

**ALMA MATER STUDIORUM – UNIVERSITY OF
BOLOGNA**

SCHOOL OF ENGINEERING AND ARCHITECTURE

(Bologna Campus)

Academic Year 2025/2026

Master Thesis

Curricular Internship performed by the Student:

Naveen Joseph Delphin (Matriculation No. 0001118682)

enrolled in the degree programme:

(8891) AUTOMATION ENGINEERING / INGEGNERIA DELL'AUTOMAZIONE

at:

FRAUNHOFER ITALIA RESEARCH KONSORTIALGESELLSCHAFT MIT
BESCHRÄNKTER HAFTUNG

on the following topic:

Single-Camera Vision System for Real-Time Manual Assembly Verification and
Adaptive Operator Guidance

Student:

Naveen Joseph Delphin

Academic Tutor:

Prof. Luigi Di Stefano

Signature

Signature approving the final report

Host Organisation Supervisor:

Dr. Marco Todescato

Signature approving the final report

Abstract

Manual assembly remains an important part of manufacturing where flexibility, low production volumes, or frequently changing operations make full automation impractical. This internship focused on the development of a vision-based system for monitoring, verifying, and assisting manual assembly operations using a single RGB camera. The objective was to develop a modular framework capable of interpreting the operator's actions, identifying the current assembly stage, verifying the correctness of the performed operations, and providing appropriate visual assistance. The development involved the investigation and comparison of different computer vision and machine-learning approaches for the main components of the system. These included ROI-based motion and interaction detection, hand and trajectory analysis, object detection, object pose estimation, assembly-stage verification, and visual guidance. Several alternative approaches were experimentally evaluated before selecting the methods that provided an appropriate balance between robustness, computational requirements, and integration with the overall system. The resulting system was integrated into a real-time assembly monitoring pipeline and equipped with configurable visual assistance levels. The complete framework was evaluated using an experimental assembly and disassembly task involving a pneumatic piston. The evaluation considers both technical and user-level aspects, including object detection and stage verification performance, task duration, trajectory characteristics, and user feedback collected through questionnaires. The developed framework demonstrates the feasibility of combining multiple RGB-based perception and temporal reasoning techniques into a modular system for real-time monitoring and adaptive assistance of manual assembly operations.

Acknowledgements

I would like to express my sincere gratitude to my academic tutor, Prof. Luigi Di Stefano, for approving this topic and supervising my work throughout this internship. I am truly grateful to have had the opportunity to work under his guidance.

I would also like to thank Leonardo, Marco, and Isabella at Fraunhofer Italia, who supported me closely throughout my time at the workplace. Their technical guidance, patience, and willingness to help at every stage of this work were invaluable, and this thesis would not have been possible without them.

I would further like to thank the volunteers who participated in the experimental evaluation of the system. Their time, patience, and willingness to engage with an early-stage prototype were essential to obtaining the results presented in this work.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	viii
List of Tables	x
List of Abbreviations	xi
1 Introduction	1
1.1 Background and Context	1
1.2 Host Organisation	2
1.3 Research Question and Original Contributions	2
1.4 Internship Objectives	3
1.5 Main Activities	3
1.6 Expected Results	3
1.7 Structure of the Thesis	4
2 Problem Definition and Requirements	5
2.1 Manual Assembly Process	5
2.2 Challenges in Manual Assembly	5
2.3 Research Problem	6
2.4 Functional and Non-Functional Requirements	7
2.5 Evaluation Criteria	11
3 Literature Review	13
3.1 Industrial Assembly Assistance Systems	13
3.2 Human Action and Hand-Object Interaction Recognition	14
3.3 Object Detection for Assembly Monitoring	14
3.4 Object Pose and Orientation Estimation	15
3.5 Assembly Stage and Procedure Verification	15

3.6	Related Trends: Egocentric, Hand-Object, and AR-Based Assembly Understanding	16
3.7	Comparative Analysis and Research Gap	16
3.8	Positioning of the Proposed Work	19
4	System Design	21
4.1	Overall System Architecture	21
4.2	Hardware and Software Architecture	22
4.3	Assembly Stage Definition	23
4.4	Workspace and Region Definition	23
4.5	Preliminary System and Motivation for Thesis Development	24
5	Stage Segmentation Development	26
5.1	Stage Segmentation Problem	26
5.2	ROI Motion Trigger	27
5.3	Color-Based Trigger	27
5.4	Depth-Based Trigger	28
5.5	Hand-Presence Trigger	29
5.6	Trajectory-Based Interaction Evidence	29
5.7	Limitations of Individual Trigger Methods	30
5.8	Hybrid ROI Interaction Trigger	31
6	Object Detection Development	34
6.1	Initial Prompt-Based Detection	34
6.2	Dataset Preparation	35
6.2.1	Real Dataset Preparation & Annotation	35
6.2.2	Synthetic Data Generation	35
6.3	YOLOv8 Models	37
6.4	RF-DETR	37
6.5	RTMDet	37
6.6	Comparative Evaluation	38
7	Object Pose Estimation Development	41
7.1	Pose Estimation Requirements	41
7.2	Dense Object Nets	41
7.3	FoundationPose Investigation	42
7.4	Pose Classification Reformulation	46
7.5	YOLO-Based Pose Classifier	47
8	Motion Understanding and Stage Management	49
8.1	MotionFSM Design	49

8.2	StageFSM Design	51
8.3	Ordered and Unordered Stage Groups	52
8.4	Active-Hand Selection Policy	53
8.5	Trajectory Acquisition & Tool-Hand Synchronisation	54
8.6	Dynamic Time Warping	54
8.7	Stage Completion and Error Handling	56
9	User Guidance Interface	57
9.1	Interface Requirements	57
9.2	Adaptive Assistance Modes	58
9.2.1	Full-Assistance Mode	58
9.2.2	Medium-Assistance Mode	59
9.2.3	No-Assistance Mode	59
9.3	Guidance Region Visualisation	60
9.4	CAD Projection	60
9.5	Feedback and Warning Messages	61
9.6	Interface Integration	62
10	Final Integrated System	63
10.1	Final Architecture	63
10.2	End-to-End Pipeline	63
10.3	Real-Time Inference Architecture	64
10.4	System Performance and Robustness	64
10.5	Limitations And Possible Improvements	65
11	Experimental Evaluation	67
11.1	Research Questions and Hypotheses	67
11.2	Participants	68
11.3	Experimental Conditions	68
11.4	Experimental Protocol	69
11.5	Objective Performance Measures	70
11.6	Subjective Measures	72
11.7	Statistical Analysis	73
11.8	Results	75
11.8.1	Task Completion Time	76
11.8.2	Assembly Errors	76
11.8.3	Trajectory and Motion Metrics	77
11.8.4	Questionnaire Results	80
11.9	Discussion	82

12 Conclusions	86
12.1 Summary of the Internship	86
12.2 Principal Results	86
12.3 Technical Contributions	86
12.4 Skills and Knowledge Acquired	87
A Assembly Procedure	88
B System Parameters	94
C Questionnaires	101
D Additional Experimental Results	102
E Software and Hardware Configuration	105
Bibliography	109

List of Figures

2.1	Components of the pneumatic piston assembly.	6
4.1	Overall Final Integrated System Architecutre	21
4.2	Camera placement	22
4.3	Regions of Interest (ROIs) in the workspace are color-coded as follows: yellow for tool box, dark pink for front cover, violet for piston rod, lime green for piston, cyan for mounting screw, bright green for cylinder tube, dark blue for rear cover, orange for bottom tie rod nut, bluish green for tie rod nut assemblies, and bright pink for piston rod nut.	24
6.1	Parts of Pneumatic Piston	36
6.2	Representative synthetic image samples generated for training the object detection model.	36
7.1	Evaluation of the Pose Z override (1.24 m) against the measured median depth and projection-to-mask ratio across frames. The persistent depth offset between the configured pose depth and sensor measurements results in quality flags.	43
7.2	Evaluation of the Pose Z override (1.14 m) against the measured median depth and projection-to-mask ratio across frames. The measured depth closely follows the configured pose depth, resulting in substantially fewer quality flags.	43
7.3	Effect of camera height on depth availability and geometric consistency. Increasing the camera height from 30 cm to 140 cm substantially improves the completeness of the measured depth and the agreement between projected and observed object geometry.	45
7.4	Variation in pose-depth estimation error at a camera height of 140 cm. Although the depth measurements are complete, the estimated pose depth remains variable across frames, indicating residual ambiguity in the estimated object orientation and geometry.	45
7.5	Incorrect pose estimation caused by symmetric geometry.	46

7.6	Yolo Pose Classification for front cover. (a) side pose, (b) flipped pose and (c) correct pose	48
8.1	Comparison of (a) Euclidean matching vs (b) Dynamic time warping matching	56
9.1	Light-blue bounding box in the lower-left region delineates the target part-pick ROI; Cyan CAD projection adjacent to component 11 indicates the target placement pose for the retrieved nut; Grey status banner in the upper-right region indicates operational progress—transitioning to green upon stage success or red upon failure; Highlighted yellow panel in the mid-right section provides real-time telemetry, including orientation compliance, motion activity, FSM state transitions, and active object detections; The instruction panel provides sequential textual guidance, while the looping reference viewport renders the pre-recorded expert baseline video for the active stage.	58
9.2	Highlighted yellow row marks the active assembly stage, dynamically advancing down the table as sequential stages are completed; The Instructions column provides procedural guidelines for executing the current manipulation; The Results column logs real-time operational feedback, including object orientation, motion activity metrics, and current FSM states; The Status column records the final execution outcome (pass/fail) upon stage completion.	59
9.3	The primary viewport displays the active stage index alongside its descriptor name, while the central Grey status tab indicates real-time operational state—rendering grey for an ongoing stage, green upon successful completion, and red upon failure.	60
9.4	(a) The ROI has a strong border and prominent color fill indicating Active, (b) ROI has only a visible boundary indicating no Active and signaling user to pick component from that box	61
11.1	Fitted learning trajectories across guidance conditions with interruptions held at the dataset mean (0.47). Full guidance produces steady improvement; mid guidance plateaus; no guidance improves from a lower baseline.	80
A.1	Comparison of Dismantle and Assembly JSON stage definitions.	88

List of Tables

2.1	Functional requirements and acceptance criteria	7
2.2	Non-functional requirements and acceptance criteria	9
2.3	Evaluation criteria and corresponding requirements	12
3.1	Comparison of representative approaches for assembly understanding.	17
5.1	Comparison of Feature Detection Methods	30
6.1	Model evaluation results.	39
7.1	Z coordinate Override Evaluation	43
7.2	Dataset Robustness Analysis	44
7.3	Camera Height Evaluation	45
7.4	Orientation classifier validation accuracy.	47
8.1	MotionFSM transition predicates.	50
11.1	Task completion time by assistance level ($n = 70$ trials).	76
11.2	Failed stages per trial by assistance level.	77
11.3	Motion-quality metrics by assistance level (mean $\pm$ SD).	77
11.4	Primary mixed-effects model: composite skill score.	78
11.5	Interaction model: composite skill score with condition $\times$ learning interaction.	79
11.6	Per-trial Likert responses by assistance level (mean $\pm$ SD).	80
11.7	Debrief preferences ($n = 8$).	81
A.1	Assembly and dismantling procedure represented by the implemented stage configuration.	89
B.1	Motion and interaction trigger parameters.	95
B.2	Colour and depth trigger parameters (disabled by default).	96
B.3	MotionFSM parameters.	97
B.4	Stage scoring, DTW, and synchronisation parameters.	98

B.5	Active-hand selection, MediaPipe, and workspace parameters. . .	99
E.1	Stage descriptor parameters used by the FSM configuration. . . .	105
E.2	Implementation and configuration elements for unordered stage progression.	108

List of Abbreviations

AI	Artificial Intelligence
CAD	Computer-Aided Design
DTW	Dynamic Time Warping
FSM	Finite-State Machine
GUI	Graphical User Interface
mAP	Mean Average Precision
RGB	Red, Green and Blue
ROI	Region of Interest

Chapter 1

Introduction

1.1 Background and Context

The advent of Industry 5.0 has shifted the focus of manufacturing from efficiency and automation to the adoption of human-centered production systems characterized by collaboration, adaptability, and inclusiveness [1]. Despite widespread automation, many assembly operations need active human interaction or are completely manual due to complexity of manipulations, low production volumes, individuality of the products and other factors. Manual assembly implies differences among the operators[2], which results in wrong assembly order, missing parts, wrong orientation of the parts and varied periods of production. Thus, human assembly affects the efficiency, accuracy and completeness of the assembled products.

Existing vision-based assistance systems typically address only one facet of this problem at a time: some focus on action or gesture recognition, others on object detection, others on trajectory-based inspection, and few combine these cues into a single reasoning [3–6]. Many also rely on wearable sensors or intrusive sensing hardware, or are validated only under controlled laboratory conditions rather than realistic industrial settings with occlusions, variable illumination, shadows, and reflections. Furthermore, systems that do provide operator guidance generally offer a single, fixed level of assistance, with no mechanism to adapt the amount of information shown to operator expertise. A computer vision system combined with artificial intelligence that integrates stage recognition, verification and adaptive guidance within a single, non-intrusive RGB-based framework can address these gaps, decreasing errors, training novice operators, and increasing consistency without wearable devices. The following thesis describes the development of such a system, applied to the assembling and disassembling of a pneumatic piston.

1.2 Host Organisation

Fraunhofer Italia Research is an applied research organisation that develops and transfers emerging technologies to industrial applications. The internship was carried out within the Inclu5ion project, which focuses on developing human-centred and inclusive workplaces through assistive Industry 5.0 technologies. The project aims to support workers with physical or cognitive disabilities by adapting industrial work environments to different worker needs. Within this context, the internship focused on the development of a vision-based system for monitoring manual assembly operations and providing adaptive operator guidance. The developed system contributes to the project objective of creating technology supported workstations by using RGB-based perception to recognise assembly actions, verify the execution of the procedure, and provide configurable assistance to the operator.

1.3 Research Question and Original Contributions

Building on the gap identified above in [Section 1.1](#), this work addresses the following research question: *How can a single-camera RGB vision system robustly recognize manual assembly stages, verify their correctness, and provide adaptive real-time guidance, without relying on wearable sensors, multi-modal hardware, or task-specific 6-DOF pose estimation?* The original contributions of this internship are:

- A **hybrid ROI interaction trigger** that fuses motion, hand-presence and trajectory evidence into a single finite-state interaction model, overcoming the individual failure modes of each cue in isolation.
- A **comparative evaluation of object detection architectures** (YOLO family [7, 8], RTMDet [9], RF-DETR [10]) under realistic assembly conditions, including a real+synthetic dataset construction pipeline built from CAD models.
- A **reformulation of object pose estimation as discrete orientation classification**, motivated by the empirical failure of DON[11] and FoundationPose[12] on symmetric, reflective, small-scale industrial parts.
- A **two-level FSM architecture** (MotionFSM / StageFSM) that integrates procedural knowledge, object detection, orientation classification, and DTW-based trajectory validation into a single stage-verification mechanism, supporting both ordered and unordered stage groups.

- A **configurable adaptive guidance interface** (full/medium/no-assistance) decoupled from the underlying perception and verification pipeline, enabling controlled experimental comparison of assistance levels.
- An **experimental evaluation** of the integrated system on a real assembly/disassembly task, combining technical performance measures with user-level trajectory and questionnaire data.

1.4 Internship Objectives

The main objective was to analyse, understand and develop a vision-based system capable of automatically recognizing assembly actions performed by a human operator. In order to achieve the above objective the following goals were set: (i) Tracking of hand movements in real time, (ii) Detection of assembly components in real time, (iii) Identification of the current assembly stage, (iv) Verification of correctness and completeness of the assembly sequence, (v) Orientation check of the assembled parts, (vi) Understanding interactions with different workspace regions, (vii) Providing real time adaptive guidance, (viii) Experimental Evaluation of the system.

1.5 Main Activities

With the main objective and goals defined, the main activities carried out during the thesis include: studying existing literature related to assembly guidance, understanding the assembly process of a pneumatic piston, defining stages schemes and workflow, defining the software architecture, developing ROI based interaction detections, implementing motion and stage FSMs, collecting and annotating real datasets, creating synthetic datasets from CAD models, investigating different object detection algorithms, evaluating multiple pose estimation approaches, integrating different software modules, developing user interfaces with multiple assistance levels, performing experimental evaluation and user studies and finally statistical analysis of the results.

1.6 Expected Results

The system is expected to achieve object detection precision and recall suitable for reliable stage verification, correctly identify the active assembly stage with a low rate of false or missed transitions, correctly classify component orientation for the parts requiring it, and operate within a processing latency compatible

with real-time operator feedback. At the user level, the adaptive guidance is expected to reduce task completion time and trajectory inefficiency relative to the no-assistance condition, with these effects reflected in the questionnaire-based usability results reported in [Chapter 11](#)

1.7 Structure of the Thesis

This report is organised as follows. [Chapter 2](#) introduces the problem and system requirements. [Chapter 3](#) presents the relevant literature. The development process is described in [Chapters 4 to 8](#). The interface and evaluation are presented in [Chapters 9 and 11](#). Finally, [Chapters 10 and 12](#) discuss the final integrated system and the conclusions.

Chapter 2

Problem Definition and Requirements

2.1 Manual Assembly Process

Although there has been significant progress in computer vision techniques in manufacturing processes, many challenges remain in achieving a real-time understanding of the manual assembly procedure [5, 6]. The case study considered in this work is the assembly and disassembly of a pneumatic piston [Figure 2.1](#). The assembling process comprises various stages that have to be performed in a specific manner, where every part should be selected, aligned, assembled, and fastened in a proper way. Some subsequences are also present in which the order of assembly stages doesn't matter. For instance, four fastening screws can be tightened in any sequence, or each screw can be individually placed and fully secured before moving to the next. The operator continuously performs actions like picking, placing, tool utilization, fastening, and inspecting. Since the process is deterministic, each stage involves expected objects, expected workspaces, and expected hand motions, which makes it a good candidate for automatic monitoring through computer vision. The thesis focuses on recognizing these assembly stages in real time, verifying the correctness of each executed stage while providing operator guidance.

2.2 Challenges in Manual Assembly

Manual assembly involves a wide range of challenges, which are discussed below:

- **Human variability:** Different users may perform the same task in different ways, with considerable variations in execution speed, hand trajectories, grasping styles, and hand dominance.[2].

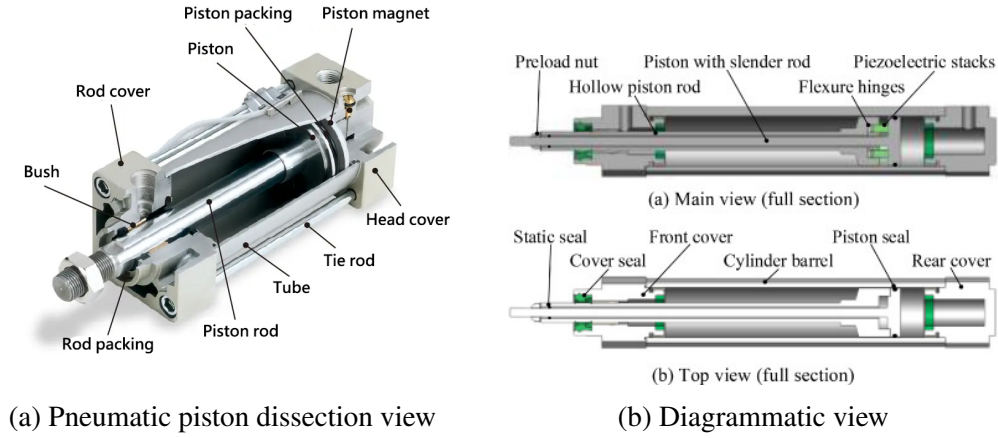


Figure 2.1: Components of the pneumatic piston assembly.

- **Visual challenges:** Occlusions caused by hands, small and reflective metal parts, similar-looking components, lightening variation, shadows, temporary object occlusions[13].
- **Temporal challenges:** Stages have no fixed duration[6], some users hesitate while others skip quickly[14], stage transitions are not explicitly defined, and the order of some stages may be interchangeable.
- **Interaction challenges:** Hands frequently pass over nearby regions, workspace contains multiple adjacent components, detecting intentional interaction is harder than detecting simple motion[6].
- **Real-time constraints:** The system must process continuous live video, decisions should be available with minimal delay, feedback must be provided while the user is still performing the task.

It is quite evident that these challenges require combining multiple vision techniques instead of relying on a single detector for achieving a proper vision based validation and guidance[3, 5, 6].

2.3 Research Problem

Building on the research question outlined in [Section 1.3](#), this thesis investigates how a vision-based system can robustly recognize manual assembly stages and provide adaptive real-time guidance under realistic industrial conditions. Formulating this central objective leads to several key sub-investigations: establishing reliable mechanisms for stage transition detection, developing robust criteria for object interaction verification, benchmarking object detection and pose estimation architectures for real-time deployment, designing multi-level assistance modes

that adapt without disrupting workflow, and defining quantitative metrics to evaluate operator learning progression and system usability. Together, these targeted sub-objectives define the functional constraints and parameters governing system design, providing a structured methodological path for technical implementation and experimental evaluation.

2.4 Functional and Non-Functional Requirements

This section defines the operations the proposed system must provide (functional requirements) and the attributes it must exhibit while providing them (non-functional requirements). Each requirement below functional, or not is given a measurable, falsifiable acceptance threshold, together with the evaluation metric from [Section 2.5](#) that verifies it. Where a numeric target could not yet be derived from pilot data, a provisional value grounded in related literature or comparable industrial systems is used and flagged accordingly.

Table 2.1: Functional requirements and acceptance criteria

ID	Functional Requirement	Re-	Acceptance Criterion	Linked (Section 2.5)	Metric
FR1	Acquire RGB images from the camera.		Continuous acquisition at ≥ 15 FPS with no more than 1 dropped frame per 1000.	System operation / effective FPS	
FR2	Detect the operator's hands.		Hand position available in $\geq 90\%$ of frames in which the hand is at least 50% visible (MediaPipe confidence ≥ 0.7).	Hand/trajectory availability	
FR3	Detect the assembly components required by the task.		Precision ≥ 0.90 and recall ≥ 0.85 on the held-out test set (provisional target, consistent with reported performance of comparable detectors in industrial small-part detection [13]).	Detection precision, recall, F1	

Continued on next page

Table 2.1 – continued from previous page

ID	Functional Requirement	Re-	Acceptance Criterion	Linked	Metric
				(Section 2.5)	
FR4	Recognize the current assembly stage.		Stage recognition accuracy $\geq 90\%$, with invalid (out-of-sequence) transitions $\leq 5\%$ of total transitions (provisional target, informed by comparable FSM/vision-based stage-verification systems [5]).	Stage recognition accuracy	
FR5	Detect operator interaction with pre-defined workspace regions.		Interaction detection accuracy $\geq 90\%$, with false-positive rate $\leq 10\%$ under nominal lighting.	Interaction detection accuracy	
FR6	Validate the assembly sequence.		100% of accepted transitions conform to the predefined procedure (ordered/unordered constraints); zero tolerance on invalid transitions being accepted.	Valid/invalid transition rate	
FR7	Verify the orientation of task-specific components.		Orientation classification accuracy $\geq 95\%$ on components requiring orientation verification.	Pose classification accuracy	
FR8	Provide configurable assistance levels.		All three assistance modes correctly reflect the active configuration in 100% of mode switches during testing.	Mode-transition correctness	
FR9	Generate feedback for stage completion and errors.		Correct feedback message displayed within ≤ 1 frame-processing cycle of the corresponding system-state change.	Feedback correctness	
FR10	Record relevant experiment data and system outputs.		100% of required fields logged for every trial, with zero missing entries.	Data completeness	

Continued on next page

Table 2.1 – continued from previous page

ID	Functional Requirement	Re-	Acceptance Criterion	Linked (Section 2.5)	Metric
FR11	Compute trajectory-based performance metrics.	met-	All defined trajectory metrics (DTW similarity, path efficiency, pause ratio, jerk index, micro-stop count) computable for $\geq 95\%$ of valid trials.	Trajectory completeness	data
FR12	Export experiment logs for subsequent analysis.		Logs exportable in structured format (CSV/JSON) with 100% field integrity, verified against source data.	Export completeness	

Table 2.2: Non-functional requirements and acceptance criteria

ID	Non-Functional Requirement		Acceptance Criterion	Linked (Section 2.5)	Metric
NFR1	Real-time operation.	opera-	End-to-end processing latency ≤ 200 ms per decision cycle (≥ 5 effective FPS for the decision pipeline), consistent with latency thresholds reported for interactive assembly-assistance systems in the literature [5] (provisional).	Processing latency / effective FPS	
NFR2	Robustness to illumination variations and shadows.		No persistent (> 2 s) false stage transition under simulated lighting disturbance ($\pm 30\%$ brightness change, added shadow); false-trigger rate $\leq 5\%$ of trials under disturbance.	False-trigger rate under disturbances	

Continued on next page

Table 2.2 – continued from previous page

ID	Non-Functional Requirement	Acceptance Criterion	Linked (Section 2.5)	Metric
NFR3	Robustness to operator variability.	Stage and interaction recognition accuracy remains $\geq 85\%$ across all tested operators, with no single operator falling more than 10 percentage points below the group mean.	Stage and interaction accuracy across users	Stage recognition and interaction accuracy across users
NFR4	Modular software architecture.	Any single perception module (detector, classifier, trigger) replaceable and no changes to downstream FSM/interface code.	Module-level integration testing	
NFR5	Adaptability to different assembly procedures.	New stage definitions addable via configuration file only, with zero source-code modification, verified on at least one alternative stage sequence (e.g. dismantle vs. assembly mode).	Configuration effort / successful task re-configuration	
NFR6	Configurability of workspace and stage definitions.	Workspace ROIs and stage constraints modifiable entirely through JSON configuration, verified by successfully reconfiguring at least 3 regions without code changes.	Configuration test	
NFR7	Maintainability and reusability.	$\geq 70\%$ of core processing modules (detector, FSMs, trajectory module) reused unmodified between assembly and dismantle configurations.	Code/module reuse assessment	

Continued on next page

Table 2.2 – continued from previous page

ID	Non-Functional Requirement	Acceptance Criterion	Linked (Section 2.5)	Metric
NFR8	Usable visual guidance.	Mean usability questionnaire score $\geq 4/5$ (Likert scale) for clarity of guidance, without reported task interference.	Usability questionnaire	
NFR9	Efficient use of computational resources.	GPU inference latency ≤ 100 ms per frame on the target GPU workstation, with peak memory usage within available hardware limits.	Inference latency / GPU utilisation	

2.5 Evaluation Criteria

The proposed system is evaluated according to the functional and non-functional requirements defined in [Section 2.4](#). The evaluation combines technical performance measures with user-level measures obtained during the experimental trials, refer [Table 2.3](#).

The object detection component is evaluated using standard metrics, including precision, recall and F1-score. Stage and interaction verification are evaluated using correct and incorrect transitions, false activations, missed detections and stage completion accuracy. The pose classification module is evaluated according to its classification accuracy.

System-level performance is assessed through processing latency, effective frame rate and, where available, GPU utilisation. User performance is evaluated through task duration and trajectory-based measures, including Dynamic Time Warping similarity, path efficiency, pause ratio, jerk index and micro-stop count. Finally, the usability of the assistance interface is assessed through the questionnaire responses collected during the user experiments.

Table 2.3: Evaluation criteria and corresponding requirements

System Module/Function	Mod-	Metrics	Requirements
Object Detection		Precision, recall, F1-score, false detections	FR3
Hand and Interaction Detection		Hand availability, interaction accuracy, false-trigger rate	FR2, FR5, NFR2
Stage Recognition		Stage accuracy, false transitions, missed transitions, completion accuracy	FR4, FR6, NFR3
Pose Verification		Classification accuracy and inference time	FR7
Assistance Interface		Mode correctness, feedback correctness, questionnaire responses	FR8, FR9, NFR8
Real-Time Performance	Perfor-	Processing latency, effective frame rate, GPU utilisation	NFR1, NFR9
Trajectory Performance	Perfor-	DTW similarity, path efficiency, pause ratio, jerk index, micro-stop count	FR11
Data Management		Data completeness and successful export	FR10, FR12
System Adaptability		Configuration effort and successful reconfiguration	NFR4, NFR5, NFR6, NFR7
User Experience		Perceived usefulness, ease of understanding, preferred assistance level and questionnaire responses	NFR8

Chapter 3

Literature Review

3.1 Industrial Assembly Assistance Systems

Manual assembly remains important where flexibility, product variety, and frequent changes in production make full automation difficult to justify. This flexibility, however, introduces variability: Folgado et al. [2] showed significant differences between operators in task completion speed even for identical assembly tasks, motivating systematic monitoring of manual operations.

The rise of Industry 4.0/5.0 has increased interest in technologies that support rather than replace operators. Mattsson et al. [15] proposed a cognitive automation strategy in which assistance level adapts to the operator's task and knowledge, emphasising that *when* and *how* information is presented matters as much as its content.

Sensing approaches for assembly monitoring range from wearable and instrumented methods (eye tracking, IMUs, RFID, force sensors) to vision-based observation. Haslgrübler et al. [16] and Kawase and Hashimoto [4] used wearable eye-tracking and gaze-hand coordination to characterise operator skill, but such equipment adds complexity and can hinder deployment. Vision-based systems avoid this by observing the operator, tools, and workspace from a single external viewpoint. Zhang et al. [5] proposed a learning-based stage-verification system that combines multiple visual cues to identify assembly states and detect execution errors, using only a camera-based setup; however, their system addresses stage classification and error detection only, without operator guidance or skill assessment a gap this thesis targets directly (§3.7).

These studies indicate that assembly assistance spans several connected functions: human-activity understanding, object/workspace understanding, procedural verification, and operator feedback.

3.2 Human Action and Hand-Object Interaction Recognition

Assembly actions are inherently temporal and object-dependent: the same hand motion can correspond to different operations depending on what is being manipulated. Skeleton-based representations offer a compact, appearance-invariant motion descriptor but do not by themselves describe the manipulated object.

Aganian et al. [3] address this by augmenting skeleton-based action recognition with detected object centres, showing on the IKEA ASM dataset that object context improves recognition over skeleton information alone. Zhou et al. [6] instead model temporal hand-trajectory transitions between predefined workspace regions, using DTW and dwell time to identify missed or incorrect operations demonstrating that temporal evolution, not single-frame object detections, carries the relevant evidence. Kawase and Hashimoto [4] further show that hand-movement patterns change measurably with operator experience, indicating trajectories carry proficiency information beyond action identity.

In practice, hand motion is noisy: trajectories overlap spatially, operators pass through regions without interacting, and occlusion interrupts tracking. Object identity, workspace location, and temporal context are therefore needed alongside motion to disambiguate genuine interaction. Aganian et al. and Zhou et al. together illustrate this progression from motion recognition toward hand-object interaction understanding, motivating the combination of motion, object, and temporal evidence rather than reliance on any single descriptor.

3.3 Object Detection for Assembly Monitoring

Object detection supplies the spatial evidence needed to identify components and tools, and to verify that the correct part is involved in an operation. Assembly environments are harder than standard detection benchmarks: parts are small, visually similar, reflective, and frequently occluded by the operator’s hands [13].

One-stage detectors such as the YOLO family [7, 8] remain the standard choice for real-time efficiency. Transformer-based detectors offer an alternative formulation: DETR [17] removed hand-designed components from detection via end-to-end set prediction, and Grounding DINO [18] extends this toward open-set, text-grounded detection. These are distinct from DINOv2 [19], a self-supervised representation model rather than a detector, though its features are used as backbones in later architectures. RF-DETR [10] applies neural architecture search to optimise detection transformers along the accuracy-latency trade-off

relevant to real-time deployment.

A bounding box, however, does not describe orientation: two detections of the same part may differ in a way that determines whether the next assembly step can proceed correctly. This motivates considering pose and orientation separately.

3.4 Object Pose and Orientation Estimation

Full 6D pose estimation recovers three translational and three rotational components from visual observation, via geometric (PnP, keypoint), correspondence-based, or learning-based methods [20].

Dense Object Nets (DON) [11] learn dense visual descriptors providing pixel-level correspondence across observations of an object, useful for manipulation; Adrian et al. [21] later proposed a more robust training procedure for DON models. FoundationPose [12] instead targets generalisation to novel objects, combining large-scale synthetic training with a transformer architecture to support both model-based and model-free 6D pose estimation and tracking.

These methods target the general pose-estimation problem, but many assembly operations only require confirming that a part is in one of a small set of valid orientations not its continuous 6D pose. This suggests reformulating orientation verification as a discrete classification problem when the downstream decision does not need full pose information, a distinction taken up in [Chapter 7](#).

3.5 Assembly Stage and Procedure Verification

Beyond recognising individual actions or objects, a monitoring system must determine which stage is active, whether required operations were completed or skipped, and whether the observed sequence matches the intended procedure.

Learning-based approaches can model this directly from data, as in Zhang et al.’s [5] stage-verification system (§3.1); such methods are flexible but depend on representative task-specific training data. Probabilistic alternatives include Hidden Markov Models: Berg et al. [22] use a layered HMM over human movement observations for assembly-subtask recognition. Finite-state machines instead encode procedural structure deterministically: Zhao et al. [23] apply an FSM to human-robot collaborative assembly, and hierarchical FSMs have been used to decompose and allocate collaborative tasks [24]. Their advantage is that procedural constraints are explicit rather than inferred from data.

DTW-based trajectory comparison provides a complementary temporal

cue: Kung and Chou [25] and Tang et al. [26] apply DTW to hand-trajectory/gesture data, and Zhou et al. [6] combine it with workspace regions and dwell time for assembly inspection. Trajectory similarity alone, however, does not establish procedural correctness different operations can share similar motion, and the same operation varies across operators. No single technique here learning-based recognition, HMMs, DTW, or FSMs simultaneously captures motion, object context, spatial interaction, and procedural validity; this motivates integrating them.

3.6 Related Trends: Egocentric, Hand-Object, and AR-Based Assembly Understanding

Recent work extends assembly understanding from isolated actions to structured, procedural activity. Assembly101 [27] is a large multi-view (static + egocentric) dataset for procedural activities, with annotations supporting action recognition, temporal segmentation, and notably mistake detection, directly relevant to stage verification. EgoProceL [28] targets unsupervised procedure/key-step learning from egocentric video across multiple demonstrations, without assuming a pre-defined procedural model.

Hand-object interaction (HOI) has emerged as a distinct sub-problem: AssemblyHands [29] provides high-quality 3D hand-pose annotations for egocentric assembly, while Shan et al. [30] propose a general-purpose model detecting hand location, side, contact state, and the contacted object’s bounding box directly from RGB a capability that could substitute for the ROI/hand-presence triggers explored in Chapter 5. As with skeleton-only motion, however, hand-contact information alone does not determine intended action without object and procedural context.

Augmented-reality (AR) research addresses the complementary problem of communicating system state back to the operator: reviews of AR-based assembly guidance [31] show widespread use of HMDs and projection to present task instructions in-situ. Such systems demonstrate the value of contextual assistance but typically decouple guidance presentation from continuous verification of the operator’s actual actions.

3.7 Comparative Analysis and Research Gap

Table 3.1: Comparison of representative approaches for assembly understanding.

Approach	Modality	Primary Task	Main Capability	Main Limitation
Aganian et al. [3]	Skeleton + object information	Action recognition	Object-augmented skeleton representation for incorporating object context into human action recognition	Does not explicitly address procedural verification
Zhou et al. [6]	RGB + hand trajectory	Procedure inspection	Combines object detection, hand trajectories and DTW-based temporal matching	Task-specific and dependent on reliable trajectory estimation
Zhang et al. [5]	RGB	Stage verification	Visual stage classification using multiple visual cues for assembly-state recognition	Task-specific training; guidance and skill assessment are not the primary focus
Berg et al. [22]	Human motion	Action recognition	Layered HMM-based modelling of temporal assembly actions	Does not provide an explicit representation of the complete assembly procedure
Zhao et al. [23]	Eye-hand information + FSM	Collaborative assembly	Uses explicit state transitions to represent assembly activities	Developed in a virtual collaborative setting rather than a standalone RGB system

Continued on next page

Table 3.1 – continued

Approach	Modality	Primary Task	Main Capability	Main Limitation
Assembly101 [27]	Multi-view RGB + 3D hand pose	Procedural understanding	Supports action recognition, temporal segmentation and assembly mistake detection	Primarily a benchmark and dataset rather than an integrated guidance system
EgoProceL [28]	Egocentric RGB	Key-step learning	Self-supervised learning for discovering procedural steps in instructional activities	Does not provide a real-time operator feedback loop
Assembly-Hands [29]	Egocentric RGB + 3D hand pose	Hand/activity understanding	Provides high-quality representations of hand-object interactions	Does not explicitly perform assembly-stage verification
Shan et al. [30]	RGB	Hand-object interaction detection	General-purpose detection of hand-object contact and interaction	Not explicitly integrated with assembly procedure logic
AR systems [31]	RGB / AR	Operator guidance	Provides contextual visual instructions within the physical workspace	Guidance is not necessarily coupled with continuous activity verification
This thesis	Overhead RGB	Detection, verification and guidance	Integrated interaction detection, orientation verification, FSM-based procedure modelling and adaptive visual guidance	Fixed viewpoint and task-specific assembly setup

Five gaps emerge from this comparison:

- **Integration of complementary evidence.** Object identity, hand motion, spatial location, and temporal activity are individually well studied but rarely combined into a single interaction decision.
- **Explicit procedure representation.** Learning-based stage recognition identifies visual states [5], and HMM/FSM approaches encode temporal structure [22, 23], but few systems explicitly represent permitted, unordered, and completion-defining transitions together.
- **Robust interaction detection.** Detection and tracking alone do not confirm an operation occurred; occlusion and transient movement require temporal confirmation rather than frame-level triggering.
- **Task-specific orientation verification.** General 6D pose estimation [12, 20] is often more than an assembly task requires; a discrete, task-specific orientation classification is frequently sufficient.
- **Monitoring-assistance connection.** AR systems [31] deliver guidance without continuous action verification, while monitoring research rarely closes the loop back to adaptive assistance.

These gaps point not to insufficiency of individual techniques, but to the absence of their integration across perception, interaction detection, procedural verification, orientation validation, and operator communication.

3.8 Positioning of the Proposed Work

This thesis is positioned as an integration effort: established perception techniques are combined with task-specific representations for assembly verification and guidance.

Four complementary information sources anchor the system: object detection for component identity/location; hand and trajectory information for operator-workspace interaction; region-based temporal evidence for meaningful-interaction detection; and an explicit procedural representation using ordered and unordered stage groups relating observed interactions to the expected sequence. Orientation verification is likewise formulated as task-specific classification rather than general 6D pose estimation, following the distinction argued in [Section 3.4](#).

The resulting framework is layered: raw visual observations become object/hand/ workspace information, which is combined into interaction and

temporal evidence, interpreted against the current stage, and used to drive operator feedback. This distinguishes the work from approaches limited to action recognition, detection, trajectory matching, or guidance alone (Table 3.1); its contribution is the integration of these capabilities into one modular, real-time, single-camera framework, developed and evaluated in the following chapters.

Chapter 4

System Design

4.1 Overall System Architecture

The main idea behind the proposed system is to use a single overhead RGB-D camera to monitor the assembly stages, collect the necessary information for skill assessment, and verify them successively as each stage is completed, all while simultaneously providing guidance to the operator based on their expertise. The system follows a modular architecture, refer [Figure 4.1](#), wherein each separate module deals with different aspects such as image acquisition, image processing, hand detection, motion analysis, stage verification, object recognition, guidance generation and interaction with the user. This approach was chosen because it provides us with the flexibility to change specific modules alone while keeping the rest of the system untouched, as several different algorithms were explored during the internship until the final one was chosen. In general terms, the system continuously takes images from the overhead camera, processes the necessary visual information, understands what the operator is doing, checks the present assembly step and gives necessary visual cues and feedback to the operator.

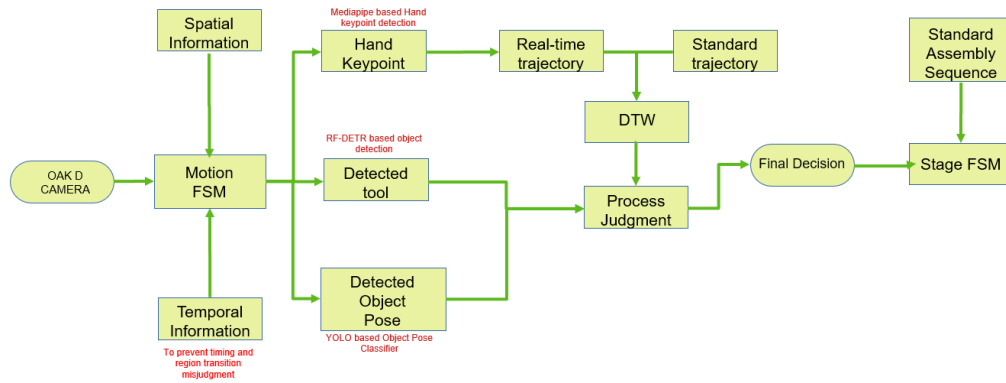


Figure 4.1: Overall Final Integrated System Architecture

4.2 Hardware and Software Architecture

A fixed workstation was first defined in which the assembly and dismantle operations of a pneumatic piston takes place. A luxonis OAK-D camera is placed overhead of the workstation in order to have a full top-down view of the assembly environment [Figure 4.2](#). OAK-D camera was chosen as it delivers a synchronized RGB and depth data, as the initial plan was to also explore several depth based algorithms. The entire workspace was further divided logically into different regions such as pick/drop area, assembly target area, tools area. These regions remain fixed through the experiments (yet configurable via a single configuration file) and were used as spatial references for interpreting operator and object interactions.



Figure 4.2: Camera placement

The software architecture was mainly built using the Python programming language and used open-source computer vision libraries. These included OpenCV[32] for image processing and visualization, MediaPipe[33] for hand detection, PyTorch[34] for deep learning inference, as well as some custom-developed software modules. Due to the increase in computational difficulty in later stages of development, deep learning inference was moved to a remote GPU server while the rest of the system ran on the local workstation. This allowed computationally intensive object detection networks to be run without impacting the performance of the user interface. These software modules are designed to exchange information via data structures that ensure they work independently

but exchange only the necessary information needed for further processing, refer [Figure 4.1](#).

4.3 Assembly Stage Definition

For automatic verification, the entire assembly of the pneumatic cylinder was broken down into a series of assembly stages, where each stage consists of a single operation that must be carried out sequentially. Special care should be taken as certain set of consecutive stages can be done in an interchangeable order while producing the correct output, therefore appropriate algorithmic handling should be implemented. Every step describes what action is expected, what component or tool should be used, what workspace areas are involved in the operation, and what conditions should be met for successful completion of the step. By describing the assembly process in terms of the ordered sequence of tasks, we are able to verify the operator's actions continuously. Having a structured representation also enables to provide simplified guidance to the operator by showing only the stage specific information, instead of presenting the whole assembly process at once. The stage definition serves as the basis of Stage FSM discussed in the later chapters, wherein transitions between stages is automated by using ROI interactions of the operators hands with the regions specified for a stage.

4.4 Workspace and Region Definition

To achieve proper monitoring of interactions carried out by the operators, the assembly workspace was split into a number of pre-defined Regions of Interest (ROIs), refer [Figure 4.3](#). Each region represents a certain physical position on the workbench, such as storage boxes for components, tools, intermediate assembly positions, and final assembly positions. In an assembly process, each particular stage has its own set of operations performed in its own defined regions, e.g., for stage 1, the bolt must be picked from the bolt box and assembled in target area and next pick should happen in the nut box. This spatial reasoning is exploited in the verification and segmenting sequential stages, discussed later in detail. Pre-defined polygonal ROIs were set up when configuring the system and saved in configuration files. In practice, the detected hands, objects, and virtual guidance layers are continually projected to these regions during the process. In this way, the system will be able to identify if the operator is working on the right spot for that particular assembly phase. One other advantage of having a

region-based implementation is the ability to provide visual guidance by allowing virtual projections and highlighted work areas to be rendered directly within the corresponding workspace regions.

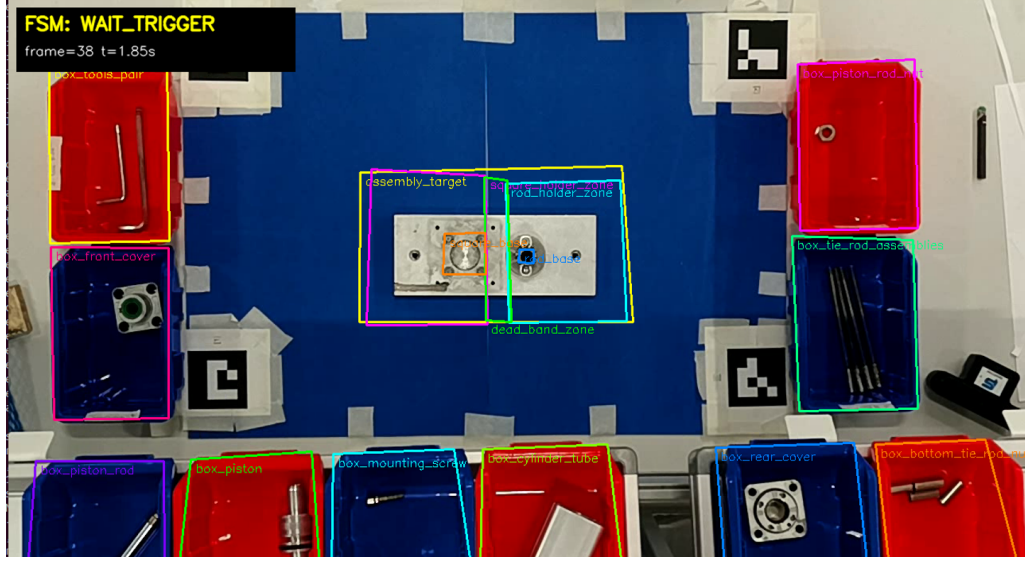


Figure 4.3: Regions of Interest (ROIs) in the workspace are color-coded as follows: **yellow** for tool box, **dark pink** for front cover, **violet** for piston rod, **lime green** for piston, **cyan** for mounting screw, **bright green** for cylinder tube, **dark blue** for rear cover, **orange** for bottom tie rod nut, **bluish green** for tie rod nut assemblies, and **bright pink** for piston rod nut.

4.5 Preliminary System and Motivation for Thesis Development

The thesis builds upon a preliminary prototype developed during a preceding internship to investigate the feasibility of vision-based monitoring for manual assembly and establish a modular software architecture. The prototype used an overhead RGB camera and MediaPipe for hand detection, with the workspace divided into predefined spatial regions to monitor hand movements and interactions. A preliminary finite-state machine represented the assembly stages and controlled their progression based on the observed interactions. The modular structure was designed to facilitate the later integration of additional perception and verification components.

However, the preliminary system relied mainly on hand motion and region-based interaction detection, making it sensitive to illumination changes, shadows, metallic reflections, and hand-detection failures. It also lacked explicit object awareness, preventing reliable identification of manipulated components

or verification of whether the correct object was used. Component-orientation estimation, comprehensive stage verification, and operator guidance were likewise not yet incorporated. Furthermore, the increasing computational requirements of more advanced vision models exposed limitations in the initial single-workstation configuration and motivated the adoption of distributed processing.

These limitations provided the starting point for the thesis development. The preliminary concepts were therefore revisited and substantially extended through improved hand and trajectory representation, explicit object and tool awareness, structured stage verification, component-orientation analysis, adaptive operator assistance, and a distributed processing architecture. The internship prototype consequently serves as the initial architectural foundation, while the thesis focuses on transforming it into a more robust and comprehensive framework for assembly monitoring and operator assistance.

Chapter 5

Stage Segmentation Development

5.1 Stage Segmentation Problem

The segmentation of assembly stages determines when one stage ends and the next begins. It therefore represents one of the crucial modules of the proposed framework, as the Motion FSM, Stage FSM, object verification, trajectory comparison, and adaptive guidance mechanisms all depend on it. The stage segmentation problem can be formulated as determining the current assembly state

$$S(t) \in \{A_1, A_2, \dots, A_n\} \quad (5.1)$$

where A_i denotes the i^{th} assembly stage among the n possible stages, and $S(t)$ denotes the assembly stage active at time t . A transition between two consecutive stages occurs whenever sufficient evidence exists that the operator has completed the current action and intentionally moved to the next assembly step. Mathematically,

$$S(t) \neq S(t - 1) \quad (5.2)$$

only if

$$E(t) \geq \tau \quad (5.3)$$

where $E(t)$ denotes the accumulated evidence supporting completion of the current stage and τ is a predefined decision threshold.

A number of independent triggering processes were studied for calculating the value of $E(t)$. Experimental evaluation showed that none of these methods could individually guarantee robust performance in practical scenarios. This finding motivated the development of a hybrid triggering system, which was ultimately adopted as the final solution.

5.2 ROI Motion Trigger

The first technique explored uses background subtraction to estimate motion occurring within predefined Regions of Interest (ROIs). A reference background image was computed during system initialization using the median of the first N calibration frames,

$$B(x, y) = \text{median}\{I_1(x, y), I_2(x, y), \dots, I_N(x, y)\}. \quad (5.4)$$

For compensating illumination variations, the current image was normalized to roughly equalize illumination. Motion was determined based on the absolute difference calculation.

$$D(x, y) = |I_t(x, y) - B(x, y)|. \quad (5.5)$$

Pixels exceeding a predefined threshold T were classified as motion,

$$M(x, y) = \begin{cases} 1, & D(x, y) > T \\ 0, & \text{otherwise.} \end{cases} \quad (5.6)$$

For each workspace region R_i , the motion score was calculated as

$$mscore_{R_i} = \frac{\sum_{(x,y) \in R_i} M(x, y)}{|R_i|}, \quad (5.7)$$

where $|R_i|$ denotes the area of the region.

Although computationally efficient and effective, during development trials, this was qualitatively observed to produce false triggers and incorrect stage transitions, particularly under lighting change and shadow. This was not captured as a recorded quantitative failure rate at this development stage (same is the case for the other individual methods discussed from 5.3 to 5.6). The threshold values used in this section are listed in Appendix B, Table B.1.

5.3 Color-Based Trigger

The second approach involved trying to determine interactions using the color variations occurring within the different workspace regions. To reduce the influence of illumination, each ROI was first converted from an RGB to an HSV color space. A median HSV color value for each ROI was then determined and compared against a baseline value that was obtained during system initialization.

The color difference was calculated as

$$\Delta = \sqrt{w_h \Delta H^2 + w_s \Delta S^2 + w_v \Delta V^2}, \quad (5.8)$$

where H, S, V represent hue, saturation, and value, and w_h, w_s, w_v are weighting coefficients (during trials the weights were set as 0.5, 1.0 and 1.0 respectively). A trigger occurred whenever

$$\Delta > \tau_c. \quad (5.9)$$

Despite being computationally cheap, the method was not feasible for use in industrial settings since metallic objects could cause color variations due to reflections and shadowing. Refer to Appendix B, Table B.1 for actual threshold values.

5.4 Depth-Based Trigger

As the OAK-D sensor is capable of providing synchronized depth information, an analysis of depth-based interaction detection was also explored. A depth difference image was computed as

$$\Delta Z = |Z_t - Z_{bg}|, \quad (5.10)$$

where Z_t represents the current depth map and Z_{bg} the reference background. Regions exceeding a depth threshold (cross-reference Appendix B, Table B.1 for τ_d) were considered active when

$$\Delta Z > \tau_d. \quad (5.11)$$

Theoretically speaking, this approach is, color and lighting independent. In practice, however, certain limitations have been observed. Depth measurement has been affected by metal components and the reconstruction of small objects has proven to be problematic. Furthermore, large depth variations within the workspace resulted in noisy depth measurements. Occlusions could also lead to missing or unreliable depth data.

Consistent with these limitations, and despite the OAK-D camera having been selected in Chapter 4 specifically for its synchronised RGB-D output, the depth trigger is disabled by default in the final configuration. The hardware's depth channel is therefore unused by the deployed interaction-detection module; this is revisited for pose estimation in Chapter 7, where the same reflective, small-part conditions are shown to limit depth-based methods there as well.

5.5 Hand-Presence Trigger

An alternative approach involved checking whether the hand of the operator was within the expected workspace region using MediaPipe hand detection (Cross-reference Appendix B, Table B.5 for the MediaPipe thresholds.). Rather than using a single hand landmark, an anchor point was computed as the midpoint between the wrist landmark and the index-finger MCP landmark provided by MediaPipe. Let W_t and I_t denote the corresponding wrist and index-finger MCP coordinates at time t . The hand position was therefore defined as

$$H_t = \frac{W_t + I_t}{2} = (x_t, y_t) \quad (5.12)$$

The interaction condition becomes

$$H_t \in R_i. \quad (5.13)$$

This idea was easy to implement and did not require heavy computations. On the other hand, this solution relied solely on accurate hand detection, which was not always possible when using MediaPipe. Specifically, MediaPipe was sometimes unable to detect partially occluded hands, hands holding a tool, or hands near metal objects.

5.6 Trajectory-Based Interaction Evidence

In contrast to frame analysis, which involves the examination of separate individual frames, trajectory analysis is concerned with temporal changes in hand movement. A trajectory is represented as

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}. \quad (5.14)$$

For two consecutive trajectory points, the travelled distance is given by

$$\Delta d_i = |(x_i, y_i) - (x_{i-1}, y_{i-1})|, \quad (5.15)$$

and the corresponding instantaneous speed is computed as

$$v_i = \frac{\Delta d_i}{\Delta t_i}, \quad (5.16)$$

where $\Delta t_i = t_i - t_{i-1}$ is the time interval between the two consecutive trajectory points. The motion direction relative to the target region is estimated from the

angle between the velocity vector ($\mathbf{v}_i$) and a reference vector ($\mathbf{r}_i$) pointing from the current hand position toward the target region. Specifically,

$$\theta_i = \cos^{-1} \left(\frac{\mathbf{v}_i \cdot \mathbf{r}_i}{|\mathbf{v}_i| |\mathbf{r}_i|} \right). \quad (5.17)$$

Here, the trajectory starts when the recording condition is activated, and the initial trajectory point defines the starting position of the recorded motion. The reference vector ($\mathbf{r}_i$) is therefore defined from the current hand position toward the relevant target region rather than from an unspecified global reference point. The use of trajectory information allows for an important distinction between intention and random movement of the hand. Yet trajectory information alone is not enough to identify an actual interaction with an object, especially if the hand just moves past an area of interest (transient hand movements).

5.7 Limitations of Individual Trigger Methods

Experimental trials were conducted after the implementation of each method. The results showed that every triggering method had its own strengths, but also major weaknesses when used independently. Motion-based triggering is fast, but still prone to environmental influences. Colour-based methods are affected by reflections and variations in lighting conditions. Depth-based segmentation is influenced by noisy sensors and inaccurate reconstructions of metallic objects. Triggering based on hand presence relies heavily on accurate hand tracking, whereas trajectory-based triggering does not provide explicit proof of the physical interaction.

Table 5.1: Comparison of Feature Detection Methods

Method	Fast	Robust	Lighting	Occlusion	Reliability
Motion	✓	Medium	Poor	Good	Medium
Colour	✓	Low	Poor	Good	Low
Depth	Medium	Medium	Good	Poor	Medium
Hand Presence	High	Medium	Good	Poor	Medium
Trajectory	Medium	High	Good	Medium	High

Based on the above results [Table 5.1](#), it can be concluded that none of the methods alone was robust enough for stage segmentation thus motivating the development of a hybrid approach.

5.8 Hybrid ROI Interaction Trigger

The final interaction trigger implemented evaluates motion, hand-presence, and trajectory evidence as a sequence of gate conditions, each with an explicit, configured threshold, instead of combining cues into a single weighted score. This is a deliberate design choice: a weighted sum can mask which cue actually caused a transition, whereas a gated combination makes the reason for every transition explicit and independently debuggable. Given how frequently individual cues failed in isolation (§ 5.7), this transparency was considered essential. Each gate condition is the direct formalisation of one of the single-cue methods evaluated earlier in this chapter, hardened against the specific failure mode identified for that cue.

Motion Evidence: The motion gate formalises the ROI motion cue of § 5.2. A workspace region is considered active when the fraction of pixels exceeding the pixel-level intensity-difference threshold within the region exceeds a region-activation threshold, computed against a rolling baseline. To address the false-activation failure mode of the single-cue version identified in § 5.2 where broad arm movement or lighting change triggers the cue without a genuine localised interaction a global-motion suppression mechanism is applied on top of the baseline score. For each frame, the frame-wide motion score is computed as the mean motion over the entire workspace:

$$m_{\text{global}} = \frac{1}{|W|} \sum_{(x,y) \in W} M(x, y), \quad (5.18)$$

where W denotes the full workspace area. A region R_i is then considered globally suppressed if the frame-wide motion exceeds a suppression threshold τ_{gs} :

$$m_{\text{global}} > \tau_{\text{gs}}. \quad (5.19)$$

Under this condition, the region’s motion score is discounted by a relative factor α_{gs} , giving the suppressed score:

$$m'_{R_i} = \alpha_{\text{gs}} \cdot mscore_{R_i}. \quad (5.20)$$

However, a region whose own local motion is unusually strong exceeding a strong-motion threshold τ_{strong} is exempted from full suppression. In that case, a

softer relative factor α_{strong} (with $\alpha_{\text{strong}} > \alpha_{\text{gs}}$) is applied instead:

$$m'_{R_i} = \begin{cases} \alpha_{\text{strong}} \cdot \text{mscore}_{R_i}, & \text{mscore}_{R_i} > \tau_{\text{strong}} \text{ and } m_{\text{global}} > \tau_{\text{gs}}, \\ \alpha_{\text{gs}} \cdot \text{mscore}_{R_i}, & \text{mscore}_{R_i} \leq \tau_{\text{strong}} \text{ and } m_{\text{global}} > \tau_{\text{gs}}, \\ \text{mscore}_{R_i}, & \text{otherwise.} \end{cases} \quad (5.21)$$

A region is finally declared active when its suppressed score exceeds the region-activation threshold (τ_{score}):

$$m'_{R_i} > \tau_{\text{score}} \quad (5.22)$$

This two-tier structure directly extends the baseline motion cue of § 5.2: the strong-local-motion exemption allows a genuine pick inside a source box to survive even when the rest of the frame is busy, while the suppression discount prevents weak incidental motion from triggering during broad arm movement. The configured values for (τ_{gs}) , (α_{gs}) , (τ_{strong}) , (α_{strong}) , and (τ_{score}) are listed in Appendix B, Table B.1.

Hand-Presence Evidence: The hand-presence gate formalises the hand-position cue of § 5.5, in which the hand anchor was tested for membership inside the expected workspace region. The single-cue version was found to be vulnerable to momentary tracking loss. To address this, the gate requires the hand anchor to remain inside the relevant region for a minimum number of consecutive frames before the evidence is considered valid, rather than triggering on a single frame. The gate is additionally suspended for an initial warm-up period after system start, to avoid spurious activations caused by detector boot noise and startup blur. The configured values for the consecutive-frame requirement and the warm-up duration are listed in Appendix B, tables B.1, B.3 and B.5.

Trajectory/Temporal Evidence: The trajectory gate formalises the temporal evidence of § 5.6, where trajectory analysis was used to distinguish intentional movement toward a region from random passing motion. The trajectory gate operates as a forgiving counterpart to the hand-presence gate, evaluated in parallel. Instead of requiring a strictly consecutive run of valid hits, it accepts a smaller number of validated hits within a rolling temporal window. This allows a brief interruption a single lost frame during a pick to be tolerated without discarding the interaction evidence. The configured values for the hit count and the rolling window size are listed in B, tables B.1, B.3 and B.5.

Combination Rule and Relationship to the MotionFSM: A transition from the idle state to the recording state requires motion evidence AND (hand-presence evidence OR trajectory evidence). In other words, motion is necessary but not sufficient; once motion has fired, either of the two remaining cues can

independently confirm intent. When strict-motion-only gating is enabled (the default configuration), the system additionally requires that region-based motion hits rather than a hand-position fallback alone drive the trigger, preventing a noisy or delayed hand detection from independently re-triggering a stage without genuine motion backing it.

The resulting gate signal feeds directly into the MotionFSM of [Chapter 8](#), which is the component responsible for tracking the progress of the manipulation once an interaction has begun. Concretely: satisfying the combination rule above is the condition that causes the MotionFSM to leave its idle state and begin recording the manipulation. The MotionFSM then applies stage-specific policies expected object, expected source region, completion condition to decide when the interaction counts as a completed stage. The gate described here therefore answers the question "has the operator begun a meaningful interaction with a region?"; the MotionFSM answers "did that interaction fulfil the requirements of the current stage?". This separation keeps the perception side (interaction detection) decoupled from the procedural side (stage management), and is described from the stage-management perspective in [Section 8.1](#). The contract validator enforces spatial constraints on top of the gate signal, rejecting interactions that occur in the wrong region for the current stage.

During development trials, the gated combination described above was observed to produce fewer spurious activations than the motion trigger alone particularly for transient hand movements near, but not committing into, a region boundary, and for brief MediaPipe tracking dropouts during an otherwise valid pick. These observations were not captured as recorded quantitative metrics at this development stage; they are reported here as qualitative development insights that motivated the final design.

Chapter 6

Object Detection Development

Building on the interaction detection of [Chapter 5](#), which establishes when an operator interacts with a workspace region, this chapter develops the object detection module required to verify which component is being manipulated. Several practical requirements were considered when selecting an object detector. These requirements guided the evaluation of different detection approaches. Some of these considerations are; the detector should correctly detect different similar looking small metallic components, it should run fast enough for real time executions, and finally it should be integrated easily into the system architecture and provide stable detections that could be used for further verification of stages and pose classification.

6.1 Initial Prompt-Based Detection

Before building a dedicated object detector, prompt-based object detection was explored to determine whether a sufficiently accurate solution could be achieved without the effort of preparing a labelled dataset. This approach used DINOv2[19], a self-supervised vision model that produces dense feature embeddings rather than object detections directly, as the underlying representation. The pipeline worked as follows: a small set of reference (prompt) images was collected for each component, each showing the object cropped and isolated; DINOv2 was used to extract a feature embedding for each reference crop. At inference time, a query frame was scanned by extracting embeddings from candidate image patches or a sliding set of regions and each candidate region's embedding was compared against the stored reference embeddings using a similarity measure (e.g. cosine similarity). A candidate region was localized as belonging to a given object class when its embedding was sufficiently similar to that class's reference embedding(s), with the highest-similarity match determin-

ing the predicted location and identity. In effect, this repurposes DINOv2 as a similarity-based visual search mechanism rather than a trained detector: it never learns object-specific bounding boxes or class boundaries, but instead relies entirely on nearest-neighbour matching against a small, manually chosen set of reference images. This method was able to detect objects only with a close-up view, with objects fully visible and fewer reflections and shade. This is because the parts of pneumatic piston are very similar and small in size, so occlusions greatly hinder the accuracy of detections. The prompt based detector was also highly computationally demanding making it hard for real time implementation, despite caching the input prompt images. The choice of reference prompt images also highly affected the performance of the detector. For this reason, it was not possible to utilize it as a practical detector in the proposed framework as the system built should be robust to viewpoint changes, illumination, and occlusions. Thus, DINOv2 was not chosen as the final detection algorithm.

6.2 Dataset Preparation

6.2.1 Real Dataset Preparation & Annotation

Since the parts of the pneumatic piston are task-specific, refer [Figure 6.1](#), and given the limitations already observed with the DINOv2 prompt-based approach in [Section 6.1](#), a custom-trained, task-specific detector was pursued rather than further evaluating additional off-the-shelf foundation models. This narrowed the available options to building a task specific object detector, for which we need to prepare the training dataset. Data collection was performed using the overhead RGB camera on the experiment stand. The images obtained encompassed different configuration, lighting, arrangements of the objects, and partial occlusions. The dataset consists of all the elements used in the pneumatic cylinder assembly, including the structural components, fasteners, and assembly tools. To speed up the annotation process, first pre-annotations were done with DINOv2 prompt image based detector, followed by manual label corrections. Finally, the annotations were saved in the YOLO object detection format, allowing for easy training of various detectors. Data augmentation was later used to generate further data for training.

6.2.2 Synthetic Data Generation

While the real data set was a great learning resource, some objects showed up only rarely and had few different poses. In order to improve the diversity of the

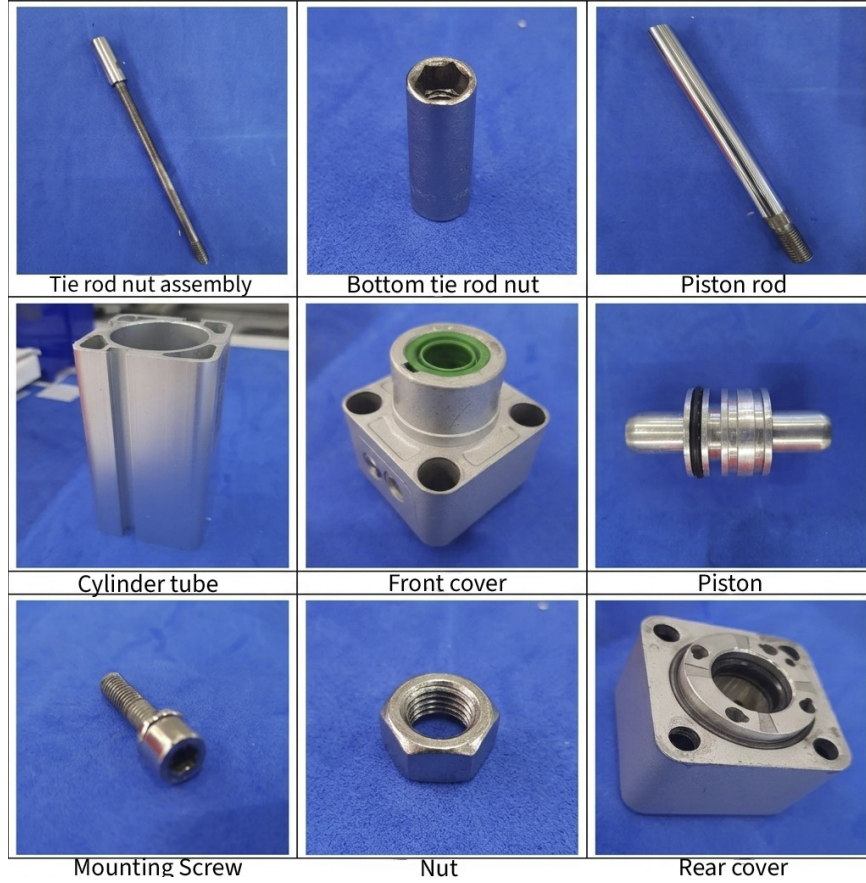


Figure 6.1: Parts of Pneumatic Piston

data set, a pipeline to generate synthetic data based on the CAD models of assembly components was built. Several scenes were created by changing object pose, scale, lighting, and background conditions, as shown in [Figure 6.2](#). As the label information is directly available during the rendering process, the generated images are perfectly annotated without any extra labeling efforts. The synthetic data set significantly improved increasing viewpoint diversity and detector robustness.



Figure 6.2: Representative synthetic image samples generated for training the object detection model.

An additional experiment was conducted to investigate how the proportion of synthetic data affected detector performance. The effect of synthetic data

supplementation varied by architecture (Table 6.1). For YOLOv8s, adding synthetic data at 20% of the real dataset size improved mAP50-95 from 0.7739 to 0.7818. For RF-DETR, the real-only configuration achieved the highest precision (0.9674), with synthetic supplementation slightly reducing precision but maintaining comparable mAP. This suggests that the benefit of synthetic data depends on the detector architecture and the specific data distribution.

6.3 YOLOv8 Models

YOLOv8[8] was chosen as the first baseline detector because of its wide usage and high-speed inferencing capability. The one-stage model structure allows for fast and real-time implementation without compromising the detection quality. The experiment showed that the performance of the method was good for large components. But there are cases of either missing small metallic parts or inaccurately locating them. Other yolo models such as YOLOv11 and YOLOv26 were also explored and they yielded the same results but the models were more computationally demanding.

6.4 RF-DETR

In contrast to traditional convolutional detectors, RF-DETR[10] poses object detection as a set prediction problem, and directly predicts the object set without needing any post-processing like Non-Maximum Suppression. Given an image I , the detector predicts

$$\mathcal{O} = \{(b_i, c_i)\}_{i=1}^N, \quad (6.1)$$

where b_i denotes the bounding box coordinates of the i^{th} detected object, c_i denotes its corresponding object class, and N is the number of predictions. Each prediction corresponds directly to one object instance. Even though RF-DETR demands more computational power compared to one-stage detectors, it provided more stable detection results especially during occluded and cluttered assembly cases.

6.5 RTMDet

RTMDet[9] was then investigated as an alternative approach to real-time object detection that would combine computational efficiency and detection accuracy. In contrast to YOLOv8, RTMDet had better localisation results for some object classes. However, detection accuracy was still unreliable for the smallest

assembly parts.

6.6 Comparative Evaluation

All detector configurations were tested under the same training dataset and experimental conditions. Different aspects of performance were considered during evaluation, such as detection precision, recall, mAP, and robustness in realistic assembling conditions.

Model performance varied across architectures during training and evaluation. RF-DETR achieved the highest detection precision (up to 0.9674) and good mAP metrics, demonstrating strong robustness in challenging environments with multiple similar or partially occluded objects. YOLOv8 models demonstrated competitive overall performance (mAP50-95 reaching up to 0.7818) while maintaining the fastest training times among all evaluated networks, taking as little as 1620 seconds for 120 epochs. In contrast, RTMDet variants exhibited lower overall mAP scores (mAP50 of 0.85) alongside intermediate training durations.

Table 6.1: Model evaluation results.

Run	Model	ImgSz	Epochs	Batch	Precision	Recall	mAP50	mAP50-95	Train Time (s)	Peak GPU (MiB)
rfdetr small 120e	small	—	120	4	0.9674	0.8700	0.9487	0.7771	6619.6050	4576
rfdetr small real+synth 20% 120e	small	—	120	4	0.9570	0.8600	0.9439	0.7726	7192.8428	4580
rfdetr small real+synth 30% 120e	small	—	120	4	0.9556	0.8700	0.9439	0.7768	7757.0188	4578
rtmdet s 120e	rtmdet_s_8xb32-300e	—	120	8	—	—	0.8540	0.6970	2627.2677	4406
rtmdet s 120e tuned	rtmdet_s_8xb32-300e	—	120	8	—	—	0.8520	0.6930	2646.8742	4428
yolo11s 120e	yolo11s.pt	960	120	8	0.9310	0.8783	0.9180	0.7625	1688.6626	6194
yolo26s 120e	yolo26s.pt	960	120	8	0.9264	0.8702	0.9283	0.7645	2105.5535	7246
yolov8n 100e	yolov8n.pt	960	100	16	0.9204	0.8304	0.9061	0.7292	—	—
yolov8s 120e	yolov8s.pt	960	120	8	0.9063	0.8725	0.9252	0.7553	—	—
yolov8s 120e	yolov8s.pt	960	120	8	0.9281	0.8813	0.9272	0.7739	1620.2919	5894
yolov8s real+synth 20% 120e	yolov8s.pt	960	120	8	0.9242	0.8912	0.9324	0.7818	1828.4333	5804

As seen from [Table 6.1](#), RF-DETR was selected as the final object detection model used throughout the remainder of this work. While YOLOv8s achieved the highest overall mAP50-95, RF-DETR demonstrated more consistent detection of the smallest components (screws, nuts) and fewer missed detections under partial occlusion observations made during qualitative evaluation of the detector outputs on validation frames. Additionally, RF-DETR’s higher precision (0.9674) reduces false positives that would trigger incorrect stage verifications. Precise per-frame latency measurements were not recorded during development. The client-server architecture was adopted specifically to decouple inference cost from the local control loop ([Section 10.3](#)), and the system operated at interactive rates during the evaluation trials. The resulting object detection module provided the accuracy and integration characteristics required by the experimental system.

Chapter 7

Object Pose Estimation Development

7.1 Pose Estimation Requirements

Most robotics manipulation problems require full six-degree-of-freedom (6DOF) pose estimation for all components. However, an assembly assistance system does not need this for all parts. Only a limited number of assembly components require their orientation to be explicitly verified. In some cases, the physical structure of the assembly constrains the possible orientation of a component, while in others, the correct orientation can be inferred from the assembly procedure. For instance, the system needs to differentiate between the front and back sides of the cylinder cover or find out the orientation of the piston rod. In contrast, for components such as screws, the physical structure of the screw puts a constrain of only one possible orientation. Let an object pose be represented by

$$P = (R, \mathbf{t}), \quad (7.1)$$

where $R \in SO(3)$ denotes the rotation matrix, and $\mathbf{t} \in \mathbb{R}^3$ represents the translation vector. However, for the considered assembly task, only specific orientation states are required for decision making. This observation significantly influenced the subsequent design choices.

7.2 Dense Object Nets

The first approach considered was Dense Object Nets (DON)[11], which is a correspondence-based approach that uses correspondence between RGB image and CAD model to learn dense feature embeddings. The availability of accurate CAD models of all the assembly components were a reason for making this

approach particularly attractive. Data sets were created synthetically and the whole pipeline of DON was trained using CAD object models.

The trained model was able to detect the object surfaces, however practical tests showed that estimated object orientation was very unstable. Rotations were usually arbitrary and differed from frame to frame even when there were no object movements. It resulted in obtaining different orientation estimates of the same scenes. As precise orientation consistency is vital for assembly validation, DON was deemed inadequate for the proposed application due to its behaviour, which indicated that the learned correspondences were insufficiently stable under the considered imaging conditions, leading to unreliable pose estimation.

7.3 FoundationPose Investigation

The FoundationPose[12] framework leverages object segmentation, geometric reasoning, and iterative pose optimization to determine the 6D pose of unseen objects. The resulting pose estimates produced orientations that were visually consistent with the underlying object geometry and markedly more stable than DON’s baseline predictions. To evaluate sensitivity to object distance, experimental analysis was conducted across synthetic scenes and real RGB-D captures by systematically overriding the Z coordinate. The projection-to-mask ratio is defined as the ratio of the projected CAD model area to the observed object mask area. A ratio near 1.0 indicates good geometric agreement between the estimated pose and the observed object; ratios far from 1.0 indicate misalignment. A frame is flagged if the projection-to-mask ratio falls below 0.5, if valid depth is below 70%, or if the estimated pose depth differs from the sensor depth by more than 75 mm. Flagged frames indicate unreliable pose estimates. To characterise the sensitivity of FoundationPose to depth input, a controlled sanity check was performed in which the Z coordinate was overridden to two values: 1.24 m and 1.14 m, refer Table 7.1. The sensor’s actual median depth was approximately 1.13 m. With $Z=1.24$ m (an 11 cm offset), the projection-to-mask ratio dropped to 0.60 and all 14 frames were flagged, confirming that pose estimation degrades sharply when the supplied depth is inconsistent with the observed scene(Figure 7.1). With $Z=1.14$ m (a 1 cm offset), the projection-to-mask ratio improved to 0.71 and only 2/14 frames were flagged (Figure 7.2). This demonstrates that the pose estimation pipeline is affected when the supplied pose depth is inconsistent with the observed scene. Full raw execution logs for both depth configurations are provided in Appendix D (Listings D.1 and D.2).

Table 7.1: Z coordinate Override Evaluation

Z coordinate	Median Projection Ratio	Flagged	Observation
1.24 m	0.60	14/14	Severe Z mismatch
1.14 m	0.71	2/14	Better projection alignment

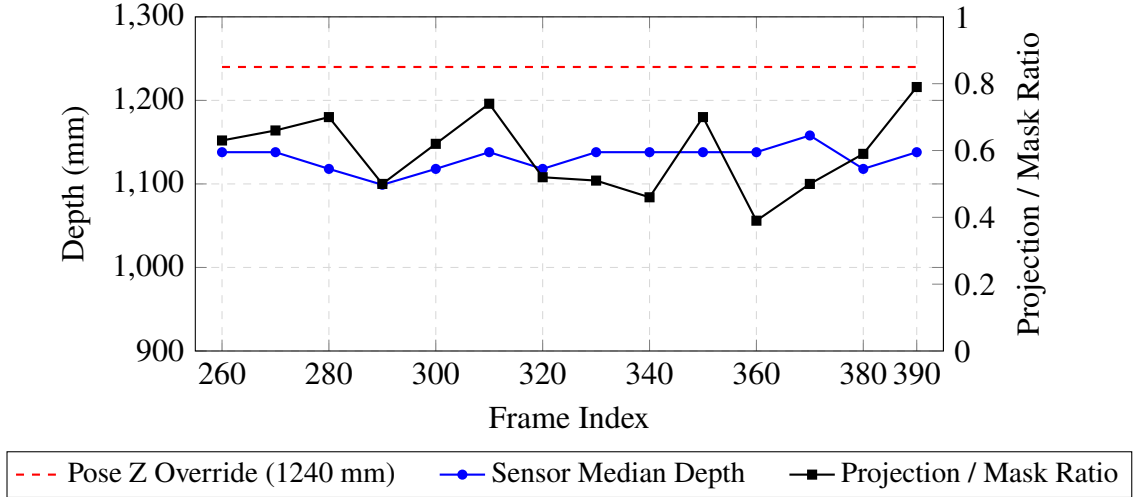


Figure 7.1: Evaluation of the Pose Z override (1.24 m) against the measured median depth and projection-to-mask ratio across frames. The persistent depth offset between the configured pose depth and sensor measurements results in quality flags.

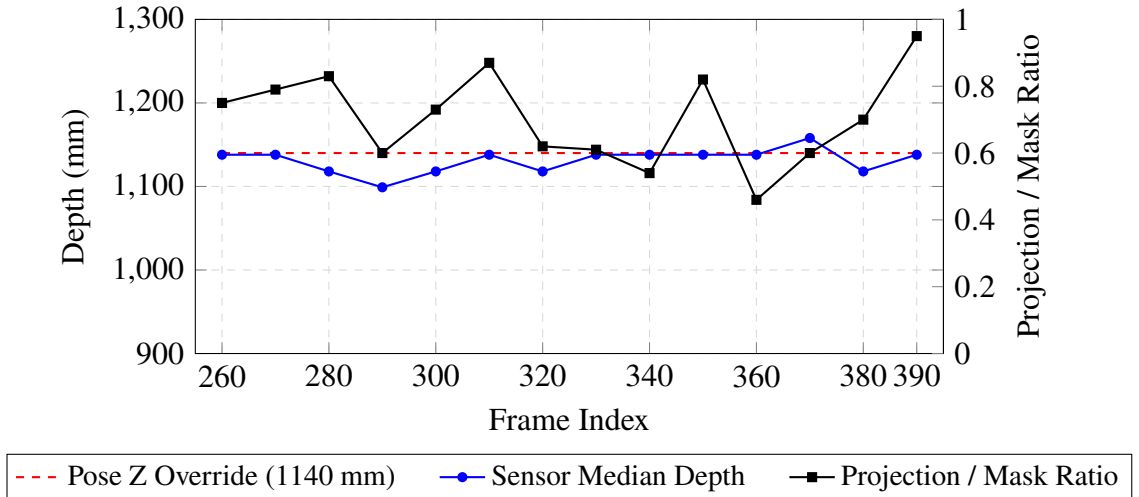


Figure 7.2: Evaluation of the Pose Z override (1.14 m) against the measured median depth and projection-to-mask ratio across frames. The measured depth closely follows the configured pose depth, resulting in substantially fewer quality flags.

The accuracy of pose estimation was largely dependent on how complete and consistent the depth measurements were. In synthetic tests, pose estima-

tion was found to be accurate where accurate depth information was available, while increasing depth corruption led to poorer pose estimation. Pose estimation became inaccurate and large geometric inconsistencies became noticeable once the percentage of valid depth measurements fell below approximately 70%. As shown in Table 7.2 (raw execution logs in Appendix D D.3, D.4 and D.5), the synthetic dataset with 100% valid depth achieved accurate pose estimation with no flagged frames, while mild depth corruption reduced the valid depth to 86% without introducing significant failures. Under heavy corruption, the valid depth decreased to approximately 66%, resulting in four flagged frames and pose-estimation collapse. The extremely large projection ratio reported for the heavily corrupted dataset is not representative of the actual geometric consistency; it results from frames in which the estimated pose depth collapsed to zero, causing the projection-to-mask ratio to become artificially large. Therefore, this value should be interpreted as an indication of pose-estimation failure rather than as a meaningful projection ratio.

Table 7.2: Dataset Robustness Analysis

Dataset	Valid Depth	Projection Ratio	Flagged	Observation
Synthetic	100%	1.36	0	Accurate pose
Synthetic + Mild Corruption	86%	1.35	0	Stable
Synthetic + Heavy Corruption	66%	3631	4	Pose collapse

In practice, the real-world tests brought out other limitations. Reflective metal surfaces led to substantial depth dropouts and incomplete geometric information at close camera distances. As shown in Fig. 7.3(a), positioning the camera at 30 cm resulted in only 42.1% median valid depth, compared with 100% at a working distance of 140 cm, refer Table 7.3. The corresponding projection-to-mask ratio also increased substantially from 0.04 to 0.63, indicating improved geometric consistency at the larger working distance (raw execution logs in Appendix D D.6 and D.7). However, the improvement in depth availability did not completely resolve pose estimation. At 140 cm, the estimated pose depth remained variable across frames, with errors ranging from approximately 0 to 49 mm in the evaluated samples (Fig. 7.4), indicating residual ambiguity in object orientation.

Table 7.3: Camera Height Evaluation

Camera Height	Median Valid Depth	Main Issue
30 cm	42%	Severe depth dropout
140 cm	100%	Pose ambiguity

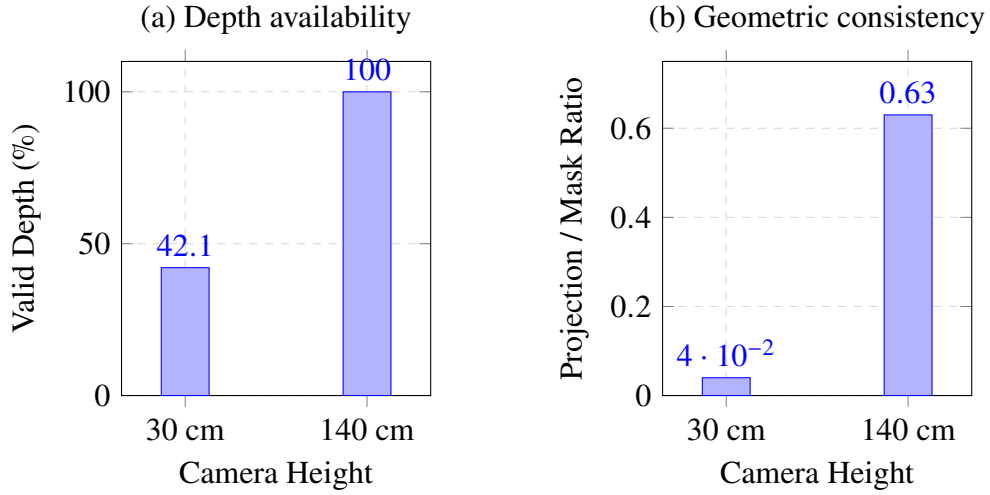


Figure 7.3: Effect of camera height on depth availability and geometric consistency. Increasing the camera height from 30 cm to 140 cm substantially improves the completeness of the measured depth and the agreement between projected and observed object geometry.

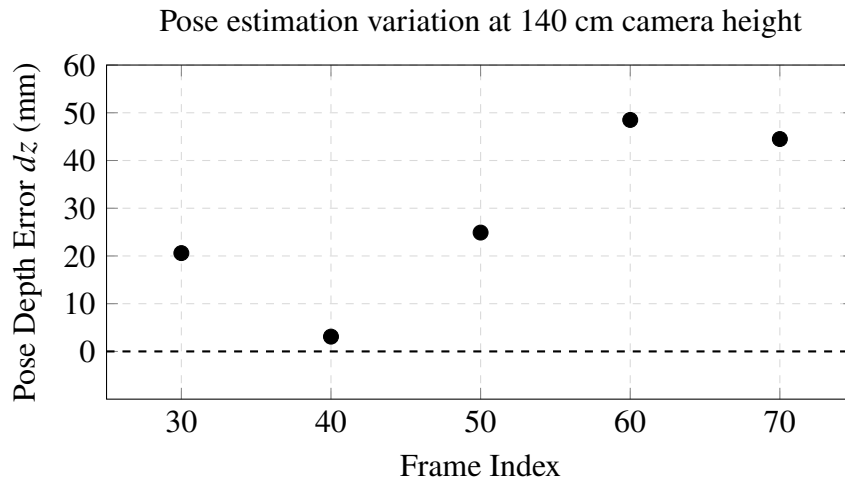


Figure 7.4: Variation in pose-depth estimation error at a camera height of 140 cm. Although the depth measurements are complete, the estimated pose depth remains variable across frames, indicating residual ambiguity in the estimated object orientation and geometry.

The depth sensitivity analysis discussed above explains when Foundation-Pose fails, but not why it fails even when depth is accurate. Further investigation

revealed that the failures stem from the symmetric geometry of the assembly components. The front and rear covers of the pneumatic piston are approximately rotationally symmetric about their central axis (Figure 6.1). As a result, multiple orientations of the CAD model produce nearly identical geometric alignments with the observed point cloud. FoundationPose, optimising for geometric consistency, converges to a pose that is geometrically valid but may be rotated by 180° relative to the true assembly orientation.

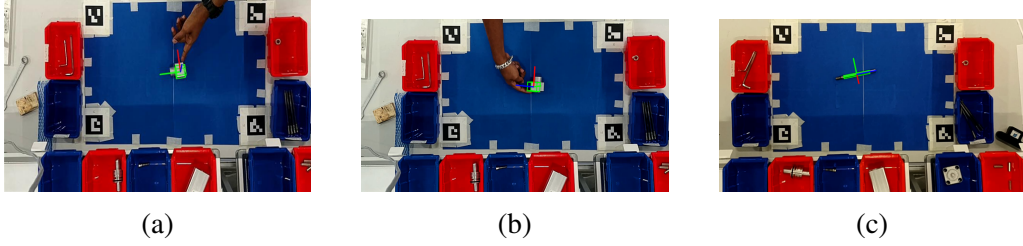


Figure 7.5: Incorrect pose estimation caused by symmetric geometry.

Figure 7.5 illustrates this failure mode. From a geometric standpoint, the estimated pose appears correct. From an assembly standpoint, it is wrong and would lead to an incorrect stage verification. This finding is the central motivation for the reformulation presented in Section 7.4. The problem is not that pose estimation is inaccurate; it is that the pose itself is ambiguous for symmetric parts under the assembly task’s requirements. No amount of pose estimation accuracy can resolve this ambiguity, because the ambiguity is inherent to the geometry.

7.4 Pose Classification Reformulation

From the experimental analysis of the two techniques, DON and FoundationPose, a key finding emerged. The implemented assembly assistance system does not need to estimate 6D pose for the considered application. It needs only requires verification that the object is in the orientation required by the corresponding assembly stage. Rather than estimating the continuous 6D pose $P = (R, \mathbf{t})$, the problem was reformulated as a discrete classification problem:

$$f(I) \rightarrow C$$

where I denotes the cropped object image and C represents one of the predefined orientation classes. For example:

Front cover:

- Correct orientation
- Flipped orientation

Rear cover:

- Correct orientation
- Flipped orientation

This reformulation significantly simplified the learning problem while directly addressing the actual verification requirements of the assembly task.

7.5 YOLO-Based Pose Classifier

Instead of making predictions based on continuous orientations, discrete classes of orientations were manually assigned to all components that required orientation validation. These classes correspond directly to assembly states that define acceptance and rejection of a particular stage. The described approach eliminates the uncertainty related to rotationally symmetrical objects and gives the exact data that is needed by the StageFSM.

The classification database for the orientation was built using both real images and synthetic images generated from the CAD model that was provided. Images were taken from different angles and lighting conditions to ensure sufficient variation among the images belonging to the same class. All images were assigned to their respective classes before being saved in the classifier’s format.

The orientation classifier was trained on a per-component dataset assembled from real images (captured under the overhead camera). During training, the classifier receives object images that have been cropped using an object detector. Given an object image I_o , the classifier predicts:

$$\hat{C} = \arg \max_{C_i} P(C_i | I_o)$$

where $P(C_i | I_o)$ denotes the probability associated with orientation class C_i . Standard image augmentation methods such as rotations, scaling, illumination changes, and background randomization were used. Training was performed for 60 epochs using stochastic gradient descent with a cosine learning-rate schedule. [Table 7.4](#) reports the peak and final top-1 validation accuracy for each component classifier. The front-cover and rear-cover classifiers converged rapidly and

Table 7.4: Orientation classifier validation accuracy.

Component	Peak top-1 accuracy	Peak epoch	Final top-1 accuracy
Front cover	99.9%	24	96.6%
Rear cover	96.1%	39	95.8%

reached peak top-1 accuracy above 95% within 25 and 40 epochs respectively.

Both exceeded the FR7 acceptance criterion of 95% orientation classification accuracy.

The orientation classifier was integrated immediately after the object detection block. For each object that needs its orientation to be checked, the corresponding image crop is passed to the classifier associated with that component. The predicted orientation is checked against the known orientation of the particular stage of assembly. This way only critical components are considered, thus saving computational resources. The predicted orientation is then sent to the StageFSM and influences the final pass/fail decision. By reformulating the problem as an orientation classification task, the system obtains a simpler and application-specific solution (Figure 7.6 shows the 3 pose classes namely correct pose, side pose and flipped pose for front cover). Precise per-frame latency measurements were not recorded during development. As with the object detectors case discussed in previous chapter, the client-server architecture was adopted specifically to decouple inference cost from the local control loop (Section 10.3), and the system operated at interactive rates during the evaluation trials. Hence, the resulting implementation integrates both the object detection and the orientation classification in a simple way that allows one to check whether the assembly is done correctly or not.

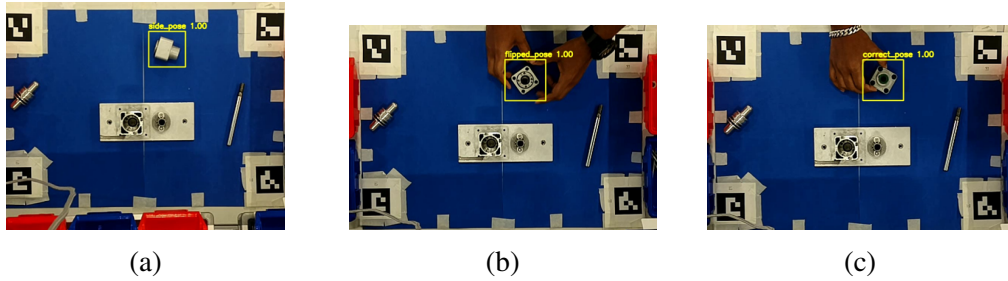


Figure 7.6: Yolo Pose Classification for front cover. (a) side pose, (b) flipped pose and (c) correct pose

Chapter 8

Motion Understanding and Stage Management

The object detector and pose classifier offer information about the spatial positions and orientations of the assembly parts, and via MediaPipe we obtain the hand trajectory of the operator. A combination of all these components on how the human operator interacts with the parts temporally is required for assembly monitoring. This motivated the development of a motion understanding module that would interpret the human actions, validate the assembly process and control the transitions among assembly stages. In contrast to processing each video frame separately, the module interprets the temporal information gained through hand movements, workspace interaction, tool usage and object perception. The interpretation of temporal information is performed through two finite state machines working on different levels of abstraction:

- **MotionFSM**, responsible for recognising the progress of individual manipulation actions.
- **StageFSM**, responsible for validating the completion of assembly stages and controlling transitions throughout the assembly sequence.

Working together, these modules are able to turn low-level visual observations into meaningful assembly events.

8.1 MotionFSM Design

MotionFSM describes the execution of individual assembly operation. It converts the gate signal of [Section 5.8](#) into a sequence of manipulation events. Rather than recognizing the operation from one image, the entire assembly is described as a

series of motion states. The state machine consists of four states:

$$\text{WAIT_TRIGGER} \rightarrow \text{RECORDING} \rightarrow \text{WAIT_COMPLETION} \rightarrow \text{COOLDOWN} \quad (8.1)$$

Wait Trigger: Monitors for a valid start action. It enforces three conditions before entering RECORDING: the interaction gate of [Section 5.8](#) must be satisfied (motion evidence with either hand-presence or trajectory confirmation), the stage-specific start role must be present in the expected source region, and the contract validator must confirm the interaction is spatially valid for the current stage.

Recording: Tracks the manipulation and evaluates the completion signal for the current stage. The completion signal depends on the stage's `completion_mode`: `next_stage_pick` (the pick that initiates the next stage), `completion_proxy` (an explicit drop role in an expected region), or `terminal_end` (no automatic completion; the stage closes only on manual advancement). Occlusion tolerance allows the hand to be briefly lost without aborting the recording.

Wait Completion: Holds the completion candidate for `completion_confirm_frames` to guard against frame-level detection flicker. If the signal drops for a frame, the FSM returns to RECORDING.

Cooldown: Enforces a brief mandatory pause (e.g., 8 frames) before allowing the system to jump back into `wait_trigger`. It gives the physical environment time to settle down immediately after an action terminates, preventing rapid double-triggering.

Table 8.1: MotionFSM transition predicates.

Transition	Predicate
WAIT_TRIGGER → RECORDING	Start-gate satisfied and contract validation passes
RECORDING → WAIT_COMPLETION_CONFIRM	Completion-gate satisfied for <code>min_consecutive_completion_hits</code> frames and <code>min_recording_frames</code> elapsed
WAIT_COMPLETION_CONFIRM → COOLDOWN	Completion-gate holds for <code>completion_confirm_frames</code> consecutive frames
WAIT_COMPLETION_CONFIRM → RECORDING	Completion-gate drops for one frame
COOLDOWN → WAIT_TRIGGER	<code>cooldown_frames</code> elapsed

[Table 8.1](#) lists the transition predicates. The implemented values of these parameters can be found in [Appendix B, Table B.3](#). The start-gate activates when policy-specific role and region conditions are met:

- **AssemblyPolicy:** Detection of `part_pick` or `tool_pick` in `start_sou`

rce_regions, OR manual_start_allowed:true with the hand in the work region.

- **DismantlePolicy:** Detection of tool_pick in start_tool_regions, OR manual_start_allowed:true with the hand visible in the workspace.

manual_start_allowed is a stage descriptor flag that permits stage advancement, when start regions are ambiguous. In the dismantle configuration, most stages set manual_start_allowed:true to implicitly chain stages from previous drop events without requiring fixed pick regions, whereas tool-specific stages set it to false due to reliable tool-retrieval detection in designated regions. Conversely, the assembly configuration sets manual_start_allowed:false by default to require explicit part picks in expected source regions, logging any manual interventions as distinct event types for trial analysis. To handle tracking interruptions, the start condition must persist for min_consecutive_start_hits consecutive frames OR reach start_recent_hit_count hits within a rolling start_recent_hit_window. Finally, the StageRegionContractValidator verifies spatial validity. The completion-gate triggers based on completion_mode:

- **next_stage_pick (Assembly):** Stage i completes when the pick for Stage $i + 1$ begins.
- **completion_proxy (Dismantle):** Completes via an explicit stage-defined role/region pair.
- **terminal_end:** Requires manual advancement.

Temporal smoothing requires min_consecutive_completion_hits consecutive frames, followed by completion_confirm_frames in WAIT_COMPLETION_CONFIRM to ensure stability. Refer to Appendix A for configuration structure and Appendix E for detailed explanations of each configuration parameters.

8.2 StageFSM Design

While the MotionFSM analyzes each individual manipulation, the StageFSM controls the entire assembly process sequence. Each assembly operation is represented as a predefined stage containing: (i) expected object/tool, (ii) expected workspace region, (iii) expected hand motion, (iv) completion conditions, (v)

expected time duration. A simplified stage descriptor is shown below [Figure A.1](#): Each stage transitions through:

$$\text{PENDING} \longrightarrow \text{ACTIVE} \longrightarrow \text{PASSED/FAILED} \quad (8.2)$$

Only one stage remains active at a given time, simplifying decision making and reducing false activations. When the MotionFSM closes a manipulation, the StageFSM evaluates the stage-level completion conditions valid workspace interaction, correct object detection, correct orientation where required, DTW trajectory validation where configured ([Section 8.6](#)), and tool verification where applicable and marks the stage PASSED or FAILED. On PASSED, the StageFSM advances; on FAILED, the stage remains active, the respective error handling occurs ([Section 8.7](#)) and the interface presents corrective feedback ([Section 9.5](#)).

8.3 Ordered and Unordered Stage Groups

While most assembly operations require a particular order, there are some operations that can take place in an arbitrary sequence. Tightening of the four tie rods can take place in any order and still result in a correct assembly. Thus, stages are categorized into: (i) Ordered Groups, (ii) Unordered Groups. An **ordered group** requires sequential execution:

$$S_i \rightarrow S_{i+1}$$

on completion of S_i , the StageFSM increments `current_stage_index` to S_{i+1} . The index itself carries the precondition the next stage cannot become active until the current stage has reached PASSED. Whereas an **unordered group** allows execution in any permutation:

$$\{S_1, S_2, S_3, S_4\}$$

provided that every required stage is eventually completed. All stages sharing the same `unordered_group_id` belong to one group. Upon completion of a member stage, the member's name is added to a per-group completed set, and the set of remaining members is recomputed. If members remain, `current_stage_index` is set to the lowest-index remaining member; otherwise, the group is complete and `current_stage_index` advances to the index immediately following the group. The lowest-index member is used only as the current FSM reference; matching remains enabled against the remaining group members, so the operator is not required to follow index order. Because the mechanism is implemented across multiple layers of the system, the description below distinguishes between

the FSM-level matching logic, the pipeline-level group progression logic, and the configuration-level group descriptors.

- **MotionFSM:** `_match_unordered_start()` and `_match_unordered_completion()` test each remaining group member in sequence and return the first whose policy is satisfied, resolving the operator’s execution order dynamically.
- **Pipeline level:** `_postprocess_unordered_segment_progress()` updates the completion set and injects the next stage, task status, and remaining stage count into the feedback structure consumed by the guidance interface.
- **Configuration level:** Each group is defined by its member list, a lock-out guard (`unordered_group_lockout`, default `True`) that prevents re-counting completed members, and a required count (`unordered_group_required_count`, default 1, set to 4 for the tie-rod-tightening group) specifying how many unique members must complete for group fulfillment.

Refer to Appendix A for configuration structure and Appendix E for detailed explanations of each configuration parameters.

8.4 Active-Hand Selection Policy

Since both hands may appear simultaneously within the camera view, the system dynamically selects the active hand responsible for the current manipulation. This also makes the system hand invariant. For each detected hand h_i , an interaction score is computed:

$$\text{Score}(h_i) = w_1 D_i + w_2 V_i + w_3 T_i$$

where D_i denotes Dominant hand of operator since an operator will always prefer the dominant hand over the other, V_i represents velocity of the hand, and T_i is the tool proximity term, i.e the distance between hand anchor point and tool. The hand with the maximum interaction score is selected:

$$h^* = \arg \max_i \text{Score}(h_i)$$

This prevents inactive supporting hands from influencing trajectory analysis. To prevent rapid oscillation between hands, a switch only occurs if the challenger exceeds the current active hand’s score by a switch-score margin for a minimum

number of confirm frames. Configured values for the weights, the normalisation scales, and the switch parameters are listed in Appendix B, Table B.5.

8.5 Trajectory Acquisition & Tool-Hand Synchronisation

During every manipulation, the trajectory of the active hand is continuously recorded. Each trajectory sample is represented as:

```
trajectory_point = {
  "x": x,
  "y": y,
  "t": timestamp,
  "hand": "left",
  "region": "front_cover_region"
}
```

A complete trajectory becomes:

```
trajectory = [
  {
    "x": ...,
    "y": ...,
    "t": ...
  },
  ...
]
```

Only samples acquired while the `MotionFSM` remains in the active manipulation states are retained. The trajectory therefore represents only the intentional manipulation rather than the complete hand motion throughout the assembly process. Every processed frame contributes information to a shared runtime structure. A simplified representation is:

```
frame_data = {
  "timestamp": t,
  "hands": {...},
  "tool": {...},
  "poses": {...},
  "region_hits": {...},
  "active_hand": "right",
  "motion_state": "INSIDE",
  "current_stage": 5
}
```

This structure serves as the communication interface between all processing modules and ensures that each subsystem operates on a consistent description of the current assembly state.

8.6 Dynamic Time Warping

Dynamic Time Warping (DTW) is employed to compare the observed trajectory with a reference trajectory, providing a measure of similarity between the observed trajectory and a reference trajectory. Unlike standard Euclidean distance

which compares points at rigid, fixed time steps and fails when speeds vary, Dynamic Time Warping (DTW) non-linearly aligns temporal sequences, making it ideal for robustly matching operator hand trajectories executed at varying speeds, as shown in [Figure 8.1](#). Given trajectory sequences:

$$P = \{p_1, p_2, \dots, p_n\}$$

and a reference sequence

$$Q = \{q_1, q_2, \dots, q_m\}$$

DTW computes the minimum cumulative alignment cost:

$$DTW(i, j) = d(i, j) + \min \begin{cases} DTW(i-1, j) \\ DTW(i, j-1) \\ DTW(i-1, j-1) \end{cases}$$

where:

$$d(i, j) = \|p_i - q_j\|$$

denotes the Euclidean distance between corresponding trajectory samples. A manipulation is accepted if:

$$DTW(P, Q) < \tau_{dtw}$$

where τ_{dtw} represents an experimentally determined acceptance threshold. The use of DTW enables comparison of trajectories performed at different speeds while preserving their spatial characteristics.

DTW serves two complementary roles. As a stage gate, for stages whose descriptor includes a reference trajectory, the StageFSM requires $DTW(P, Q) < \tau_{dtw}$ for the stage to be marked PASSED. The acceptance threshold τ_{dtw} and the stage-score weights are listed in [Appendix B, Table B.4](#). As an evaluation metric, DTW distance is recorded for every completed manipulation regardless of pass/fail status, providing [Chapter 11](#) with a continuous measure of trajectory quality. The two roles are complementary: the same distance gates the stage at runtime and is logged for offline analysis.

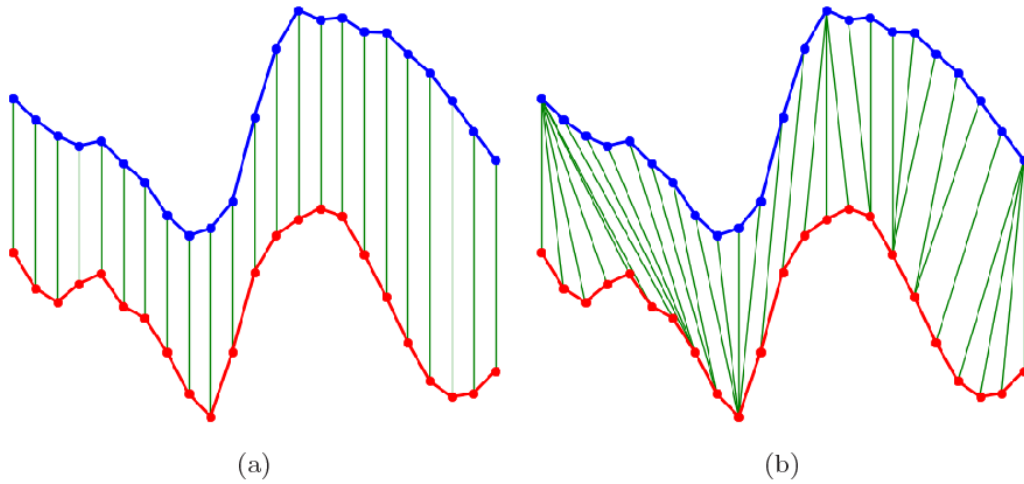


Figure 8.1: Comparison of (a) Euclidean matching vs (b) Dynamic time warping matching

8.7 Stage Completion and Error Handling

A stage is considered complete only when all required validation conditions are simultaneously satisfied. These conditions may include: successful MotionFSM completion, valid workspace interaction, correct object detection, correct object orientation, successful DTW trajectory validation, tool verification (if applicable). If one of the conditions is not fulfilled, the stage will remain active and provide immediate visual feedback about the condition that failed.

In order to improve usability, some recovery procedures are provided, including re-execution of the stage, manual advancement of the stage (as an intervention fallback when correct actions are misflagged as FAILED) and resetting of the system without starting the whole assembly process again.

Chapter 9

User Guidance Interface

Building upon the completed perception, state tracking, and stage management modules including object detection, pose classification, interaction triggers, and trajectory verification the system requires a dedicated user-facing operational layer. Chapter 9 presents the design and implementation of the User Guidance Interface, which transforms underlying algorithmic outputs into real-time, non-intrusive visual feedback for the human operator.

9.1 Interface Requirements

The user guidance interface is meant to be the visual communication layer between the monitoring system based on vision and the operator. Its main function is to communicate the current state of the assembly process, highlight the interaction zones expected by the system, give feedback on the detected actions, and alert the operator about an incorrect action. As the assembly process will be manual, the interface should be informative but not interfere with the camera view and overload the operator with information. The main requirements of the interface are: provide real-time feedback with minimal delay; clearly identify the regions relevant to the current assembly stage; distinguish between inactive, active, completed, and erroneous interactions; provide different levels of assistance according to the selected assistance mode; display CAD-derived virtual guides when required; communicate warnings and stage transitions clearly; remain integrated with the output of the StageFSM, MotionFSM, object detector, pose classifier, and interaction-detection modules.

9.2 Adaptive Assistance Modes

The system allows configuring of the assistance modes as follows: full assistance, medium assistance, and no assistance. The rationale behind this feature is that there is a need to configure the amount of information displayed based on the necessary degree of assistance needed. There is no change in the core perception and verification process; however, assistance modes affect the display of guidance information. This separation is especially crucial for the user experiment, as it allows to assess the effect of the guidance system on the assembly process without altering the basic assembly-verification pipeline.

9.2.1 Full-Assistance Mode

The full-assistance mode (Figure 9.1) contains the most information in visual form. In this mode, the interface displays the relevant workspace regions, virtual CAD projections, interaction states, looping reference video, and feedback messages in relation to the current state of assembly. As a result, the operator will be able to use visual cues to understand from which place a component or a tool needs to be taken. It is mainly aimed at inexperienced operators.

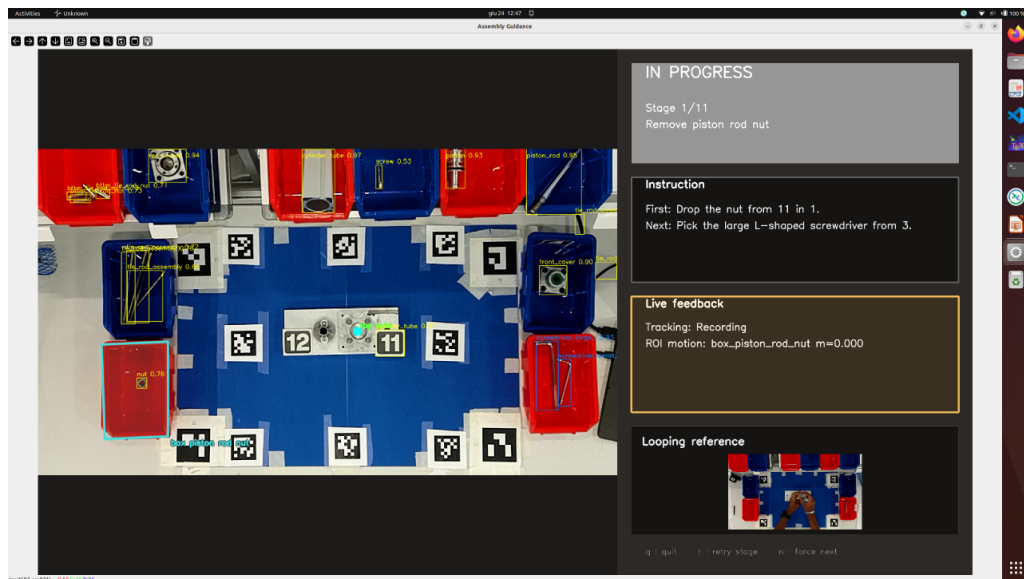


Figure 9.1: Light-blue bounding box in the lower-left region delineates the target part-pick ROI; Cyan CAD projection adjacent to component 11 indicates the target placement pose for the retrieved nut; Grey status banner in the upper-right region indicates operational progress—transitioning to green upon stage success or red upon failure; Highlighted yellow panel in the mid-right section provides real-time telemetry, including orientation compliance, motion activity, FSM state transitions, and active object detections; The instruction panel provides sequential textual guidance, while the looping reference viewport renders the pre-recorded expert baseline video for the active stage.

9.2.2 Medium-Assistance Mode

The medium-assistance (Figure 9.2) method cuts down on the amount of information provided, but retains the amount of information that is needed in order to execute the task. Rather than showing all the visual instructions for each step, the medium-assistance system shows information selectively, such as the instruction for the current step, warning information, or feedback related to the current operation. The idea is to provide help to the user without totally relying on the visual instruction.

Stage	Instruction	Result	Status
CURRENT 1. Remove piston rod nut	First: Drop the nut from 11 in 1. Next: Pick the large L-shaped screwdriver from 3.	Live: RECORDING Waiting...	PENDING
NEXT 2. Loosen tie rod nuts	First: Pick the large L-shaped screwdriver from 3.(ignore if don't/coming Then: ONLY Loosen the four tie rod nuts in 11. Next: Return the tool to 3.		PENDING
NEXT 3. Remove tie rod assemblies	Pick each tie rod assembly from 11 and drop ALL together AT ONCE in 2.	Upcoming	PENDING

q : quit r : retry stage n : next

Figure 9.2: Highlighted yellow row marks the active assembly stage, dynamically advancing down the table as sequential stages are completed; The Instructions column provides procedural guidelines for executing the current manipulation; The Results column logs real-time operational feedback, including object orientation, motion activity metrics, and current FSM states; The Status column records the final execution outcome (pass/fail) upon stage completion.

9.2.3 No-Assistance Mode

In no-assistance mode (Figure 9.3), the visual guidance elements are not present. The underlying perception, stage verification, and motion analysis modules continue to operate normally, but their outputs are not used to provide continuous guidance to the operator. At the end of each stage, the operator is provided with a pass/fail status of the executed stage, providing information on whether they can proceed or redo the stage. This mode helps the system to observe an operator assembling the object on their own and provides a baseline against which the assisted modes can be compared.

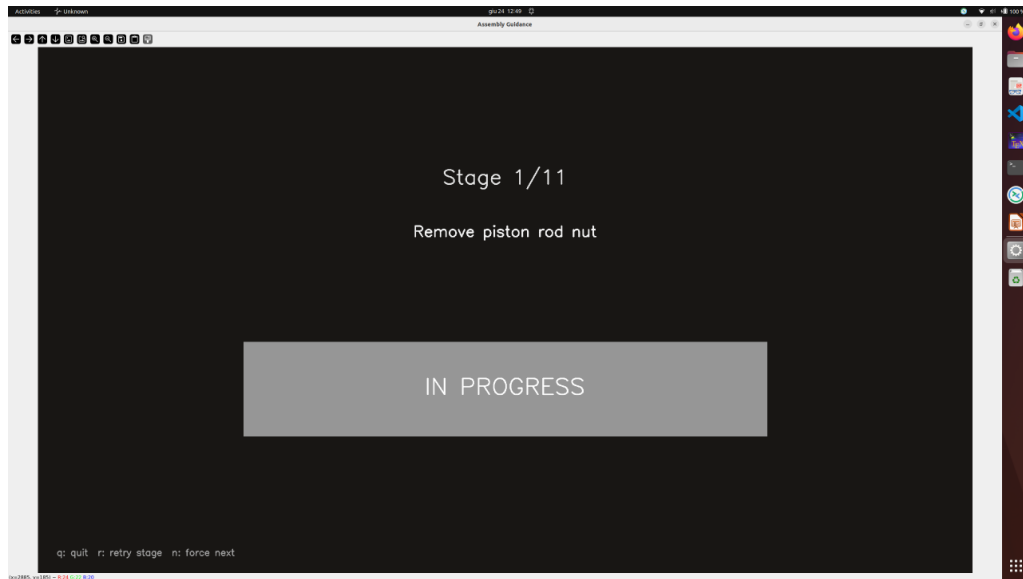


Figure 9.3: The primary viewport displays the active stage index alongside its descriptor name, while the central Grey status tab indicates real-time operational state—rendering grey for an ongoing stage, green upon successful completion, and red upon failure.

9.3 Guidance Region Visualisation

Workspace regions are used by the interaction-detection module and the stage-management module to define important regions of the assembly workspace. The regions are related to picking and placing of components and tools, depending on the stage of assembly being executed. However, only the relevant regions for the current stage of assembly are highlighted by the guidance interface (§9.2.1). This makes sure that the workspace does not get crowded with irrelevant information. The display of the regions depends on whether there is any interaction occurring within the particular region or not. The inactive region is defined by a clearly visible boundary while the active region has a stronger border and prominent colour fill (Figure 9.4). This state-dependent representation allows the interface to communicate the assembly progression without requiring textual instructions for every operation.

9.4 CAD Projection

Where communication of the position/orientation of the component is required in an operation, the system can use virtual projections derived from CAD models on the camera frame. The CAD model of the parts forming the assembly is converted to the top view sprite and geometrically transformed to align it with the appropriate workspace region. The projection can be represented as a



Figure 9.4: (a) The ROI has a strong border and prominent color fill indicating Active, (b) ROI has only a visible boundary indicating no Active and signaling user to pick component from that box

transformation:

$$I'_{CAD} = T(I_{CAD}, R_i)$$

where I_{CAD} is the rendered CAD sprite and T represents the geometric transformation required to map it onto the target workspace region R_i . In the implementation, T is a fixed 2D affine transform (scale, rotation, translation) parameterised per region in the configuration file.

This projection will help the user understand where the part should be placed (figures 9.1 and 9.4). The CAD projection will be especially helpful when the region provided by a simple rectangle does not suffice to provide the placement of the part. In addition to knowing where the part should be placed, the projection will give some clues regarding the expected shape/position of the part.

9.5 Feedback and Warning Messages

Apart from spatial cues, the interface gives textual feedback based on the current state of the assembly FSMs. Feedback messages can communicate events such as: successful completion of an operation; transition to a new assembly stage; incorrect component interaction; incorrect tool interaction; incorrect component orientation; unexpected stage progression; failed or incomplete interaction.

It should be noted that the feedback messages are formed based on the internal state of the system and not independently interpreted by the interface. This ensures consistency between the visual feedback and the actual state of the monitoring system.

9.6 Interface Integration

The guidance interface serves as the final visualization stage of the complete pipeline for monitoring. The data coming from different subsystems is merged in a single frame prior to presentation to the user. The guidance manager gets the current stage, expected regions, detected interactions, object states, pose-verification results, and assistance configuration. On the basis of this data, the guidance manager decides what should be rendered. The final frame can therefore be considered as:

$$I_{\text{out}} = G(I_{\text{RGB}}, S_{\text{stage}}, S_{\text{motion}}, R, O, P, M)$$

where:

- I_{RGB} is the original camera frame;
- S_{stage} is the current StageFSM state;
- S_{motion} is the current MotionFSM state;
- R represents the workspace-region states;
- O represents detected objects;
- P represents pose-classification results;
- M represents the selected assistance mode;
- G is the guidance-rendering function.

This approach guarantees that the guidance interface is still a consumer of decisions made by the system, but not a decision-making element. As a result, any changes in perception and/or stage verification algorithms will not affect the user interface design.

Chapter 10

Final Integrated System

10.1 Final Architecture

The final level of the architecture processes the stream of RGB images coming from one camera. The image is analyzed to obtain visual information necessary for assembly inspection such as object detections, hand or pose information, evidence of region interaction, orientations of components, and other temporal information. This information is then used by the motion and stage managers for deciding the current state of the assembly process.

The system architecture is hierarchical. At the low level, the information about the scene is extracted by perceptual modules independently from the image stream, whereas at the high level, temporal modules use this information for checking whether the meaningful interaction took place and whether it corresponds to the specified assembly process.

The final architecture also maintains a separation between perception, decision making, and operator guidance. In this way, different perception modules can be easily updated or replaced independently of the stage management and interface modules. The selected pose-classification module is integrated only at stages where component orientation is relevant. Object detector, Hand tracker and DTW are employed at every stage as each stage has its own tool usage with a desired hand trajectory.

10.2 End-to-End Pipeline

The whole processing chain begins with RGB frame acquisition and ends when either there will be a change in the assembly state or there will be new visual feedback for the operator. These findings are combined with information related to time from the previous frames. MotionFSM considers the interaction data

and checks if the observed motion indicates the valid operator's action. Having enough evidence, the interaction state is passed to the StageFSM. StageFSM checks if the detected interaction meets the expectations regarding actions and restrictions at the current stage of assembly. In case the necessary conditions are met, the stage can move to the next one. Additional checking of the object orientation is carried out if needed by the particular stage.

The frame data representation discussed before, allows the different modules to operate as components of the same processing pipeline rather than as independent algorithms. At the end of each processing cycle, the current state is used by the guidance system to determine what should be displayed to the operator. Depending on the current stage and selected assistance level, the interface may display workspace regions, CAD-based projections, textual instructions, warnings, or completion feedback. The pipeline thus creates a feedback loop:

$$\text{Visual Input} \rightarrow \text{Perception} \rightarrow \text{Interaction} \rightarrow \text{Stage Verification} \rightarrow \text{Guidance} \rightarrow \text{Operator} \quad (10.1)$$

Any actions of the operator create new visual inputs that are then processed by the next pipeline cycle.

10.3 Real-Time Inference Architecture

In the final system, the video acquisition and control functions have been separated from the computationally intensive inference operations. This design enables the application at the camera side to continue running uninterrupted while some specific deep learning models can be run on a more powerful machine.

There is communication between the local and remote systems using network connection. The local application is in charge of keeping the temporal state of assembly process as well as coordination of the entire process. The computationally intensive neural network inference operations may alternatively be performed in the GPU system. The information exchanged between the systems consists primarily of the camera frame or the required inference input and the resulting predictions. These predictions are subsequently incorporated into the local frame_data structure and used by the temporal decision-making components.

10.4 System Performance and Robustness

The performance of the final integrated system is determined not only on the inference performance of individual computer vision modules but also on the

interaction of frame acquisition, pre-processing, inference through the neural network, networking, temporal computation, and visualization. Computational cost is mostly affected by the object detection and pose classification modules, which are the most computationally intensive parts of the whole pipeline. On the other hand, FSM-based stage management, ROI computation, trajectory acquisition, and other rule-based computations incur relatively small computational cost.

Robustness is an important consideration since the proposed system uses the RGB observations that are gathered in a realistic assembly setting and variations in lighting, shadows, reflections, partial occlusion, and temporary loss of visual detection can have influence on particular perception modules. However, the design of the final system does not depend solely on the single visual measure in one frame. The information from multiple sources is combined through temporal processing and procedural constraints. Similarly, temporary loss of hand detection does not necessarily result in an immediate loss of the current assembly state. Previously accumulated trajectory and interaction information can remain available to the temporal processing layer, allowing the system to tolerate short detection gaps. The procedural nature of StageFSM provides another way for robustness. Transitions between the stages are constrained by the sequence of the assembly. Thus, an isolated erroneous observation does not mean the arbitrary transition between the stages.

The robustness analysis should distinguish between recovery from temporary perception failures and complete failure of a perception module. The former can potentially be handled by temporal integration, whereas the latter may require additional information or operator intervention.

10.5 Limitations And Possible Improvements

There are a number of task-specific constraints associated with the current system. The fact that only one RGB camera is used means that the system is vulnerable to issues like heavy occlusions, bad viewpoints, and conditions under which the relevant objects or the hands of the operator may be poorly visible. Further, the steps in the assembly process, work area areas, object classes, and orientations of the components are pre-defined for the assembly process under investigation.

The pose verification module is similarly specifically crafted as a classification problem over the finite set of poses needed for the task instead of a generic 6-DOF pose estimator. This simplification decreases computational effort but confines the algorithm's applicability to domains where the relevant orientations are known beforehand.

Potential for future advancements can be seen in enhancing the generalization ability of the perception model using a varied range of training data, better interaction detection using extra temporal and spatial clues, and minimizing the task-specific customization needed during system transfer to a different assembly process. Future realization may also include studying the usage of depth information and/or more complex temporal models when only RGB data falls short.

Chapter 11

Experimental Evaluation

11.1 Research Questions and Hypotheses

The preceding chapters developed a single-camera vision system that monitors a manual assembly task, verifies each stage, and provides configurable visual assistance at three levels. Chapter 11 evaluates the system in an experimental setting to determine whether and how the assistance level affects operator performance and learning. The evaluation is guided by four research questions.

RQ1 — Does the level of visual assistance affect motion quality during the assembly task?

Hypothesis. Full guidance will produce the highest composite motion quality, mid guidance an intermediate level, and no guidance the lowest. The expectation is that richer guidance reduces deviations from the expert reference trajectory, and that this effect is measurable in the composite motion-quality score described in [Section 11.8](#).

RQ2 — Do the three assistance levels produce different learning trajectories?

Hypothesis. Operators will improve over successive trials in all conditions, and the rate of improvement will differ across levels. Full guidance is expected to support the steepest learning curve, on the grounds that continuous visual prompts let operators internalise the correct sequence and motion pattern more quickly than trial-and-error learning.

RQ3 — Does the assistance level affect task completion time or the rate of detected assembly errors?

Hypothesis. Task completion time will be longer under full guidance, reflecting more deliberate action and greater adherence to the reference trajectory; error rate is not expected to differ substantially across conditions because the task structure is identical and the FSM validation criteria are the same.

RQ4 — How do operators perceive the three assistance levels, and which

level would they prefer for learning versus daily use?

Hypothesis. Full or mid guidance will be rated as more helpful and easier to understand than no guidance. For daily use, operators will prefer less assistance than they prefer for learning, reflecting a shift from instruction-following to independent performance once the task is mastered.

11.2 Participants

Eight volunteers participated in the study. All were employees of Fraunhofer Italia and had no prior experience with the pneumatic piston assembly task. Participation was voluntary and took place after an internal invitation distributed by the host organisation. No financial compensation was provided.

The participant group comprised a mix of engineers, researchers, and administrative staff, with a range of prior exposure to industrial assembly ranging from none to occasional. Six participants reported some form of corrected vision condition (myopia, hyperopia, or astigmatism); all worked with their normal correction. This is noted here because the guidance interface is visual and individual differences in visual perception could affect how individual operators interact with the assistance prompts.

One additional participant the author, who conducted the experiment completed a small number of trial runs while validating the system but did not complete the questionnaire and is excluded from the analysis. The final dataset therefore contains nine participants, 90 trials, and 896 stage-level observations.

Participation was anonymous with respect to the analysis: each participant was assigned a numeric identifier that was used to link their sessions to the recorded metadata. The experiments were conducted in a single location at the Fraunhofer Italia laboratory over multiple sessions on June 2026.

11.3 Experimental Conditions

Three assistance levels were evaluated, corresponding to the three modes of the guidance interface introduced in [Section 9.2](#). The assistance level was assigned per trial in a pseudo-randomised order, so that across the nine participants each condition was exercised a comparable number of times, and no participant received the same level on every trial. All other aspects of the system object detection, hand tracking, stage verification, and FSM behaviour were identical across the three conditions. The only variable was the amount of information presented by the interface.

11.4 Experimental Protocol

Setup. The experiment was conducted at the fixed workstation described in [Section 4.2](#), with the OAK-D camera mounted overhead and the guidance display positioned beside the workspace. The pneumatic piston components were arranged in the workspace regions defined in [Section 4.4](#).

Initial demonstration. Before their first trial, each participant was given a short demonstration of the three assistance levels by the experiment conductor. During the demonstration, the conductor performed one complete assembly sequence in each mode, describing aloud what the interface displayed and what the expected action was. Participants were told that the three modes represented different amounts of information and that their task was to complete the assembly as correctly as possible in each trial.

Instructions to participants. Participants were asked to:

- Complete the assembly using the correct procedure at each stage.
- Follow the guidance displayed on the screen when guidance was available.
- Perform the assembly from their own understanding when no guidance was displayed, again following the correct procedure.

They were explicitly instructed not to treat the sessions as a competition and not to attempt to maximise the displayed score at the expense of correct procedure. Despite this instruction, several participants adopted a competitive attitude toward the score, and this is discussed in [Section 11.9](#) as a limitation on the interpretation of the results.

Trial sequence. Each participant completed between six and fifteen trials, depending on their availability. Trials were executed consecutively within a session, with the assistance level shown on the display at the beginning of each trial. A trial consisted of one complete assembly sequence from the initial pick to the final tightening, and ended when the StageFSM reached the terminal stage. Between trials, the workspace was reset by the experiment conductor and the components were returned to their designated boxes or the participants were alternated between assembly and dismantle operations.

Experimenter interventions. When a participant became stuck, asked for clarification, or performed a stage in a way that the FSM could not resolve automatically, the experiment conductor intervened. Interventions were of three broad kinds: clarification of the current instruction, correction of an incorrect order or action, and manual advancement of a stage that the FSM had flagged as failed but that had been performed correctly. The number of interventions per trial was

logged manually and is treated as a control covariate in the statistical analysis ([Section 11.8](#)).

Data collection. For each trial, the pipeline recorded:

- A metadata JSON containing the system configuration, the FSM decisions for each stage, and per-stage metrics.
- Video of the assembly workspace.
- The assistance level assigned to that trial.

After each trial, participants completed a short questionnaire on perceived helpfulness and clarity of the assistance. At the end of the session, they completed a debrief questionnaire covering preferences across the three modes, mental demand, and the number of trials after which they felt confident without assistance. The questionnaire responses are analysed in [Section 11.8.4](#).

Order of trials. The assistance level for each trial was assigned pseudo-randomly, with the constraint that each participant experienced all three levels within their first three trials. This ensured that every participant had direct experience of all three modes early in the session and could respond to the debrief questions on the basis of comparison rather than speculation.

11.5 Objective Performance Measures

Four categories of objective measures were extracted from the pipeline for each trial: task completion time, assembly errors, trajectory-level motion metrics, and stage-level composite motion quality. All measures were computed automatically from the metadata recorded during the trial and require no manual annotation.

Task completion time. For each trial, the completion time was computed as the elapsed time between the first detected stage segment and the last. A segment is the time interval during which the MotionFSM transitions from RECORDING to WAIT_COMPLETION_CONFIRM for a given stage. Time spent waiting between stages is excluded. Task completion time is therefore a measure of active manipulation time, not total session time.

Assembly errors. Two error measures were recorded. The first is the count of stage-level failures, defined as any stage for which the StageFSM marked `stage_passed=false` at the end of the trial. A stage may fail because object detection did not find the expected component, orientation verification rejected the observed orientation, the DTW similarity of the trajectory fell below the acceptance threshold, or the expected tool was not detected. The second measure is the count of manual interventions per trial, logged by the experiment conductor

when a stage had to be corrected or advanced manually. Manual interventions are treated as a control covariate rather than a performance measure, since they reflect both operator errors and experimenter decisions.

Trajectory-level motion metrics. For each stage, the active-hand trajectory during RECORDING is processed to extract four motion-quality metrics, each capturing a different aspect of movement quality:

- **Jerk index** measures the smoothness of the motion. Higher values indicate abrupt changes in acceleration, which typically occur when the operator hesitates, corrects, or moves in a non-purposeful manner.
- **Pause ratio** is the fraction of the trajectory during which the hand velocity remains below a small threshold. Higher values indicate more time spent stationary within the segment, typically associated with uncertainty or search behaviour.
- **Micro-stop count** counts the number of brief velocity dips within the trajectory. It complements the pause ratio by capturing short interruptions that might not contribute much to the total pause time but still indicate fragmented motion.
- **Path inefficiency ratio** is the ratio of the path length actually travelled to the straight-line displacement between the start and end of the trajectory. A value of 1 indicates a straight path; higher values indicate deviation, curvature, or backtracking.

All four metrics are lower-is-better. They were chosen because they are computable directly from the hand trajectory, independent of the reference trajectory, and therefore not confounded with the DTW comparison that already contributes to the stage-level score.

Composite Motion-Quality Score. To synthesize performance across the four motion metrics $m \in \{\text{jerk, pause, stops, ineff}\}$, a unified composite score was derived in three steps: First, to reconcile differing raw units (acceleration units, fraction, integer count, and dimensionless ratio), each raw metric $x_{i,m}$ for stage row i was standardized globally over N total dataset rows:

$$z_{i,m} = \frac{x_{i,m} - \bar{x}_m}{\sigma_m} \quad (11.1)$$

where $\bar{x}_m$ and σ_m represent the global mean and standard deviation of metric m . Second, because all raw metrics are lower-is-better, standardized values were sign-inverted to express positive motion quality ($q_{i,m} > 0$ indicates above-average

performance):

$$q_{i,m} = -z_{i,m} \quad (11.2)$$

Third, the sign-inverted values were averaged row-wise over available metrics M_i :

$$\text{skill_score}_i = \frac{1}{|M_i|} \sum_{m \in M_i} q_{i,m} \quad (11.3)$$

Rows with missing metrics were excluded prior to modeling, rendering $|M_i| = 4$ globally. The resulting `skill_score` is a dimensionless metric centered at zero, expressed in standard deviation units of quality-adjusted motion. It assesses relative performance differences across conditions and trials rather than absolute motion levels.

Using a single composite rather than four distinct metrics prevents inflation of false-positive rates, captures shared variance among correlated metrics, and provides a unified "higher-is-better" scale. Global standardization preserves between-participant variability, supporting the random-intercept mixed-effects model in [Section 11.7](#).

11.6 Subjective Measures

Two questionnaires were administered. A short per-trial questionnaire captured the participant's immediate perception of the assistance during the trial. A longer debrief questionnaire at the end of the session captured overall preferences and mental demand. Both were delivered in electronic form.

Per-trial questionnaire. After each completed trial, participants rated four statements on a 7-point Likert scale (1 = strongly disagree, 7 = strongly agree, unless noted):

- The assistance provided during this trial helped me complete the task.
- The assistance was easy to understand and follow.
- I could perform this trial from memory without relying on the assistance.
- How often did you refer to the assistance during this trial? (rated on a frequency scale from Never to Constantly)

The four items were designed to capture three dimensions: perceived usefulness (items 1 and 2), self-assessed independence (item 3), and reliance behaviour (item 4). Because the no-guidance condition had no assistance available, items 3 and 4 have a different meaning in that condition working from memory and

not referring to assistance are the natural consequences of the condition, not independent observations. This is noted in the interpretation of the results.

Debrief questionnaire. At the end of the session, participants answered six questions:

- Which assistance level best supported your learning of the assembly process?
- If you were learning this assembly again for the first time, which assistance level would you prefer?
- Approximately after which trial did you feel confident performing the assembly without relying on the assistance?
- If you were to perform this assembly every day as part of your job, which assistance level would you prefer?
- Which assistance level did you find least distracting?
- Overall, how mentally demanding was the experiment? (Low / Mid / High)

These questions distinguish between the three assistance modes as experienced during the session (full, mid, no) and the participant's preference for a hypothetical real-world use case. The distinction between learning-time preference and daily-use preference is important, because the two are not necessarily the same.

Baseline and final state. The Stanford Sleepiness Scale was administered at the start and end of each session to record the participant's alertness level. These measures are not part of the primary analysis but are reported descriptively as a check that the results are not driven by within-session fatigue.

Questionnaire limitations. All questionnaire responses are subjective. The per-trial responses were collected after the trial ended, so they may be influenced by the outcome of the trial and by the participant's sense of how well they performed. The debrief responses were collected once per participant and reflect an aggregate impression rather than a condition-specific judgement. Given the small sample, these responses are reported descriptively rather than tested inferentially, and no causal claims are drawn from them.

11.7 Statistical Analysis

Structure of the data. The dataset is nested: multiple stages within each trial, and multiple trials within each participant. Treating stage-level observations as independent would inflate the effective sample size and produce biased standard

errors. To account for this structure, the analysis uses a linear mixed-effects model with a random intercept per participant. The participant is the cluster unit; stages within a trial and trials within a participant are treated as repeated observations from the same participant.

Dependent variable. The composite motion-quality score defined in [Section 11.5](#) is the dependent variable in the primary analysis. Task completion time and error counts are analysed descriptively; the sample size is not sufficient for a reliable inference test on their differences across conditions, and the interpretation of these measures is complicated by the behavioural context discussed in [Section 11.9](#).

Fixed effects. Three fixed predictors were included:

- **Condition:** A categorical predictor with three levels (full, mid, no). Full is the reference level, so the coefficients on mid and no represent the estimated difference in composite score relative to full at trial 1.
- **log global:** The natural logarithm of the trial number within the session, treated as a continuous covariate to model the learning effect. The logarithmic form was chosen because the observed improvement in the earlier sessions decelerates over trials, and a linear specification would not capture that shape.
- **interruptions:** The number of manual interventions logged for that trial, treated as a continuous covariate. This variable controls for experimenter intervention frequency, which differed across participants and conditions.

Model specification. Three nested models were fitted:

1. **Reduced model:** `skill_score~log_global+interruptions+(1|subject)`. This is the baseline model without condition, testing whether learning and interruptions alone explain the data.
2. **Primary model:** `skill_score~C(condition)+log_global+interruptions+(1|subject)`. This adds the categorical condition predictor and is the model used for the primary result.
3. **Interaction model:** `skill_score~C(condition)+log_global+interruptions+C(condition):log_global+(1|subject)`. This adds a condition-by-learning interaction, allowing the learning rate to differ across conditions.

Estimation. All three models were fitted using maximum likelihood (ML) rather than restricted maximum likelihood (REML). ML was chosen because the

analysis compares nested fixed-effect specifications via likelihood-ratio tests, and the REML likelihood is not directly comparable across models with different fixed-effect structures. The participant random intercept is estimated jointly with the fixed effects.

Hypothesis tests. Two likelihood-ratio tests are reported:

- The **assistance effect test** compares the primary model against the reduced model. A significant result indicates that the categorical condition predictor explains a meaningful amount of additional variance beyond learning and interruptions.
- The **interaction test** compares the interaction model against the primary model. A significant result indicates that the rate of learning differs across conditions.

The likelihood-ratio statistic is computed as $2(\ell_{\text{full}} - \ell_{\text{reduced}})$, with degrees of freedom equal to the difference in the number of fixed-effect parameters. Under the null hypothesis, the statistic follows a chi-squared distribution (χ^2). A significance threshold of $\alpha = 0.05$ is used.

Robustness. Several alternative specifications were tested during the analysis and are not reported here because they did not change the substantive conclusions. In particular, the choice of standardization (global vs within-subject) affects the identifiability of the random intercept, and the global specification was selected because it preserves between-participant variation. The number of interruption covariates (raw count vs severity-weighted) was also explored; the raw count is reported because it is the most straightforward operationalisation and the additional severity information did not materially change the estimated condition effect.

Software. All statistical analyses were performed in Python 3.11 using statsmodels version 0.14 and scipy version 1.11. The scripts and processed datasets accompany the thesis as supplementary material.

11.8 Results

This section reports the experimental results in the order established in [Section 11.5](#). Descriptive statistics are reported for all four categories of measures. Inferential statistics are reported only for the composite motion-quality score, where the mixed-effects model of [Section 11.7](#) provides a valid test given the study's design and sample size.

11.8.1 Task Completion Time

Task completion time was computed for each trial as the elapsed time between the first and last detected stage segment. [Table 11.1](#) reports mean, standard deviation, and median per assistance level. The pattern is monotonic: full

Table 11.1: Task completion time by assistance level ($n = 70$ trials).

Assistance	Trials	Mean (s)	SD (s)	Median (s)
Full	21	273.03	138.81	212.29
Mid	22	197.61	61.15	182.93
No	27	177.66	80.81	157.27

guidance was associated with the longest mean completion time, mid guidance an intermediate value, and no guidance the shortest. The gap between full and no was approximately 60 seconds on the mean, or about 34% of the no-guidance mean.

The standard deviation in the full condition (138.81 s) was more than twice the SD in the mid condition (61.15 s). This higher variance is relevant to the interpretation in [Section 11.9](#): it reflects substantial variability in how much operators engaged with the full-guidance prompts. Some participants completed the assembly efficiently; others paused, re-checked their own work, or re-attempted stages after a failure flag, extending their session.

Task completion time is reported descriptively. Given the study's small sample (nine participants) and the departure of several participants from the intended task framing they were instructed not to treat the sessions as a competition but several did a formal significance test on this measure would conflate task performance with behavioural variation that is not the focus of the study. [Section 11.9](#) discusses the interpretation of this measure in detail.

11.8.2 Assembly Errors

Two measures of assembly error were recorded: the number of stages flagged as failed by the StageFSM at the end of a trial, and the number of manual interventions logged by the experiment conductor. [Table 11.2](#) reports the failure count per trial by assistance level. The three conditions are nearly identical on this measure. The mean failed stages per trial ranged from 0.33 to 0.52 across conditions, and the totals ranged from 9 to 11 out of 70 trials. No condition showed a systematic advantage in error rate.

This similarity should be interpreted with care. The failure counts reflect stages that the StageFSM marked stage passed = false at the end of the trial.

Table 11.2: Failed stages per trial by assistance level.

Assistance	Trials	Mean per trial	SD	Total failed stages
Full	21	0.52	0.81	11
Mid	22	0.41	0.59	9
No	27	0.33	0.62	9

Several participants in the full and mid conditions responded to a failure flag by immediately re-attempting the stage, and a re-attempt often resolved the failure before the next stage was triggered. Conversely, some failures in the no-guidance condition may have been resolved by the operator silently, without triggering another FSM decision. The failure counts therefore reflect the FSM’s state at trial end rather than the operator’s actual accuracy, and the near-parity across conditions is a property of the detection-and-recovery behaviour as much as of the operators themselves.

The manual intervention counts are reported in [Section 11.8.3](#) as a co-variate in the model. They ranged from a maximum of 16 interventions for one participant to zero for another. The distribution was highly skewed, with the majority of stage rows having zero interventions.

11.8.3 Trajectory and Motion Metrics

Four motion-quality metrics were extracted per stage: jerk index, pause ratio, micro-stop count, and path inefficiency ratio. [Table 11.3](#) reports mean and standard deviation per assistance level. Full guidance showed the lowest mean

Table 11.3: Motion-quality metrics by assistance level (mean $\pm$ SD).

Metric	Full ($n = 207$)	Mid ($n = 227$)	No ($n = 243$)
Jerk index	3.55 ± 2.49	11.48 ± 9.20	13.74 ± 10.12
Pause ratio	0.247 ± 0.187	0.235 ± 0.159	0.209 ± 0.179
Micro-stop count	3.85 ± 6.45	4.63 ± 5.52	5.98 ± 6.97
Path inefficiency	4.35 ± 3.90	4.03 ± 2.81	4.63 ± 3.65

jerk index (3.55) and the lowest micro-stop count (3.85). Mid guidance showed the lowest path inefficiency (4.03). No guidance showed the highest jerk index (13.74), the highest micro-stop count (5.98), and the highest path inefficiency (4.63). Pause ratio showed a different pattern: no guidance had the lowest value (0.209), mid guidance an intermediate value (0.235), and full guidance the highest (0.247).

The similarity in pause ratio across conditions is worth noting. Unlike jerk and micro-stops, pause ratio measures the fraction of the segment during which the hand is essentially stationary. Operators in all three conditions paused during the assembly to check a component, to locate the next part, or to consider the next step. The full-guidance interface may have prompted additional brief pauses as operators verified the displayed instruction against the physical state of the workpiece.

Because the four metrics measure different aspects of motion quality and their scales differ widely, they were combined into a composite score as described in [Section 11.5](#). The composite is the primary dependent variable in the statistical analysis.

Mixed-effects model. The composite score was analysed with a linear mixed-effects model that includes condition, log trial number, and the per-trial interruption count as fixed predictors, and a random intercept per participant. [Table 11.4](#) reports the fixed-effect coefficients from the primary model. Full

Table 11.4: Primary mixed-effects model: composite skill score.

Predictor	Coefficient	SE	z	p
Intercept	0.020	0.064	0.308	0.758
Condition: mid (vs full)	-0.262	0.046	-5.713	< 0.001
Condition: no (vs full)	-0.430	0.048	-8.977	< 0.001
log_global	0.164	0.029	5.699	< 0.001
interruptions	-0.006	0.025	-0.218	0.828
Participant random-intercept variance	0.014	—	—	—
Residual variance	0.216	—	—	—

guidance serves as the baseline reference condition. Relative to full guidance, both mid guidance ($\beta = -0.262$) and no guidance ($\beta = -0.430$) were associated with significantly lower composite scores. The magnitude of the no-guidance deficit is roughly 1.6 times that of the mid-guidance deficit, demonstrating a clear monotonic ordering: full > mid > no.

The learning covariate was positive and statistically significant ($\beta = 0.164$, $p < 0.001$). Each unit increase in log trial number corresponded to an improvement of 0.16 units in the composite score. Across the study's scope (approximately trials 1 to 20, or log 1 to log 3), this indicates an overall improvement of approximately 0.49 units.

The interruption covariate was not statistically significant in the primary model ($\beta = -0.006$, $p = 0.828$). Controlling for assistance condition and trial number, the frequency of experimenter interventions per trial showed no measur-

able impact on the composite score.

Finally, the likelihood-ratio test comparing the primary model against a reduced model omitting the assistance condition yielded $LR = 77.22$ ($df = 2$, $p < 1.7 \times 10^{-17}$). This confirms that the assistance condition exerts a substantial and highly significant effect on overall motion quality.

Interaction model. A second model added a condition-by-log-trial interaction, allowing the learning rate to differ across conditions. Table 11.5 reports the coefficients. In this interaction model, the coefficients on mid and no rep-

Table 11.5: Interaction model: composite skill score with condition $\times$ learning interaction.

Predictor	Coefficient	SE	z	p
Intercept	-0.018	0.066	-0.278	0.781
Condition: mid (vs full)	-0.064	0.092	-0.694	0.488
Condition: no (vs full)	-0.534	0.131	-4.088	< 0.001
log_global	0.198	0.035	5.647	< 0.001
mid $\times$ log_global	-0.149	0.062	-2.413	0.016
no $\times$ log_global	0.054	0.080	0.674	0.500
interruptions	-0.008	0.026	0.018	0.986

resent the estimated condition effects at trial 1 (where $\log_global = 0$). At this baseline, mid guidance was not statistically different from full guidance ($\beta = -0.064$, $p = 0.488$), while no guidance produced a significantly lower composite score ($\beta = -0.534$, $p < 0.001$).

The interaction terms model how condition-specific performance gaps evolve with increasing $\log_global$. The mid $\times$ log_global coefficient is negative and significant ($\beta = -0.149$, $p = 0.016$), indicating that mid-guidance operators improved at a significantly slower rate than full-guidance operators. The no $\times$ log_global coefficient is positive but not significant ($\beta = +0.054$, $p = 0.500$), indicating that no-guidance operators improved at approximately the same rate as full-guidance operators.

A likelihood-ratio test evaluating the interaction model against the primary model yielded $LR = 7.46$ ($df = 2$, $p = 0.024$). The null hypothesis of equivalent learning trajectories across conditions is rejected.

Learning trajectories. Figure 11.1 plots the fitted learning curves across the three assistance conditions, with interruptions held constant at the dataset mean. Full guidance exhibits the highest baseline performance and demonstrates steady, consistent improvement. No guidance starts at a substantially lower baseline but improves at a rate comparable to full guidance, so the gap narrows

only slowly. Mid guidance starts closer to full guidance at trial 1 but improves at a significantly slower rate, so the mid-guidance trajectory flattens over the study while full guidance continues to rise.

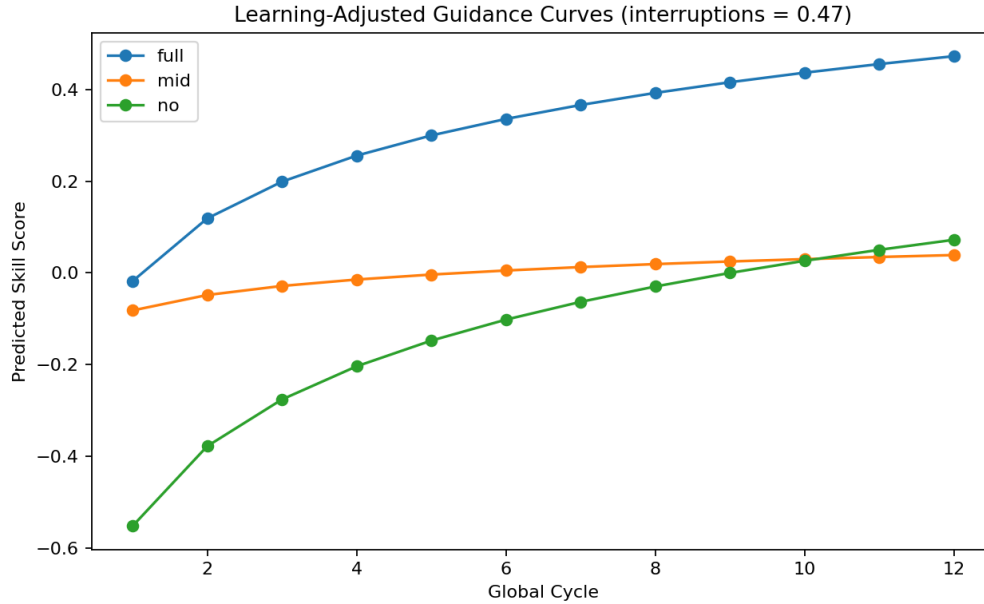


Figure 11.1: Fitted learning trajectories across guidance conditions with interruptions held at the dataset mean (0.47). Full guidance produces steady improvement; mid guidance plateaus; no guidance improves from a lower baseline.

11.8.4 Questionnaire Results

Per-trial responses. The per-trial Likert items were rated on a 1–7 scale after each trial. Table 11.6 reports the mean and standard deviation per item and per condition. Because the questions "I could perform this trial from memory" and "How often did you refer to the assistance" are only meaningful when assistance is available, they are reported here for the full and mid conditions only; the no-guidance values are shown for completeness but reflect the design of the condition, not an independent observation.

Table 11.6: Per-trial Likert responses by assistance level (mean $\pm$ SD).

Item	Full ($n = 21$)	Mid ($n = 21$)	No ($n = 27$)
Assistance helped me complete the task	5.40 $\pm$ 1.98	6.00 $\pm$ 1.33	2.12 $\pm$ 1.39
Assistance was easy to understand	4.85 $\pm$ 1.81	5.53 $\pm$ 0.96	2.36 $\pm$ 1.85
Could perform from memory	3.95 $\pm$ 2.04	3.63 $\pm$ 1.77	4.64 $\pm$ 1.96
How often referred to assistance	5.40 $\pm$ 2.14	6.05 $\pm$ 1.51	3.12 $\pm$ 2.24

Full and mid guidance were rated similarly on perceived helpfulness (5.40 vs 6.00) and ease of understanding (4.85 vs 5.53). Both were rated substantially higher than no guidance on these items (2.12 and 2.36).

On self-assessed ability to work from memory, no guidance scored highest (4.64), mid guidance lowest (3.63), and full guidance intermediate (3.95). This ordering is a direct consequence of the experimental design: in the no-guidance condition, operators necessarily worked from memory, whereas in the guided conditions the assistance was available and its use was part of the intended workflow.

Reference frequency followed the opposite pattern: high under full and mid (5.40 and 6.05), lower under no guidance (3.12). The no-guidance value is not zero because some participants reported occasional glances at the display for the pass/fail status. The two most notable findings from the per-trial data are:

1. Full and mid guidance were perceived as nearly equivalent on helpfulness and ease of understanding, despite producing measurably different objective outcomes in the mixed-effects model.
2. Operators did not rate the interface as difficult to follow even at the full-information level, where the amount of on-screen content was greatest.

Debrief responses. The debrief questionnaire was completed once per participant at the end of the session. [Table 11.7](#) reports the frequency of each response option for the four preference questions. Mid guidance was selected

Table 11.7: Debrief preferences ($n = 8$).

Question	Full	Mid	No
Best supported learning	3	5	0
Preferred if learning again	5	3	0
Preferred for daily use	1	3	4
Least distracting	2	3	3

most often as the mode that best supported learning (5 of 8), while full guidance was the most common preference for re-learning the task from scratch (5 of 8). No participant selected no guidance for either learning question.

For daily use, participants preferred lower assistance (4 of 8) or mid guidance (3 of 8), with only one preferring full. When asked which mode was least distracting, no guidance and mid guidance were chosen equally often (3 of 8 each), followed by full guidance (2 of 8).

Mental demand and sleepiness. Overall mental demand was reported as moderate by most participants (6 of 8), low by one, and high by one.

The Stanford Sleepiness Scale was administered at the start and end of each session. Six of the eight participants reported no change over the session; one reported an increase of one level, and one reported an increase of two. No participant reported a decrease. This suggests that the results are not driven by within-session fatigue, although the sample is too small to test this formally.

Because the sample size is small ($N = 8$) and the responses are subjective, no inferential tests are reported for the questionnaire data. The results are reported descriptively.

11.9 Discussion

The experimental evaluation set out to answer four research questions. This section addresses each in turn, using both the objective and subjective results, and discusses several limitations that qualify the interpretation.

RQ1: Does the level of visual assistance affect motion quality? Yes, and the effect is large. The mixed-effects analysis found a highly significant effect of condition on the composite motion-quality score ($LR = 77.22, p < 0.001$), with a monotonic ordering: full guidance produced the highest composite quality, mid guidance an intermediate value, and no guidance the lowest. The no-guidance deficit (-0.430) was substantially larger than the mid-guidance deficit (-0.262).

This supports the system’s design intent. The assistance interface is meant to reduce deviations from the expert reference trajectory by providing visual prompts at the point of action. The result confirms that it does: operators under full guidance exhibited substantially lower jerk and fewer micro-stops than operators under no guidance, while path inefficiency was similar between the two conditions.

The result also confirms that the mid-guidance mode, while less effective than full guidance, still improves motion quality relative to no guidance. This is relevant to the practical deployment of the system: even a reduced-information interface produces measurable improvements, which may be useful in contexts where a full-guidance overlay would be impractical or distracting.

RQ2: Do the three assistance levels produce different learning trajectories? Yes. The primary model demonstrates a robust learning effect across all conditions combined ($\log_{\text{global}} \beta = 0.164, p < 0.001$), and the interaction test confirms that the rate of learning differs significantly across conditions ($LR = 7.46, p = 0.024$).

The interaction is driven primarily by the mid-guidance condition. Full-guidance operators improve steadily throughout the study. No-guidance operators improve at a similar rate from a much lower baseline, so the gap between

full and no guidance narrows slowly but does not close within the observed range. Mid-guidance operators, however, plateau after the initial trials—the $\text{mid} \times \log_{\text{global}}$ coefficient is significantly negative ($\beta = -0.149, p = 0.016$), indicating that mid-guidance operators improve at approximately a quarter of the rate of full-guidance operators.

This finding is consistent with the interpretation that the mid-guidance interface provides enough information for the operator to complete the task correctly, but not enough to support continued skill acquisition. Full guidance provides the CAD projection, the region highlighting, and the reference video, which together may function as an effective training signal. Conversely, mid guidance provides only the stage instruction and pass/fail feedback—sufficient for immediate task execution, but inadequate for promoting learning beyond the baseline level.

RQ3: Does the assistance level affect task completion time or assembly errors? Task completion time varied substantially across conditions. Full guidance had the longest mean completion time (237.73 s), mid guidance an intermediate value (197.61 s), and no guidance the shortest (177.66 s). But the interpretation of this measure is complicated by two factors that emerged during the trials and are discussed in the limitations below. First, several participants treated the session as a competition despite being asked not to. Full-guidance operators in particular tended to spend extra time verifying their work against the displayed prompts, extending their trials. Second, the number of stages detected by the FSM differed across conditions: no-guidance trials averaged 9.19 stages detected, compared to 10.48 (full) and 10.78 (mid). A trial with fewer detected stages measures a shorter span of activity, which partially accounts for the shorter duration.

Error counts were similar across conditions: 0.52 failed stages per trial in full guidance, 0.41 in mid, and 0.33 in no. The near-parity is expected given the design: the StageFSM applies the same validation criteria in all conditions, and the difficulty of the task is identical. Failure counts reflect the FSM’s state at trial end, not the operator’s underlying accuracy, and participants in the guided conditions often re-attempted a stage immediately after a failure flag and resolved the failure before the next stage was triggered.

The correct summary of the error result is not that guidance fails to reduce errors, but that the measured error counts do not distinguish operator accuracy from FSM behaviour. A more direct measure would require manual inspection of the FSM transition logs for each trial, which was outside the scope of this analysis.

RQ4: How do operators perceive the three assistance levels? The per-trial and debrief responses reported in §11.8.4 address this question. Operators perceived

full and mid guidance as nearly equivalent. On the two items that rate the assistance directly helpfulness and ease of understanding the two conditions scored within 0.7 points of each other on a 1–7 scale. This is despite the significant objective difference between them: full guidance produced a mid-vs-full composite deficit of -0.262 , which the mixed model finds highly significant.

This dissociation between perception and performance is one of the more interesting findings of the study. It suggests that operators who used mid guidance did not perceive themselves as performing worse than operators who used full guidance. If confirmed in a larger study, this would have practical implications: users of a reduced-information interface may not know when they are deviating from the reference trajectory, which is precisely the kind of error the system is designed to catch.

The debrief responses reveal a further nuance. Mid guidance was selected most often as the mode that best supported learning, while full guidance was the most common preference for re-learning from scratch. This suggests that operators found mid guidance sufficient to learn the task in the first place, but valued full guidance when they needed to refresh their knowledge. For daily use, most operators preferred less assistance than they preferred for learning, consistent with the observation that assistance is most valuable during the acquisition phase and less so once the task is internalised.

Limitations: Several limitations qualify the results. *Sample size.* Eight participants is a small sample for a mixed-effects analysis, and the study is underpowered for detecting moderate interaction effects. The significant results reported in [Section 11.8.3](#) are robust to this limitation because the effect size is large, but they should still be interpreted in light of this constraint.

Protocol adherence. Participants were instructed not to treat the sessions as a competition and to follow the guidance displayed on the screen. Many did not. Some participants adopted a competitive attitude and re-performed actions to improve their displayed score; others skipped or ignored guidance prompts and completed the assembly from their own understanding. The three conditions therefore represent the availability of guidance at three levels rather than compliance with guidance at three levels. The measured condition effects reflect the value of having access to guidance, not of using it.

Emotional response to failure states. Several participants reacted emotionally to the failure flags displayed by the interface, and this appeared to affect subsequent performance in the trial. The effect was not systematically logged and cannot be quantified from the current data, but it plausibly contributes to the variance observed in the full and mid conditions.

Completion time as a measure. The longer completion times under full guidance

are not a reliable indicator of inefficiency. They reflect extra verification actions, in-the-moment consultation of the interface, and re-attempts after a failure flag. The absence of a stage-level timing breakdown prevents a direct decomposition of the extra time into these components.

Error measurement. The failure counts reflect the FSM's decision at trial end rather than the operator's actual accuracy. Re-attempts, false failure flags, and silent recoveries all affect the recorded count. A more reliable error measure would require a per-stage manual review of the FSM transition logs, which was not performed.

Subjective data. The questionnaire results are self-reported and were collected after each trial (per-trial items) or at the end of the session (debrief). The sample is too small for inferential testing, and the responses may be influenced by the outcome of the trial or by the participant's overall impression of the session rather than a stable assessment of the assistance level.

Task generalisability. The study used a single assembly task (a pneumatic piston) with a fixed number of stages and a single workstation layout. Whether the observed patterns transfer to other assembly tasks, larger workpieces, or production environments with higher tempo remains open.

Summary: The system's assistance is effective: full guidance produces measurably better motion quality relative to no guidance with a composite-score deficit of approximately 0.43 for no guidance relative to full, and operators perceive both guided conditions as helpful. The learning trajectory differs by condition: full-guidance operators improve steadily, no-guidance operators improve at a similar rate from a lower baseline, and mid-guidance operators plateau after the initial trials. Operators prefer less assistance for daily use than for learning. The main caveats concern the small sample, the partial non-adherence to the study protocol, and the difficulty of measuring errors and completion time in a way that reflects operator accuracy rather than system behaviour.

Chapter 12

Conclusions

12.1 Summary of the Internship

The thesis focused on designing an image-based system that would monitor and assist manual assembly processes. This process included researching various computer vision techniques and machine learning techniques and combining the chosen techniques into a system using one RGB camera. The internship included all stages of system development including the literature review, system design, data acquisition, object detection, pose verification, staging, trajectory analysis and operator guidance.

12.2 Principal Results

The final system is able to monitor the assembly process using object detection, hand-trajectory detection, interaction detection based on ROI, state modelling, and pose classification tailored to the task at hand.

The system was evaluated at each stage of its development: the object detector achieved precision 0.9674, the orientation classifiers exceeded 95% accuracy, and the assistance effect was significant in the experimental evaluation ($LR = 73.01$, $p < 0.001$). The obtained system allows to verify stages of the process in real time and provide configurable guidance for the studied assembly task.

12.3 Technical Contributions

The key technical aspect of the internship is the combination of a variety of vision and temporal-processing methods into a modular assembly monitoring framework. In particular, the system combines: RGB-based object detection

and hand tracking; hybrid ROI and hand-based interaction detection; FSM-based motion and assembly-stage management; trajectory analysis using Dynamic Time Warping; task-specific pose classification; configurable visual guidance with CAD-based region projections; and synthetic data generation for detector training.

12.4 Skills and Knowledge Acquired

The internship provided practical experience in computer vision, deep learning, dataset preparation, object detection, pose analysis, trajectory processing, and finite-state modelling. In addition to the above, significant experience was gained with Python, OpenCV, machine learning libraries, CAD-based synthetic data generation, and real-time system integration. Beyond the technical aspects, the work provided experience in experimental design, evaluation of alternative approaches, debugging complex vision pipelines, and developing a research-oriented engineering solution.

Appendix A

Assembly Procedure

Assembly and dismantling procedures are defined as ordered sequences of structured JSON configuration files. Each stage specifies start and completion conditions, workspace regions, expected tools, active-object candidates, and validation parameters. [Table A.1](#) summarises the complete assembly and dismantling sequences in human-readable form. The JSON representation ([Figure A.1](#)) entries below illustrate the underlying schema used for all stages.

```
"modes": {
  "dismantle": {
    "stages": [
      {
        "stage_name": "
          D01_remove_piston_rod_nut_drop",
        "manual_start_allowed": true,
        "start_tool_regions": [],
        "completion_regions": [
          "box_piston_rod_nut"
        ],
        "completion_roles": [
          "part_drop"
        ],
        "cooldown_frames": 1,
        "hand_scope": "any",
        "expected_parts": [
          "nut",
          "tie_rod_assembly",
          "front_cover",
          "piston_rod"
        ],
        "expected_tools": [
          "NONE"
        ],
        "active_part_candidates": [
          "nut"
        ],
        "active_tool_candidates": [
          "NONE"
        ],
        "enforce_active_part_match": false,
        "enforce_active_tool_match": false
      },
    ]
  }
}
```

(a) Dismantle Stage Configuration

```
"assembly": {
  "stages": [
    {
      "stage_name": "A01_place_bottom_tie_rod_nuts",
      "manual_start_allowed": false,
      "start_roles": [
        "part_pick"
      ],
      "start_source_regions": [
        "box_bottom_tie_rod_nuts"
      ],
      "completion_mode": "next_stage_pick",
      "next_stage_trigger_roles": [
        "part_pick"
      ],
      "next_stage_source_regions": [
        "box_rear_cover"
      ],
      "cooldown_frames": 1,
      "min_stage_valid_time": 6.0,
      "start_recent_hit_window": 12,
      "start_recent_hit_count": 3,
      "hand_scope": "any",
      "expected_parts": [
        "bttm_tie_rod_nut"
      ],
      "expected_tools": [
        "NONE"
      ],
      "active_part_candidates": [
        "bttm_tie_rod_nut"
      ],
      "active_tool_candidates": [
        "NONE"
      ],
      "enforce_active_tool_match": false,
      "enforce_active_part_match": false
    },
  ]
}
```

(b) Assembly Stage Configuration

Figure A.1: Comparison of Dismantle and Assembly JSON stage definitions.

Table A.1: Assembly and dismantling procedure represented by the implemented stage configuration.

ID	Operation	Stage activation and completion	Correspondence with physical procedure
Assembly			
A01	Place bottom tie-rod nuts	Starts on a part-pick event in the bottom tie-rod nut source region. Completion is associated with the subsequent pick from the rear-cover region.	Pick the four bottom tie-rod nuts and place them in the square holder.
A02	Place rear cylinder cover	Starts on a part-pick event from the rear-cover region. Completion is associated with the subsequent pick from the cylinder-tube region.	Place the rear cylinder cover over the four tie-rod nuts in the square holder. The cover orientation is also checked.
A03	Place cylinder tube	Starts on a part-pick event from the cylinder-tube region. Completion is associated with the subsequent pick from the piston-rod region.	Place the cylinder tube (housing) over the rear-cover assembly.

Continued on next page

ID	Operation	Stage activation and completion	Correspondence with physical procedure
A04	Place piston rod in rod holder	Starts on a part-pick event from the piston-rod region. Completion is associated with the subsequent pick from the piston region.	Pick the piston rod and place it in the rod holder. The piston-rod orientation is checked.
A05	Place piston on piston rod	Starts on a part-pick event from the piston region. Completion is associated with the subsequent pick from the mounting-screw region.	Pick the piston and place it onto the piston rod.
A06	Place mounting screw on piston	Starts on a part-pick event from the mounting-screw region. Completion is associated with the subsequent tool pick from the tool region.	Pick the piston mounting screw and place it in the piston.
A07	Tighten mounting screw and insert subassembly	Starts on a tool-pick event from the tool region and requires the small L screwdriver. The stage combines tightening and subassembly insertion into a single monitored stage.	Use the small L screwdriver to tighten the piston mounting screw, then insert the resulting piston-rod subassembly into the cylinder-tube assembly.

Continued on next page

ID	Operation	Stage activation and completion	Correspondence with physical procedure
A08	Place front cylinder cover	Starts on a part-pick event from the front-cover region. Completion is associated with the subsequent pick from the tie-rod-assembly region.	Pick the front cylinder cover and place it on the assembled setup. The cover orientation is checked.
A09	Place long tie-rod assemblies	Starts on a part-pick event from the tie-rod-assembly region. Completion is associated with the subsequent tool pick from the tool region.	Pick the four long tie-rod assemblies and place them through the corresponding holes. The operation may be performed individually or as a grouped manipulation.
A10	Tighten tie-rod assemblies	Starts on a tool-pick event from the tool region and requires the large L screwdriver. Completion is associated with the subsequent pick from the piston-rod-nut region.	Use the large L screwdriver to tighten the four tie-rod assemblies and return the screwdriver to the tool region.
A11	Tighten top bolt	Starts on a part-pick event from the piston-rod-nut region. The stage uses a terminal completion mode and is completed at the assembly target.	Pick the top bolt and tighten it onto the piston rod. This is the terminal assembly stage.

Continued on next page

ID	Operation	Stage activation and completion	Correspondence with physical procedure
Dismantling			
D01	Remove piston-rod nut	The stage is manually startable and is completed by a part-drop event in the piston-rod-nut region.	Remove the piston-rod nut and drop it into its designated box.
D02	Unscrew tie-rod assemblies and return large L screwdriver	Starts with a tool pick from the tool region and requires the large L screwdriver. Completion is a tool-drop event in the tool region.	Pick the large L screwdriver, unscrew the four tie-rod assemblies, and return the screwdriver to its drop region.
D03	Remove tie-rod assemblies	Completed by a part-drop event in the tie-rod-assembly region.	Remove the four tie-rod-nut/tie-rod assemblies and drop them into their designated box.
D04	Remove front cylinder cover	Completed by a part-drop event in the front-cover region.	Remove the front cylinder cover and place it in its drop box.
D05	Remove rear cylinder cover	Completed by a part-drop event in the rear-cover region.	Remove the rear cylinder cover and place it in its drop box. This corresponds to the rear-cover removal within the physical procedure involving the cylinder-tube and piston-rod assembly.

Continued on next page

ID	Operation	Stage activation and completion	Correspondence with physical procedure
D06	Remove bottom tie-rod nuts	Completed by a part-drop event in the bottom tie-rod-nut region.	Remove the four bottom tie-rod nuts remaining on the square holder and drop them into their designated box.
D07	Unscrew mounting screw and return small L screwdriver	Starts with a tool pick from the tool region and requires the small L screwdriver. Completion is a tool-drop event in the tool region.	Pick the small L screwdriver, remove the piston mounting screw, and return the screwdriver to its drop region.
D08a	Drop mounting screw	Completed by a part-drop event in the mounting-screw region. It belongs to an unordered group with D08b.	Remove the piston mounting screw and drop it into its designated box.
D08b	Drop piston	Completed by a part-drop event in the piston region. It belongs to the same unordered group as D08a.	Remove the piston and drop it into its designated box. D08a and D08b are intentionally not forced into a fixed order.
D09	Remove cylinder tube	Completed by a part-drop event in the cylinder-tube region.	Remove the cylinder tube (housing) and drop it into its designated box.
D10	Remove piston rod	Completed by a part-drop event in the piston-rod region.	Remove the piston rod and drop it into its designated box.

Appendix B

System Parameters

All numeric parameters used by the system are centralised in a single configuration module. Tables [B.1](#) through [B.5](#) list the parameter names, values, units, and descriptions for the deployed configuration. Parameters marked “disabled” are implemented but not active in the final system (see Chapters [5–7](#) for rationale).

Table B.1: Motion and interaction trigger parameters.

Parameter	Value	Unit	Description
MOTION_TRIGGER_ENABLED	True	—	Enable ROI motion trigger
MOTION_BASELINE_FRAMES	40	frames	Rolling baseline window size
MOTION_DIFF_THRESHOLD	12	8-bit grey	Pixel intensity difference threshold T
MOTION_SCORE_THRESHOLD	0.01	fraction	Region motion score threshold
MOTION_DWELL_FRAMES	4	frames	Frames a region must remain active
MOTION_HISTORY_FRAMES	6	frames	Motion history window
MOTION_BLUR_KERNEL	(5, 5)	pixels	Gaussian blur kernel
MOTION_GLOBAL_SUPPRESS_THRESHOLD	0.02	fraction	Frame-wide motion suppression threshold
MOTION_GLOBAL_RELATIVE_FACTOR	1.0	—	Discount factor under global motion
MOTION_STRONG_SCORE_THRESHOLD	0.20	fraction	Strong local motion override threshold
MOTION_STRONG_RELATIVE_FACTOR	1.5	—	Softer discount under strong local motion
MOTION_LOG_INTERVAL_FRAMES	10	frames	Debug log interval
MOTION_FOCUS_IGNORE_FRAMES	40	frames	Startup frames ignored

Table B.2: Colour and depth trigger parameters (disabled by default).

Parameter	Value	Unit	Description
COLOR_TRIGGER_ENABLED	False	—	Colour trigger disabled in final config
COLOR_BASELINE_FRAMES	30	frames	Colour baseline window
COLOR_DELTA_THRESHOLD	18.0	HSV distance	Colour trigger threshold τ_c
COLOR_DELTA_RANGE	40.0	HSV distance	Normalisation range
DEPTH_TRIGGER_ENABLED	False	—	Depth trigger disabled in final config
DEPTH_BASELINE_FRAMES	30	frames	Depth baseline window
DEPTH_DELTA_MM	15.0	mm	Depth difference threshold τ_d
DEPTH_DWELL_FRAMES	6	frames	Frames depth must remain active
DEPTH_FALLBACK_TO_RGB	True	—	Fall back to RGB if depth unavailable
DEPTH_ROI_OVERLAY_ENABLED	True	—	Depth ROI overlay for debugging
FUSED_TRIGGER_THRESHOLD	0.65	—	Fused trigger threshold
FUSED_WEIGHT_COLOR	0.6	—	Colour weight in fused trigger
FUSED_WEIGHT_DEPTH	0.2	—	Depth weight in fused trigger
FUSED_WEIGHT_RGB	0.2	—	RGB weight in fused trigger

Table B.3: MotionFSM parameters.

Parameter	Value	Unit	Description
MOTION_FSM_WARMUP_FRAMES	10	frames	Startup frames ignored
MOTION_FSM_MIN_CONSECUTIVE_VISIBLE	3	frames	Consecutive in-work frames for start
MOTION_FSM_START_NO_ACTIVITY_FRAMES	10	frames	Low-activity start tolerance
MOTION_FSM_MIN_CONSECUTIVE_DROP_HITS	2	frames	Drop region hits for completion
MOTION_FSM_MIN_RECORDING_FRAMES	15	frames	Minimum recording duration
MOTION_FSM_MAX_HAND_JUMP	0.35	norm. units	Maximum one-frame hand displacement
MOTION_FSM_ACTIVITY_V_MIN	0.003	norm. units/frame	Minimum activity threshold $v_{\min}$
MOTION_FSM_PAUSE_TOLERANCE_FRAMES	20	frames	Low-activity tolerance
MOTION_FSM_OCCLUSION_TOLERANCE_FRAMES	30	frames	Invisibility tolerance
MOTION_FSM_DONE_COOLDOWN_FRAMES	8	frames	Post-completion cooldown
MOTION_FSM_DROP_WINDOW_FRAMES	10	frames	Drop-hit rolling window
MOTION_FSM_VERSION	“v2”	—	FSM version
MOTION_FSM_START_RECENT_HIT_WINDOW	12	frames	Recent-hit window

Continued on next page

Parameter	Value	Unit	Description
MOTION_FSM_START_RECENT_HIT_COUNT	3	hits	Recent-hit count threshold
MOTION_FSM_REQUIRE_ANCHOR_BEFORE_COMPLETION	True	—	Require anchor before completion
MOTION_FSM_ANCHOR_STARTUP_GRACE_FRAMES	20	frames	Anchor startup grace
MOTION_FSM_STRICT_MOTION_HITS_ONLY	True	—	Require motion hits only
MOTION_FSM_DEBUG_LOGGING_ENABLED	True	—	Debug logging

Table B.4: Stage scoring, DTW, and synchronisation parameters.

Parameter	Value	Unit	Description
STAGE_SCORE_THRESHOLD	0.90	—	Stage pass/fail threshold τ
STAGE_WEIGHTS[“dtw”]	1.0	—	DTW weight in stage score
STAGE_WEIGHTS[“tool”]	0.0	—	Tool score weight
STAGE_WEIGHTS[“smoothness”]	0.05	—	Trajectory smoothness weight
STAGE_WEIGHTS[“efficiency”]	0.03	—	Path efficiency weight
STAGE_WEIGHTS[“temporal_stability”]	0.02	—	Temporal stability weight

Continued on next page

Parameter	Value	Unit	Description
SYNC_MAX_TIME_DIFF	0.05	s	Maximum hand-tool timestamp mismatch
TOOL_HAND_DISTANCE_THRESHOLD	0.10	norm. units	Hand-tool association distance
STAGE_THRESHOLDS_BY_STAGE	{“StageA”: 0.80, “StageB”: 0.70, “StageC”: 0.70}	—	Per-stage threshold overrides (example)

Table B.5: Active-hand selection, MediaPipe, and workspace parameters.

Parameter	Value	Unit	Description
DOMINANT_HAND	“RIGHT”	—	Configured dominant hand
ANCHOR_MODE_DEFAULT	“wrist_index_midpoint”	—	Hand anchor definition
ACTIVE_HAND_POLICY_DEFAULT	“dominant”	—	Active-hand selection policy
ACTIVE_HAND_WEIGHT_DOMINANT	0.10	—	Dominant-hand weight w_1
ACTIVE_HAND_WEIGHT_VELOCITY	0.80	—	Velocity weight w_2
ACTIVE_HAND_WEIGHT_TOOL	0.10	—	Tool-proximity weight w_3
ACTIVE_HAND_VELOCITY_NORM_SCALE	0.06	norm. units	Velocity normalisation scale

Continued on next page

Parameter	Value	Unit	Description
ACTIVE_HAND_TOOL_DISTANCE_NORM	0.30	norm. units	Tool proximity normalisation
ACTIVE_HAND_SWITCH_SCORE_MARGIN	0.10	—	Switch hysteresis margin
ACTIVE_HAND_SWITCH_CONFIRM_FRAMES	2	frames	Switch confirmation frames
HANDEDNESS_FALLBACK_IMAGE_MIRRORED	True	—	Image mirroring for handedness fallback
HANDEDNESS_SWAP_LABELS	True	—	Swap detector handedness labels
WORK_REGION	(0.05, 0.05, 0.95, 0.95)	norm.	Work region bounds
DROP_REGION	(0.60, 0.00, 1.00, 0.38)	norm.	Drop region bounds
MP_MAX_NUM_HANDS	2	—	Maximum hands tracked
MP_MODEL_COMPLEXITY	0	—	MediaPipe model complexity
MP_MIN_DETECTION_CONFIDENCE	0.5	—	MediaPipe detection confidence
MP_MIN_TRACKING_CONFIDENCE	0.5	—	MediaPipe tracking confidence
NO_HAND_FRAMES_TO_SPLIT	200	frames	Full-video split threshold
REARM_NO_HAND_FRAMES	1	frames	Drop-exit rearm gate
MIN_SEGMENT_POINTS	50	points	Minimum segment size
DELAY_MS	40	ms	Display delay

Appendix C

Questionnaires

Appendix D

Additional Experimental Results

quality_summary: frames=14 median_valid_depth=100.0% median_proj_mask_ratio=0.60 flagged=14											
idx	frame	conf	mask_wh	area	valid_depth	med_depth_mm	pose_z	dz_mm	proj_wh	proj/ mask	flags
00	260	0.95	66x59	2564	100.0	1138.0	1.240	102.0	44.4x55.4	0.63	Z_DEPTH_MISMATCH
01	270	0.95	66x59	2565	100.0	1138.0	1.240	102.0	51.0x50.7	0.66	Z_DEPTH_MISMATCH
02	280	0.95	66x60	2588	100.0	1118.0	1.240	122.0	53.8x51.7	0.70	Z_DEPTH_MISMATCH
03	290	0.94	65x64	2632	100.0	1099.0	1.240	141.0	42.3x49.2	0.50	CAD_TOO_SMALL, Z_DEPTH_MISMATCH
04	300	0.94	68x67	2931	100.0	1118.0	1.240	122.0	56.1x50.3	0.62	Z_DEPTH_MISMATCH
05	310	0.96	59x67	2628	100.0	1138.0	1.240	102.0	48.3x60.3	0.74	Z_DEPTH_MISMATCH
06	320	0.96	59x66	2625	100.0	1118.0	1.240	122.0	45.9x44.4	0.52	CAD_TOO_SMALL, Z_DEPTH_MISMATCH
07	330	0.95	60x68	2774	100.0	1138.0	1.240	102.0	46.7x44.4	0.51	CAD_TOO_SMALL, Z_DEPTH_MISMATCH
08	340	0.91	63x70	3019	100.0	1138.0	1.240	102.0	51.2x39.3	0.46	CAD_TOO_SMALL, Z_DEPTH_MISMATCH
09	350	0.55	75x59	2773	99.5	1138.0	1.240	102.0	56.1x54.9	0.70	Z_DEPTH_MISMATCH
10	360	0.93	74x64	2881	100.0	1138.0	1.240	102.0	44.4x41.5	0.39	CAD_TOO_SMALL, Z_DEPTH_MISMATCH
11	370	0.95	65x58	2588	100.0	1158.0	1.240	82.0	46.7x40.5	0.50	CAD_TOO_SMALL, Z_DEPTH_MISMATCH
12	380	0.94	65x58	2577	100.0	1118.0	1.240	122.0	43.6x50.9	0.59	Z_DEPTH_MISMATCH
13	390	0.94	65x58	2571	100.0	1138.0	1.240	102.0	51.5x58.1	0.79	Z_DEPTH_MISMATCH

Listing D.1: Execution Log: Pose Z = 1.24 m

quality_summary: frames=14 median_valid_depth=100.0% median_proj_mask_ratio=0.71 flagged=2											
idx	frame	conf	mask_wh	area	valid_depth	med_depth_mm	pose_z	dz_mm	proj_wh	proj/ mask	flags
00	260	0.95	66x59	2564	100.0	1138.0	1.140	2.0	48.4x60.2	0.75	-

01	270	0.95	66x59	2565	100.0	1138.0	1.140	2.0	55.7x55.0	0.79
		-								
02	280	0.95	66x60	2588	100.0	1118.0	1.140	22.0	58.8x56.1	0.83
		-								
03	290	0.94	65x64	2632	100.0	1099.0	1.140	41.0	46.5x53.7	0.60
		-								
04	300	0.94	68x67	2931	100.0	1118.0	1.140	22.0	60.8x54.6	0.73
		-								
05	310	0.96	59x67	2628	100.0	1138.0	1.140	2.0	52.4x65.7	0.87
		-								
06	320	0.96	59x66	2625	100.0	1118.0	1.140	22.0	49.7x48.4	0.62
		-								
07	330	0.95	60x68	2774	100.0	1138.0	1.140	2.0	51.3x48.5	0.61
		-								
08	340	0.91	63x70	3019	100.0	1138.0	1.140	2.0	55.6x42.5	0.54
		CAD_TOO_SMALL								
09	350	0.55	75x59	2773	99.5	1138.0	1.140	2.0	61.2x59.6	0.82
		-								
10	360	0.93	74x64	2881	100.0	1138.0	1.140	2.0	48.6x45.1	0.46
		CAD_TOO_SMALL								
11	370	0.95	65x58	2588	100.0	1158.0	1.140	-18.0	51.2x44.1	0.60
		-								
12	380	0.94	65x58	2577	100.0	1118.0	1.140	22.0	47.8x55.2	0.70
		-								
13	390	0.94	65x58	2571	100.0	1138.0	1.140	2.0	56.3x63.4	0.95
		-								

Listing D.2: Execution Log: Pose Z = 1.14 m

quality_summary: frames=4 median_valid_depth=100.0% median_proj_mask_ratio=1.36 flagged=0										
idx	frame	conf	mask_wh	area	valid_depth	med_depth_mm	pose_z	dz_mm	proj_wh	proj/ mask flags
00	0	na	61x74	3761	100.0	952.0	0.974	22.2	62.0x79.9	1.10
		-								
01	1	na	90x92	5485	100.0	779.0	0.799	19.6	100.1x105.3	
	1.27	-								
02	2	na	88x88	4889	100.0	853.0	0.874	21.5	103.9x107.2	
	1.44	-								
03	3	na	81x80	4041	100.0	937.0	0.952	14.6	100.8x98.4	
	1.53	-								

Listing D.3: Execution Log: Clean Synthetic Data

quality_summary: frames=4 median_valid_depth=85.8% median_proj_mask_ratio=1.35 flagged=0										
idx	frame	conf	mask_wh	area	valid_depth	med_depth_mm	pose_z	dz_mm	proj_wh	proj/ mask flags
00	0	na	61x74	3761	85.9	954.0	0.977	22.6	61.3x78.5	1.07
		-								
01	1	na	90x92	5485	86.3	780.0	0.803	22.8	99.9x105.1	
	1.27	-								
02	2	na	88x88	4889	85.6	855.0	0.877	21.8	103.5x106.7	
	1.43	-								
03	3	na	81x80	4041	85.2	941.0	0.953	11.7	100.2x98.1	
	1.52	-								

Listing D.4: Execution Log: Synthetic + Mild Depth Corruption

```

quality_summary: frames=4 median_valid_depth=65.7% median_proj_mask_ratio
                 =3631.24 flagged=4
idx frame conf mask_wh area valid_depth med_depth_mm pose_z dz_mm proj_wh proj/
mask flags
00 0      na  31x44   906  66.1          951.0          0.000 -951.0 2611.6x5002.7
    9578.51 LOW_DEPTH, POSE_Z_ZERO, CAD_TOO_LARGE, Z_DEPTH_MISMATCH
01 1      na  56x55   1729 65.6          776.0          0.820  43.8 97.5x103.2
    3.27    LOW_DEPTH, CAD_TOO_LARGE
02 2      na  54x48   1512 65.7          851.0          0.895  44.0 101.2x104.9
    4.09    LOW_DEPTH, CAD_TOO_LARGE
03 3      na  45x40   1102 65.4          934.0          0.000 -934.0 2611.6x5002.7
    7258.38 LOW_DEPTH, POSE_Z_ZERO, CAD_TOO_LARGE, Z_DEPTH_MISMATCH

```

Listing D.5: Execution Log: Synthetic + Heavy Depth Corruption

```

quality_summary: frames=90 median_valid_depth=42.1% median_proj_mask_ratio=0.04
                 flagged=90
idx frame conf mask_wh area valid_depth med_depth_mm pose_z dz_mm proj_wh proj/
mask flags
00 100    0.97 143x129 12857 67.0          1274.0          1.342  67.6 45.1x50.1 0.12
    LOW_DEPTH, CAD_TOO_SMALL, Z_DEPTH_MISMATCH
01 110    0.97 164x163 16897 55.7          1677.0          1.938  261.2 31.7x36.9 0.04
    LOW_DEPTH, CAD_TOO_SMALL, Z_DEPTH_MISMATCH
02 120    0.97 168x173 17447 46.0          1770.0          2.026  256.3 32.8x26.8 0.03
    LOW_DEPTH, CAD_TOO_SMALL, Z_DEPTH_MISMATCH
03 130    0.97 183x187 20326 36.9          1931.0          1.931  -0.3 33.9x39.6 0.04
    LOW_DEPTH, CAD_TOO_SMALL
04 140    0.97 192x192 22284 35.8          1448.0          1.721  273.0 34.3x43.5 0.04
    LOW_DEPTH, CAD_TOO_SMALL, Z_DEPTH_MISMATCH
05 150    0.98 209x208 26218 46.9          1554.0          1.683  129.3 36.5x46.7 0.04
    LOW_DEPTH, CAD_TOO_SMALL, Z_DEPTH_MISMATCH
06 160    0.97 209x210 26678 35.7          1300.0          1.508  208.3 44.9x38.1 0.04
    LOW_DEPTH, CAD_TOO_SMALL, Z_DEPTH_MISMATCH

```

Listing D.6: Execution Log: Camera Height = 30 cm

```

quality_summary: frames=37 median_valid_depth=100.0% median_proj_mask_ratio=0.63
                 flagged=10
idx frame conf mask_wh area valid_depth med_depth_mm pose_z dz_mm proj_wh proj/
mask flags
00 30     0.93 70x71   2680 100.0          1138.0          1.159  20.6 54.6x63.8 0.70
    -
01 40     0.93 70x72   2893 100.0          1118.0          1.121   3.1 57.6x55.3 0.63
    -
02 50     0.94 71x72   2870 100.0          1138.0          1.163  24.9 57.1x62.0 0.69
    -
03 60     0.93 89x78   4615 100.0          1118.0          1.166  48.5 62.3x62.4 0.56
    -
04 70     0.94 74x78   4086 100.0          1118.0          1.163  44.5 56.3x62.8 0.61
    -
05 80     0.94 82x78   4654 100.0          1138.0          1.138  -0.2 60.7x57.0 0.54
    CAD_TOO_SMALL
06 90     0.93 68x79   3258 100.0          1118.0          1.140  22.4 60.3x61.2 0.69
    -

```

Listing D.7: Execution Log: Camera Height = 140 cm (Actual Workstation View)

Appendix E

Software and Hardware Configuration

Table E.1 summarises the principal fields used in the stage descriptors, and Table E.2 summarises StageFSM and Configuration Elements. Parameters that define numerical temporal thresholds are configured according to the values reported in Appendix B.

Table E.1: Stage descriptor parameters used by the FSM configuration.

Parameter	Applicable stages	Description
stage_name	All	Unique identifier assigned to the stage.
manual_start_allowed	All	Specifies whether the stage may be initiated manually rather than through automatic start detection.
start_roles	Assembly	Defines the interaction roles accepted as valid stage-start events, such as <code>part_pick</code> or <code>tool_pick</code> .
start_source_regions	Assembly	Specifies the workspace regions from which the expected part or tool must be picked to initiate the stage.
start_tool_regions	Dismantle	Specifies the regions in which the expected tool must be detected for stage initiation.
completion_mode	Assembly	Defines how stage completion is determined. The supplied configuration uses <code>next_stage_pick</code> and <code>terminal_end</code> .

Continued on next page

Table E.1 – continued

Parameter	Applicable stages	Description
next_stage_trigger_roles	Assembly	Defines the interaction role that triggers the following stage and therefore completes the current stage when next_stage_pick is used.
next_stage_source_regions	Assembly	Specifies the source region associated with the trigger of the following stage.
completion_regions	Dismantle / terminal	Defines the workspace regions in which a valid completion event must occur.
completion_roles	Dismantle / terminal	Defines the interaction roles accepted as completion events, such as part_drop, tool_drop, or assembly_target.
completion_support_regions	Terminal	Defines additional regions used to support validation of a terminal completion condition.
cooldown_frames	All	Number of frames for which a new stage interaction is suppressed after a stage transition.
min_stage_valid_time	Selected stages	Minimum elapsed time required before the stage can be considered valid for completion.
start_recent_hit_window	Selected assembly stages	Number of recent frames considered when applying the rolling-window start condition.
start_recent_hit_count	Selected assembly stages	Number of valid start detections required within the rolling start window.
hand_scope	All	Specifies which tracked hand(s) are eligible to satisfy the stage conditions.
expected_parts	All	Lists the parts expected to be relevant or present during the stage.
expected_tools	All	Lists the tools expected to be relevant during the stage.
active_part_candidates	All	Defines the parts considered as candidates for the active manipulation target.
active_tool_candidates	All	Defines the tools considered as candidates for the active manipulation tool.

Continued on next page

Table E.1 – continued

Parameter	Applicable stages	Description
<code>expected_poses</code>	Selected assembly stages	Defines the expected orientation class together with the temporal window and confidence/matching requirements used for pose verification.
<code>enforce_expected_pose_match</code>	Selected stages	Specifies whether the configured pose condition is enforced as a stage constraint.
<code>enforce_active_part_match</code>	All	Specifies whether the detected active part must match the configured part candidate.
<code>enforce_active_tool_match</code>	All	Specifies whether the detected active tool must match the configured tool candidate.
<code>unordered_group_id</code>	Unordered stages	Identifies the unordered group to which a stage belongs.
<code>unordered_group_members</code>	Unordered stages	Lists all stage descriptors belonging to the same unordered group.

Table E.2: Implementation and configuration elements for unordered stage progression.

Element	Type	Description and role
unordered_group_lockout	Implementation mechanism	Prevents already completed members from being counted again during group progression.
unordered_group_required_count	Implementation mechanism	Defines the number of unique members required to complete the unordered group.
_match_unordered_start()	MotionFSM function	Tests the remaining members and dynamically identifies which unordered stage has been started.
_match_unordered_completion()	MotionFSM function	Tests completion conditions for remaining members and identifies the completed stage independently of index order.
_postprocess_unordered_segment_progress()	Pipeline function	Processes unordered-stage completion, updates group progress, selects the next FSM reference stage, and updates interface feedback.
completed_unordered_members	Runtime state	Stores completed group members and prevents them from being considered again.
current_stage_index	Runtime state	Identifies the current FSM reference; it advances normally for ordered stages and is reassigned to a remaining member for unordered groups.
Remaining-member set	Runtime state	Set of group members not yet completed and therefore still eligible for matching.
Next-stage injection	Pipeline mechanism	Updates the active stage after completion of an unordered member while preserving arbitrary execution order.
Task-status feedback	Interface mechanism	Propagates current task state and unordered-group progress to the guidance interface.

Bibliography

- [1] European Commission, Directorate-General for Research and Innovation, M. Breque, L. De Nul, and A. Petridis, *Industry 5.0 – Towards a sustainable, human-centric and resilient European industry*. Publications Office of the European Union, 2021.
- [2] R. Folgado, P. Peças, and E. Henriques, “Mapping workers’ performance to analyse workers heterogeneity under different workflow policies,” *Journal of Manufacturing Systems*, vol. 36, pp. 27–34, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0278612515000187>
- [3] D. Aganian, M. Kohler, S. Baake, M. Eisenbach, and H. M. Groß, “How object information improves skeleton-based human action recognition in assembly tasks,” in *2023 International Joint Conference on Neural Networks*. IEEE, 2023, pp. 1–9.
- [4] Y. Kawase and M. Hashimoto, “Analysis of skill improvement process based on movement of gaze and hand in assembly task,” in *Computer Analysis of Images and Patterns: 18th International Conference, CAIP 2019*. Springer, 2019, pp. 15–26.
- [5] X. Zhang, Y. Duan, and Z. Chen, “Learning-based stage verification system in manual assembly scenarios,” in *Proceedings of the International Conference on Smart and Advanced Manufacturing (ICSAM 2024)*, 2025.
- [6] S. Zhou, X. Wang, Z. Liu, Y. Jiang, and L. Huang, “Research on worker assembly procedure inspection based on improved yolo and hand trajectory,” *Preprints.org*, 2024.
- [7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016. [Online]. Available: <https://arxiv.org/abs/1506.02640>
- [8] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics yolov8,” 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>

- [9] C. Lyu, W. Zhang, H. Huang, Y. Zhou, W. Liu, Y. Pan, K. Lin, H. Xie, K. Chen, and D. Lin, “Rtmdet: An empirical study of designing real-time object detectors,” 2022.
- [10] I. Robinson, P. Robicheaux, M. Popov, D. Ramanan, and N. Peri, “Rf-detr: Neural architecture search for real-time detection transformers,” 2026. [Online]. Available: <https://arxiv.org/abs/2511.09554>
- [11] P. R. Florence, L. Manuelli, and R. Tedrake, “Dense object nets: Learning dense visual object descriptors by and for robotic manipulation,” 2018. [Online]. Available: <https://arxiv.org/abs/1806.08756>
- [12] B. Wen, W. Yang, J. Kautz, and S. Birchfield, “Foundationpose: Unified 6d pose estimation and tracking of novel objects,” 2024. [Online]. Available: <https://arxiv.org/abs/2312.08344>
- [13] A. Kupcsik, M. Spies, A. Klein, M. Todescato, N. Waniek, P. Schillinger, and M. Buerger, “Supervised training of dense object nets using optimal descriptors for industrial robotic applications,” 2021. [Online]. Available: <https://arxiv.org/abs/2102.08096>
- [14] U. Sailer, J. R. Flanagan, and R. S. Johansson, “Eye–hand coordination during learning of a novel visuomotor task,” *The Journal of Neuroscience*, vol. 25, no. 39, pp. 8833–8842, Sep 2005.
- [15] S. Mattsson, Åsa Fast-Berglund, D. Li, and P. Thorvald, “Forming a cognitive automation strategy for operator 4.0 in complex assembly,” *Computers and Industrial Engineering*, vol. 139, p. 105360, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0360835218303838>
- [16] M. Haslgrübler, B. Gollan, C. Thomay, A. Ferscha, and J. Heftberger, “Towards skill recognition using eye-hand coordination in industrial production,” 06 2019, pp. 11–20.
- [17] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” 2020. [Online]. Available: <https://arxiv.org/abs/2005.12872>
- [18] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” 2024. [Online]. Available: <https://arxiv.org/abs/2303.05499>

- [19] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khali-dov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, “Dinov2: Learning robust visual features without supervision,” *arXiv preprint arXiv:2304.07193*, 2023.
- [20] G. Marullo, L. Tanzi, P. Piazzolla, and E. Vezzetti, “6d object position estimation from 2d images: a literature review,” *Multimedia Tools and Applications*, vol. 82, no. 16, pp. 24 605–24 643, Jul 2023. [Online]. Available: <https://doi.org/10.1007/s11042-022-14213-z>
- [21] D. B. Adrian, A. G. Kupcsik, M. Spies, and H. Neumann, “Efficient and robust training of dense object nets for multi-object robot manipulation,” 2022. [Online]. Available: <https://arxiv.org/abs/2206.12145>
- [22] J. Berg, T. Reckordt, C. Richter, and G. Reinhart, “Action recognition in assembly for human-robot-cooperation using hidden markov models,” *Procedia CIRP*, vol. 76, pp. 205–210, 2018, 7th CIRP Conference on Assembly Technologies and Systems (CATS 2018). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2212827118300945>
- [23] X. Zhao, Y. He, X. Chen, and Z. Liu, “Human–robot collaborative assembly based on eye-hand and a finite state machine in a virtual environment,” *Applied Sciences*, vol. 11, no. 12, p. 5754, 2021.
- [24] I. Makrini, M. Omid, F. Fusaro, E. Lamon, A. Ajoudani, and B. Vanderborght, “A hierarchical finite-state machine-based task allocation framework for human-robot collaborative assembly tasks,” 10 2022, pp. 10 238–10 244.
- [25] C. Y. Kung and T.-Y. Chou, “Skeleton based action analysis on manufacturing assembly site,” in *PACIS 2022 Proceedings*, 2022, paper 87. [Online]. Available: <https://aisel.aisnet.org/pacis2022/87>
- [26] J. Tang, H. Cheng, Y. Zhao, and H. Guo, “Structured dynamic time warping for continuous hand trajectory gesture recognition,” *Pattern Recognition*, vol. 80, pp. 21–31, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320318300621>
- [27] F. Sener, D. Chatterjee, D. Shelepov, K. He, S. Singhania, R. Wang, and A. Yao, “Assembly101: A large-scale multi-view video dataset for understanding procedural activities,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 21 096–21 106.

- [28] S. Bansal, C. Arora, and C. V. Jawahar, “My view is the best view: Procedure learning from egocentric videos,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022, pp. 658–675.
- [29] T. Ohkawa, K. He, F. Sener, T. Hodan, L. Tran, and C. Keskin, “Assemblyhands: Towards egocentric activity understanding via 3d hand pose estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 12 999–13 008.
- [30] D. Shan, J. Geng, M. Shu, and D. F. Fouhey, “Understanding human hands in contact at internet scale,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9869–9878.
- [31] X. Wang, S. K. Ong, and A. Y. C. Nee, “Augmented reality in assembly systems: State of the art and future perspectives,” *e-Factory and Assembly Systems / Virtual and Augmented Reality in Manufacturing*, 2021.
- [32] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [33] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, “Mediapipe: A framework for building perception pipelines,” 2019. [Online]. Available: <https://arxiv.org/abs/1906.08172>
- [34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” 2019. [Online]. Available: <https://arxiv.org/abs/1912.01703>