object pose, graspable surfaces, support contacts, or robot reachability. For this reason, segmentation is valuable for target grounding, but it cannot by itself guarantee that the selected object can be safely or successfully manipulated [28].

For this reason, segmentation in the dissertation is best understood as the first target hypothesis in a larger validation chain. The mask or polygon makes the system’s interpretation visible: it shows which image region the model associated with the spoken command. This is useful precisely because it can be inspected. If the mask selects the wrong instance, covers only part of the object, follows the wrong boundary, or corresponds to an ambiguous command, the operator can reject or refine the interpretation before downstream 3D reconstruction or robot execution is considered.

This interpretation is consistent with the broader literature on grounded robot interaction. Ichter, Brohan, Chebotar, *et al.* [41] argue that language-conditioned robot action should be constrained

by what the robot can actually do in the current environment, rather than by language plausibility alone. Although their system uses affordance functions rather than XR validation, the underlying principle is the same: linguistic interpretation must be grounded in physical context. In the present dissertation, the segmentation overlay provides the first explicit bridge between speech and scene, enabling the operator to reject or refine the target before downstream 3D reconstruction and robot control are invoked.

3.5 Object-centric 3D reconstruction and spatial validation

After the target has been isolated in the image, the relevant question is not only how to reconstruct it, but what the reconstruction is expected to support. A 3D representation used for visual inspection does not need to satisfy the same requirements as a representation used for grasp planning, collision checking, or physical manipulation. For AR feedback, the reconstruction must mainly be understandable and spatially comparable with the object in front of the operator. For manipulation, the demands become stricter: scale, pose, support contact, graspable surfaces, and coordinate-frame consistency become part of the problem. In this dissertation, the object-centric reconstruction sits between these two uses. It is not treated as a ground-truth model of the object, but as an inspectable 3D hypothesis that can help the operator judge whether the system has understood the target before the validated state is forwarded to the simulated execution branch.

Classical 3D reconstruction relies on multiple calibrated views, feature correspondences, stereo geometry, or depth fusion [29]. These methods are powerful when enough observations and accurate calibration are available, and they are the natural reference point when metric consistency, scale recovery, and pose estimation are required. However, they can be difficult to fit into a fast operator-facing loop where the system must react from one or a few egocentric observations. Recent object-centric reconstruction models take a different path by using learned priors to infer plausible 3D structure from limited visual evidence. SAM 3D Objects is directly relevant to this dissertation because it targets visually grounded 3D object reconstruction from a single image. The SAM 3D work presents a generative approach that predicts object geometry, texture, and layout from natural images, including scenes with clutter and occlusion [37].

Figure 3.2 illustrates this object-centric view of reconstruction. Instead of treating the scene only as a single global reconstruction problem, SAM 3D decomposes visual input into individual object hypotheses that can be reconstructed, displayed, and inspected separately.

For inspection, this type of reconstruction is valuable because it gives the operator more than a mask. A mask shows the visible pixels associated with the instruction; an object-centric 3D artifact can be rotated, placed in the workspace, compared against the physical object, and checked for obvious failures such as missing parts, implausible shape, wrong object identity, or poor alignment. Splat-based representations, such as 3D Gaussian Splatting, follow a similar visual logic: they can provide high-quality real-time rendering and novel-view inspection from compact scene representations [38]. Their strength is visual plausibility and inspectability, rather than direct suitability for robot collision checking or grasp synthesis.

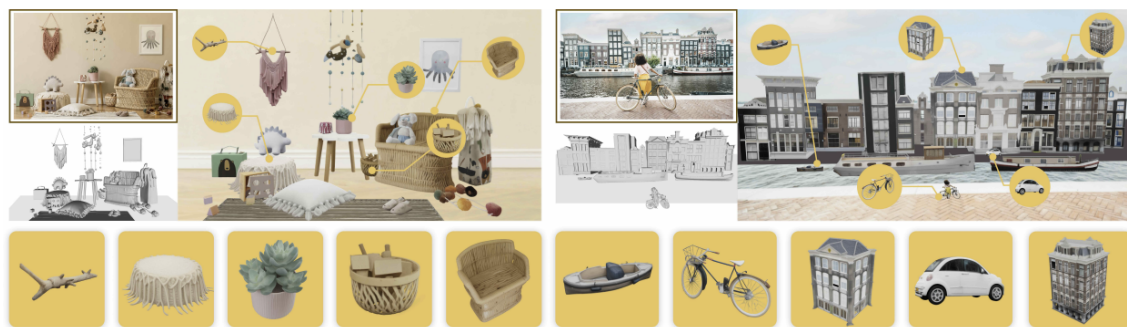


Figure 3.2: **SAM** 3D object-centric reconstruction from a single image [37].

Manipulation changes the meaning of reconstruction. In robotic grasping, **3D** information is often used to estimate object pose, infer grasp candidates, reason about contacts, or avoid collisions. Reviews of vision-based grasping and learning-based grasp synthesis show that depth, point clouds, meshes, object pose, and grasp representations are central to manipulation pipelines, but also that small errors in pose, scale, occlusion handling, or graspable geometry can directly affect execution feasibility [28], [60], [61]. A visually convincing mesh is therefore not automatically a manipulation-ready model. It may be sufficient to help a human inspect the target, while still being too uncertain for autonomous grasp planning without additional validation.

Scale and pose are especially delicate in single-view reconstruction. Hidden surfaces are inferred rather than observed, object scale may be ambiguous, symmetric objects may have uncertain orientation, and the local object frame may not correspond to a frame that is useful for grasping. Monocular geometry models such as MoGe illustrate the value of learned **3D** priors from limited visual input, while also reinforcing that open-domain single-image geometry estimation still depends on assumptions about scale, camera geometry, and coordinate-frame interpretation [62]. In an **AR**-assisted picking workflow, this means that a reconstruction should be checked not only as a shape, but also as a spatial object: whether its apparent scale is reasonable, whether its pose agrees with the physical object, whether its projected silhouette is compatible with the segmentation, and whether its placement is consistent with local depth evidence and the support surface.

This is where operator validation becomes part of the reconstruction story rather than a separate interface detail. In **AR** robot programming and mixed-reality waypoint generation, the operator is often kept in the loop so that robot intentions, waypoints, or task elements can be inspected and corrected before execution [63], [64]. The same principle is applied here to the reconstructed object. The operator should not only see a final mesh; the operator should be able to reject it when it corresponds to the wrong object, refine it when the view is insufficient, or manually correct its placement when the pose, scale, or centroid is visibly wrong.

In this dissertation, the resulting validation chain is intentionally layered. The **2D** mask identifies the visible target region. Local depth and camera geometry on the **AR** client provide metric evidence for estimating and anchoring a world-space centroid. The object-centric reconstruction adds a richer visual hypothesis that can be compared with the physical object. If the artifact is misaligned, scaled incorrectly, oriented ambiguously, or reconstructed with implausible geometry, the validation

interface must allow cancellation, guided refinement, or manual correction before the result is accepted. The reconstructed object is therefore not the final proof that manipulation is possible; it is an intermediate representation that makes the system’s interpretation visible, questionable, and correctable before the simulated **VLA**-based execution branch receives the validated target state.

3.6 Natural language, mixed reality and grounded robot programming

Once perception has produced a visible target hypothesis, the next question is how a human instruction becomes a robot task without hiding the intermediate interpretation from the operator. In language-based robot programming, the critical issue is not only whether a command can be converted into waypoints, plans, or actions, but whether the operator can inspect what the system understood before execution. This distinction is especially important in collaborative picking: a fluent natural-language interface is only useful if the selected object, the intended action, and the resulting robot behaviour remain visible and correctable.

Earlier work on **AR**-assisted **PbD** already shows the value of making robot-programming artefacts spatially inspectable. Soares, Petry, and Moreira [63] present an **AR**-based robot **PbD** system in which users can visualise taught paths and refine robot motions. The relevance of this work is not that it solves language grounding, but that it treats robot programming as something the operator should be able to see, adjust, and verify in the workspace. In the context of this dissertation, that idea is moved earlier in the pipeline: before a robot motion is accepted, the perception result that will condition the motion must also be made inspectable.

More recent work connects natural language and mixed reality more directly. Fang, Zieliński, Maes, *et al.* [64] propose an **LLM**-based mixed-reality framework that converts natural-language commands into robot waypoints, visualises the generated actions in **AR**, and executes them after user confirmation. Gkournelos, Konstantinou, and Makris [65] discuss an **LLM**-based approach for human–robot collaboration in assembly, while Yin, Fan, Liu, *et al.* [66] propose a scene-centric mixed-reality approach for robot programming in collaborative manufacturing. These works show how language and **MR** can reduce programming effort and support preview, but they also reveal an important boundary: the confirmation step is often centred on the planned task, waypoint, or robot action, while the grounding of the commanded object in a live, cluttered scene is less explicitly exposed.

This difference matters because guidance, preview, validation, and correction are not the same interaction. Guidance tells the operator what to do. Preview shows what the robot is expected to do. Validation asks whether the system’s interpretation is acceptable. Correction gives the operator a way to change that interpretation before it affects execution. For an **AR**-assisted picking system, previewing a future robot motion is useful, but it is not sufficient if the selected target object has not first been made visible as a grounded perception hypothesis. A planned trajectory can look reasonable while still being attached to the wrong object, an ambiguous instruction, or a poorly localised target.

Robotics-focused surveys of **LLM** integration describe modular patterns in which language models support task interpretation, planning, explanation, and code generation, while also stressing that deployment requires grounding, verification, and safety-aware integration [67]. Grounded language-to-action systems provide a complementary perspective. SayCan combines language-model reasoning with robot affordance estimates so that selected actions are both linguistically appropriate and feasible for the robot [41]. Although this architecture differs from the present dissertation, the underlying principle is relevant: a command should not be executed only because it is plausible in language. It must be constrained by the objects, geometry, and feasible actions available in the current environment.

The dissertation follows this concern, but places the emphasis on target inspectability rather than on language planning alone. The speech transcript is not treated as a direct robot program. It first becomes a target-grounding problem: the edge server resolves a target phrase against the current image, returns a segmentation overlay, produces an object-centric reconstruction when possible, and lets the operator validate the result through **AR** feedback. Only after this visual grounding loop does the confirmed state become suitable input for the robot execution branch. In this sense, the contribution is not a new **LLM** planner, but a workflow that places grounded perception, operator validation, and correction between speech and robot action.

3.7 VLA policies and simulation-based adaptation

Recent **VLA** policies such as **RT-2**, OpenVLA, and the π_0 family demonstrate increasingly general natural-language manipulation by connecting visual observations and language instructions directly to robot actions. However, this also introduces a transparency problem in collaborative settings: if target selection and action generation are coupled inside an opaque learned policy, the operator may not know which object the model selected, how the instruction was interpreted, or why a particular action is being generated.

The first of these references, **RT-2**, is important because Zitkovich, Yu, Xu, *et al.* [43] co-fine-tune **VLMs** on web-scale visual-language tasks and robot trajectory data, representing robot actions in a tokenised form that the model can generate. This showed that semantic knowledge acquired from web-scale visual-language data can transfer to robotic control. OpenVLA extends this direction in an open-source setting, providing a generalist **VLA** model trained on diverse robot demonstrations and designed for downstream fine-tuning [44]. These models make language-conditioned manipulation more flexible, but they do not remove the need to decide what part of the scene should be trusted as the target of the command.

The π_0 family is especially relevant to this dissertation because the robotic execution branch is based on $\pi_{0.5}$. Black, Brown, Driess, *et al.* [45] introduce π_0 as a vision-language-action flow model for general robot control, using a pre-trained vision-language backbone and flow matching to model continuous robot actions. Black, Brown, Darpinian, *et al.* [46] extend this direction toward broader open-world generalisation through co-training across heterogeneous tasks, robots, and data sources. The strength of this family is that it can condition continuous manipulation on

visual observations and language. Its limitation, for the purpose of collaborative picking, is that performance still depends on embodiment, camera viewpoint, gripper configuration, workspace distribution, and the amount and quality of task-specific adaptation data.

Large-scale robot datasets and demonstration-generation methods are therefore part of the same story. The Open X-Embodiment dataset shows the value of pooling demonstrations across many robots and tasks to train more generalist policies [68]. MimicGen addresses a complementary problem: generating additional manipulation data from a smaller number of human demonstrations, which is relevant when task-specific trajectories are expensive to collect [69]. These works support the use of simulation and dataset expansion as a practical adaptation route, but they also show that policy performance is inseparable from the data distribution used during training and fine-tuning.

For this dissertation, this motivates a conservative role for the **VLA** branch. The robot policy is not treated as a direct speech-to-action system that receives an unverified operator command and decides both the target and the motion. Instead, the **AR** perception loop first turns the spoken instruction into a visible target hypothesis through segmentation, centroid estimation, and object-centric reconstruction when available. The operator then validates, refines, corrects, or cancels that interpretation. Only after this step does the confirmed target state become part of the task context used by the execution branch.

This separation is important because a correct action policy cannot compensate for an incorrectly grounded target. In a cluttered picking scene, two objects may be visually similar, partially occluded, or described ambiguously by the operator. If a **VLA** policy receives only raw observations and a language instruction, the selected object may remain implicit. By placing target grounding before action generation, the dissertation does not claim to eliminate all ambiguity; rather, it makes ambiguity in target selection visible and correctable before simulated robot execution is attempted.

The simulation setting also defines the scope of the contribution. The **VLA**-based execution branch is evaluated in a robotic **DT**, where task-specific demonstrations, dataset expansion, and policy fine-tuning can be explored before transfer to a physical workcell. This is useful because simulation allows controlled testing of object placement, camera observations, action representations, and policy behaviour. However, it is not equivalent to validated physical deployment. Transfer to the real robot would still require calibration, hardware integration, safety validation, contact robustness, and additional testing under real sensing and actuation conditions.

The role of the **VLA** literature in this dissertation is therefore twofold. It provides the execution model that can transform language-conditioned visual context into robot actions, and it exposes the reason why an intermediate validation layer is needed. As **VLA** policies become more capable, the need to make their inputs and intended targets inspectable becomes more important in collaborative scenarios. The proposed workflow responds to this by decoupling target validation from action generation: the operator first approves the grounded object state in **AR**, and only then is the task passed to the simulation-based **VLA** execution branch.

3.8 Synthesis and gap addressed by this dissertation

Taken together, the works discussed in this chapter show that the main challenge is not the absence of capable components, but how they are connected before robot execution. **AR** and **XR** interfaces already support instructions, spatial cues, robot state, safety information, and motion intent in the operator’s workspace [5], [6], [14], [48], [52], [53]. However, many picking systems assume that the target item is already known, while many **HRC** visualisation systems focus on guidance, safety, trajectory preview, or legibility rather than on validating a live target-grounding result from a spoken command.

Edge computing, promptable segmentation, object-centric **3D** reconstruction, natural-language robot programming, and **VLA** policies address different parts of this pipeline [11], [30], [31], [35]–[37], [41], [43]–[46], [63]–[66]. Edge processing makes heavier perception feasible near the workcell, but does not by itself make model outputs understandable or correctable. Segmentation and reconstruction can produce candidate object representations, but a mask is not a validated target and a reconstructed object is not ground truth. Similarly, language-based and mixed-reality programming systems can simplify task specification and preview, while **VLA** policies connect language, vision, and action more directly. This directness can reduce inspectability when target selection and action generation are coupled inside a learned policy.

The gap addressed in this dissertation is therefore an *inspectability gap*. The contribution is not defined by simply combining **AR**, language, edge perception, **2D** segmentation, **3D** reconstruction, and robotic execution. Rather, the relevant gap is the lack of a workflow in which the perceptual interpretation of a spoken picking command becomes an explicit intermediate state: visible to the operator, spatially related to the workspace, and correctable before it is passed to robot execution.

This framing requires distinguishing between four interaction functions that are often conflated in **AR** and robotics. *Guidance* helps the operator perform a task through instructions, routes, object cues, or safety information. *Preview* shows future system behaviour, such as a waypoint, path, trajectory, action, or generated task sequence. *Validation* asks whether the system’s interpretation is correct, while *Correction* allows the operator to reject, refine, manually adjust, or cancel that interpretation before execution. In this dissertation, guidance and preview are useful but insufficient: the central requirement is validation and correction of the grounded target state before the robotic branch is allowed to act.

Table 3.1 summarises this comparison across interaction, sensing, edge processing, grounding, reconstruction, operator correction, robot execution, **VLA** control, and evaluation. When a row aggregates several related works, the status indicates the representative capability of that work or system class rather than exact feature parity. The purpose is to show which parts of the proposed validation chain are already well supported and where the inspectable target-state workflow remains underdeveloped.

Table 3.1: Systematic comparison of representative works and system classes against the inspectable spoken-command picking workflow.

Work/system	HMD	Speech/NL	Ego RGB	Edge	2D	3D	Corr.	Robot	VLA	Evaluation
AR pick-by-vision systems [14], [52], [53]	Y	–	P	–	P	–	P	–	–	User/task evaluation of picking guidance and interface design.
XR/HRC legibility and intent communication [5], [6], [48]	P	P	P	–	–	–	P	P	–	Surveys, prototypes, and formal accounts of legible robot behaviour.
Edge-assisted perception and video analytics [11], [30], [31]	P	–	P	Y	P	P	–	–	–	System-level evaluation of latency, offloading, and video analytics.
Promptable segmentation models [35], [36]	–	P	P	–	Y	–	–	–	–	Segmentation benchmarks and model-level evaluation.
Object-centric reconstruction and learned geometry [37], [38], [62]	–	–	P	–	P	Y	P	–	–	Reconstruction, rendering, and geometry evaluation.
AR-based PbD and motion refinement [63]	Y	–	–	–	–	P	Y	Y	–	Prototype evaluation of AR-supported demonstration and refinement.
LLM/MR robot programming [64]–[66]	Y	Y	P	P	P	P	P	P	–	Prototype or case-study evaluation of language-based robot programming.
Grounded language-to-action robotics [41]	–	Y	P	–	P	–	–	Y	–	Robot task evaluation with language-conditioned skill selection.
RT-2 [43]	–	Y	Y	–	P	–	–	Y	Y	Robot/model evaluation of vision-language-action generalisation.
OpenVLA [44]	–	Y	Y	–	P	–	–	Y	Y	Robot/model evaluation of open VLA policies.
π_0 and $\pi_{0.5}$ [45], [46]	–	Y	Y	–	P	–	–	Y	Y	Robot/model evaluation of generalist action policies.
Robot-learning datasets and adaptation [68], [69]	–	P	P	–	–	–	–	P	P	Dataset-scale or data-generation evaluation for robot learning.
This dissertation	Y	Y	Y	Y	Y	Y	Y	P	P	AR prototype and simulation-based execution in a DT.

Legend: Y = explicitly addressed; P = partially or indirectly addressed; – = not central. HMD = head-mounted display; NL = natural language or speech; Ego RGB = egocentric RGB observation; 2D = language-conditioned 2D grounding; 3D = object-centric 3D reconstruction/hypothesis; Corr. = operator rejection/refinement/manual correction; Robot = robotic execution; VLA = vision-language-action policy.

The comparison shows that existing work provides strong building blocks for **AR** guidance, **XR/HRC** legibility, edge-assisted perception, language-conditioned segmentation, object-centric **3D** reconstruction, mixed-reality robot programming, and **VLA**-based manipulation. However, these elements are rarely organised around the requirement that a spoken picking command should first become an explicit, inspectable, spatially anchored, and correctable target state before robot execution.

This is the position adopted in this dissertation. The **AR** interface is therefore used not only as a guidance display, but as a validation and correction layer for a language-conditioned target hypothesis produced from egocentric visual evidence and edge-side perception. The proposed workflow mitigates ambiguity by making the system's interpretation visible before it enters the robot-control branch.

This workflow does not eliminate all uncertainty. Speech recognition may be incomplete, commands may be underspecified, similar objects may be confused, depth alignment may be noisy, segmentation may be incomplete, single-view reconstruction may be geometrically inaccurate, and a **VLA** policy may still generate an unsuitable action. For this reason, the dissertation does not claim a fully autonomous or fully validated physical robot deployment. The robotic execution branch is evaluated through simulation-based digital-twin execution, while transfer to the physical workcell would require additional calibration, safety validation, contact robustness, and hardware testing.

Chapter 4

Methodology and System Design

This chapter presents the methodological and design decisions behind the proposed XR-assisted intelligent picking system. The system is conceived as a human-supervised pipeline that connects operator intent, edge-assisted perception, spatial validation in XR, and simulation-based robotic manipulation.

The work follows an engineering design methodology in which a distributed system is decomposed into functional modules, integrated progressively, and evaluated through increasingly complete stages of an intelligent picking workflow. The central design choice is to avoid a direct speech-to-action architecture. Instead, the operator’s spoken intent is first grounded in visual observations, represented through spatial XR feedback, validated by the operator, and only then made available to the downstream manipulation branch. This structure is motivated by the ambiguity of natural-language commands in real environments and by the need to make learning-based robotic decisions inspectable before they affect the physical or simulated execution pipeline [41], [45], [46].

The methodology therefore treats perception outputs as intermediate hypotheses rather than final commands. A segmentation mask, reconstructed object, estimated centroid, or candidate object pose can support robotic reasoning, but it must first be exposed to the operator in an interpretable spatial form. This design makes the human operator part of the grounding and validation loop, while still allowing the system to benefit from modern perception, reconstruction, simulation, and robot-learning tools.

4.1 System Design

The proposed system is designed as a closed-loop architecture linking three main domains: the operator-side XR client, the edge perception server, and the robotic simulation and VLA branch. These domains address different requirements of the intelligent picking workflow. The headset provides situated interaction and real-time feedback, the edge server executes computationally demanding perception and reconstruction processes, and the simulation branch provides a controlled environment for testing manipulation behaviours and preparing learning-based policies.

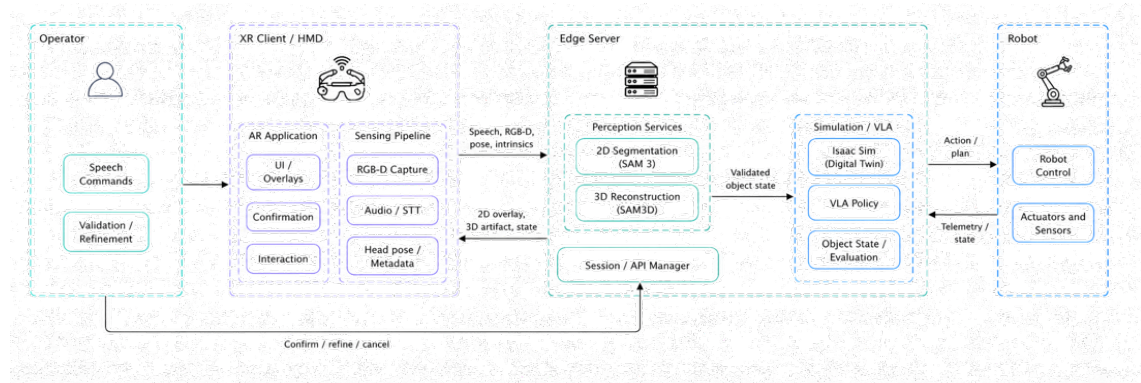


Figure 4.1: Closed-loop architecture diagram of the proposed XR-assisted intelligent picking system.

The **XR** client is the operator-side part of the system. It captures the operator’s speech, egocentric **RGB** observations, depth measurements, headset pose, camera metadata, and interaction events. It also renders the system interpretation in the user’s spatial context through segmentation overlays, object markers, reconstructed artifacts, state panels, and validation controls. The headset is therefore both a sensing platform and an interaction medium: it allows the operator to observe how the system interprets the scene and to intervene when that interpretation is incomplete or incorrect.

The edge server is responsible for perception, reconstruction, session management, and artifact serving. It receives speech and image streams from the headset, associates them with the corresponding interaction session, derives an object-level target prompt, performs language-conditioned **2D** segmentation, generates object-centric **3D** reconstruction artifacts, and exposes the resulting state and artifacts to the **XR** client. This follows an edge-assisted design in which latency-sensitive interaction remains on the headset, while computationally demanding inference is executed on a nearby machine with stronger **GPU** resources [30], [31], [54].

The robotic execution branch is simulation-first. In the current work, it is represented by a **DT** in NVIDIA Isaac Sim containing the KUKA workcell, a Robotiq gripper, task-relevant cameras, reconstructed objects, and a data-generation and policy-evaluation workflow. The validated object state produced by the **XR** and edge-perception stages is used as grounded input for **VLA**-based manipulation. This allows the manipulation pipeline to be developed, inspected, and tested before the full loop is transferred to a physical robot cell.

A key aspect of the design is that the system does not collapse perception and action into a single automatic step. The user instruction gives semantic intent, the perception stack proposes an interpretation of the target object, and the **XR** layer makes that interpretation visible. The operator can then confirm, refine, correct, or cancel the result. This creates a controlled transition from uncertain perception to structured object state.

Operational workflow

1. **Capture operator intent and scene context** — the operator starts a recording interaction on the **HMD**. The **XR** client creates a unique session identifier and buffers the corresponding speech, **RGB** frames, depth observations, camera intrinsics, timestamps, and headset pose metadata. This session identifier is used as the main key for associating all data produced during the same interaction.
2. **Transmit session-linked data to the edge server** — the **XR** client sends visual frames, metadata, and speech transcription events to the edge server through the communication layer. **ROS** topics are used for time-sensitive event, frame, segmentation, pose, and refinement messages, while **HTTP** endpoints are used to poll session state, retrieve artifact URLs, cancel sessions, and access reconstruction results.
3. **Ground the spoken target in the image stream** — the edge server associates the transcript and visual frames through the session identifier. A target phrase is derived from the transcript and used to perform language-conditioned **2D** segmentation on recent **RGB** observations. The resulting polygonal outline is returned to the **XR** client, where it is shown as a visual overlay and used as geometric evidence for object localisation.
4. **Estimate and visualise the object hypothesis in XR** — the **XR** client combines the returned **2D** segmentation with locally cached depth, camera pose, and camera intrinsics to estimate a world-space object centroid. In parallel, the edge server uses the selected **RGB** image and its corresponding segmentation mask to produce an object-centric **3D** reconstruction. This reconstruction is not generated from a depth-derived point cloud; it is treated as a visual object hypothesis that must be spatially placed, inspected, and validated in **XR**. When the artifact is ready, the **XR** client downloads it, applies the required transform and convention metadata, and displays it in the operator's physical workspace.
5. **Validate, refine, or cancel the interpreted target** — the operator inspects the transcript, captured frames, segmentation overlay, centroid estimate, and reconstructed **3D** artifact through the **XR** interface. The interpreted target can be confirmed, refined through additional guided scan frames, manually adjusted, or cancelled. This validation step is central to the workflow because it makes the system's interpretation visible before robotic execution.
6. **Execute the validated task in the robotic branch** — once the target state is accepted, the confirmed object information is made available to the robotic execution branch. In the current dissertation, the manipulation stage is evaluated in a **DT** using a fine-tuned **VLA** policy. This allows the grounded target state and natural-language instruction to be tested in a controlled pick-and-place scenario before transfer to the physical workcell.

Assumptions and constraints

- The **XR** client remains responsible for interaction, sensing, spatial alignment, visual feedback, and operator validation, while computationally expensive perception and reconstruction run on the nearby edge server.
- The workflow is session-centric: speech, **RGB-D** observations, segmentation messages, reconstruction artifacts, object pose estimates, refinement inputs, and validation decisions must remain linked to the same interaction episode.
- The **2D** segmentation and **3D** reconstruction outputs are treated as hypotheses that require operator inspection, not as automatically executable commands.
- The object-centric **3D** reconstruction is generated from the selected **RGB** view and segmentation mask, rather than from a depth-derived point cloud. It is therefore used primarily as an inspectable visual hypothesis, while depth, camera intrinsics, and headset pose are used by the **XR** client to support spatial placement and centroid estimation.
- Spatial consistency depends on timestamped camera metadata, depth/**RGB** alignment, camera intrinsics, headset pose estimates, and transform conventions between the edge server and Unity.
- The robotic execution branch is currently validated in simulation; transfer to the physical workcell requires calibration, hardware integration, safety validation, and additional robustness testing.
- The architecture assumes that the local network can support the exchange of selected visual observations, metadata, perception outputs, and reconstruction artifacts with sufficient responsiveness for interactive validation.
- The reconstructed object artifact may support downstream manipulation reasoning, but it is not assumed to be a metrically perfect model of the physical object. Uncertainty may remain in scale, hidden geometry, pose, alignment, and reconstruction quality, especially when the available visual evidence is limited.
- The closed-loop design assumes that uncertain perception, ambiguous language, object similarity, occlusion, or reconstruction artifacts can be handled through operator validation, refinement, manual correction, or cancellation before execution.
- The system separates target grounding from robot action generation. The **VLA** branch receives a validated object state and task context rather than acting directly on an unverified speech command.
- The current implementation prioritises integration, inspectability, and simulation-based validation over full physical deployment. Real-world operation would require additional

safety-layer integration, robot-cell calibration, and testing under industrial operating conditions.

Following these assumptions and constraints, the prototype is intended for close-range **XR**-assisted object interaction rather than long-range scene perception. In the implemented setup, the **XR** client captures **RGB** frames at 1280×720 , encodes them as JPEG images, and sends them with timestamps, camera intrinsics, and headset pose metadata. The capture and buffering interval is 0.2 s, corresponding to approximately 5 Hz, while the live streaming interval is 0.05 s, corresponding to up to 20 Hz. Depth observations are cached locally at 5 Hz and are used by the headset to estimate the object centroid after the edge server returns the **2D** segmentation outline.

The spatial grounding stage assumes that the **RGB** frame used for segmentation and the depth frame used for centroid estimation are temporally consistent. For this reason, the implemented prototype rejects **RGB**–depth matches above 120 ms during centroid estimation. Depth samples are also filtered to the interval between 0.15 m and 7.5 m. This interval defines the coded acceptance range used by the grounding module, not a guarantee of uniform depth quality across that full range. In practice, the validation scenarios focus on close-range interaction distances, where the target object occupies enough of the camera image and provides enough valid depth samples. The preferred operating distance is therefore approximately between 0.3 m and 1.0 m from the headset, with the most stable operation expected around 0.3–0.5 m. Objects that are too distant, too small in the image, heavily occluded, reflective, transparent, or poorly lit are outside the preferred operating conditions of the current prototype.

Latency is considered at the level of individual stages rather than as a single end-to-end value. The interaction separates local headset feedback, speech transcription, frame transmission, edge-side **2D** segmentation, asynchronous **3D** reconstruction, artifact retrieval, and operator validation. This is important because the segmentation overlay should provide relatively fast feedback for target inspection, while the reconstructed **3D** artifact can take longer and is treated as an asynchronous result. The prototype therefore assumes a local network connection to the edge server, sufficient illumination, stable headset tracking, visible target objects, limited occlusion, and an operator positioned close enough to inspect and correct the returned hypothesis.

4.2 Design Principles

The methodology is structured around the following design principles:

Human-supervised grounding

Natural language is used to express intent, but it is not treated as a sufficient basis for robot execution. A spoken instruction such as selecting or manipulating an object must first be grounded in visual observations of the current scene. The system therefore transforms the instruction into a target-object hypothesis, presents that hypothesis through spatial **XR** feedback, and waits for operator validation before the result is forwarded to the manipulation branch. This reduces the risk of acting on an incorrect semantic interpretation, especially in scenes where several objects may satisfy a similar description.

Session-based traceability

Each operator interaction is represented as a session. The session identifier links the spoken instruction, **RGB** frames, frame metadata, depth-derived spatial evidence, segmentation output, reconstruction artifacts, refinement observations, and validation decisions. This is important because the pipeline is asynchronous: the transcript, image stream, segmentation outline, **3D** artifact, and user decision are produced at different times and by different modules. A session-centric design preserves coherence between these data products and supports cancellation, supersession, recovery, and later analysis of failure cases.

Edge-assisted perception

The perception pipeline is distributed between the headset and the edge server. The headset performs sensing, timestamping, local interaction, depth-based spatial grounding, and rendering. The edge server performs heavier inference stages, including text-prompted segmentation and object-centric reconstruction. This division allows the **XR** application to remain responsive while still using perception models that would be impractical to run entirely on the headset. It also keeps the system extensible, since perception components can evolve on the edge server without redesigning the whole interaction layer.

Spatially explicit validation

The system is designed to expose intermediate results in the same spatial context in which the task is defined. Instead of presenting only numerical states or detached image previews, the **XR** interface displays segmentation outlines, object centroids, reconstructed artifacts, pose cues, and validation controls in relation to the physical workspace. This allows the operator to compare the system's interpretation with the real scene and to identify errors in object selection, alignment, scale, or orientation before the result is used downstream.

Simulation-first robotic validation

The robotic execution branch is developed first in simulation. The **DT** allows reconstructed objects, camera observations, trajectories, grasp candidates, demonstrations, and policy outputs to be tested without exposing a physical operator or robot to unvalidated behaviour. This does not replace later physical calibration and real-cell validation, but it provides a controlled environment for developing the manipulation pipeline and analysing failures under repeatable conditions.

Explicit interfaces and measurable outputs

Each subsystem exchanges explicit data products: transcripts, image frames, camera metadata, segmentation polygons, **3D** artifacts, object poses, demonstration datasets, and policy actions. This makes the pipeline easier to inspect and evaluate because failures can be associated with a specific stage, such as speech transcription, target extraction, segmentation, depth alignment, reconstruction, validation, policy inference, or robot execution. The design therefore supports both implementation debugging and experimental validation.

4.3 System Decomposition

The system is decomposed into functional modules rather than monolithic software blocks. This decomposition reflects the different responsibilities involved in the pipeline: sensing and interaction, perception, spatial grounding, validation, simulation, and robotic execution. Table 4.1 summarises the role of each module and the representative data products exchanged throughout the workflow.

Table 4.1: Main system modules and exchanged data products.

Module	Main responsibility	Representative data products
XR client	Captures operator intent and egocentric context; renders validation feedback; supports confirmation, refinement, and cancellation.	Audio segments, transcripts, RGB frames, depth frames, camera poses, intrinsics, segmentation overlays, centroid markers, validation decisions.
Edge perception server	Performs language-conditioned 2D segmentation, object-centric 3D reconstruction, session management, and artifact serving.	Target prompt, segmentation polygon, mask, Binary glTF (GLB) mesh, Polygon File Format (PLY) representation, preview renders, transform metadata, session state.
Spatial grounding layer	Converts image-space evidence into world-space object hypotheses using camera metadata, depth, and projection geometry.	Object centroid, support-plane evidence, world-space candidate pose, reprojection checks, alignment diagnostics.
Operator validation layer	Makes model outputs inspectable and actionable before execution.	Confirmation, refinement request, manual pose correction, cancellation, accepted object state.
DT and VLA branch	Uses the validated object state to generate, train, test, and evaluate robot actions in simulation.	USD scene, camera observations, demonstrations, actions, policy checkpoints, simulated execution results.
Robot execution interface	Represents the target manipulation platform and receives validated states or policy actions.	Object pose, gripper state, end-effector commands, robot telemetry, execution status.

This modular structure also makes the system adaptable. Individual components can be replaced or extended without changing the overall methodology. For example, the segmentation model can be improved independently from the **XR** interaction layer, and the simulation branch can evolve from demonstration recording to policy evaluation without changing the way the operator validates reconstructed objects.

4.4 Validation Strategy

The validation strategy follows the research questions defined in Chapter 1 and the modular structure of the proposed workflow. The system transforms a spoken picking instruction into a session-linked object hypothesis, exposes it through XR feedback, allows operator validation or correction, and uses the accepted target state in a simulated robotic execution branch. The validation therefore considers both individual stages and their role in an inspectable and correctable picking workflow.

A single end-to-end success case would not be sufficient, since failures may originate from speech transcription, target extraction, segmentation, depth-based grounding, 3D reconstruction, XR alignment, operator validation, or simulated robot execution. The validation is therefore organised by research question and supported by repeated tests: 100 complete interaction sessions for session integration, ten trials per controlled perception, reconstruction, and validation condition, and 50 rollouts for the simulated VLA execution branch.

- **RQ1: Session integration.** The first validation objective is to verify whether the proposed XR–edge–robot architecture can associate all data products generated during the same picking interaction. This is evaluated over 100 complete sessions to test the consistency of the session-management protocol across asynchronous executions. A session is successful when the same identifier is preserved across the operator’s speech event, RGB frames, frame metadata, camera intrinsics, depth-related spatial information, segmentation output, reconstruction state, returned artefacts, validation actions, refinement frames, and cancellation or supersession events. It fails if outputs from different commands are mixed, if an artefact is loaded for the wrong session, or if a stale or superseded result is presented as active.
- **RQ2: Inspectable perception.** The second validation objective is to determine whether the perception pipeline produces outputs that are inspectable before execution: the speech transcript, language-conditioned 2D segmentation outline, estimated world-space centroid, and reconstructed 3D artefact.

For speech and intent capture, each condition is evaluated over ten trials. A trial is successful when the transcript preserves the target object, or a semantically equivalent phrase, so that the edge server can derive the visual grounding prompt. Minor errors are acceptable if they do not change the target; failures occur when the target is omitted, replaced, or corrupted.

For segmentation, each scene condition is evaluated over ten trials. A trial is successful when the mask or outline corresponds to the intended object and is sufficiently aligned for operator recognition. It is partially successful when the correct object is selected but the mask is incomplete, slightly imprecise, or includes small neighbouring regions while remaining inspectable. It fails when the wrong object is selected, no usable mask is produced, or the mask is too inaccurate for target confirmation. Tested conditions include a clearly visible target, nearby clutter, partial occlusion, visually similar objects, and multiple possible targets.

For centroid estimation, each spatial condition is evaluated over ten trials using the Euclidean distance, in metres, between the estimated world-space centroid and the measured or manually

marked object position. A centroid estimate is successful when it supports object-level XR inspection and target anchoring, using a reference tolerance of approximately 0.05 m for close-range validation. Estimates above this tolerance are partial when they still indicate the correct object region but require manual correction, and failed when unstable, missing, or associated with the wrong location.

For 3D reconstruction, each condition is evaluated over ten trials. A trial is successful when the edge server generates an artefact, the XR client loads it, and the reconstructed shape remains recognisable as the intended object. The reconstruction is inspectable when its main geometry, approximate scale, orientation, and placement allow the operator to understand what was reconstructed. Partial success covers recognisable artefacts with scale mismatch, orientation offset, incomplete geometry, or visible distortion requiring correction. Failure occurs when the artefact cannot be loaded, the object is not recognisable, or the geometry is too incomplete or distorted for meaningful inspection.

- **RQ3: Operator validation.** The third validation objective is to verify whether XR feedback makes the system's interpretation visible, correctable, and cancellable before robotic execution. Each validation action is evaluated over ten trials. A trial is successful when the operator can inspect the interpreted target and confirm a correct hypothesis, cancel an incorrect or uncertain one, request refinement, or manually adjust a close but insufficiently aligned target state. The accepted state must result from explicit validation rather than from the initial speech command or segmentation output alone. A trial fails if the hypothesis cannot be distinguished, if the relevant perception output is not exposed, or if an incorrect result cannot be cancelled or corrected before acceptance.
- **RQ4: VLA execution in simulation.** The fourth validation objective is to evaluate the simulated robotic execution branch after target grounding and validation. The manipulation stage is tested in the robotic DT, where the VLA-based policy attempts the pick-and-place task using the available task and object representation. This branch is evaluated over 50 simulated rollouts. A rollout is successful when the robot picks the intended object and places it in the target container according to the simulation success condition. Failed rollouts are categorised by observed cause, including grasp failure, object slip or drop, timeout, reachability problem, placement failure, collision or contact mismatch, or policy behaviour inconsistent with the intended task. This separates execution failures from failures originating earlier in perception or validation.
- **RQ5: Failure modes and limitations.** The final validation objective is to identify the main limitations of the complete workflow. Failure cases are analysed by stage, triggering condition, and effect on the rest of the pipeline. The analysis considers ambiguous or incorrectly transcribed commands, wrong-object segmentation, incomplete masks, insufficient depth samples, unstable centroid estimates, distorted reconstructions, scale or orientation mismatch, XR alignment issues, refinement limitations, and simulated execution failures. This makes

the validation explanatory rather than only descriptive, since unsuccessful trials indicate which part of the workflow failed and what mitigation would be required.

The validation results are interpreted through two complementary views: first, by the criterion appropriate to each technical stage, including transcript usability, segmentation correctness, centroid error, reconstruction inspectability, validation action completion, and simulated task outcome; and second, by the five research questions, covering session coherence, inspectable perception, operator validation or correction, simulated task execution, and limiting failure modes.

Overall, this strategy verifies whether the workflow can transform a spoken instruction into a session-linked, inspectable, and correctable object hypothesis, and whether that hypothesis can support simulated robotic manipulation. The repetition counts capture both protocol-level reliability and condition-specific behaviour across perception, reconstruction, validation, and simulated execution.

4.5 Architectural and Technological Design Choices

The technological choices in this dissertation follow from the needs of the proposed workflow. The operator must be able to express intent in the physical workspace, inspect how the system interpreted that intent, and validate the resulting object hypothesis before it is used downstream. For this reason, the architecture combines an optical see-through XR interface, edge-side perception and reconstruction, session-based coordination, and a simulation-first robotic branch.

This structure keeps perception and execution separated by an explicit validation step. Rather than treating segmentation or reconstruction as final commands, the system returns them to the operator as intermediate hypotheses that can be inspected, corrected, or rejected. Table 4.2 summarises the technological choices that enable this workflow, while the concrete implementation stack and deployment configuration are detailed later in Section 5.1.

Table 4.2: Architectural and technological choices supporting the proposed pipeline.

Pipeline stage	Technological option	Methodological justification
Operator interaction	Magic Leap 2 optical see-through XR headset	Allows the operator to remain in the physical workspace while receiving spatial feedback over the real scene. This supports in-situ inspection and keeps validation within the same task context.
Intent acquisition	Speech-to-text model for free-form spoken commands	Speech is a natural modality for an XR picking scenario, allowing the operator to describe the target while observing the scene. The recognised intent can then be displayed for confirmation or correction.

Pipeline stage	Technological option	Methodological justification
Scene observation	Egocentric RGB capture from the headset	Captures the scene from the operator's viewpoint, linking the spoken command to the visual context in which it was issued and supporting object-level language grounding.
Spatial grounding	Headset pose, camera metadata, and depth-assisted projection	Combines image-space perception with spatial information to convert segmentation results into world-space object evidence, supporting centroid estimation, overlay, and artefact placement in XR .
Target identification	SAM 3 language-conditioned segmentation	Allows the referred object to be grounded from language without restricting the task to predefined classes. The result remains inspectable through an outline or mask in XR .
Object reconstruction	SAM 3D object-centric reconstruction	Provides a 3D hypothesis of the selected object that can be inspected, aligned, refined, or rejected, supporting the transition from 2D identification to a simulation-ready spatial representation.
Edge perception	Edge-side execution of segmentation and reconstruction	Offloads computationally demanding perception stages from the headset, preserving interaction responsiveness while giving the pipeline access to stronger computational resources.
Runtime coordination	Session-centric data association	Keeps speech, image frames, metadata, perception outputs, artefacts, refinement actions, and validation decisions associated with the same command, preventing delayed outputs from being used in the wrong context.
Communication model	ROS -based streaming and HTTP -based artefact access	Uses streaming for time-sensitive events and request-response access for heavier artefacts such as masks, meshes, previews, and metadata, balancing responsiveness with reliable retrieval.
Operator supervision	Inspectable XR validation before downstream use	Treats segmentation, centroid estimation, and reconstruction as hypotheses to be inspected before downstream use, reducing the risk of propagating incorrect perception outputs into manipulation.
Robotic execution context	Isaac Sim and Isaac Lab digital-twin environment	Provides a simulation-first branch where the validated object state can be tested in a controlled and repeatable robotic environment before physical deployment is considered.

Pipeline stage	Technological option	Methodological justification
Policy execution	VLA-oriented simulated manipulation pipeline	Connects language-conditioned intent, visual observations, and robot actions in simulation, supporting evaluation of the validated object state in downstream pick-and-place behaviour.

These choices define the methodological structure of the system. The Magic Leap 2 provides an in-situ **XR** interface for command input and validation, while Speech-to-Text (**STT**) connects natural spoken instructions to the target-identification stage. SAM 3 and SAM 3D provide the perception and reconstruction components needed to generate inspectable object hypotheses. Edge processing keeps these computational stages outside the headset, while the session-centric communication model keeps asynchronous data products linked to the same interaction.

The architecture therefore supports the main research focus of the dissertation: integrating operator intent, inspectable perception, **XR** validation, and simulation-based manipulation into a single workflow. The concrete implementation of these choices is described in Chapter 5.

4.6 Chapter Summary

This chapter defined the methodology and system design of the **XR**-assisted intelligent picking system. The proposed approach is based on a closed loop in which speech-based intent is grounded by edge-side perception, made inspectable through **XR**, validated by the operator, and then used by the simulation and **VLA** manipulation branch.

The system design is organised around a set of principles: human-supervised grounding, session-based traceability, edge-assisted perception, spatially explicit validation, simulation-first robotic validation, and explicit interfaces with measurable outputs. These principles support a pipeline in which uncertain perception results are not hidden from the user, but transformed into spatial feedback that can be inspected and corrected.

The chapter also described the main system modules and the validation dimensions that will later be used to assess the system. The following chapter describes the concrete implementation of this design.

Chapter 5

Implementation

This chapter describes how the system design introduced in Chapter 4 was implemented as a session-centric XR–edge–simulation workflow. The focus is on the concrete software components, communication interfaces, data products, and runtime decisions that connect operator intent to object grounding, reconstruction, validation, and simulated manipulation.

A central implementation decision is the use of session identifiers. Each operator command creates a unique session identifier, which is attached to the speech transcript, RGB frames, frame metadata, segmentation output, depth-derived object evidence, reconstruction artefacts, refinement data, and validation decisions. Since speech transcription, frame transfer, segmentation, reconstruction, artefact loading, and user validation are asynchronous, this identifier preserves the relation between all data products generated by the same interaction and prevents delayed outputs from previous commands from being treated as current results.

5.1 Implementation Overview and Deployment Configuration

The implementation translates the system design into a working XR–edge–simulation pipeline. This section introduces the concrete runtime structure of the prototype before the following sections describe each stage of the workflow in detail. The system is organised around a session-centric interaction loop: the operator issues a spoken command through the headset, the edge server grounds the target object in the visual scene, the reconstructed and spatially grounded hypothesis is returned to XR, and the accepted object state is then made available to the simulation/VLA branch.

The prototype is implemented across three runtime environments. The first is the XR client running on the Magic Leap 2, which is responsible for operator interaction, speech capture, RGB capture, local depth use, spatial feedback, artefact loading, and validation controls. The second is the edge perception server, which receives session-linked speech and image data, performs language-conditioned segmentation, prepares object-centric reconstruction jobs, stores artefacts, and exposes session state. The third is the simulation/VLA branch, which represents the robotic workcell in a digital twin and is used to evaluate downstream pick-and-place behaviour.

Table 5.1 summarises the main technologies used to implement these environments. The table acts as an implementation map: it identifies the main hardware, software frameworks, perception models, simulation tools, and learning components that form the prototype backbone, while the paragraphs that follow provide the deployment details needed to reproduce the implemented configuration.

Table 5.1: Main technologies used in the implemented system.

Category	Technology	Role in the implementation
XR hardware	Magic Leap 2	Optical see-through headset used for operator interaction, camera capture, depth sensing, spatial tracking, and world-locked feedback.
Simulation teleoperation hardware	Meta Quest 3	Device used in the implemented setup as the operator interface for teleoperating the simulated KUKA/Robotiq workcell during seed-demonstration collection.
XR client development	Unity, Magic Leap Unity Software Development Kit (SDK), OpenXR, AR Foundation, XR Interaction Toolkit	Runtime environment for the headset application, including rendering, input handling, spatial anchors, controller/hand interaction, and Android deployment.
Speech input	Magic Leap Voice Intents and Vosk	Discrete voice intents are used for control and validation commands, while Vosk provides local offline transcription of free-form picking instructions.
Middleware	ROS 2 Humble	Exchanges session events, image frames, metadata, segmentation outlines, guided scan frames, and validated object poses between Unity and the edge-side services.
Edge Application Programming Interface (API) layer	FastAPI and Gradio	Exposes health checks, session state, reconstruction artefacts, preview assets, cancellation, active-client state, and refinement interfaces.
2D perception	SAM 3	Performs text-prompted object segmentation from the target phrase derived from the speech transcript.
3D reconstruction	SAM 3D Objects, GLB , PLY	Generates object-centric 3D artefacts from a selected masked image and provides mesh or point-based representations for XR validation.
Spatial grounding	Depth camera, camera intrinsics, headset pose, projection geometry	Converts image-space segmentation into world-space object evidence and supports centroid estimation, alignment, and validation.

Category	Technology	Role in the implementation
Digital twin	NVIDIA Isaac Sim and USD	Simulates the workcell, including the KUKA iiwa, Robotiq gripper, task cameras, reconstructed object, physics, and rendering. USD is used as the scene and asset representation.
Learning environment	Isaac Lab	Defines the KUKA pick-to-box task, observations, action space, recorders, teleoperation, imitation-learning workflows, and policy wrappers.
Data preparation	Demonstration recording, annotation, Isaac Lab Mimic, grasp generation, cuRobo/cuMotion	Supports demonstration collection and dataset preparation. Mimic-style generation and automatic grasp-based generation are treated as augmentation or experimental acceleration paths unless validated separately.
VLA training and evaluation	OpenPI/LeRobot-compatible pipeline and $\pi_{0.5}$ -based policy	Prepares multimodal demonstrations and evaluates language-conditioned visuomotor policies for the simulated manipulation task.
Containerisation	Docker with NVIDIA GPU runtime	Provides reproducible execution environments for simulation, perception, data generation, and training workloads.

The simulation entries in Table 5.1 separate Isaac Sim from Isaac Lab because they play different implementation roles. Isaac Sim provides the simulator itself: physics, rendering, cameras, USD scene representation, articulated robots, and the digital-twin workcell. Isaac Lab provides the robotics-learning layer built on top of the simulator, including the KUKA task definition, observations, action semantics, recorders, teleoperation utilities, imitation-learning tools, and policy interfaces. The XR-edge workflow and the simulation/VLA branch are therefore connected at the level of object and task representation, while robotic execution remains simulation-based.

The same implementation can now be described in terms of deployment. The prototype runs as a local XR-edge system supported by a separate simulation branch. The following points record the hardware, software versions, communication services, capture parameters, and processing-time measurements used by this configuration.

- **XR device and client.** The operator-facing application runs on a Magic Leap 2 optical see-through headset as a Unity 2022.3.45f1 Android x86_64 application. The client uses Magic Leap Unity SDK 2.6.0, Unity OpenXR 1.13.0, AR Foundation 5.1.0, XR Interaction Toolkit 2.1.0, Unity Input System 1.8.2, glTFast 6.13.1, and Unity Robotics ROS-TCP-Connector. In the implemented workflow, the headset hosts the operator-facing part of the session loop, including capture, spatial rendering, local depth processing, artefact loading, and validation.
- **Speech, image, and depth capture.** Free-form speech is captured through a hold-to-record interaction and transcribed locally with Vosk using the `vosk-model-en-us-0.22` model.

Audio is recorded at 16 kHz for up to 30 s. **RGB** frames are captured from the Magic Leap computer-vision camera, encoded as JPEG, and associated with frame metadata. The configured **RGB** resolution is 1280×720 . Periodic frame buffering uses a 0.2 s interval, corresponding to 5 Hz, while the local streaming interval is 0.05 s, corresponding to a 20 Hz cap. The camera is configured for 30 FPS. Depth is consumed locally by the **XR** client rather than published as a raw depth topic; the implementation uses the Magic Leap LongRange depth stream at 5 FPS, with depth values expressed in metres.

- **Edge server hardware.** The edge-side workload runs on an Ubuntu 24.04.4 LTS workstation equipped with an Intel Core Ultra 9 285K CPU, 62 GiB of RAM, and an NVIDIA GeForce RTX 5090 GPU with 32 GB of VRAM. The workstation uses NVIDIA driver 580.159.03 and CUDA 13.0, and runs the **ROS** bridge, SAM 3 segmentation service, SAM 3D capture and worker processes, preview/API services, and GPU inference workload.
- **Network and services.** The Magic Leap 2 and the edge workstation operate on the same local network, with the headset client configured with the edge host address used by the deployment. ROS-TCP is used for streamed session messages, with the **ROS** endpoint on port 10000. The SAM 3 segmentation service runs on port 8008, and the SAM 3D preview/API service runs on port 7861. This configuration keeps perception and reconstruction on the local edge machine rather than relying on external cloud services.
- **Runtime communication.** **ROS** topics are used for session events, **RGB** frames, frame metadata, segmentation outlines, guided scan frames, and validated object poses. **HTTP** endpoints are used for health checks, session polling, cancellation, refinement control, active-client state, and artefact retrieval. This division separates streamed observations from persistent access to heavier outputs such as masks, previews, meshes, splats, and metadata.
- **Segmentation and reconstruction runtime.** SAM 3 is served through FastAPI/Uvicorn using CUDA and the checkpoint `SAM3/checkpoints/facebook_sam3/sam3.pt`. SAM 3D Objects runs as a GPU worker with a preview/API service. A base reconstruction job is created from one selected **RGB** image, one rasterised segmentation mask, and one masked RGBA crop. Generated artefacts may include `input.jpg`, `mask.png`, `rgba.png`, `metadata.json`, `mesh.glb`, `splat.ply`, rendered previews, and refinement/orientation metadata.
- **Processing-time measurements.** The server-side SAM 3 bridge reports segmentation handling times of approximately 130.6 ms median and 732.2 ms p95. SAM 3D reconstruction metadata reports durations of approximately 7.39 s median, 10.83 s mean, and 33.92 s p95. These values characterise the edge-side segmentation and reconstruction stages, not full end-to-end operator latency.
- **Simulation/VLA branch.** The robotic execution branch is evaluated in a digital twin using NVIDIA Isaac Sim, Isaac Lab, USD assets, a KUKA iiwa manipulator, a Robotiq gripper,

task cameras, destination containers, and the OpenPI/LeRobot-compatible policy pipeline. The physical KUKA workcell is not used for real robot execution in this dissertation.

The deployment details identify where each component runs, but the workflow also depends on how data moves between components during a command session. Since the implementation is asynchronous, speech transcription, buffered images, metadata, segmentation results, reconstruction artefacts, refinement requests, and validation decisions can be produced at different moments. The session identifier links these data products to the same operator command and prevents delayed outputs from being interpreted as current results.

Figure 5.1 summarises this session-level information flow. Solid arrows represent **ROS**-based message exchange, dashed arrows represent **HTTP** requests and responses, and dotted arrows represent operator-facing interaction or validation actions. The diagram also makes explicit that perception outputs are not sent directly to the robotic branch. Instead, segmentation, centroid estimation, and reconstruction are returned to the operator as inspectable hypotheses before a final object state is accepted.

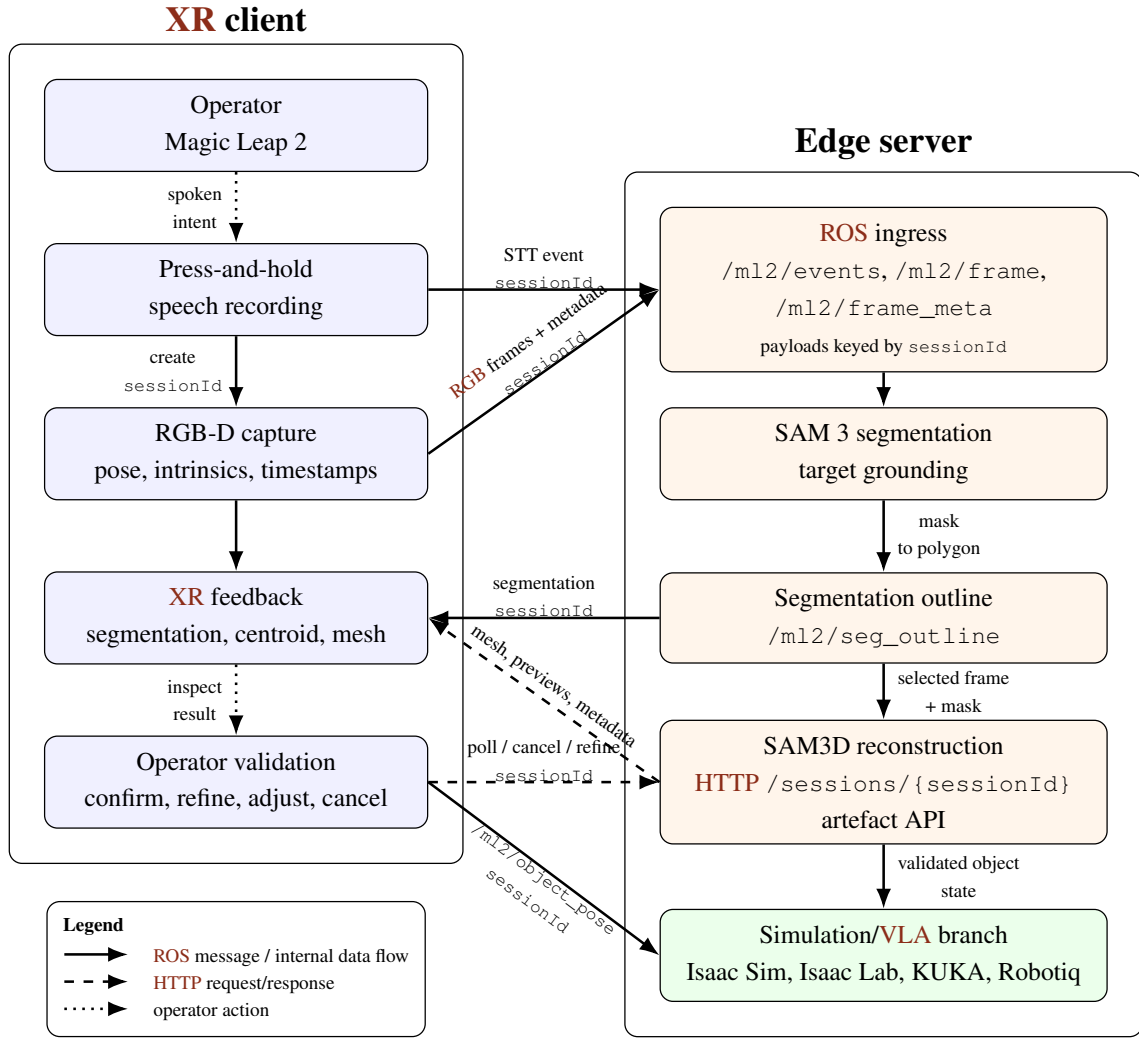


Figure 5.1: Session-level information flow in the implemented pipeline. Solid arrows denote **ROS**-based messages or internal data flow, dashed arrows denote **HTTP** request/response interactions, and dotted arrows denote operator-facing actions. The `sessionId` is generated on the **XR** client and propagated in the ROS payloads and HTTP session endpoints, where it correlates speech, RGB frames, metadata, segmentation, reconstruction artefacts, and validation state.

Figure 5.1 closes the overview by showing how the deployed components operate as a single session-based pipeline. The key point is that the headset, edge server, and simulation branch are not independent modules connected only at the end of the process. They exchange session-linked data throughout the workflow, from the initial spoken command to the validated object state. This session-level structure is what allows the following sections to describe each stage separately while preserving the same end-to-end interaction logic. The next section begins with the first stage of this loop: acquiring the operator's intent in **XR**.

5.2 Operator Intent Acquisition

The implemented pipeline begins with the operator wearing the Magic Leap 2 headset and initiating a command session. This stage corresponds to the first step of the flow introduced in Figure 5.1. Its purpose is to capture not only the spoken command, but also the visual and spatial context in which the command is given. This is important because object-level instructions are often ambiguous when detached from the scene. A command such as “pick the bottle” only becomes meaningful when it is associated with the object arrangement observed by the headset at that moment.

The system uses free-form speech as the operator intent input. The operator describes the target object or task in natural language, for example with commands such as “pick the red box” or “grab the bottle”. This speech input is transcribed locally with offline speech-to-text and later grounded against the visual observations captured during the same session. In this way, the spoken command is not interpreted as an isolated instruction, but as a language prompt associated with the RGB scene observed by the headset.

Free-form operator intent is acquired through a hold-to-record interaction. The operator presses and holds the configured input while speaking the command. When recording begins, the XR client creates a new session identifier of the form `ml2-<guid>`. While the operator speaks, the client buffers RGB observations and associated metadata under the same session identifier. When the recording is released, the audio clip is trimmed to the actual recorded samples and processed locally with Vosk. The implementation uses offline speech recognition so that free-form transcription does not depend on cloud services.

The audio pipeline is configured for short single-command utterances. Audio is recorded through Unity’s microphone interface at 16 kHz, with a maximum recording duration of 30 s. The active speech recognition configuration uses the language code `en-US` and the Vosk model `vosk-model-en-us-0.22`. Before recognition, the recorded audio is converted to mono, resampled when required, quantized to 16-bit PCM little-endian, and passed to the Vosk recognizer. The command duration is computed from the number of recorded microphone samples and the configured sampling rate, while the transcription step records its own processing time as `sttSeconds`. This makes each command session traceable from the press/release interaction to the structured speech event sent to the edge server.

Figure 5.2 shows the operator-side intent acquisition stage. Figure 5.2a highlights the controller input used to trigger the hold-to-record interaction, while Figure 5.2b shows the XR summary window after the command is sent, including the recognised intent and the captured visual frames associated with the active session. Together, these views document how the system links user input to the session-based processing pipeline.

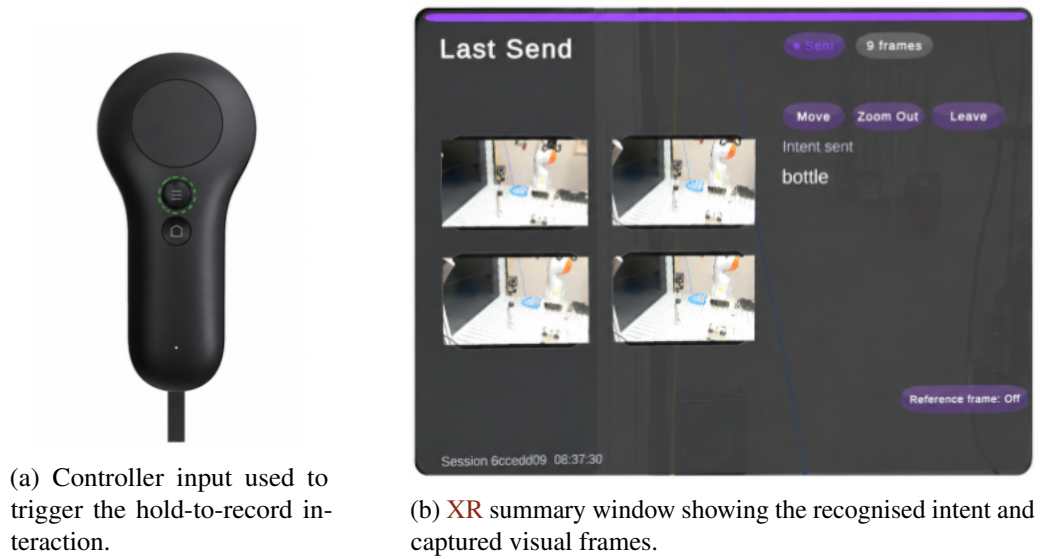


Figure 5.2: Operator intent acquisition in XR. Panel (a) shows the controller input used for recording. Panel (b) shows the summary window generated after the command is sent.

The transcript is published to the edge server as a structured **STT** session event, not as a continuous audio stream. This distinction is important because the normal perception pipeline consumes the recognised text and associated session metadata, while raw or chunked audio streaming is only an auxiliary/debug path. The **STT** event is published on `/ml2/events` as a JSON payload and includes the event type, the active `sessionId`, the recognised transcript, client/device identifiers when available, timestamps, language information, and transcription latency metadata. In the configured workflow, the client publishes visual frames before sending the **STT** event. This increases the likelihood that the edge server already has scene context available when the transcript arrives and the target object must be identified.

The transcript is treated as an operator intent hypothesis rather than as an immediately executable command. This is necessary because free-form speech can be correct, partial, ambiguous, or incorrect. For example, a transcript such as “pick the red box” provides both an action and a target object, while a shorter transcript such as “red box” can still provide a usable object phrase for visual grounding. An ambiguous transcript such as “pick the bottle” may refer to more than one object in the scene, and an incorrect transcript may cause the perception stage to focus on the wrong target. These cases are handled by coupling the transcript with the captured visual frames and by requiring the returned segmentation, centroid, and reconstruction to be inspected in XR before the object state is accepted.

Interaction is not limited to speech. The headset application also supports controller rays, trigger input, trackpad interaction, hand aim, hand pinch, hover feedback, reticles, and haptics. These inputs become important after the perception result is returned, when the operator must inspect, refine, correct, or validate the object state. Controller rays and hand-aim rays define the pointing direction used to select world-space UI elements. Trigger input is used for hold-to-record, button activation, click/drag operations, and capture/release actions. Hand pinch provides an

alternative selection gesture when hand interaction is enabled. Trackpad input supports fine manual adjustments, including centroid or placement correction. Hover feedback, reticles, and haptic pulses indicate which UI element or spatial target is active, confirm clicks or snaps, and reduce accidental selections.

In practice, these inputs are used in the XR summary and validation interface shown after a command is sent. The operator can inspect the recognised transcript, captured frame thumbnails, segmentation outline, reconstructed mesh, centroid marker, and pipeline state. The same interaction layer is used to confirm the result, cancel the session, request refinement, retake the capture, select a target anchor, drag or adjust the centroid, and interact with manual mesh placement controls. Recording triggers are blocked while the operator is interacting with validation or placement controls, preventing a new recording from interrupting the current session.

At the end of this stage, the system has created a session containing a transcript, timing metadata, and a set of visual observations. This session becomes the unit of processing for the rest of the pipeline.

5.3 Target Identification and Grounding

After the command session is created, the system must determine which object in the scene corresponds to the operator's intention. This stage corresponds to the communication, segmentation, centroid estimation, and reconstruction steps introduced in Figure 5.1. It connects the spoken command, the **RGB** observations, the depth stream, and the edge-side perception pipeline into a spatial object hypothesis that can later be inspected by the operator.

The Magic Leap 2 client captures **RGB** frames from the computer-vision camera. Frames are encoded as compressed images and published through **ROS**, together with metadata containing the session identifier, timestamp, frame index, image dimensions, and the camera intrinsic parameters when available. The client also captures depth frames locally and stores them with their own timestamps, intrinsics, pose, depth values, confidence information, and validity flags. The depth stream is used later by the client to convert segmentation results into spatial evidence.

On the edge server, the transcript received from the headset is converted into a target phrase suitable for segmentation. The goal of this step is not to perform full task planning, but to extract the object-level prompt that should be grounded in the image. The edge server then applies this prompt to recent **RGB** frames associated with the same session.

Segmentation is performed using SAM 3. The bridge node sends the selected image and target prompt to the segmentation service, receives the corresponding mask, converts it into a polygonal outline, and publishes the result back to the system on `/ml2/seg_outline`. The segmentation service uses the language prompt and the **RGB** image to obtain the object mask, applies a score threshold of 0.15 and a minimum mask area of 0.002 of the image, extracts the largest contour, and simplifies it into a polygon. The message returned to the headset includes the session identifier, image dimensions, polygon points, frame timestamp, and tracking information.

Because the segmentation output is tagged with the session identifier, the headset can ignore results that do not belong to the active interaction.

The runtime interfaces used in this stage are formalised in Table 5.2. The table separates the transport mechanism, direction, mandatory fields or dependencies, and implementation role. This makes explicit that `sessionId` is not only a local headset variable: it is generated by the **XR** client and propagated through the **ROS** payloads and **HTTP** session endpoints so that speech, images, segmentation, reconstruction, refinement, and validation decisions remain associated with the same interaction.

Table 5.2: Runtime interface contracts used during target identification, grounding, reconstruction, and refinement.

Interface	Direction	Required fields, dependency, and role
<code>/ml2/events</code> <code>std_msgs/String</code> JSON	XR client → edge server	Requires <code>kind=stt</code> , <code>sessionId</code> , and <code>transcript</code> . This message sends the recognised speech command as a session event and provides the text used by the edge server to derive the segmentation target.
<code>/ml2/frame</code> <code>sensor_msgs</code> <code>/CompressedImage</code>	XR client → edge server	Requires a compressed RGB image payload. The image depends on the corresponding <code>/ml2/frame_meta</code> message to recover the session identifier, timestamp, image size, and camera information.
<code>/ml2/frame_meta</code> <code>std_msgs/String</code> JSON	XR client → edge server	Requires <code>kind=frame</code> , <code>sessionId</code> , <code>timestamp</code> , <code>width</code> , and <code>height</code> . This metadata associates each image with the active session and provides the information needed for segmentation, reconstruction input selection, and spatial grounding.
<code>/ml2/seg_outline</code> <code>std_msgs/String</code> JSON	edge server → XR client	Requires <code>kind=seg_outline</code> , <code>sessionId</code> , <code>image size</code> , <code>polygon points</code> , and <code>frame timestamp</code> . This message returns the segmented target outline to the headset and links the polygon to the frame used for grounding.
<code>/ml2/guided_scan</code> <code>_frame</code> <code>/ml2/guided_scan</code> <code>_frame_meta</code>	XR client → edge server	Requires guided-scan image data and metadata associated with the same <code>sessionId</code> . These messages provide additional operator-guided observations for refinement of pose, orientation, or scale.
<code>/ml2/object_pose</code> <code>std_msgs/String</code> JSON	XR client → edge server / simulation branch	Requires <code>sessionId</code> and an estimated or operator-accepted object pose or transform. This message represents the accepted object hypothesis produced by the XR validation workflow; it is not by itself a robot-safe execution command.
GET <code>/api/sam3d/sessions/</code> <code>sessionId</code>	XR client ← edge server	Requires <code>sessionId</code> in the path. This endpoint returns the session state, job state, reconstruction status, result metadata, and artifact references used by the XR client to monitor progress.

Interface	Direction	Required fields, dependency, and role
POST /api/sam3d/sessions/ {sessionId}/cancel	XR client → edge server	Requires <code>sessionId</code> in the path. This endpoint cancels or invalidates a pending or active session so that delayed results are not used after the operator cancels or starts a newer command.
Refinement API group /refinement/start /refinement/{scanId}/ frames /refinement/{scanId}/ finish /refinement/{scanId}/ run	XR client ↔ edge server	Requires <code>sessionId</code> , and also <code>scanId</code> after a refinement scan has been created. These endpoints start a guided refinement scan, append frames, finish the scan, and run the refinement procedure associated with the same session.
GET /files/outputs/ path	XR client ← edge server	Requires an artifact path returned by the session/result API. This endpoint is used to download generated outputs such as meshes, point-based representations, masks, previews, rendered views, and metadata for XR placement.

The `/ml2/seg_outline` payload is the central return message for 2D grounding. It carries the polygon in full-resolution image coordinates and enough frame information to associate the result with the RGB/depth data stored on the headset. A typical payload has the following structure:

```
{
  "kind": "seg_outline",
  "sessionId": "ml2-8f1a2b3c4d5e67890abcdeffedcba987",
  "prompt": "bottle",
  "score": null,
  "w": 1280,
  "h": 720,
  "poly": [[490.0, 210.0], [620.0, 215.0],
           [640.0, 360.0], [500.0, 355.0]],
  "frameTsUnixMs": 1772112000123,
  "frameIndex": 12,
  "tracking": false,
  "trackingFast": false,
  "clear": false
}
```

The `w` and `h` fields define the image resolution used by the polygon, and `poly` contains the ordered outline points in pixel coordinates. The fields `frameTsUnixMs`, `frameIndex`, and, when available, `frameId` are used to associate the outline with the RGB metadata stored by

the client. The `clear` flag is used to remove stale outlines from the display when the tracked segmentation should no longer be shown.

The segmentation polygon is first used as visual feedback. It allows the operator to see whether the system selected the intended object. However, the polygon is also used geometrically. The **XR** client combines the image-space polygon with the locally cached depth frames to estimate a world-space object centroid. For this association, the client resolves the RGB metadata first by frame identifier when available, and otherwise by timestamp or frame index. It then searches for the nearest depth frame by timestamp. Depth/RGB matches with a temporal difference above 120 ms are rejected, which avoids fusing a polygon from one viewpoint with depth captured from a different viewpoint.

The centroid estimation is projection-based. Candidate depth pixels are sampled from the depth image with a stride of 3 pixels, which reduces computation while preserving spatial coverage across the segmented object. Each valid depth sample is converted into a **3D** point using the depth camera intrinsics and pose. These points are transformed into the Unity world frame and then reprojected into the **RGB** image using the **RGB** camera pose and intrinsics. If the reprojected pixel falls inside the segmentation polygon, the corresponding world point is treated as object evidence.

The accepted depth range is limited to values between 0.15 m and 7.5 m. Depth samples outside this interval, invalid depth flags, out-of-bounds reprojections, and unreliable confidence values are filtered before the centroid is computed. The implementation requires at least 40 accepted object points before producing a centroid. This prevents sparse depth observations, holes in the depth map, or small accidental polygon overlaps from generating an unstable object position. The remaining points are filtered using confidence statistics when available and spatial outlier removal based on percentiles.

The final centroid is computed from the filtered **3D** point set using robust statistics. The horizontal coordinates are derived from the median of the accepted world points, while the vertical coordinate uses the configured percentile value of 0.38. This avoids making the centroid depend on a simple mean of all visible depth points, which can be affected by boundary pixels, missing depth, or unevenly visible object surfaces. The centroid is then rendered in the headset and used as a spatial anchor for placing the reconstructed object.

Figure 5.3 presents the transition from edge-side target grounding to operator-facing spatial feedback. Figure 5.3(a) shows the edge-side selected **RGB** frame, the binary mask, and the object crop used as reconstruction input. Figure 5.3(b) shows the corresponding **XR** overlay, where the segmented object is highlighted and the centroid is reported back to the user. This figure is important because the segmentation is not only a visual overlay; it also defines which depth samples are considered object evidence for centroid estimation.



(a) Selected RGB frame, binary segmentation mask, and cropped object representation generated on the edge side.



(b) Corresponding XR overlay, where the target object is highlighted and the centroid calculation is reported to the operator.

Figure 5.3: Segmentation and centroid feedback during target grounding.

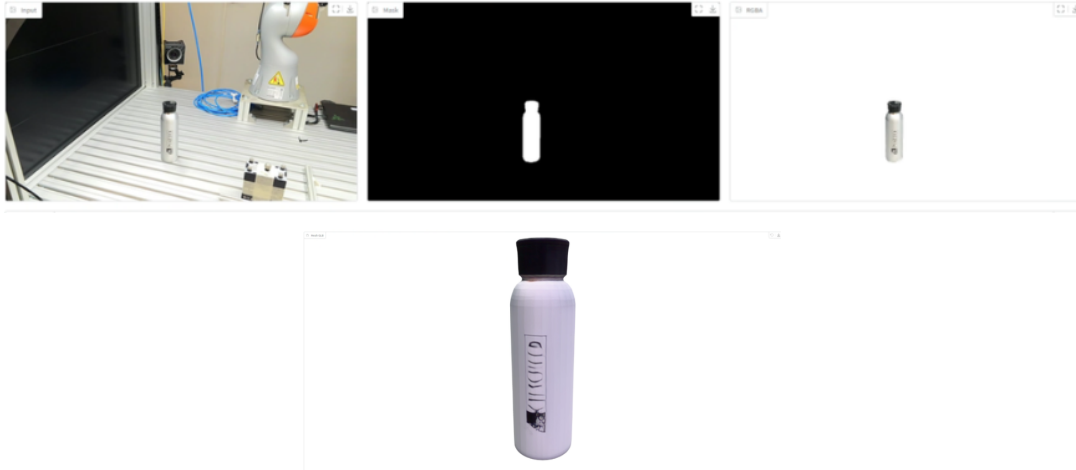
The same segmentation result is also used to create the reconstruction input. The edge server stores the selected **RGB** image, a rasterised binary mask, and an Red, Green, Blue, and Alpha (**RGBA**) object crop or masked image representation. These files are written into a pending SAM 3D job together with metadata describing the session, prompt, frame, and available camera information.

The reconstruction worker processes the pending job and generates object-centric artifacts. These may include a **GLB** mesh, a **PLY** or splat representation, preview renders, a turntable animation, masks, input images, and metadata describing coordinate conventions, transforms, orientation candidates, scale estimates, warnings, and runtime information. In the current implementation, the main reconstruction input is single-view: one selected image and one mask are used to create the object representation. Guided scan frames can support later refinement of pose, orientation, or scale, but they are not treated as a fully validated multi-view reconstruction pipeline.

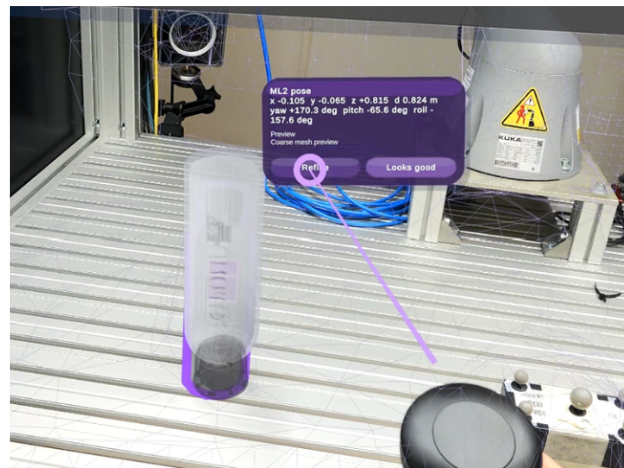
The **XR** client monitors session state through **HTTP**. The server exposes endpoints for health checking, session status, active-client status, cancellation, refinement, and artifact download. When a reconstruction result becomes available, the client downloads the artifact and its metadata. Mesh artifacts are preferred because they can be loaded into Unity and inspected as spatial objects. The client applies the transform metadata and local coordinate corrections before rendering the object as part of the validation interface.

Figure 5.4 shows the reconstruction stage and the return of the generated artifact to the headset. Figure 5.4(a) presents the server-side SAM 3D outputs, including the selected image, mask, object crop, and mesh preview. Figure 5.4(b) shows the reconstructed mesh loaded into the **XR** client

as a spatial hypothesis aligned with the real object. This makes clear that reconstruction is not only a server-side output, but a hypothesis that is re-imported into the operator's spatial frame for inspection.



(a) SAM 3D reconstruction inputs and preview, including the selected image, mask, object crop, and generated mesh.



(b) Reconstructed mesh loaded into the XR client as a spatial hypothesis aligned with the real object.

Figure 5.4: Reconstruction artifacts and mesh loading in the XR client.

At the end of this stage, the system has produced three linked object representations: the semantic target phrase, the image-space segmentation, and the spatial object hypothesis represented by a centroid and reconstructed artifact. These outputs are not treated as final commands. They are passed to the operator validation interface, where they can be inspected, accepted, refined, corrected, or cancelled.

5.4 Operator Validation Interface

The validation stage is the point at which the automatic perception result becomes a human-approved object state. This stage is necessary because errors may occur at several levels. The transcript may not fully capture the operator’s intention, the segmentation may select the wrong object, the centroid may be displaced by depth noise or projection errors, and the reconstructed mesh may have incorrect scale, orientation, or local origin. For this reason, the implementation does not send perception output directly to the manipulation branch.

The headset presents the result through a world-locked XR interface. The operator can inspect the transcript, the captured visual context, the current processing state, the segmentation outline, the centroid marker, the reconstructed artifact, and the available decisions. These elements are not separate outputs from the user’s point of view; they form a single validation workflow. The segmentation shows whether the system selected the intended object, the centroid shows whether the selected object was grounded in a plausible spatial position, and the mesh shows whether the reconstructed object can be meaningfully aligned with the real object.

The interface is implemented as a combination of a world-space summary canvas and object-level spatial overlays. The summary canvas displays the recognised transcript, captured frame thumbnails, the session or pipeline state, and the contextual validation controls. The segmentation outline is rendered over the selected object, the centroid is shown as a world-locked marker, and the reconstructed mesh is loaded as an interactive spatial artifact in the same XR scene. Additional contextual controls appear as pose or target balloons near the object when the operator enters placement, refinement, manual correction, or target-anchor selection modes.

Interaction is performed through the XR input layer rather than through a desktop-style interface. The operator can use controller rays and trigger input to point at and activate world-space buttons, and optional hand-aim and hand-pinch input provide an alternative pointing and selection mechanism. Hover feedback, reticles, and haptic pulses indicate which element is active and confirm interaction events such as clicks, drags, and snaps to interface targets. These mechanisms are used for confirming a result, cancelling a session, requesting refinement, selecting orientation or scale options, entering manual correction, dragging the centroid marker, and specifying a target anchor.

This visual inspection is central to the implementation. A segmentation result may appear correct in image space but still produce an unstable centroid if the depth evidence is noisy or incomplete. Similarly, a reconstructed mesh may be recognisable as the intended object while still requiring pose, scale, or orientation correction before it becomes useful for simulation. Displaying these outputs in the headset allows the operator to judge the result in the same physical context in which the command was given.

Table 5.3 summarizes the session states exposed by the validation workflow. These states are not independent applications; they represent the progression of one `sessionId` from command capture to an accepted, cancelled, or superseded object hypothesis.

Table 5.3: Session states in the XR validation workflow.

State	Implementation role
Recording	The operator is holding the recording input. A new <code>sessionId</code> is active and RGB frames are being buffered with the spoken command.
Processing	The command and visual frames have been sent, and the client is waiting for segmentation, reconstruction, or session status updates from the edge server.
Preview available	A segmentation outline, centroid estimate, coarse mesh, or placement preview is available for inspection in XR . The operator can accept the preview, retake the capture, request refinement, or enter manual review.
Refinement requested	The operator has requested refinement or guided scanning. The client captures additional guided-scan frames and sends them to the edge server with the active <code>sessionId</code> .
Manual adjustment	The operator is adjusting the reconstructed object or centroid through the manual correction controls. Rotation, scale, centroid, reset, and apply controls are available in this state.
Confirmed	The operator accepts the displayed target interpretation and placement. The client can publish the accepted object state, including the final transform and the user-accepted flag.
Cancelled	The operator rejects or stops the current session. The local UI clears or ignores the result, and the server cancellation endpoint can mark pending work as cancelled.
Superseded	A newer command from the same client replaces the previous active session. Delayed results from the older session are ignored and pending work can be marked as superseded.

If the result is close but not reliable enough, the operator can request refinement or perform manual correction. Figure 5.5 illustrates this stage through a sequence of **XR** views. Figure 5.5(a) shows the initial coarse preview of the reconstructed object. Figure 5.5(b) shows the interface for selecting among orientation candidates. Figure 5.5(c) shows the manual controls used to adjust scale, centroid, and orientation before the final object state is accepted. This is important because refinement is part of the human-supervised design: it mitigates perception and reconstruction errors without treating the automatic output as ground truth.

Manual correction is implemented through explicit adjustment modes. In rotation mode, the operator selects the rotation axis and applies discrete nudges around pitch, yaw, or roll. The implemented axes correspond to the local right, up, and forward directions, and each rotation nudge applies a 6 degree increment. In scale mode, the mesh is adjusted using uniform multiplicative

scaling, with step factors of 1.06 or $1/1.06$; the implementation therefore changes the global object scale rather than scaling independently along each axis. In centroid mode, the centroid marker can be dragged in space. The controller/trackpad interaction supports an XZ-plane mode and a vertical Y-axis mode, allowing horizontal repositioning and height correction to be handled separately. Fine centroid adjustment is smoothed and bounded per frame, which avoids abrupt jumps during manual dragging.

The manual correction state also provides reset and apply actions. Reset restores the original placement pose from the current preview, while Apply commits the current accumulated adjustment and exits manual placement mode. The correction workflow is therefore based on an editable current placement state: rotation, uniform scale, and centroid changes are accumulated into the active object transform until the operator applies or resets them. The interface does not treat manual correction as a robot command; it updates the **XR** object hypothesis that will later be confirmed or rejected by the operator. Support-plane and placement-validation feedback can be used to judge whether the adjusted pose is plausible, while final robot-side feasibility and collision checks remain outside this operator-facing validation step.

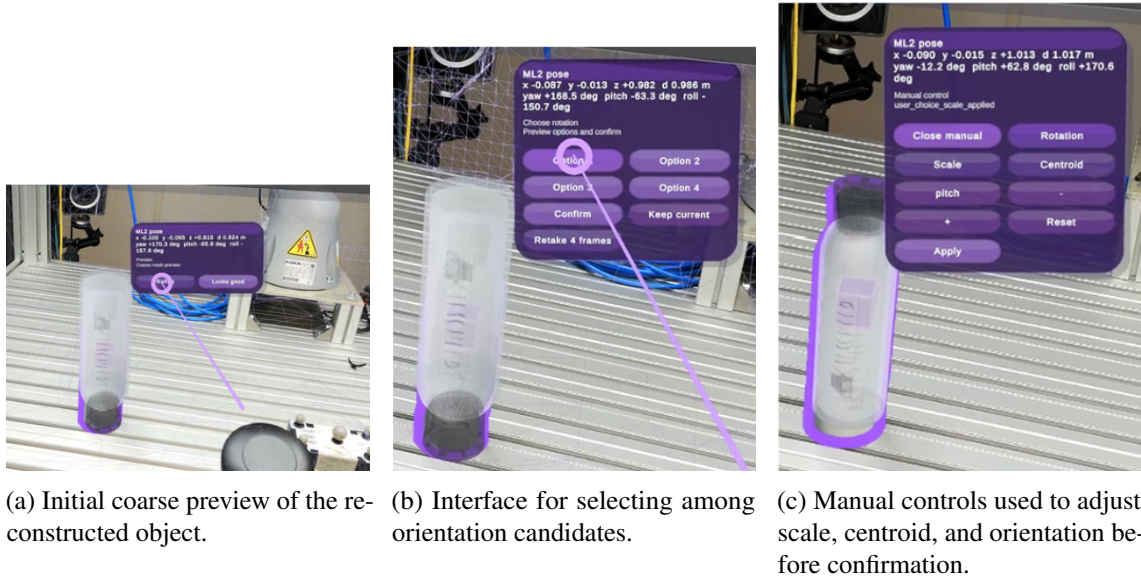


Figure 5.5: Refinement and manual correction of the reconstructed object hypothesis.

Refinement is handled as a guided extension of the current session. When refinement is requested, the client starts a guided scan associated with the same `sessionId`, captures additional **RGB** frames from new viewpoints, and sends the corresponding image data and metadata to the edge server. The guided-scan metadata includes the scan identifier, frame index, timestamps, image dimensions, camera intrinsics when available, camera-frame convention, and world-camera or mesh-placement transforms used to relate the scan frames to the current preview. The edge server can then use these frames to refine orientation, pose, or scale before the result is returned to the **XR** client for further inspection.

If the operator considers the result correct, the session can be confirmed. Confirmation produces a validated object state that includes the target identity, the accepted reconstruction artifact, the centroid or pose estimate, transform metadata, and the operator's decision. In implementation terms, this state corresponds to the object hypothesis accepted in the XR interface and can be published through `/ml2/object_pose` with a user-accepted flag and the relevant object/world transform information. This accepted state is the interface between perception and the downstream manipulation branch.

If the result is wrong, ambiguous, or no longer relevant, the operator can cancel the session. Cancellation has both local and server-side effects. Locally, the headset clears or ignores the displayed result and prevents it from being used. On the server, the cancellation endpoint can mark the session as cancelled and move pending work to the cancelled state when possible. If a job is already being processed, the session metadata still records that the result is no longer active for the client, so delayed outputs from that session are not treated as valid interface results. Supersession handles the related case in which a newer command from the same client replaces an older active session; in that case, pending work can be marked as superseded and the older result is ignored by the current validation workflow.

The interface also supports target-pose or target-anchor placement. This extends the validation stage from object inspection toward task specification: the operator may not only approve the object, but also indicate a desired final placement or target pose. In this mode, contextual controls such as confirm, cancel, and keep-moving actions allow the operator to refine the target anchor before accepting it. Figure 5.6 shows this transition from object validation to task specification, where the accepted object state and target pose become the inputs passed to the simulation/action-generation branch.

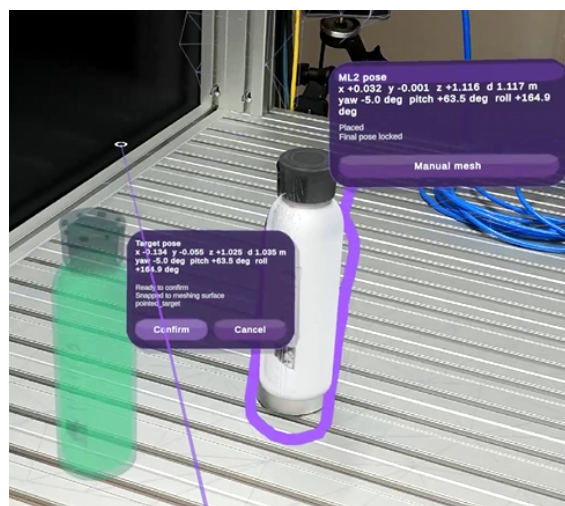


Figure 5.6: Final target pose specification through the XR validation interface. The accepted reconstructed object state is shown together with a target anchor, allowing the operator to specify the intended placement before the result is passed to the simulation/action-generation branch.

This stage turns the implementation into a supervised pipeline rather than an automatic perception chain. The operator confirms the intended target interpretation, while robot-side feasibility, collision, and safety checks remain required before execution.

5.5 Simulation Asset and VLA Training Pipeline

After the object hypothesis is validated in **XR**, the next stage prepares the simulation-side representation used for demonstration collection and **VLA** policy training. It covers the conversion of the reconstructed object into a simulation asset, its insertion into the KUKA digital twin, the collection and expansion of demonstrations, and the fine-tuning of the policy later executed in simulation.

The validated **XR** state and the Isaac Sim scene do not share the same runtime frame by default. The Magic Leap 2 client represents poses in the Unity/**XR** world frame, while the KUKA environment in Isaac Sim has its own world origin, robot base frame, table layout, asset hierarchy, and object-local conventions. Both systems use metric units, but simulation placement still requires a consistent mapping between the accepted object interpretation and the digital-twin workcell. In this implementation, the validated object state provides the target identity and spatial interpretation, while the simulation environment instantiates the corresponding object asset in the KUKA pick-and-place scene using the environment placement configuration.

The object used in the simulation pipeline is the SAM 3D reconstruction of the water bottle. The reconstruction output is first available as a visual **GLB** mesh, which is converted into a USD asset using the Isaac Lab mesh conversion tool. The prepared asset is registered in the IsaacLab Arena object library as `sam3d_real_water_bottle_variant` and can then be selected as the active object in the KUKA pick-to-box environment. This step turns the reconstructed object from an **XR** visualization artifact into an object that can be loaded by the simulation runtime.

The converted asset also contains the collision and rigid-body information required for interaction with the gripper, table, and destination container. In this implementation, the SAM 3D **GLB** mesh is converted to USD with a convex-hull collision approximation generated from the input mesh, rather than a manually drawn bounding box, a triangular mesh collider, or a convex decomposition with multiple parts. The conversion command fixes the object mass, density is not estimated separately, and the object registration uses the Isaac Lab default rigid-body material, with static friction 0.5, dynamic friction 0.5, and restitution 0.0. These values make the asset usable for repeated simulation experiments, but remain implementation parameters rather than measured physical properties of the real bottle.

The KUKA digital twin is implemented in Isaac Sim through Isaac Lab and the local IsaacLab Arena environment. The scene contains the KUKA iiwa arm, the Robotiq gripper, the workcell, the SAM 3D bottle asset, the destination container, optional distractors, and the cameras used for policy learning. The task is a pick-to-box task in which the robot must pick up the water bottle and place it in the black box; the environment defines the active object, destination, robot embodiment, observation terms, action interface, and success conditions used during recording and evaluation.

The policy action space used throughout this pipeline is `kuka_differential_ik_absolute`. The arm command is expressed as an end-effector target pose for the differential inverse-kinematics controller, while the Robotiq gripper is controlled by a scalar open/close command. Values above the configured gripper threshold map to the closed gripper target, and lower values map to the open target. Preserving this action space across recording, dataset generation, export, training, and serving is necessary because changing the action semantics between stages would make the learned policy incompatible with the execution environment.

In the implemented setup, demonstrations were collected in the simulated KUKA workcell using a Meta Quest 3 teleoperation interface. The headset was used as the operator input device for recording successful pick-and-place episodes. However, the recording pipeline is not intrinsically tied to this specific device and could be adapted to another compatible teleoperation interface, provided that the same action representation and dataset structure are preserved. Additional demonstrations can be recorded whenever broader coverage of object poses, workspace regions, task variants, or object categories is required. The resulting episodes are stored as dense HDF5 data with actions, processed actions, robot state, object state, camera observations, and initial state. A companion JSONL manifest stores readable metadata such as the task description, active object, destination container, visible objects, initial object pose, action key, action space, camera names, and source information. For policy learning, the external camera captures the global relation between the robot, object, and destination container, while the wrist camera provides local feedback near the gripper. During export, these observations are mapped into the LeRobot dataset format together with joint position, gripper position, action vectors, timestamps, indices, and task identifiers.¹ The task text used for the exported dataset is: “Pick up the water bottle and place it in the black box.”

The collected demonstrations are annotated for Isaac Lab Mimic using the grasp event, represented as `grasp_1`, as the key subtask boundary. This identifies the transition into the grasping phase and allows Mimic to reuse the structure of successful demonstrations while preserving physically meaningful timing with the Robotiq gripper and simulated object. Starting from these annotations, Mimic generates additional visual episodes with object-position variation; the generated episodes are validated and filtered before export, increasing spatial coverage while keeping the data grounded in the same task, action space, object, and camera configuration.

The final dataset used for policy training is exported in LeRobot format under the local KUKA SAM 3D bottle dataset. The training configuration used for the final policy combines 35 teleoperated seed demonstrations with 350 Mimic-generated episodes, including external camera images, wrist camera images, joint position, gripper position, action vectors, timestamps, indices, and task identifiers.

The **VLA** model is trained using the OpenPI framework with a $\pi_{0.5}$ base model fine-tuned via LoRA for the KUKA pick-to-box task.² In this setup, OpenPI refers to the training and serving

¹<https://github.com/huggingface/lerobot>

²OpenPI repository: <https://github.com/Physical-Intelligence/openpi>. $\pi_{0.5}$ model description: <https://www.pi.website/blog/pi05>.

stack, while $\pi_{0.5}$ refers to the base model configuration used for fine-tuning. The configuration uses a two-camera visual input setup, an action horizon of 16, an action dimension of 32 in the OpenPI model interface, and a LoRA configuration based on the $\pi_{0.5}$ checkpoint. The fine-tuning process computes normalization statistics for the exported dataset, trains the policy, and stores a checkpoint that can later be loaded by the remote policy server.

Table 5.4 summarizes the complete preparation flow from validated object to trained policy checkpoint.

Table 5.4: Simulation asset and VLA training pipeline.

Stage	Implementation role
Validated object state	The XR interface provides the accepted object interpretation, including target identity, centroid or pose information, reconstruction artifact, and operator decision.
SAM 3D asset conversion	The reconstructed water-bottle mesh is converted from GLB to USD so that it can be loaded by Isaac Sim as a simulation asset.
Collision and physics setup	The converted USD uses a convex-hull collision approximation and the Isaac Lab default rigid-body material parameters.
KUKA task setup	Isaac Lab/Arena instantiates the KUKA iiwa, Robotiq gripper, workcell, SAM 3D bottle asset, black-box destination, cameras, and task logic.
Teleoperated demonstrations	The operator records successful simulated pick-and-place episodes. The collector stores HDF5 data and a JSONL manifest.
Mimic annotation	The demonstrations are annotated with the grasp subtask boundary, allowing Isaac Lab Mimic to reuse the structure of successful episodes.
Mimic generation	Annotated demonstrations are expanded into additional visual episodes with object-position variation, camera observations, validation, and filtering.
LeRobot export	The validated episodes are exported to LeRobot format with camera images, robot state, gripper state, actions, timestamps, and task text.
OpenPI fine-tuning	The exported dataset is used to fine-tune an OpenPI $\pi_{0.5}$ -based VLA policy through LoRA. The resulting checkpoint is used for simulated execution.

This stage therefore prepares action generation through demonstration learning. The system does not compute a robot trajectory directly from the **XR** command or from the reconstructed

mesh. Instead, it builds a simulation dataset in which language, vision, robot state, gripper state, and actions are aligned. The action is generated later by the trained policy, which observes the simulated scene and returns action chunks for the KUKA environment. That execution loop is described in the next section.

5.6 Simulated Robotic Execution

The final stage of the implementation corresponds to the execution of the pick-and-place task inside the simulated robotic workcell. At this point, the object has already been prepared as a simulation asset, the demonstration dataset has been exported, and the **VLA** policy checkpoint has been produced. This section therefore focuses on the runtime rollout loop: how observations are collected, how the policy is queried, how actions are applied to the KUKA environment, and how success or failure is determined.

The simulated workcell is implemented using Isaac Sim, Isaac Lab, and the project-specific IsaacLab Arena extension. At the beginning of an episode, the environment instantiates the KUKA iiwa manipulator, the Robotiq gripper, the workcell background, the active object, the destination container, optional distractor objects, cameras, contact sensors, and task state. The task follows a pick-to-box formulation: the active object must be grasped and transported into the selected destination container. The environment also supports object-layout presets and sampled object positions, which are used to evaluate the task under different initial placements rather than only in a single fixed scene.

The robot is represented as a USD-backed KUKA iiwa model with an integrated Robotiq gripper. In the **VLA** branch, the control interface is the same `kuka_differential_ik_absolute` action space used during dataset preparation. The first seven action dimensions encode an absolute end-effector target, composed of Cartesian position and quaternion orientation, while the final scalar controls the Robotiq gripper. The gripper command is interpreted as a binary joint-position target, where low values correspond to an open gripper and values above the configured threshold command closure. Using the same action representation during training and execution ensures that the policy output remains compatible with the Isaac Lab action terms.

The simulation configuration used for the KUKA/**VLA** branch is based on an Isaac Lab simulation step of $\text{dt} = 1/200$, a render interval of 2, an action decimation of 4, and a default episode horizon of 50 s. This corresponds to a policy/control frequency of approximately 50 Hz. The visual policy pipeline uses an external RGB camera and a wrist-mounted RGB camera. In the Mimic/OpenPI data path, these camera observations are exported at 320×180 pixels and 50 frames per second. The policy observation bundle also includes the KUKA arm joint position and the Robotiq gripper position.

During policy execution, the simulator process acts as a client of a remote OpenPI policy server. The OpenPI server loads the trained checkpoint and receives the current observation bundle together with the natural-language task instruction. The client sends the external RGB image, wrist RGB image, arm joint position, gripper position, and task text to the server. The server

returns a fixed-horizon sequence of actions, and the simulator client consumes this action chunk for a configured number of environment steps before requesting a new one. For the absolute differential-IK configuration, the first seven returned action dimensions are applied as the KUKA end-effector target, while the eighth dimension is applied as the Robotiq gripper target.

The execution loop assumes that the active object has already been prepared as a simulation-compatible asset. The repository contains support for SAM 3D-derived object variants and GLB-to-USD preparation, but the runtime loop does not repeat the reconstruction-to-asset conversion at every policy step. This separation keeps the rollout focused on policy execution: the environment receives a registered object asset with defined scale, collision geometry, rigid-body properties, support pose, and gripper-compatible geometry, and the policy is evaluated on the resulting simulated scene.

Task success is defined at the simulation level rather than by visual inspection alone. For the pick-to-box task, a rollout is counted as successful when the active object reaches the selected destination container and triggers the corresponding object-container contact condition. A grasped object is therefore only an intermediate state, not the final success criterion. A rollout is counted as failed when the object is not placed in the destination container before the episode timeout, when the object is dropped below the support surface, when the gripper misses or releases the object prematurely, or when the returned action sequence cannot produce a valid manipulation under the configured action space. This definition provides a direct criterion for the quantitative validation in Chapter 6.

The success rate used for simulated robotic execution is defined as

$$\text{Success Rate} = \frac{N_{\text{success}}}{N_{\text{rollouts}}}, \quad (5.1)$$

where N_{success} is the number of rollouts ending with valid object-container contact, and N_{rollouts} is the total number of attempted simulated rollouts. Additional failure labels can be logged to support diagnosis, including missed grasp, early gripper closure, object drop, timeout, missing camera observations, invalid object or destination configuration, failed grasp validation, unreachable scripted motion, and policy runtime errors.

Figure 5.7 illustrates the distinction between intermediate and final execution outcomes. In Fig. 5.7(a), the bottle is grasped by the Robotiq gripper, which is an intermediate manipulation success. In Fig. 5.7(b), the object is placed inside the target container, which corresponds to the final task success condition. In Fig. 5.7(c), the gripper closes without establishing a stable grasp, and the rollout is classified as a failure unless the policy later recovers and completes the placement.

This stage closes the implementation pipeline in the digital twin by connecting the prepared simulation asset, the KUKA pick-to-box environment, the visual observation interface, and the trained policy checkpoint in a single rollout loop. Each episode starts from a reset scene, collects camera and robot-state observations, queries the OpenPI policy server, applies the returned action chunks through Isaac Lab action terms, and terminates according to the task conditions defined above. The resulting rollout outcome is evaluated from simulator state and contact events, rather

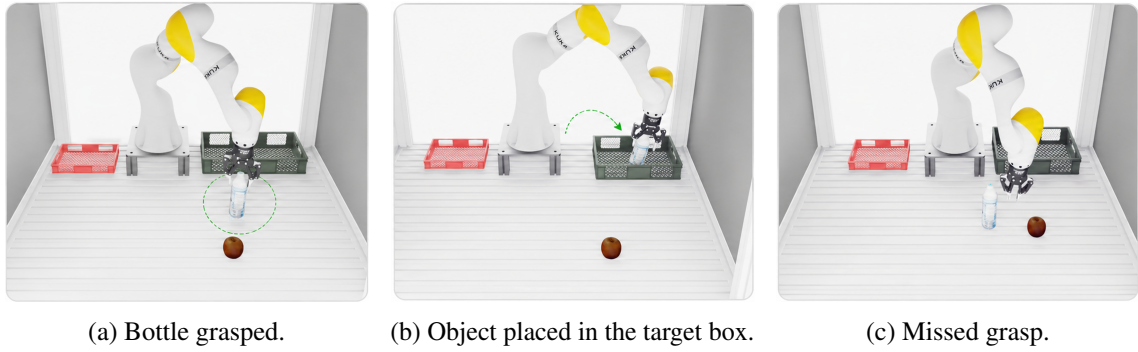


Figure 5.7: Examples of simulated KUKA pick-and-place execution outcomes. A grasped object is considered an intermediate manipulation state, while final task success requires the object to be placed inside the target container. A missed grasp or an object left outside the gripper is classified as a failed rollout unless the policy recovers before timeout.

than from visual inspection alone. A video demonstration of the complete prototype pipeline is also available, showing the XR intent capture, edge-side grounding, object reconstruction, operator validation, and simulated execution sequence.³

5.7 Summary

This chapter presented the implementation of the proposed system as a session-centric XR–edge workflow, complemented by a simulation/VLA branch for downstream preparation and simulated evaluation. The most mature and operational part of the implementation is the operator-facing loop that connects intent acquisition, visual capture, edge-side perception, object-centric reconstruction, spatial feedback, and human validation. A session identifier is used throughout this loop to associate the speech transcript, RGB frames, metadata, segmentation outputs, reconstruction artefacts, refinement actions, and validation decisions with the same operator command.

On the perception side, the Magic Leap 2 client captures the spoken instruction and the visual context of the scene, while the edge server uses the recognised target phrase to perform language-conditioned segmentation and to prepare SAM 3D reconstruction jobs. The returned segmentation is not treated as a final command. Instead, it is combined on the XR client with depth and camera metadata to estimate a world-space centroid, and the reconstructed mesh is loaded into the headset as an inspectable spatial hypothesis.

The validation interface is therefore a central part of the implemented workflow. It allows the operator to inspect the transcript, captured frames, segmentation outline, centroid estimate, reconstructed artefact, and session state before accepting, refining, correcting, retaking, or cancelling the result. In this context, acceptance should be understood as an operator-approved object interpretation or object hypothesis, rather than as a robot-safe execution state. Final manipulation feasibility, collision checking, and execution safety remain outside the XR validation step.

³Video demonstration: <https://www.youtube.com/watch?v=4RzoDm-bBD0>

The simulation/**VLA** branch extends this implementation by preparing a corresponding digital-twin representation for the KUKA/Robotiq pick-to-box task. The SAM 3D reconstruction can be converted into a USD asset, inserted into an Isaac Sim/Isaac Lab environment, and used for demonstration collection, Mimic-based dataset expansion, LeRobot export, OpenPI fine-tuning, and simulated rollout execution. This branch shows how the validated **XR** object interpretation can inform a downstream simulated manipulation workflow, but it should not be interpreted as a fully validated end-to-end deployment from headset command to physical robot execution.

Overall, the chapter demonstrates a human-supervised prototype in which natural-language intent, egocentric perception, edge-side reconstruction, and **XR** validation are integrated into a coherent operator-in-the-loop workflow. The simulation and **VLA** components provide the basis for evaluating downstream robotic manipulation in a digital twin, while the quantitative evidence for robustness, latency, reconstruction quality, and simulated task success is left to Chapter 6.

Chapter 6

Validation and Tests

This chapter presents the validation of the proposed XR-assisted perception and simulation pipeline. The objective is to evaluate whether the system can transform a spoken picking instruction into a session-linked object hypothesis, expose that hypothesis through XR feedback, allow the operator to validate or correct it, and use the accepted state in a simulated robotic execution branch.

The validation follows the research questions and validation strategy defined in Chapter 4.4. Since the workflow is composed of asynchronous and modular stages, a single end-to-end example would not be sufficient to characterise the system. A failure may originate from speech transcription, target extraction, segmentation, depth-based grounding, 3D reconstruction, XR alignment, operator validation, or simulated execution. For this reason, the validation is organised by stage, while the results are interpreted in relation to the complete workflow.

The evaluation includes session integration, speech-to-text intent capture, target segmentation, centroid estimation, 3D reconstruction, operator validation, simulated VLA execution, and failure-mode analysis. The repetition counts follow the validation strategy: session integration is evaluated over 100 complete interaction sessions, perception and reconstruction conditions are evaluated using ten trials per condition, operator validation is evaluated over ten validation trials, and the simulated VLA execution branch is evaluated over 50 rollouts.

6.1 Validation Overview

Table 6.1 summarises the validation groups, objectives, and number of trials used in this chapter. The purpose of this structure is to separate the source of each failure. For example, a wrong 3D artifact may result from an incorrect transcript, an ambiguous segmentation, insufficient depth information, or limitations of the reconstruction model. Evaluating each stage separately makes it possible to understand which part of the workflow failed and whether the operator could detect or correct the problem before simulated execution.

Table 6.1: Summary of the validation groups.

Validation group	Objective	Trials
Session integration	Verify that data products generated during the same interaction remain associated with the same session.	100 sessions
STT and intent capture	Verify whether the spoken command preserves the target object needed for visual grounding.	10 trials per condition
Target segmentation	Verify whether the intended object is selected in the RGB image.	10 trials per condition
Centroid estimation	Verify whether the segmented object can be converted into a usable world-space position.	10 trials per distance
3D reconstruction	Verify whether the generated artifact is recognisable, loadable, and useful for XR inspection.	10 trials per condition
Operator validation	Verify whether the operator can inspect, accept, cancel, refine, or correct the returned hypothesis.	10 trials
Simulated VLA execution	Evaluate the pick-and-place task in the simulated KUKA workcell.	50 rollouts
Failure analysis	Identify the stage, condition, and impact of unsuccessful or partial trials.	All failed and partial trials

6.2 Session Integration Validation

The session integration validation evaluates whether the system preserves a coherent session state across the asynchronous stages of the workflow. This aspect is essential because the system does not process all data products at the same time. Speech transcription, frame transmission, segmentation, reconstruction, artifact loading, validation actions, and possible refinement or cancellation events may occur at different moments. The session identifier must therefore remain consistent so that outputs from one operator command are not mixed with outputs from another command.

This validation was not performed as a separate isolated experiment with 100 different spoken intents. Instead, session consistency was checked throughout the remaining validation tests, whenever a new interaction session was created. Each time the operator started a new command, the generated session identifier was followed across the available stages of the pipeline, including the speech event, **RGB** frames, frame metadata, segmentation output, reconstruction state, returned artifacts, and **XR** validation workflow. When refinement, cancellation, or replacement of an active session occurred, the corresponding session state was also checked.

A session was considered successful when the data products generated during the same interaction remained associated with the same session identifier and the active hypothesis shown to the operator corresponded to the most recent valid session. A session would be considered failed if outputs from different commands were mixed, if an artifact was loaded for the wrong session, or if a stale or superseded result was presented as the active hypothesis.

Across the validation campaign, session integration remained consistent. No case was observed where a transcript, frame batch, segmentation result, reconstruction artifact, or validation action was associated with the wrong interaction session. Likewise, no stale artifact was presented as the active hypothesis after a newer session had been started. This indicates that the session-based protocol was sufficient to maintain coherence between the XR client, the edge-side perception and reconstruction stages, and the validation interface during the tested workflow.

This outcome affects the interpretation of the following validation stages, since the segmentation, centroid estimation, reconstruction, and operator validation results must correspond to the same operator command and visual context. The absence of observed session-mixing errors therefore supports the use of the session identifier as the main correlation mechanism for the implemented prototype.

6.3 Speech-to-Text Validation

The STT validation evaluates whether the headset-based speech capture preserves the target object in the operator command. In the implemented system, free-form speech is recorded on the XR headset and transcribed locally. The resulting transcript is then used by the edge server to derive the target object and segmentation prompt. For this reason, the transcript does not need to be perfect word by word; it must preserve the object reference with enough semantic clarity for the following perception stage.

The validation considered two types of spoken input. The first type consisted of short object-focused commands, where the operator only pronounced the object name or a minimal phrase containing it. The second type consisted of longer task instructions, where the target object was embedded in a complete sentence. Each type was evaluated over ten trials. Background noise was not treated as a separate controlled condition in this validation, but it was considered during the interpretation of the results because the quality of the local transcript depends on the acoustic environment, the clarity of the operator's pronunciation, and the complexity of the spoken instruction.

A trial was considered successful when the transcript preserved the target object or a semantically equivalent target phrase. A trial was considered failed when the transcript omitted, replaced, or corrupted the target object in a way that prevented the correct segmentation prompt from being derived.

The results, summarized in Table 6.2, show that short object-focused commands were more reliable than longer instructions. In the short-command condition, the target object was preserved in 8 out of 10 trials. This indicates that the local STT path is suitable when the operator uses concise commands focused on the object to be selected. The failed cases were associated mainly with pronunciation and acoustic capture conditions, which affected the way the object name was transcribed.

Longer instructions were less reliable, with 6 successful trials out of 10. This behaviour is expected because the selected STT model is a compact model intended to run directly on the Magic

Leap 2, rather than a larger model executed on the edge server. Running the model locally reduces dependence on external speech services and keeps the speech stage close to the headset, but it also limits recognition robustness. As the sentence becomes longer and more complex, the probability of one or more words being incorrectly recognised increases.

Table 6.2: STT validation results.

Input type	Trials	Successful	Failed
Short object-focused command	10	8	2
Longer instruction	10	6	4

The most relevant errors were cases where the target word became ambiguous or was replaced by a phonetically similar word. This can occur with short and similar-sounding terms, for example words such as “mouse”, “most”, or “mouth”, depending on pronunciation and noise. In object-selection tasks, these errors are important given that the perception pipeline depends on the target phrase to derive the segmentation prompt. If the wrong object word is produced, the following segmentation stage may search for the wrong target.

Overall, the **STT** validation shows that the implemented local speech interface is functional for concise object-focused commands, which are the most suitable form of input for the current prototype. However, the results also show that longer and more complex instructions are more likely to introduce transcription errors. In practical use, this motivates the operator to use short target-oriented commands when interacting with the system. For noisier environments or more complex language input, a possible improvement would be to send the recorded audio to the edge server and run a larger speech-recognition model there, while preserving the same session-based workflow.

6.4 Target Segmentation Validation

The segmentation validation evaluates whether the perception pipeline selects the object intended by the spoken instruction. This stage is critical since centroid estimation and **3D** reconstruction depend on the returned mask or outline. If the wrong object is segmented, the following stages will also refer to the wrong target.

The tested conditions were selected to represent the main scene-level difficulties expected in an **XR**-assisted picking workflow. Each condition was evaluated over 10 trials. The tested cases include a clearly visible target, a target near clutter, a partially occluded object, and a scene with multiple possible targets.

Table 6.3: Segmentation validation conditions.

Condition	Description	Trials
Clearly visible target	One target object is clearly visible and separated from other elements.	10
Nearby clutter	The target object is close to other objects or background elements.	10
Partial occlusion	Part of the target object is hidden by another object or surface.	10
Multiple possible targets	More than one visible object could plausibly match the spoken instruction.	10

A segmentation trial was considered successful when the returned mask or outline corresponded to the intended object and was sufficiently aligned with the visible target to be recognised by the operator. A partially successful segmentation selected the intended object but produced an incomplete, slightly imprecise, or mildly contaminated mask. A failed segmentation occurred when the wrong object was selected, no usable mask was produced, or the mask was too inaccurate to support target confirmation.

Table 6.4: Segmentation validation results.

Condition	Trials	Success	Partial	Failure
Clearly visible target	10	10	0	0
Nearby clutter	10	8	1	1
Partial occlusion	10	4	5	1
Multiple possible targets	10	7	0	3

The results in Table 6.4 show a clear degradation pattern as the visual grounding problem becomes less constrained. In the clearly visible target condition, the system selected the correct object in all ten trials. This confirms that, when the spoken intent is unambiguous and the object is well separated from the background, the returned outline is reliable enough to support the following centroid and reconstruction stages.

The nearby clutter condition produced 8 successful trials, 1 partial result, and 1 failure. This shows that clutter is not necessarily a problem when the surrounding objects are visually distinct from the requested target and do not obstruct the operator’s view of the object. In these cases, the model can still isolate the correct target. However, clutter can reduce mask quality when neighbouring regions are visually close to the object boundary or when the background makes the object contour less clear.

Partial occlusion had a stronger effect on segmentation quality. In this condition, only 4 out of 10 trials were fully successful, while 5 produced partial segmentations and 1 failed. The partial cases occurred when the system still selected the correct object but only segmented the visible region or produced an outline with small inaccuracies. These outputs can still be useful

for operator inspection and may allow the workflow to continue, especially when the occlusion is small. However, when the object is heavily occluded, the segmentation becomes less reliable and the error propagates to the following stages. An incomplete or shifted mask can affect the centroid estimate, and the 3D reconstruction may generate an incomplete or distorted mesh because the input evidence no longer represents the full object.

The multiple possible targets condition resulted in 7 successful trials and 3 failures. These failures were mainly caused by scene ambiguity rather than by the segmentation model alone. When more than one object can plausibly match the spoken instruction, the system may select an object that is visually plausible but not the one intended by the operator. This reinforces the importance of the XR overlay: the operator can inspect the selected mask before accepting the target and can repeat or refine the command when the selected object is not the intended one.

Overall, the segmentation validation shows that the system is reliable in simple and visually clear scenes, remains usable with moderate clutter, and becomes less stable under occlusion or ambiguity. The main limitation is not only whether a mask is produced, but whether that mask remains sufficiently aligned with the intended object to support centroid estimation, reconstruction, and operator validation.

6.5 Centroid Validation

After segmentation, the system estimates a world-space centroid by combining the segmentation polygon with depth information. This centroid is used as a spatial anchor for the object and supports the placement of the reconstructed artifact in the XR scene. The centroid validation evaluates whether this estimate remains close enough to the physical object to support object-level inspection and target anchoring.

The centroid was evaluated at four distances from the headset: 0.30 m, 0.50 m, 1.00 m, and 5.00 m. Each distance was tested over ten trials. The main metric is the Euclidean distance between the estimated centroid and the measured or manually marked object position, reported in metres. A centroid estimate was considered successful when the error was close enough to support XR inspection and approximate object anchoring. For close-range validation, a reference tolerance of approximately 0.05 m was used. Estimates above this tolerance were classified as partial when they still indicated the correct object region but required manual correction, and as failed when they were missing, unstable, or clearly associated with the wrong spatial location.

Table 6.5: Centroid validation results by distance from the headset.

Distance	Trials	Mean error (m)	Success	Partial/failed
0.30 m	10	0.012	10	0
0.50 m	10	0.017	10	0
1.00 m	10	0.051	7	3
5.00 m	10	0.087	6	4

The results, shown in Table 6.5, indicate that the centroid estimation was most reliable at short distances. At 0.30 m and 0.50 m, all ten trials were successful, with mean errors of 0.012 m and 0.017 m, respectively. These errors are small enough for the intended XR workflow, since the estimated centroid remains visually close to the physical object and can be used as a practical anchor for displaying and inspecting the reconstructed artifact. In these close-range cases, only minor manual adjustment was required, and in some trials the placement was already sufficient without correction.

At 1.00 m, the centroid estimation became less consistent. Seven out of ten trials were successful, while three were classified as partial or failed. The mean error increased to 0.051 m, which is close to the reference tolerance used for this validation. This behaviour is expected because the object occupies a smaller region in the image at this distance. As a result, small segmentation inaccuracies or depth variations have a stronger effect on the estimated centroid.

At 5.00 m, the limitations of the current centroid estimation became clearer. Six out of ten trials were successful, while four were partial or failed, and the mean error increased to 0.087 m. The observed behaviour depended strongly on the apparent size of the object in the image. When the segmented object occupied only a small image region, the centroid could become unstable or fail to be computed. In those cases, the reconstructed mesh could still be generated, but its placement in the XR world was no longer anchored to the expected physical location. For larger objects, the centroid could still be estimated at this distance, but with a larger spatial error.

Overall, the centroid validation shows that the implemented depth-based grounding is suitable for close-range object interaction, especially around 0.30 m and 0.50 m from the headset. At approximately 1.00 m, the estimate remains usable but benefits more from operator inspection and possible manual correction. At longer distances, particularly around 5.00 m, the method becomes less reliable, mainly because the target occupies a smaller image region and the segmentation-depth combination provides less stable spatial evidence. This confirms that the current workflow is better suited to close-range XR-assisted object validation than to distant object selection.

6.6 3D Reconstruction Validation

The 3D reconstruction validation evaluates whether the generated artifact can support operator inspection when displayed in XR. A reconstruction should not be considered successful only because a mesh file is produced. The artifact must also be loadable, recognisable as the intended object, correctly associated with the active session, and sufficiently aligned in scale and orientation to allow visual comparison with the physical object.

Each reconstruction condition was evaluated over ten trials. The tested conditions include a clear object under good lighting, partial occlusion, capture-angle difficulty, scale or orientation alignment difficulty, and object-prior mismatch. A separate lighting condition was not used as an independent test group. However, lighting and image quality were considered during the interpretation of the results because they influence the visual evidence available to the reconstruction model.

Table 6.6: 3D reconstruction validation conditions.

Condition	Description	Trials
Clear object under good lighting	The object is clearly visible, well segmented, and captured under suitable lighting.	10
Partial occlusion	Part of the object is hidden or the mask does not cover the complete object.	10
Capture-angle difficulty	The available view does not provide enough visual evidence for the full object geometry.	10
Scale or orientation alignment difficulty	The artifact is recognisable but may require pose, scale, or orientation correction in XR.	10
Object-prior mismatch	The requested object differs from the type of objects for which the reconstruction model produces reliable priors.	10

A reconstruction trial was considered successful when the edge server generated an artifact, the XR client loaded it, and the reconstructed shape remained recognisable as the intended physical object. A partially successful reconstruction was recognisable but presented issues such as scale mismatch, orientation offset, incomplete geometry, or visible distortion that required operator correction. A failed reconstruction occurred when the artifact could not be loaded, the object was not recognisable, or the geometry was too incomplete or distorted to support meaningful inspection.

Table 6.7: 3D reconstruction validation results.

Condition	Trials	Success	Partial	Failure
Clear object under good lighting	10	10	0	0
Partial occlusion	10	7	2	1
Capture-angle difficulty	10	8	2	0
Scale or orientation alignment difficulty	10	6	4	0
Object-prior mismatch	10	9	0	1

The results in Table 6.7 show that reconstruction quality degraded mainly when the input evidence became incomplete or geometrically ambiguous. In the clear-object condition, all ten trials produced successful reconstructions. This suggests that, when the input image contains sufficient visual evidence and the object belongs to a common or visually familiar category, the reconstruction model can generate meshes that are recognisable and suitable for XR inspection.

Partial occlusion reduced reconstruction quality. In this condition, seven trials were successful, two were partial, and one failed. When only a small part of the object was hidden, the generated mesh often remained recognisable, although minor geometric imperfections could appear. However, stronger occlusion reduced the reliability of the output. In those cases, the segmentation mask no longer represented the full object, the centroid could become less accurate, and the reconstructed

mesh could become incomplete or distorted. This confirms that reconstruction quality is strongly affected by the amount of visible object evidence available in the input image.

Capture angle also influenced the result. Eight out of ten trials were successful, while two were partially successful. The partial cases occurred when the input view did not provide enough information about important object surfaces. Since the reconstruction model must infer the unseen parts of the object from a limited image, difficult viewpoints can lead to small deformations, missing details, or less reliable geometry. This effect is especially visible when the object has asymmetric geometry or when important parts are hidden by perspective rather than by another object.

Scale and orientation alignment remained one of the most relevant practical issues. In this condition, six trials were successful and four were partially successful. In several cases, the reconstructed object was visually recognisable but required correction before being considered well aligned in the XR scene. This occurs because the reconstructed mesh is generated in an object-centric coordinate system, while the XR application must display it in the headset world frame. Differences between mesh coordinates, camera coordinates, Unity world coordinates, scale conventions, and object orientation can cause the artifact to appear offset, rotated, too large, or too small. These issues are not always failures of the visual reconstruction itself, but they affect the usefulness of the artifact for spatial validation.

The object-prior mismatch condition produced nine successful trials and one failure. This condition tested different objects rather than only repeating the same object type. Most objects were reconstructed successfully, but one object produced a poor result, likely because its shape or visual appearance was less represented in the model's learned priors. The same limitation can also become more visible when the input image quality is weak, when the object occupies a small region of the image, or when the image does not expose the object's most distinctive structure. In these cases, the reconstruction model has less reliable evidence from the image and weaker prior knowledge to complete the missing geometry. As a result, the generated mesh may lose important structural details, simplify the shape excessively, or produce a visually plausible but inaccurate object.

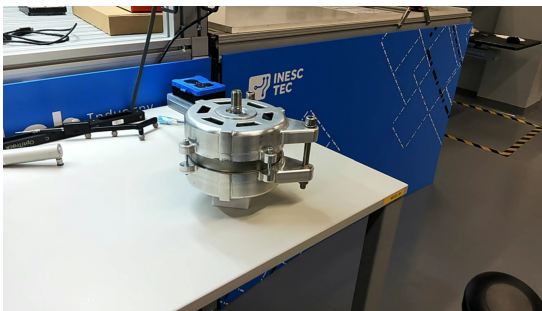
Figure 6.1 illustrates two qualitative behaviours observed during reconstruction. The first pair shows a case where the generated mesh is affected by object-prior mismatch and limited visual evidence. The second pair shows a successful reconstruction of a visually complex object. This demonstrates that complex geometry can still be reconstructed when the input image is sufficiently clear and the object appearance is compatible with the model's learned priors.



(a) Input object with weaker model prior support.



(b) Reconstruction affected by object-prior mismatch.



(c) Input view of a complex object.



(d) Successful reconstruction of the complex object.

Figure 6.1: Qualitative examples of 3D reconstruction behaviour. The first pair illustrates degradation caused by object-prior mismatch and limited visual evidence, while the second pair shows that complex objects can still be reconstructed successfully when the image is clear and the object is compatible with the model's learned priors.

The refinement process is used to reduce placement, scale, and orientation issues after the initial artifact is loaded. Its effectiveness depends on the visual evidence available to the system. If the guided images are captured from unsuitable angles, under poor lighting, or without enough visible object structure, the refinement may be less effective and the operator may need to repeat the capture process. This reinforces the role of the operator as a validation step rather than assuming that the first generated artifact is always final.

Overall, the 3D reconstruction validation shows that the system can produce recognisable and inspectable artifacts when the object is clearly visible, sufficiently segmented, and compatible with the reconstruction model's priors. The main limitations are occlusion, insufficient view coverage, scale and orientation alignment, image quality, and object categories that are less represented by the model. These limitations do not necessarily invalidate the artifact in the XR workflow, but they require operator inspection, refinement, and, when needed, repetition of the capture process.

6.7 Operator Validation

The operator validation test verifies whether the XR interface exposes the system interpretation in a way that can be inspected before simulated execution. This includes confirming a correct hypothesis, cancelling an incorrect or uncertain result, requesting refinement when more visual evidence is needed, and manually adjusting the returned target state when the result is close but not sufficiently aligned.

This validation was evaluated over ten trials. The objective was not to perform a user study, but to verify the functional availability of the validation workflow. When the SAM 3D artifact reaches the XR client correctly and can be loaded in the scene, the operator can inspect the mesh, compare it with the real target, and perform the available validation actions. Therefore, the main dependency of this validation stage is the successful arrival and loading of the reconstructed artifact from the previous stage.

Table 6.8: Operator validation test.

Validation workflow	Trials	Successful	Notes
Inspect, confirm, cancel, refine, or manually adjust the returned hypothesis	10	10	All validation actions were available and functional.

The results show that the operator validation workflow was available and functional in all ten trials. Once the SAM 3D artifact was correctly delivered to the XR client and loaded in the scene, the operator could inspect the returned mesh, compare it with the real object, confirm correct hypotheses, cancel incorrect or uncertain results, request refinement, and manually adjust the returned target state when needed. This confirms that the accepted target state results from an explicit validation action rather than from the initial speech command, segmentation result, or reconstruction output alone.

The only validation action with more variable behaviour was refinement. The refinement option was always available, but it did not always produce the best alignment on the first attempt. In most cases, the first refinement improved the placement or orientation of the artifact, but this depended on the quality of the additional images, the viewpoints from which they were captured, the lighting conditions, and the geometry of the object itself. More complex objects or poorly chosen capture angles can provide insufficient visual evidence for refinement, making it necessary to repeat the process with better views.

Overall, the operator validation test confirms that the XR interface supports the intended human-in-the-loop workflow. The operator can inspect the system’s interpretation before simulated execution and can decide whether to accept, reject, refine, or manually correct the result. This validation step is essential because the previous stages can produce partial or uncertain outputs, especially under occlusion, ambiguous segmentation, difficult reconstruction angles, or scale and orientation mismatch.

6.8 Simulation and VLA Validation

The simulation and VLA validation evaluates the downstream robotic execution branch in the digital twin. In this branch, the KUKA manipulator and the Robotiq gripper attempt a pick-and-place task using the simulated object and task representation. The objective is to evaluate whether the learned policy can complete the task after the perception and validation stages have produced an accepted target state, and to analyse how this behaviour changes with different fine-tuning stages, object assets, and workspace distributions.

The simulated execution branch was evaluated through headless rollouts in the KUKA simulation environment. A rollout was considered successful when the robot picked the intended object and placed it inside the target container. The tests therefore evaluate the complete manipulation sequence, including reaching the object, establishing a stable grasp, transporting the object, and completing the task without triggering a failed gripper close–open sequence.

Two types of objects were considered in these experiments. The first type corresponds to the water-bottle mesh reconstructed through the SAM 3D-based perception branch. This object was converted into a simulation-compatible asset and used to evaluate whether a reconstructed object could be integrated into the robotic execution pipeline. The second type corresponds to existing simulation dataset assets already available in the local object library, namely the dataset water bottle and the dataset apple. These assets are not SAM 3D reconstructions; they are pre-existing objects from the simulation dataset ecosystem used by the manipulation environment.

This distinction is important because the dataset objects are more likely to be close to the visual, geometric, and physical distribution expected by the base VLA model and by the simulation pipeline. Their meshes, textures, scales, and collision models are stable and already adapted to robotic simulation. In contrast, the SAM 3D object is a reconstructed mesh and is therefore more likely to differ from the distribution represented in the original policy data and in the simulation asset library. The better performance observed with the dataset bottle should therefore not be interpreted only as the result of fine-tuning, but also as evidence that dataset-like assets are easier for the policy and environment than reconstructed meshes.

Three fine-tuning stages were considered. The first policy was fine-tuned for the SAM 3D water bottle. The second policy was fine-tuned for the dataset water bottle. The third policy continued from the dataset-bottle policy and was further fine-tuned with data for both the dataset bottle and the dataset apple. Most evaluations were performed inside a restricted workspace region of approximately 0.50 m by 0.50 m, corresponding to the region represented in the demonstrations. A final evaluation expanded the object sampling area to test spatial generalisation.

Table 6.9: Fine-tuning stages used for simulated VLA validation.

Policy	Training object data	Episodes	Initialisation
SAM 3D bottle policy	SAM 3D reconstructed bottle	386	Base VLA policy
Dataset bottle policy	Dataset water bottle	385	Base VLA policy
Bottle and apple policy	Dataset water bottle and dataset apple	655	Dataset bottle policy

The first policy was fine-tuned for the SAM 3D water bottle. When evaluated on that same object, it achieved 18% success. In many failed rollouts, the robot moved close to the object, but the grasp was not established with sufficient precision or stability. The gripper often closed near the object and later opened without the success condition being reached. This indicates that the main limitation was not the absence of motion towards the target, but rather the instability of the grasping and de-grasping sequence around the reconstructed object.

As a diagnostic comparison, the same SAM 3D-fine-tuned policy was evaluated using the dataset bottle. This achieved approximately 47% success, even though the policy had not been fine-tuned for that specific asset. This result shows that the dataset bottle was easier for the policy and simulation setup than the SAM 3D bottle. The improvement suggests that object geometry, scale, collision behaviour, material appearance, and proximity to the dataset asset distribution influenced the final manipulation performance. At the same time, the result was still below the later dataset-bottle fine-tuning result, showing that object-specific fine-tuning also remained important.

The second stage fine-tuned a policy directly for the dataset bottle. In this case, the success rate increased to 74%, confirming that fine-tuning with the dataset asset produced a much more reliable pick-and-place behaviour. However, when this same policy was evaluated on the SAM 3D bottle, the success rate dropped to 12%. This supports the interpretation that the dataset bottle and the SAM 3D bottle were not interchangeable from the policy’s perspective, even though both represented water-bottle-like objects. The dataset bottle belonged to a more familiar and simulation-ready asset distribution, while the SAM 3D bottle introduced a reconstructed mesh with different geometric and contact properties.

The third stage continued from the dataset-bottle policy and added demonstrations and generated data for the dataset apple, while also keeping additional data for the dataset bottle. This means that the final policy was not trained again from the base model, but fine-tuned on top of the dataset-bottle policy. This mixed fine-tuning stage increased the dataset bottle result to 81%. This improvement is consistent with the larger amount of data available for the bottle across the fine-tuning process. The policy had more examples of the robot, gripper, object, and task behaviour, which made the learned manipulation sequence more stable inside the trained workspace region.

Table 6.10: Simulated VLA evaluation across fine-tuning stages.

Policy	Fine-tuning focus	Evaluation object	Workspace	Success
SAM 3D bottle policy	SAM 3D bottle	SAM 3D bottle	Restricted	18%
SAM 3D bottle policy	SAM 3D bottle	Dataset bottle	Restricted	$\approx 47\%$
Dataset bottle policy	Dataset bottle	Dataset bottle	Restricted	74%
Dataset bottle policy	Dataset bottle	SAM 3D bottle	Restricted	12%
Bottle and apple policy	Dataset bottle and apple	Dataset bottle	Restricted	81%
Bottle and apple policy	Dataset bottle and apple	Dataset apple	Restricted	18%
Bottle and apple policy	Dataset bottle and apple	SAM 3D bottle	Restricted	44%
Bottle and apple policy	Dataset bottle and apple	Dataset bottle	Expanded	20%

The same mixed policy achieved 44% success on the SAM 3D bottle. This was a clear improvement over the previous 12% result, even though the final policy was not specifically fine-tuned for the SAM 3D object. This suggests that increasing the amount and diversity of manipulation data improved the general behaviour of the policy and allowed partial transfer to a related object. Nevertheless, the performance remained much lower than for the dataset bottle, showing that the reconstructed SAM 3D asset remained more difficult to grasp reliably.

The dataset apple achieved 18% success with the mixed policy. The policy was able to interact with the apple and the object started to become graspable, but the complete task behaviour remained unreliable. This result is consistent with the smaller amount of apple-specific data compared with the bottle data. The policy therefore started to learn that the apple could be manipulated, but did not yet learn a robust grasping and task-completion strategy for that object.

A final evaluation tested the dataset bottle in an expanded workspace. This test was used to understand whether the policy generalized beyond the restricted region represented in the demonstrations. The success rate dropped to 20%, showing that the learned behaviour was strongly dependent on the spatial distribution of the training data.

Table 6.11: Effect of testing outside the trained workspace region.

Region during expanded evaluation	Successful	Failed	Success rate
Inside the trained region	7	1	87.5%
Outside the trained region	3	39	7.1%

The expanded-area evaluation shows that the policy was reliable when the object appeared inside the region covered by the demonstrations, but failed in most cases outside that region. The policy did succeed in a small number of cases outside the original area, which shows that the behaviour was not completely restricted to the demonstrated square. However, these successes were rare and some occurred close to the boundary of the trained region. This indicates overfitting to the spatial distribution of the dataset.

The dominant failure mode also changed with the evaluation condition. In the restricted workspace, the robot often approached the correct object, but the grasping and de-grasping sequence

was not always completed with sufficient precision or stability. In the expanded workspace, this limitation became more severe because the object appeared in positions that were poorly represented in the demonstrations. Therefore, the restricted-area tests mainly exposed grasping and de-grasping limitations, while the expanded-area test exposed both grasping instability and spatial generalisation limitations.

Overall, the simulated **VLA** validation shows that the robotic execution branch is functional but strongly data-dependent. The best result was obtained with the dataset bottle after mixed fine-tuning, reaching 81% success in the restricted workspace. The SAM 3D bottle remained more difficult, but improved to 44% after the final mixed fine-tuning stage, suggesting that broader training data can improve transfer to related objects. The apple result remained lower at 18%, mainly due to the smaller amount of apple-specific data and the remaining instability of the grasping behaviour. Finally, the expanded-area evaluation showed that broader and more varied demonstrations are necessary to reduce overfitting and improve generalisation across the full tabletop.

6.9 Failure Mode Analysis

The failure-mode analysis groups unsuccessful and partial trials according to the stage where they originate. This is necessary because the same final symptom can have different causes. For example, a wrong mesh may result from a speech error, a segmentation ambiguity, a depth-based grounding problem, or a reconstruction limitation. Similarly, a simulated manipulation failure may be caused by the policy, but it may also be affected by the object representation, contact behaviour, or scene configuration.

Table 6.12: Main failure modes considered during validation.

Failure mode	Typical cause	Mitigation
Incorrect transcript	The target object is omitted or replaced in the local STT output.	Repeat command or use clearer object-focused speech.
Wrong target selected	The command is ambiguous or several objects can match the instruction.	Inspect the XR overlay and use a more specific command.
Incomplete segmentation	The object is partially occluded, too small, or close to clutter.	Move closer, improve visibility, or capture another view.
Unstable centroid	Depth samples are insufficient or the segmentation region is unstable.	Use closer range or manual correction.
Artifact loading failure	The reconstruction job fails or the returned artifact cannot be loaded in XR .	Repeat the reconstruction or inspect edge-server state.
Wrong scale or orientation	The reconstructed mesh and the XR world use different conventions or require alignment.	Refinement and manual correction.
Poor reconstruction geometry	The object is not well represented by the reconstruction model priors or the input view is insufficient.	Use a clearer view, better lighting, or a more suitable object.
Refinement limitation	Guided images are captured from poor angles or under unsuitable lighting.	Repeat refinement with better viewpoints and lighting.
Simulated grasp failure	Collider, contact, reachability, or grasp stability problems.	Tune simulation asset and validate grasp/contact behaviour.
Policy failure	The policy does not generalise to the tested object pose or scene condition.	Increase dataset coverage and evaluate additional rollouts.

6.10 Chapter Summary

This chapter evaluated the proposed **XR**-assisted perception and simulation workflow according to the validation strategy defined in Chapter 4.4. The validation covered session integration, speech-to-text intent capture, target segmentation, centroid estimation, **3D** reconstruction, operator validation, simulated **VLA** execution, and failure-mode analysis.

Overall, the results show that the pipeline is functional across the complete perception-to-simulation workflow. The session integration remained consistent across the asynchronous stages, and the operator validation interface allowed reconstructed hypotheses to be inspected, accepted, refined, or rejected before simulated execution. The perception stages were most reliable under

clear and controlled conditions: short object-focused speech commands were more robust than longer instructions, segmentation performed best when the target was visible and unambiguous, centroid estimation was more accurate at close range, and reconstruction quality decreased with occlusion, poor viewpoints, scale or orientation misalignment, and object-prior mismatch.

The simulated **VLA** validation showed that robotic execution is strongly dependent on the object asset, the amount of fine-tuning data, and the workspace distribution. The dataset bottle achieved the best performance, reaching 74% after object-specific fine-tuning and 81% after the final mixed fine-tuning stage. The SAM 3D reconstructed bottle remained more difficult, although it improved to 44% with the final mixed policy. The dataset apple achieved 18%, indicating that additional object-specific data would be required for robust manipulation. When the dataset bottle was tested outside the trained workspace region, the success rate dropped to 20%, confirming limited spatial generalisation.

The main limitations identified were transcription errors in longer spoken commands, segmentation ambiguity under occlusion or multiple possible targets, reduced centroid accuracy at longer distances, reconstruction alignment errors, grasping/de-grasping failures during simulated manipulation, and overfitting to the demonstrated workspace region. These results confirm that the proposed workflow is viable as an **XR**-assisted perception and simulation pipeline, while also showing that broader data collection, more varied demonstrations, and improved handling of reconstructed objects are required for more robust generalisation.

Chapter 7

Conclusions and Future Work

This dissertation addressed the development of an **XR**-assisted intelligent picking workflow that connects operator intent acquisition, edge-side visual grounding, human-supervised validation, and simulation-based **VLA** robotic execution. The work was motivated by the need to make language-conditioned robot interaction more transparent before action execution. Instead of treating a spoken command as a direct robot program, the proposed system converts the command into an inspectable target hypothesis, displays the system’s interpretation through **XR**, and allows the operator to confirm, refine, correct, or cancel the result before it is used by the robotic branch.

The implemented prototype connects a Magic Leap 2 **XR** client, an edge perception server, and a simulation-first robotic execution environment. The **XR** client captures speech, **RGB-D** observations, camera metadata, spatial feedback, and validation decisions. The edge server performs session management, language-conditioned target segmentation, object-centric reconstruction, artifact storage, and refinement support. The simulation branch uses Isaac Sim and Isaac Lab to represent a KUKA/Robotiq pick-and-place workcell and to evaluate **VLA**-based execution in a controlled digital twin.

The validation results in Chapter 6 show that the complete perception-to-simulation workflow is functional under the tested conditions. They also show that reliability is uneven across stages: session integration and operator validation were robust in the prototype, while speech recognition, segmentation, reconstruction, and simulated manipulation remained sensitive to ambiguity, object visibility, reconstructed-asset quality, fine-tuning data, and workspace coverage.

7.1 Conclusions

The main objective of this dissertation was to design and implement a human-supervised intelligent picking workflow in which natural-language intent is grounded in the visual scene and made inspectable before robotic execution. This objective was achieved through a modular **XR**–edge–simulation architecture that separates sensing and operator interaction, edge-side perception, explicit human validation, and simulated robot execution.

A first conclusion is that the proposed session-centric architecture provides a practical mechanism for linking asynchronous data products within the same picking interaction. Speech events, image frames, camera metadata, segmentation outlines, reconstruction artifacts, refinement frames, validated object states, and cancellation events are produced at different times and by different modules. The validation campaign did not reveal cases where outputs from different sessions were mixed or stale artifacts were presented as active hypotheses. This supports the use of the session identifier as the main correlation mechanism for the implemented prototype.

A second conclusion is that **XR** feedback can make perception outputs inspectable before action. The system exposes the transcript-derived target, the segmentation outline, the estimated world-space centroid, the reconstructed object artifact, and the available validation actions to the operator. This turns target grounding into an explicit decision point rather than a hidden intermediate result. The operator validation workflow was functional in all tested trials, confirming that the accepted target state results from a deliberate validation action rather than from the initial speech command alone.

A third conclusion is that the edge-assisted perception branch is functional but condition-dependent. The local **STT** path was most reliable for short object-focused commands, while longer instructions produced more transcription errors. Segmentation performance decreased under occlusion and when multiple possible targets matched the spoken instruction. Centroid estimation met the defined acceptance criteria more consistently at close range, but degraded as the object occupied a smaller region in the headset view. Reconstruction produced artifacts that could be loaded, recognised, and inspected in **XR** under clear conditions, but remained affected by occlusion, capture viewpoint, object-prior mismatch, and scale or orientation alignment issues.

A fourth conclusion is that simulated **VLA** execution is feasible, but strongly dependent on the object asset, training data, and workspace distribution. The dataset bottle achieved the strongest result, reaching 74% success after object-specific fine-tuning and 81% after the final mixed fine-tuning stage. The reconstructed SAM 3D bottle remained harder to manipulate, although it improved to 44% with the final mixed policy. The dataset apple reached 18%, indicating that the policy had started to interact with the object but did not yet learn a robust complete behaviour. The expanded-workspace experiment further showed limited spatial generalisation: the dataset bottle achieved 87.5% success inside the trained region but only 7.1% outside it.

A final conclusion is that reconstructed objects and dataset-like simulation assets should not be treated as equivalent from the policy’s perspective. The dataset bottle and apple are existing simulation assets, with stable mesh, texture, scale, and collision properties, while the SAM 3D bottle is a reconstructed mesh converted into a simulation-compatible asset. The lower performance on the reconstructed bottle suggests that geometry, collision behaviour, visual appearance, and asset distribution have a direct effect on manipulation reliability. Therefore, successful integration of reconstructed objects into the digital twin is an important step, but robust manipulation of those reconstructed assets remains an open challenge.

7.2 Answering the Research Questions

RQ1: Session integration

How can an **AR-edge-robot architecture associate speech, **RGB-D** observations, spatial metadata, perception hypotheses, reconstruction artefacts, and validation decisions within the same picking session?**

The architecture answers this question through a persistent session identifier assigned to each operator command. This identifier is propagated through the **XR** client, **ROS** messages, edge-side segmentation and reconstruction services, artifact retrieval, refinement, validation, and cancellation. In the validation campaign, session consistency was preserved across the asynchronous workflow, with no observed mixing of transcripts, frames, segmentation results, reconstruction artifacts, or validation actions from different sessions. This confirms that session-centric orchestration is an appropriate design choice for maintaining traceability in the implemented prototype.

RQ2: Inspectable perception

To what extent can language-conditioned **2D segmentation and object-centric **3D** reconstruction provide inspectable hypotheses for the requested target?**

The results show that the perception pipeline can provide useful inspectable hypotheses when the input command and scene are sufficiently clear. Short speech commands preserved the target object in 8 out of 10 trials, while longer instructions succeeded in 6 out of 10 trials. Segmentation reached 10 out of 10 successes for clearly visible targets and remained mostly usable under nearby clutter, but performance decreased with partial occlusion and multiple possible targets. Centroid estimation was most reliable at close range, with mean errors of 0.012 m at 0.30 m and 0.017 m at 0.50 m, increasing at longer distances. Reconstruction was also effective in favourable conditions, reaching 10 out of 10 successes for clear objects, but was affected by viewpoint, occlusion, scale and orientation alignment, and object-prior mismatch. Therefore, the perception branch supports inspectability, but not unconditional autonomy.

RQ3: Operator validation

How can **AR feedback make the system's interpretation of a natural-language picking command visible, correctable, and cancellable before robotic execution?**

The implemented **XR** interface answers this question by displaying the system's interpretation as layered spatial evidence and by exposing explicit validation controls. The operator can inspect the segmentation outline, the estimated target position, the reconstructed artifact, the session state, and the returned hypothesis before confirming it. The validation test showed that inspection, confirmation, cancellation, refinement, and manual adjustment were available and functional in all 10 tested trials. This demonstrates that the workflow does not rely on direct speech-to-action execution; instead, it introduces an operator-approved target state before the simulation branch is allowed to use the result.

RQ4: VLA execution in simulation

How does the VLA-based execution branch perform in simulation after receiving a previously grounded and operator-validated target state?

The simulation branch was able to execute the pick-and-place task, but its performance varied substantially with the object and training distribution. The dataset bottle achieved 74% success after direct fine-tuning and 81% after the final mixed fine-tuning stage. The reconstructed SAM 3D bottle was more difficult, achieving 18% with the SAM 3D-specific policy, 12% with the dataset-bottle policy, and 44% with the final mixed policy. The dataset apple achieved 18% with the mixed policy. These results indicate that the VLA execution branch is functional in the digital twin, but also that policy robustness depends on the amount and diversity of demonstrations, the familiarity and physical stability of the object asset, and the match between training and evaluation conditions.

RQ5: Failure modes and limitations

What are the main limitations of the proposed workflow, particularly regarding ambiguous language, visually similar objects, segmentation errors, reconstruction uncertainty and transfer from simulation to the physical workcell?

The main observed limitations are consistent with the modular validation strategy. Speech errors occur more often for longer or less object-focused commands. Segmentation is less reliable with occlusion, clutter near the object boundary, or multiple plausible targets. Centroid estimation depends on depth quality, object distance, and mask stability. Reconstruction remains sensitive to input viewpoint, object-prior mismatch, and scale or orientation alignment. In simulation, the dominant issues are grasping/de-grasping failures, reconstructed-asset difficulty, limited object-specific data, and overfitting to the demonstrated workspace region. Since the robotic branch was validated only in simulation, transfer to a physical workcell remains a major limitation rather than a demonstrated outcome.

7.3 Main Contributions

The main contributions of this dissertation can be summarised as follows:

1. A closed-loop **XR-assisted intelligent picking workflow** that connects operator speech, egocentric visual observations, edge-side perception, human validation, and simulation-based robotic execution.
2. A **session-centric orchestration mechanism** that links transcripts, **RGB-D** observations, camera metadata, segmentation outputs, reconstruction artifacts, refinement data, cancellation events, and validation decisions within the same interaction episode.
3. An **edge-assisted target-grounding pipeline** that uses language-conditioned **2D** segmentation and object-centric **3D** reconstruction to produce inspectable target hypotheses instead of directly generating robot actions.

4. An **XR-based validation interface** that exposes segmentation overlays, world-space centroid estimates, reconstructed artifacts, status feedback, and controls for confirmation, refinement, manual correction, retaking, and cancellation.
5. A **simulation-first robotic execution branch** that connects validated object states to a KUKA/Robotiq digital twin and supports demonstration collection, Mimic-based dataset expansion, LeRobot export, OpenPI fine-tuning, and **VLA**-based rollout evaluation.
6. A **validation framework and experimental analysis** covering session coherence, speech intent capture, target segmentation, centroid estimation, reconstruction inspectability, operator validation, simulated **VLA** execution, and failure-mode diagnosis.
7. An **XR-based intelligent picking prototype** that supported research dissemination through a poster and participation in the Neurodesign in Human–Robot Interaction workshop at ICRA.¹

Together, these contributions show how existing perception, reconstruction, **XR**, and **VLA** components can be organised into a transparent and correctable picking workflow. The implementation also separates **XR** devices according to their functional roles. The Magic Leap 2 supports operator-facing intent capture, egocentric sensing, spatial feedback, and validation in the physical workspace, whereas the Meta Quest 3 was used as a teleoperation interface for collecting demonstrations in the simulated learning branch. This separation supports the modularity of the architecture and shows that different devices can be integrated through role-specific interfaces. However, it should not be interpreted as a formal cross-device portability evaluation of the complete application. The contribution is therefore primarily architectural and integrative: it demonstrates how the system can show what it understood, where it grounded the target, what object representation it reconstructed, and what state is passed to the simulated robot branch.

7.4 Limitations

Several limitations remain. First, the current system depends on the quality of the speech transcript and on the extracted target phrase. The local **STT** model is suitable for concise object-focused commands, but longer instructions increase the probability of corrupted object names or ambiguous prompts. Since segmentation depends on this text, speech errors can propagate to the visual grounding stage.

Second, segmentation and centroid estimation depend on scene visibility and depth quality. The system performs best when the target is visible, close to the headset, and visually distinct from surrounding objects. Occlusion, clutter near object boundaries, multiple plausible targets, reflective or transparent surfaces, small apparent object size, and unstable depth samples can reduce the reliability of the returned hypothesis.

Third, object-centric reconstruction supports operator inspection, but the generated mesh remains affected by the limitations of the reconstruction model. The reconstructed artifact may

¹Workshop website: <https://neurodesign-in-hri.webflow.io/>

contain incomplete geometry, distorted surfaces, incorrect scale, orientation offsets, or local coordinate conventions that differ from those expected by the simulation environment. These issues are relevant when the same artifact is later converted into a simulation asset, since visual plausibility in **XR** does not necessarily imply accurate physical or geometric behaviour in simulation. Operator validation and refinement reduce this risk, but they cannot fully remove the uncertainty introduced by the reconstruction model.

Fourth, the simulated **VLA** branch remains data-sensitive. The best performance was obtained with the dataset bottle inside the trained workspace region, while the reconstructed SAM 3D bottle, the dataset apple, and expanded-area tests showed lower reliability. This indicates that object asset properties, dataset balance, demonstration diversity, and workspace coverage strongly affect the learned policy.

Fifth, the dissertation does not validate physical robot deployment. The digital twin provides a safe and repeatable environment for development and testing, but it does not fully represent physical contact dynamics, real gripper behaviour, calibration drift, object compliance, sensor noise, safety constraints, or human proximity. For this reason, simulated success should be interpreted as evidence of feasibility in the implemented digital twin, not as proof of physical deployment readiness.

Finally, the validation does not include a formal user study with multiple operators. The operator-facing interface was functionally tested, but comparative usability, workload, trust, task-time reduction, and safety-related outcomes remain future evaluation dimensions.

7.5 Future Work

The results obtained in this dissertation show that the proposed workflow is functional, but also identify several aspects that require further development before the system can be considered robust in less controlled scenarios. Future work should therefore focus on increasing the scale of the validation, improving the robustness of the perception and reconstruction stages, extending the robotic execution branch, and evaluating the end-to-end latency and responsiveness of the complete workflow, testing it with users, and, eventually, deploying it in a physical robotic cell.

Broader validation and user studies

Future work should expand the number and diversity of tests. On the perception side, this includes more object categories, look-alike instances, cluttered scenes, varied lighting, reflective or transparent objects, different user viewpoints, and noisier acoustic environments. On the **XR** side, a structured user study should evaluate usability, workload, trust, perceived safety, correction behaviour, and the usefulness of the validation interface. Such a study should include both simple object-selection tasks and ambiguous cases where the system must ask for confirmation or the user must correct a wrong hypothesis. Comparison against a baseline interface, such as a tablet, fixed screen, menu-based selector, or direct command interface, would make it possible to quantify the practical value of the **XR** application.

End-to-end latency evaluation

Future work should evaluate the end-to-end latency of the complete workflow, from the moment the operator initiates intent capture to the availability of the validated object state and the onset of simulated robotic execution. This evaluation should distinguish the time required to obtain the first inspectable **2D** feedback from the time required to obtain the complete **3D** object hypothesis, since reconstruction is performed asynchronously.

The evaluation should separately measure intent capture and speech transcription, RGB acquisition and transmission, target extraction, segmentation, centroid estimation, **3D** reconstruction, artefact retrieval and loading, **XR** rendering, operator validation, policy inference, and simulated action onset. Relevant metrics include mean and median latency, worst-case latency, variability or jitter, timeout and failure rates, and the relative contribution of each stage to the total processing time. Human decision time should be reported separately from computation and network latency.

Improved language disambiguation

The language interface should be extended beyond a single local transcript. The current local **STT** path can remain on the headset because it is responsive and independent from external services, but the raw audio could also be sent to the edge server within the same session. A larger edge-side **STT** model could then produce a second transcript or correct the headset transcript when confidence is low. The **XR** interface could compare both interpretations, show the extracted target phrase, and allow the operator to accept or edit the target before segmentation. This would preserve the low-latency local path while using the edge machine to improve robustness for longer commands, noisy speech, or ambiguous object names.

Active target disambiguation

When multiple objects match the same command, the system should not force a single segmentation result. Instead, the edge server could return multiple candidate masks, and the **XR** interface could display them as numbered or coloured alternatives. The operator could then select the intended object directly in the scene, provide an additional spoken constraint, or point to the correct candidate. This would make ambiguous language and visually similar objects recoverable interaction cases rather than hard failures.

Reconstruction scale, rotation, and asset calibration

Further work is needed to study the scale and rotation problems observed during mesh inspection and simulation preparation. Possible directions include using calibrated reference objects, adding known-size markers, estimating scale from depth and camera intrinsics, applying mesh canonicalisation rules, validating object orientation against the **XR** world frame, and testing alternative conversion or alignment pipelines before USD export. A dedicated calibration benchmark with

known object dimensions and poses would make it possible to measure reconstruction accuracy instead of relying only on visual inspection.

Stronger VLA policies and dataset coverage

The VLA branch should be made more robust through larger and more diverse datasets. The existing teleoperation and recording workflow can be reused whenever additional seed demonstrations are required to address gaps in object, pose, workspace, or task coverage. Future demonstrations should cover a wider workspace area, more object poses, different approach directions, varied camera viewpoints, object distractors, and multiple object categories. The final mixed policy showed that additional data can improve transfer, but the expanded-area test showed strong overfitting to the demonstrated region. Future datasets should therefore be designed deliberately to cover the full intended tabletop, and evaluation should separate in-distribution from out-of-distribution object placements.

Physical deployment and sim-to-real validation

A major next step is deployment on the physical KUKA/Robotiq workcell. This requires calibration between the headset, robot, workspace, cameras, and simulation frame; validation of the reconstructed asset against the physical object; safety monitoring; real gripper integration; contact-aware grasp testing; and recovery behaviours. Human-in-the-loop calibration will be important because the operator can verify whether the XR target, simulated object, and physical robot frame are aligned before motion. The sim-to-real gap should be evaluated progressively, starting with constrained motions and supervised grasp trials before moving to autonomous pick-and-place.

7.6 Final Remarks

This dissertation contributes to more transparent robot interaction by placing inspectable perception and human validation between natural-language commands and robotic execution. The central idea is that an intelligent picking system should not act only on a hidden interpretation of a spoken instruction. It should show what it understood, where it grounded the target, what 3D object representation it generated, and what state is passed to the robot branch.

The resulting prototype demonstrates a coherent path for combining XR, edge perception, object-centric reconstruction, operator validation, and simulated VLA execution. The results confirm that this workflow is viable under controlled conditions, while also identifying the main barriers to robust deployment: language ambiguity, visual grounding uncertainty, reconstruction scale and orientation errors, reconstructed-asset simulation fidelity, policy overfitting, and the remaining sim-to-real gap. These findings provide a clear basis for future work towards a more robust, user-tested, and physically deployed intelligent picking system.

References

- [1] A. Keshvarparast, D. Battini, O. Battaia, and A. Pirayesh, “Collaborative robots in manufacturing and assembly systems: Literature review and future research agenda,” *Journal of Intelligent Manufacturing*, 2023. DOI: [10.1007/s10845-023-02137-w](https://doi.org/10.1007/s10845-023-02137-w).
- [2] H. Kagermann, W. Wahlster, and J. Helbig, *Recommendations for implementing the strategic initiative industrie 4.0: Final report of the industrie 4.0 working group*, acatech, 2013. [Online]. Available: <https://en.acatech.de/publication/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/>.
- [3] European Commission, Directorate-General for Research and Innovation, *Industry 5.0: Towards a sustainable, human-centric and resilient european industry*, European Commission, 2021. [Online]. Available: <https://op.europa.eu/o/opportal-service/download-handler?format=pdf&identifier=468a892a-5097-11eb-b59f-01aa75ed71a1>.
- [4] U. Othman and E. Yang, “Human–robot collaborations in smart manufacturing environments: Review and outlook,” *Sensors*, vol. 23, no. 12, p. 5663, 2023. DOI: [10.3390/s23125663](https://doi.org/10.3390/s23125663).
- [5] G. d. M. Costa, M. R. Petry, and A. P. Moreira, “Augmented reality for human–robot collaboration and cooperation in industrial applications: A systematic literature review,” *Sensors*, vol. 22, no. 7, p. 2725, 2022. DOI: [10.3390/s22072725](https://doi.org/10.3390/s22072725).
- [6] C. T. Chang and B. Hayes, “A survey of augmented reality for human–robot collaboration,” *Machines*, vol. 12, no. 8, p. 540, 2024. DOI: [10.3390/machines12080540](https://doi.org/10.3390/machines12080540).
- [7] P. Milgram and F. Kishino, “A taxonomy of mixed reality visual displays,” *IEICE Transactions on Information and Systems*, vol. E77-D, no. 12, pp. 1321–1329, 1994.
- [8] R. T. Azuma, “A survey of augmented reality,” *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997. DOI: [10.1162/pres.1997.6.4.355](https://doi.org/10.1162/pres.1997.6.4.355).
- [9] I. Soares, R. B. Sousa, M. Petry, and A. P. Moreira, “Accuracy and repeatability tests on hololens 2 and htc vive,” *Multimodal Technologies and Interaction*, vol. 5, no. 8, p. 47, 2021. DOI: [10.3390/mti5080047](https://doi.org/10.3390/mti5080047).
- [10] G. M. Costa, M. R. Petry, J. G. Martins, and A. P. G. M. Moreira, “Assessment of multiple fiducial marker trackers on hololens 2,” *IEEE Access*, vol. 12, pp. 14 211–14 226, 2024. DOI: [10.1109/ACCESS.2024.3356722](https://doi.org/10.1109/ACCESS.2024.3356722).
- [11] R. Xu, S. N. Razavi, and R. Zheng, “Edge video analytics: A survey on applications, systems and enabling techniques,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2951–2982, 2023. DOI: [10.1109/COMST.2023.3323091](https://doi.org/10.1109/COMST.2023.3323091).

- [12] J. G. Martins, G. d. M. Costa, M. R. Petry, P. G. Costa, and A. P. Moreira, "A multi-user, multi-robot collaboration through augmented reality," in *2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2024, pp. 144–150. DOI: [10.1109/ICARSC61747.2024.10535948](https://doi.org/10.1109/ICARSC61747.2024.10535948).
- [13] D. Aschenbrenner, J. S. I. Rieder, D. van Tol, *et al.*, "Mirrorlabs—creating accessible digital twins of robotic production environment with mixed reality," in *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2020, pp. 43–48. DOI: [10.1109/AIVR50618.2020.00017](https://doi.org/10.1109/AIVR50618.2020.00017).
- [14] B. Schwerdtfeger, R. Reif, W. A. Günthner, and G. Klinker, "Pick-by-vision: There is something to pick at the end of the augmented tunnel," *Virtual Reality*, vol. 15, pp. 213–223, 2011. DOI: [10.1007/s10055-011-0187-9](https://doi.org/10.1007/s10055-011-0187-9).
- [15] C. Tadjine, A. Ouafi, A. Benlamoudi, and A. Taleb-Ahmed, "Computer vision in warehouse management automation: A survey on implemented methods with prototyping hardware," *Engineering Applications of Artificial Intelligence*, vol. 160, p. 111 886, 2025. DOI: [10.1016/j.engappai.2025.111886](https://doi.org/10.1016/j.engappai.2025.111886).
- [16] P. K. Gollapudi, S. R. Kannan, S. N. S. E. Prasad, K. Balaji, S. B. K. Reddy, and S. Sujatha, "Optimizing warehouse picking and sorting with augmented reality and cnn-based item recognition," in *2025 3rd IEEE International Conference on Knowledge Engineering and Communication Systems (ICKECS)*, Hyderabad, India, Apr. 2025.
- [17] *Robotics – safety requirements – part 1: Industrial robots*, Geneva, Switzerland: International Organization for Standardization, 2025. [Online]. Available: <https://www.iso.org/standard/73933.html>.
- [18] *Robots and robotic devices – collaborative robots*, Geneva, Switzerland: International Organization for Standardization, 2016.
- [19] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018. DOI: [10.1016/j.mechatronics.2018.02.009](https://doi.org/10.1016/j.mechatronics.2018.02.009).
- [20] W. Li, Y. Hu, Y. Zhou, and D. T. Pham, "Safe human–robot collaboration for industrial settings: A survey," *Journal of Intelligent Manufacturing*, vol. 35, pp. 2235–2261, 2024. DOI: [10.1007/s10845-023-02159-4](https://doi.org/10.1007/s10845-023-02159-4).
- [21] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2020, ISBN: 9781119523994.
- [22] P. A. Rauschnabel, R. Felix, C. Hinsch, H. Shahab, and F. Alt, "What is xr? towards a framework for augmented and virtual reality," *Computers in Human Behavior*, vol. 133, p. 107 289, 2022. DOI: [10.1016/j.chb.2022.107289](https://doi.org/10.1016/j.chb.2022.107289).
- [23] J. Egger and T. Masood, "Augmented reality in support of intelligent manufacturing – a systematic literature review," *Computers & Industrial Engineering*, vol. 140, p. 106 195, 2020. DOI: [10.1016/j.cie.2019.106195](https://doi.org/10.1016/j.cie.2019.106195).
- [24] R. Palmarini, J. A. Erkoyuncu, R. Roy, and H. Torabmostaedi, "A systematic review of augmented reality in maintenance," *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 215–228, 2018. DOI: [10.1016/j.rcim.2017.06.002](https://doi.org/10.1016/j.rcim.2017.06.002).
- [25] T. Masood and J. Egger, "Adopting augmented reality in the age of industrial digitalisation," *Computers in Industry*, vol. 115, p. 103 112, 2020. DOI: [10.1016/j.compind.2019.07.002](https://doi.org/10.1016/j.compind.2019.07.002).

- [26] XR4HUMAN Consortium, *Code of conduct for the human-centered and ethical development of immersive technologies*, XR4HUMAN project deliverable, Living guidance document on ethical and human-centered XR, 2025. [Online]. Available: <https://xr4human.eu/>.
- [27] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents). Cambridge, MA: MIT Press, 2005, ISBN: 9780262201629.
- [28] G. Du, K. Wang, S. Lian, and K. Zhao, “Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review,” *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2020. DOI: [10.1007/s10462-020-09888-5](https://doi.org/10.1007/s10462-020-09888-5).
- [29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, United Kingdom: Cambridge University Press, 2004. DOI: [10.1017/CBO9780511811685](https://doi.org/10.1017/CBO9780511811685).
- [30] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. DOI: [10.1109/JIOT.2016.2579198](https://doi.org/10.1109/JIOT.2016.2579198).
- [31] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30–39, 2017. DOI: [10.1109/MC.2017.9](https://doi.org/10.1109/MC.2017.9).
- [32] Y. Guan, X. Hou, N. Wu, B. Han, and T. Han, “Deepmix: Mobility-aware, lightweight, and hybrid 3d object detection for headsets,” in *Proceedings of the 20th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '22)*, 2022, pp. 1–14. DOI: [10.1145/3498361.3538945](https://doi.org/10.1145/3498361.3538945).
- [33] R. Szeliski, *Computer Vision: Algorithms and Applications* (Texts in Computer Science), 2nd ed. Cham, Switzerland: Springer Cham, 2022, ISBN: 9783030343729. DOI: [10.1007/978-3-030-34372-9](https://doi.org/10.1007/978-3-030-34372-9).
- [34] R. Bommasani *et al.*, “On the opportunities and risks of foundation models,” *arXiv preprint arXiv:2108.07258*, 2021. arXiv: [2108.07258](https://arxiv.org/abs/2108.07258) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2108.07258>.
- [35] A. Kirillov, E. Mintun, N. Ravi, *et al.*, “Segment anything,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 4015–4026. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Kirillov_Segment_Anything_ICCV_2023_paper.html.
- [36] N. Carion, L. Gustafson, Y.-T. Hu, *et al.*, “SAM 3: Segment anything with concepts,” *arXiv preprint arXiv:2511.16719*, 2025. arXiv: [2511.16719](https://arxiv.org/abs/2511.16719) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2511.16719>.
- [37] SAM 3D Team, X. Chen, F.-J. Chu, *et al.*, “SAM 3D: 3Dfy anything in images,” *arXiv preprint arXiv:2511.16624*, 2025. arXiv: [2511.16624](https://arxiv.org/abs/2511.16624) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2511.16624>.
- [38] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, 2023. DOI: [10.1145/3592433](https://doi.org/10.1145/3592433).
- [39] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

- [40] S. Harnad, “The symbol grounding problem,” *Physica D: Nonlinear Phenomena*, vol. 42, no. 1–3, pp. 335–346, 1990. DOI: [10.1016/0167-2789\(90\)90087-6](https://doi.org/10.1016/0167-2789(90)90087-6).
- [41] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” in *Proceedings of The 6th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 205, PMLR, 2023, pp. 287–318. [Online]. Available: <https://proceedings.mlr.press/v205/ichter23a.html>.
- [42] A. Radford, J. W. Kim, C. Hallacy, *et al.*, “Learning transferable visual models from natural language supervision,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 8748–8763. [Online]. Available: <https://proceedings.mlr.press/v139/radford21a.html>.
- [43] B. Zitkovich, T. Yu, S. Xu, *et al.*, “RT-2: Vision-language-action models transfer web knowledge to robotic control,” in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229, PMLR, 2023, pp. 2165–2183. [Online]. Available: <https://proceedings.mlr.press/v229/zitkovich23a.html>.
- [44] M. J. Kim, K. Pertsch, S. Karamcheti, *et al.*, “OpenVLA: An open-source vision-language-action model,” in *Proceedings of the 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270, PMLR, 2025, pp. 2679–2713. [Online]. Available: <https://proceedings.mlr.press/v270/kim25c.html>.
- [45] K. Black, N. Brown, D. Driess, *et al.*, π_0 : A vision-language-action flow model for general robot control, 2024. arXiv: [2410.24164](https://arxiv.org/abs/2410.24164) [cs.RO]. [Online]. Available: <https://arxiv.org/abs/2410.24164>.
- [46] K. Black, N. Brown, J. Darpinian, *et al.*, “ $\pi_{0.5}$: A vision-language-action model with open-world generalization,” in *Proceedings of the 9th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 305, PMLR, 2025, pp. 17–40. [Online]. Available: <https://proceedings.mlr.press/v305/black25a.html>.
- [47] *Robotics – safety requirements – part 2: Industrial robot applications and robot cells*, Geneva, Switzerland: International Organization for Standardization, 2025. [Online]. Available: <https://www.iso.org/standard/73934.html>.
- [48] A. D. Dragan, K. C. T. Lee, and S. S. Srinivasa, “Legibility and predictability of robot motion,” in *Proceedings of the 8th ACM/IEEE International Conference on Human–Robot Interaction (HRI)*, 2013, pp. 301–308. DOI: [10.1109/HRI.2013.6483603](https://doi.org/10.1109/HRI.2013.6483603).
- [49] J. G. Martins, K. Nutonen, P. Costa, *et al.*, “Ar/vr digital twin for simulation and data collection of robotic environments,” in *2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Funchal, Portugal, 2025, pp. 198–203. DOI: [10.1109/ICARSC65809.2025.10970158](https://doi.org/10.1109/ICARSC65809.2025.10970158).
- [50] A. Rejeb, J. G. Keogh, S. F. Wamba, and H. Treiblmaier, “The potentials of augmented reality in supply chain management: A state-of-the-art review,” *Management Review Quarterly*, vol. 71, pp. 819–856, 2021. DOI: [10.1007/s11301-020-00201-w](https://doi.org/10.1007/s11301-020-00201-w).
- [51] R. Elbert and T. Sarnow, “Augmented reality in order picking—boon and bane of information (over-) availability,” in *Intelligent Human Systems Integration 2019 (IHSI 2019)*, Springer, Cham, 2019, pp. 400–406. DOI: [10.1007/978-3-030-11051-2_61](https://doi.org/10.1007/978-3-030-11051-2_61).

- [52] A. Windhausen, J. Heller, T. Hilken, D. Mahr, R. Di Palma, and L. Quintens, “Exploring the impact of augmented reality smart glasses on worker well-being in warehouse order picking,” *Computers in Human Behavior*, vol. 155, p. 108 153, 2024. DOI: [10.1016/j.chb.2024.108153](https://doi.org/10.1016/j.chb.2024.108153).
- [53] D. Gentile, F. Musolino, M. Dastan, and M. Fiorentino, “Augmented reality glasses for order picking: A user study comparing numeric code, 2d-map, and 3d-map visualizations,” *J*, vol. 8, no. 3, p. 32, 2025. DOI: [10.3390/j8030032](https://doi.org/10.3390/j8030032).
- [54] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, “An edge-computing based architecture for mobile augmented reality,” *IEEE Network*, vol. 33, no. 4, pp. 162–169, 2019. DOI: [10.1109/MNET.2018.1800132](https://doi.org/10.1109/MNET.2018.1800132).
- [55] J. Chen and X. Ran, “Deep learning with edge computing: A review,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019. DOI: [10.1109/JPROC.2019.2921977](https://doi.org/10.1109/JPROC.2019.2921977).
- [56] M. Minderer, A. Gritsenko, A. Stone, *et al.*, “Simple open-vocabulary object detection with vision transformers,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.
- [57] S. Liu, Z. Zeng, T. Ren, *et al.*, “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.
- [58] T. L'uddecke and A. S. Ecker, “Image segmentation using text and image prompts,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [59] X. Lai, Z. Tian, Y. Chen, *et al.*, “Lisa: Reasoning segmentation via large language model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 9579–9589.
- [60] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, “A survey on learning-based robotic grasping,” *Current Robotics Reports*, vol. 1, pp. 239–249, 2020. DOI: [10.1007/s43154-020-00021-6](https://doi.org/10.1007/s43154-020-00021-6).
- [61] R. Newbury, M. Gu, L. Chumbley, *et al.*, “Deep learning approaches to grasp synthesis: A review,” *arXiv preprint arXiv:2207.02556*, 2023.
- [62] R. Wang, S. Xu, C. Dai, *et al.*, “MoGe: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 5261–5271. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2025/html/Wang_MoGe_Unlocking_Accurate_Monocular_Geometry_Estimation_for_Open-Domain_Images_with_CVPR_2025_paper.html.
- [63] I. Soares, M. Petry, and A. P. Moreira, “Programming robots by demonstration using augmented reality,” *Sensors*, vol. 21, no. 17, p. 5976, 2021. DOI: [10.3390/s21175976](https://doi.org/10.3390/s21175976).
- [64] C. M. Fang, K. Zieliński, P. Maes, J. Paradiso, B. Blumberg, and M. B. Kjærgaard, *Enabling waypoint generation for collaborative robots using LLMs and mixed reality*, Published in the VLMNM 2024 Workshop at ICRA 2024, 2024. DOI: [10.48550/arXiv.2403.09308](https://doi.org/10.48550/arXiv.2403.09308). arXiv: 2403.09308 [cs.HC].
- [65] C. Gkournelos, C. Konstantinou, and S. Makris, “An llm-based approach for enabling seamless human–robot collaboration in assembly,” *CIRP Annals*, vol. 73, no. 1, pp. 9–12, 2024. DOI: [10.1016/j.cirp.2024.04.002](https://doi.org/10.1016/j.cirp.2024.04.002).

- [66] Y. Yin, J. Fan, A. Liu, and P. Zheng, “A mixed reality-assisted scene-centric robot programming approach for human–robot collaborative manufacturing,” *Robotics and Computer-Integrated Manufacturing*, vol. 98, p. 103 146, 2026. DOI: [10.1016/j.rcim.2025.103146](https://doi.org/10.1016/j.rcim.2025.103146).
- [67] Y. Kim, D. Kim, J. Choi, J. Park, N. Oh, and D. Park, “A survey on integration of large language models with intelligent robots,” *Intelligent Service Robotics*, vol. 17, pp. 1091–1107, 2024. DOI: [10.1007/s11370-024-00550-5](https://doi.org/10.1007/s11370-024-00550-5).
- [68] Open X-Embodiment Collaboration, A. O’Neill, A. Rehman, *et al.*, “Open x-embodiment: Robotic learning datasets and RT-X models,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024. DOI: [10.1109/ICRA57147.2024.10611477](https://doi.org/10.1109/ICRA57147.2024.10611477).
- [69] A. Mandlekar, S. Nasiriany, B. Wen, *et al.*, “MimicGen: A data generation system for scalable robot learning using human demonstrations,” in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229, PMLR, 2023, pp. 1820–1864. [Online]. Available: <https://proceedings.mlr.press/v229/mandlekar23a.html>.