CHALMERS
UNIVERSITY OF TECHNOLOGY

Video as Observational Data for Machine Resetting

An Exploratory Case Study of Process Information Extraction and AI Support

Master's thesis in Production Engineering and Quality and Operations Management

SAMUEL JONASSON
BENJAMIN RASK

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Video as Observational Data for Machine Resetting

An Exploratory Case Study of Process Information Extraction and AI Support

SAMUEL JONASSON
BENJAMIN RASK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Chalmers University of Technology
Gothenburg, Sweden 2026

Video as Observational Data for Machine Resetting
An Exploratory Case Study of Process Information Extraction and AI Support
SAMUEL JONASSON
BENJAMIN RASK

Supervisor: Magnus Wahlgård
Examiner: Peter Krajnik

Master's Thesis 2026
Department of Industrial and Materials Science
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: View from inside the SGP 320 grinding machine during grinding operation, adapted with ChatGPT (GPT-5.5), OpenAI, 2026.

Use of AI tools: During this thesis, the authors used Claude Opus 4.6-4.7 and ChatGPT 5 series for the following purposes: generating and editing figures illustrating the lab environment at SKF. Writing, debugging, and refactoring Python code. Proofreading for grammar and language. All AI-generated content was reviewed, verified, and edited by the authors. AI-generated figures are cited individually where they appear.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Video as Observational Data for Machine Resetting
An Exploratory Case Study of Process Information Extraction and AI Support
SAMUEL JONASSON
BENJAMIN RASK
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

Machine resetting is a growing share of lost production time in high-mix, low-volume manufacturing. At SKF, resets on the SGP 320 shoe centerless grinding machine have become more frequent, longer, and more variable, motivating the search for new observational data sources to inform a future multimodal AI-based operator assistance system. Within this context, this exploratory single-case study evaluates video as such a source. Reset events were recorded with a wide-angle RGB-D camera and an action camera. These recordings were analyzed at four levels: manual phase annotation in BORIS, operator tracking using YOLOv8 with depth-based zone analysis, R3D-18 video embeddings with cosine similarity and PCA, and visibility detection using Qwen3-VL 8B. Video analysis was further complemented by two expert interviews, the official machine manual, an OPC parameter-snapshot pipeline, and a VNC-based HMI screen recording. While video reliably captured the observable execution of a reset, it missed internal machine states and experience-based decisions. Its usefulness depended on camera placement, visibility, and analytical method. The findings position video as a valuable observational layer for capturing the execution of a reset, but one that must be combined with machine-state and interface data to support a future multimodal system.

Keywords: Machine resetting, shoe centerless grinding, video-based observation, vision-language model, video embeddings, RGB-D, multimodal AI.

Acknowledgements

We want to thank everyone at SKF Gothenburg for their support of this project. We are particularly grateful to the machine operators who agreed to be filmed and interviewed, and to the colleagues who set up the OPC integration and gave us access to the machine and its data systems. We also thank the people at UVA Lidköping for the open and detailed interview, which provided the manufacturer's perspective on the reset process and what cameras can realistically capture inside the machine. Finally, we would like to thank our supervisor at Chalmers, Magnus Wahlgård, and our examiner, Peter Krajnik, for their supervision and valuable insights.

SAMUEL JONASSON BENJAMIN RASK, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
BORIS	Behavioral Observation Research Interactive Software
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
FIM	Foundation Industrial Model
FN	False Negative
FP	False Positive
FPR	False-Positive Rate
GDPR	General Data Protection Regulation
HMI	Human-Machine Interface
IMU	Inertial Measurement Unit
LLM	Large Language Model
OPC	Open Platform Communications
PCA	Principal Component Analysis
RGB	Red, Green, and Blue
RGB-D	Red, Green, Blue, and Depth
SMED	Single-Minute Exchange of Die
TN	True Negative
TP	True Positive
ViT	Vision Transformer
VLM	Vision-Language Model
VNC	Virtual Network Computing
YOLO	You Only Look Once

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Manufacturing environments are shifting towards increased product variety and smaller batch sizes to meet changing customer requirements and increase their responsiveness. There are multiple reasons for this shift, one being higher customer demand and customization options [1]. As companies expand their product portfolios and serve segmented markets, production demand is distributed across more variants, contributing to low-volume, high-mix conditions in which batch sizes tend to be smaller [2]. A direct operational consequence of this is more frequent resets, since production equipment must be adapted more often between product variants. Machine resetting is non-productive time that may increase machine idle time and throughput time. The loss of effective capacity can contribute to longer lead times [3]. In addition, resets often involve calibration, trial runs, and verification steps, which makes the downtime caused by resetting unpredictable and hard to foresee. When the process depends on situational judgment and experience, it can also introduce considerable variability and quality risks [4]. As a result, improving changeover-related performance and supporting decision-making through planning tools and decision-support systems can increase productivity and operational performance in production. [5]

1.1.1 SKF Context

At SKF in Gothenburg, the SGP 320 shoe centerless grinding machine is used to process bearing rings across an expanding range of variants. In line with the broader shift towards high-mix, low-volume production. Internal follow-up at SKF indicates that resetting on this machine has become both more frequent and more variable in recent years, with reset durations also trending upward (Appendix C.2). This means that resetting has become an increasingly important contributor to lost production time in recent years. At SKF, resetting refers to the activity required to transition the machine from one variant condition to another, including both mechanical changeover, such as swapping out tools and components. Moreover, adjustment changeover, by switching parameters and loading a new receipt. In line with the shift towards smaller production runs and greater product variety, the frequency of machine changeovers increases. With that said, resetting is no longer an occasional disruption, but a recurring part of daily operations with a direct operational impact.

In addition to becoming more frequent, the internal follow-up also suggests that reset durations have gradually increased over the past years. Taken together, these two developments imply that the total time spent on resets has increased because resets occur more often and take longer to complete. The data also indicate that reset performance varies considerably across occasions, with recurring peaks suggesting the need for iterative troubleshooting and fine-tuning. This variation suggests that parts of the resetting process, particularly adjustment and verification, may depend on situational judgment and operator experience. Practical know-how could therefore be a relevant driver of performance, and resets may be affected by uneven skill or the absence of experienced operators. Consequently, there is value in investigating how reset knowledge can be captured and translated into reusable guidance for future operator support.

A more detailed analysis of resetting shows that mechanical changeover tasks do not solely determine the reset time. Instead, a substantial portion of the total reset duration is devoted to adjustment and verification activities. These activities typically involve parameter tuning, calibration, trial runs, and quality checks to ensure the process produces acceptable, stable results. Importantly, this part of the reset is iterative and often depends on the operator's judgment and experience, which can help explain why reset duration varies across cases and why improvements can be difficult to achieve through standard documentation and follow-up metrics.

This motivates a closer investigation of reset work as performed in practice. Although existing performance follow-up provides a useful overview of how the frequency and duration of resets develop over time, it offers limited insight into the underlying sequence of work steps. Video recordings are one form of observational data that can enable systematic analysis of resetting activities, variations, and opportunities for improvement.

1.1.2 Video-based data

In manufacturing and industrial operations, video recordings are increasingly viewed as a promising source of observational data for understanding manual and semi-manual work [6]. Compared to purely system-generated data, video can capture the work process as performed, including the sequence of actions and the use of tools and interfaces [7]. When combined with system-generated data, it can capture interactions between operator decisions and machine behavior. In recent years, this has also been linked to the development of AI-based assistance systems. The system's handling of visual information can help detect process steps, identify deviations, and provide contextual support [8].

Structured observations can be conducted using SMED tools such as AVIX, which supports documenting work steps and associated times for process analysis and standardization [9]. However, based on internal experience at SKF, such tools are rarely used in practice, partly because it is time-consuming to carry out and maintain. As a result, existing follow-up data and occasional manual analysis provide useful indicators of reset performance outcomes but still offer limited insight into how the work

unfolds in detail under real operating conditions. This creates a clear motivation for exploring alternative or complementary forms of observational data. Resetting work, particularly adjustment and verification activities, often involves iterative loops, situational judgment, and micro-adjustments that are difficult to analyze. With that said, video recordings may provide a richer way to capture such details, enabling analysis of the process at the level of actions, sequences, and decision points. Importantly, the intention is not to claim that video alone would be sufficient for an eventual AI-powered optimization help tool. A long-term practical solution would likely combine several data sources, such as machine parameters, sensor data, and visual observations. In this context, the information that can be extracted reliably and how such data could be structured to support future model development and operator guidance are of interest.

1.2 Aim and Purpose

1.2.1 Aim

This thesis aims to investigate video recordings as an observational data source for the resetting process. More specifically, it evaluates what information about operator actions, sequences, and decision points can be reliably extracted from video, and how such data could be structured to inform the future development of a multimodal AI-based operator assistance system at SKF.

1.2.2 Purpose

The purpose of this work is twofold. From an industrial perspective, it supports SKF's ambition of developing AI-based operator support by exploring whether video data can complement existing system-generated data with the level of detail needed to capture real operator actions and decision points. From an academic perspective, the purpose is to test and evaluate different tools and methodological approaches for video-based observation of semi-manual reset work in a real industrial environment, and to assess what such approaches can realistically deliver under practical constraints.

1.3 Research Questions

RQ1: What process-relevant information can be extracted from video recordings of the machine resetting process, and at what level of detail?

RQ2: To what extent can video-derived information support a future AI-based operator assistance system, and what complementary data sources are needed to contextualize it?

1.4 Scope and Delimitations

1.4.1 Scope

This thesis examines a specific case at SKF, focusing on the resetting process of a particular machine. The scope is limited to the activities required to transition the machine from treating one variant to another, including both the mechanical changeover steps and the subsequent adjustment and verification activities needed to achieve stable, acceptable quality output. The study specifically focuses on using video recordings as an observational data source to capture resetting work performed on the shop floor.

The empirical scope comprises video recordings of selected reset events captured with external and internally mounted cameras, complemented by a qualitative study based on interviews with operators of varying experience levels. The interviews are included to capture operator perspectives on challenges, decision points, tacit knowledge, and support needs during the reset process. Together, these data sources are used to analyze the resetting process at the level of phases, actions, and iterative loops, and to explore what information can be derived from video to support future operator guidance.

The technical scope of the thesis is exploratory, focusing on assessing the feasibility and usefulness of video recordings rather than developing a production-ready AI solution. Therefore, the study emphasizes the identification and characterization of reset activities, potential labeling schemes, and the types of derived features that video could provide as inputs for a future AI-based operator assistance system. While the long-term SKF project may integrate additional data sources such as machine logs and sensor signals, this thesis focuses on evaluating the added value and practical constraints of video data in that context.

1.4.2 Delimitations

Because this is a single-case study focused on one machine and one resetting context at SKF, the findings are context-specific and analytical rather than statistically generalizable to other machine contexts without further validation. The study aims to identify and characterize reset activities rather than to establish a universally applicable standard.

The empirical material is based on a limited set of recorded reset events due to time, operator availability, and logistical constraints. This limits the ability to capture the full range of variation in products, machine states, operators, and shift conditions. The recordings are also subject to practical constraints such as camera placement, field-of-view limitations, and lighting, which can influence what actions are observable and how reliably the steps can be interpreted from video alone.

The qualitative material consists of two interviews: one with a machine operator working on the SGP 320 and one with a group of three representatives from UVA Lidköping. The reset recordings were conducted in SKF's laboratory environment,

where only a few operators work on this machine. Unfortunately, access to the live production line, where the SGP 320 operates under typical conditions, was unavailable. The qualitative material, therefore, offers focused, contrasting viewpoints, one operator perspective and one manufacturer perspective, rather than a broad survey of operator practices. Therefore, interview accounts are subject to recall bias and to differences in how participants describe their own work. In addition, the manual annotation, ground-truth labeling, and video interpretation steps were performed by the authors without reliability checks. This is acknowledged as a further source of variability in the study.

The models used in this thesis were selected based on two main criteria. First, they had to run locally to ensure data security, which limited the available computational resources. Second, the selection was constrained by the GPUs available on the computers for the thesis project.

The thesis does not include the development or deployment of a production-ready AI assistance system, nor does it evaluate real-time guidance in live operations. Consequently, the study cannot conclude whether video-based support would directly reduce reset time in practice. The contribution is limited to assessing what information about operator actions, sequences, and decision points can be extracted, what process insights such data can provide, and how these insights could inform future operator support.

Finally, workplace video recording raises ethical and privacy considerations that may limit what can be recorded, how long data can be stored, and who can access it. These constraints may reduce the level of detail that can be analyzed and may also influence the extent to which the recordings represent the work as a whole.

2

Theoretical Background

This chapter shows the theoretical basis for the thesis. It first introduces machine resetting in high-mix manufacturing and the specific machine context studied. It then discusses video as a source of observational data in industrial settings, followed by the cameras used to capture such data. The chapter continues with computer vision methods for operator tracking, video embeddings for activity representation, and vision-language models. It concludes with an overview of multimodal data sources and their relevance to future AI-based operator assistance systems.

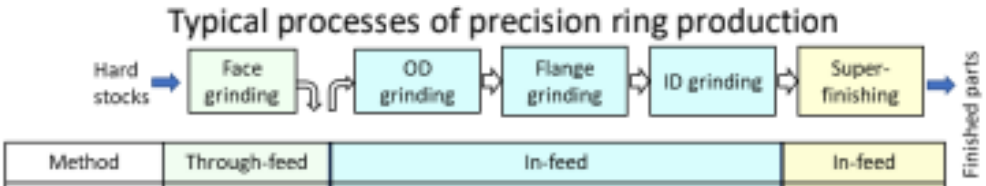
2.1 Machine Resetting in High-Mix Manufacturing

Manufacturing environments characterized by high product variety and small batch sizes require frequent transitions between product variants on the same equipment [2]. This production paradigm, commonly referred to as high-mix, low-volume (HMLV) manufacturing, has grown as companies respond to rising demand for product customization [2, 1]. As a direct consequence, industrial machines must be reset more often, and reset efficiency becomes a significant factor in machine utilization, since setup time directly contributes to machine idle time and throughput time [3].

In precision grinding, the shoe centerless grinding has historically been associated with mass-production settings, with typical changeover times reported in the range of 3 to 6 hours, as shown in Figure 2.1. The trend towards smaller production runs, therefore, poses particular challenges in this context. Although the integration of CNC technology and advances in machine design have improved flexibility and shortened setup times [10], the reset process remains complex due to the many interacting parameters that must be reconciled during the adjustment phase to achieve stable, precise grinding conditions.

2.1.1 Changeover and Adjustment Activities

In this thesis, resetting is therefore understood as comprising two distinct types of activities. The first is the *mechanical changeover*, which includes physical tasks such as replacing tools, fixtures, or other components required to accommodate the next product variant. The second is the *adjustment and verification* phase, in which the



Method	Through-feed	In-feed	In-feed
Machine	Double disk grinder	Shoe centerless grinders	Plunge superfinshing
Batch size	1k-100k [pcs]	1,000 – 100,000	1,000 – 100,000
Production rate	10-100 [pcs/min]	2-5	2-5
Changeover time	1-4 hours	3-6 hours	1-2 hours

Figure 2.1: The time for changeover times for the shoe centerless grinding is in the range of 3–6 hours. Adapted from [10].

operator tunes process parameters, performs trial runs, and checks output quality until the machine produces stable, acceptable output. While mechanical changeover tasks tend to follow a relatively defined sequence of steps, the adjustment phase is typically iterative, less structured, and strongly dependent on the operator's experience and judgment. This distinction is central to the present work and motivates a closer look at how each type of activity is typically addressed in the literature.

One well-established approach to improving changeover performance is the Single-Minute Exchange of Die (SMED) methodology, originally developed by Shingo to reduce the time machines spend idle during a changeover [11]. The method distinguishes between internal activities, which require the machine to be stopped, and external activities, which can be performed while the machine is running. By systematically converting internal activities into external ones and streamlining the remaining tasks, SMED reduces the time the machine is stopped during a changeover [11, 12].

In practice, a large part of the achievable improvement comes from preparation and workplace organization rather than from the changeover steps themselves. In a case study combining SMED with 5S in a metal-machining SME, the main interventions were organizational: pre-staging and clearly identifying the required tools, keeping them in a dedicated cabinet close to the machines, and mapping which tools each product variant requires, so that operators no longer spend changeover time searching for or fetching tooling [12]. SMED is therefore most effective for the mechanical and organizational aspects of a changeover, where tasks can be planned, sequenced, and prepared in advance.

In shoe centerless grinding, the adjustment phase is particularly challenging because the machine can run stably only when several interdependent conditions are met simultaneously [10]. The most important condition is workholding stability, shown in Figure 2.2. The workpiece must remain firmly in place against the two support shoes while it rotates and is ground [13]. To achieve this, the workpiece is rotated

by a magnetic drivehead. The pulling force of the drivehead, which depends on how fast it spins, how strong its magnet is, and how far it is offset from the workpiece center, must be balanced against the resistance from the shoes and the grinding wheel [13]. If this balance is wrong, the workpiece can slip, lift, or vibrate.

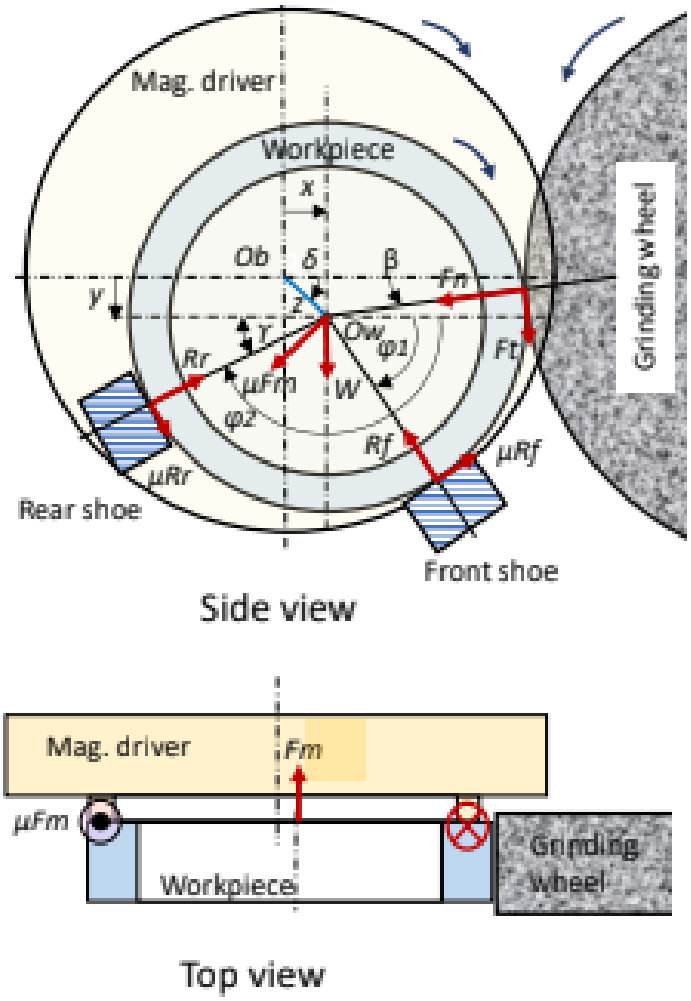


Figure 2.2: Geometry and main forces involved in the workholding balance in OD shoe centerless grinding. Adapted from [10].

In addition to workholding stability, the setup must also satisfy two further conditions known as geometrical and dynamic rounding stability, which together ensure that small shape errors on the workpiece do not grow over time, and that the process does not start to chatter during grinding [14, 10].

In practice, reaching these stability conditions is not a single adjustment but a sequence of linked setup steps. Typical steps include making sure the support surfaces behind the workpiece sit flat, placing the front shoe as close to the grinding wheel as possible, choosing the angle of the rear shoe so that the workpiece sits at a suitable height, setting how far the workpiece center is offset from the drivehead, adjusting the strength of the magnet, and finally running a short test to confirm that the workpiece seats firmly on both shoes within a single rotation [10]. These parameters

are not independent: changing one usually forces changes to others. For example, increasing the offset gives the drivehead a stronger pull but reduces its turning capability, whereas decreasing the offset has the opposite effect [13]. Although the underlying physics of these interactions has been studied and modeled in the literature [14, 13], in shop-floor practice, the adjustment phase still relies heavily on the operator's experience and judgment rather than on a fixed procedure.

2.1.2 The Grinding Machine(SGP 320)

The machine studied in this thesis is a Lidköping High Volume SGP 320 shoe centerless grinding machine. It is designed for high-throughput grinding of the outer surfaces of cylindrical components such as bearing rings and rollers, and can handle workpieces with outer diameters from 50 to 320 mm and grinding widths up to 120 mm. The SGP 320 is manufactured by UVA Lidköping, a Swedish manufacturer specializing in precision grinding systems [15]. Although the machine was originally developed for high-volume production, it is now also used in more variant-rich production environments, such as the one studied here, where resetting between product variants is a recurring operational concern.

Workpieces enter and leave the machine through inlet and outlet chutes, and the operator interacts with the machine through a Siemens Sinumerik HMI. An electronic gauging head provides in-process measurement, and the grinding wheel is periodically reconditioned and dressed to restore its profile and cutting ability. These elements together define the main interfaces and steps that an operator works with during a reset.

The SGP 320 uses shoe centerless grinding, in which the workpiece is supported on two pads (the front and rear shoes) and rotated by a magnetic drivehead, with no fixed center of rotation [14]. Because the workpiece is free to move slightly on the shoes, the rotational axis effectively re-centers itself during grinding, which helps reduce roundness errors, but also makes the process sensitive to the support setup [14]. This sensitivity is the practical reason why setting the shoe geometry, the drivehead offset, and the magnetic force is correctly applied during a reset.

Although the SGP 320 was originally designed for high-volume production, it is increasingly used in environments with greater product variety, including the case studied here. In such environments, the combination of tight precision requirements and a setup that depends on several interacting parameters means that resetting between product variants is rarely a fixed procedure, which makes the machine a relevant case for studying how video-based observational data can support the reset process.

2.2 Video as Observational Data in Manufacturing

Building on the previous section, this section examines video as a source of observational data for studying manual work in manufacturing. Compared with aggregated performance indicators such as average cycle times or output counts, video preserves the temporal structure of work as it unfolds: the sequence of activities, the transitions between them, idle periods, and the variation between prescribed and actual work [6].

It also retains operator-specific strategies and contextual cues that are typically lost when work is summarized by numerical KPIs alone, because observable behavior can be systematically captured and analyzed using video-based coding approaches [7]. These properties make video a suitable form of observational data when the object of study is the process itself, and not only its outcomes. At the same time, raw recordings are not directly analyzable: they must be interpreted, segmented, or otherwise transformed into structured representations before they can support process analysis [6]. Because video can only capture what is visible from the chosen camera perspectives, it is also discussed throughout this chapter as one component of a broader, multimodal view of the reset process [16].

2.2.1 Prior Applications of Video Analysis in Industrial Settings

A first line of evidence for using video to study manual industrial work comes from studies that apply video analysis directly to multi-step manufacturing and maintenance processes. Vybornova et al. [17] propose a foundation model for temporal action segmentation in maintenance and manufacturing videos, targeting multi-step manual processes such as engine oil changes and tire replacements. The cited work reports that industrial process videos share several structural characteristics with the present study: they consist of sequential manual activities with iterative subtasks, they vary in duration across operators and occasions, and they are difficult to capture using aggregated performance metrics alone. The same study structures the recorded processes into more than ten distinct event categories, indicating that fine-grained temporal segmentation of complex manual work from video is feasible in industrial settings.

Other applications of video analysis in industrial settings extend beyond temporal segmentation of work. Automated visual inspection systems analyze video streams to detect defects or monitor process states on production lines, typically with greater consistency and throughput than human inspectors [18]. Worker activity recognition from video has been used to assess ergonomic risk, identify unsafe postures, and estimate workload based on observable behavior without requiring wearable sensors on the operator [7]. Closer to the focus of this thesis, video has been applied in assembly operations to verify the correctness of procedural steps, to identify differences between actual and prescribed work, and to estimate how time is distributed

across subtasks [6, 7]. The first of these applications is primarily outcome-oriented, whereas the others target the process itself, a perspective adopted in the present study.

A recurring challenge across these applications is the natural variation in how manual work is carried out. Unlike machine-controlled processes, manual operations vary in pace, technique, and sequencing across operators and across repetitions of the same task [7]. Video captures this variation directly, whereas sensor- or log-based signals require additional assumptions to be interpreted as behavior [6]. This makes video valuable both as a primary observational source and as the basis for the labeled, or ground-truth, datasets used to train and evaluate automated monitoring methods [7]. The usefulness of such datasets, in turn, depends on how the underlying video is annotated and interpreted.

2.2.2 Manual Video Annotation

In a video-based study of manual work, the recorded video has to be interpreted and labeled before it can be analyzed as process data. This process is referred to here as *video annotation*: a trained observer watches the recording, identifies meaningful units of work such as individual actions or activity phases, and assigns each unit a category from a predefined set of activity labels [7]. The output is a timestamped event log in which the start and end times, and the category of each labeled segment, are recorded [6]. This event log serves as the primary data structure for subsequent analysis. It enables the computation of process measures, such as time-on-task distributions, activity transition frequencies, and comparisons across operators or reset occasions. In an industrial activity analysis context, the credibility of such an annotation depends on the clarity of the activity definitions and on the degree to which independent observers, applying the same definitions to the same recording, reach the same conclusions [19].

Manual video annotation, however, is time-consuming. Reported annotation times in the literature suggest that a single hour of video typically requires several hours of annotation effort, with the exact ratio depending on the complexity of the activity scheme and the density of events [6]. In manufacturing studies, where data collection may yield many hours of recordings, this cost is a significant practical constraint. It motivates both the careful selection of which video segments to annotate in detail and the development of semi-automated annotation methods that can reduce the manual burden while preserving the quality of a human-checked dataset [18].

2.2.3 Levels of Structure in Video-Based Analysis

Raw video recordings contain rich information about how work is performed, but this information is not directly available in a form suitable for systematic analysis [6]. For a video to be useful as process data, it must be structured. In this thesis, video-derived information is therefore considered at four analytical levels: manually annotated phases, spatial movement signals, object-level visibility, and latent visual–temporal representations. The levels differ in how much they rely on human

interpretation, what information they reveal, and how readily they can be applied to long recordings.

At the most explicit level, video is structured into manually annotated phases or events, in which an observer assigns activity labels to video segments and produces a timestamped event log that can be analyzed for duration, sequence, and transitions [7]. The reliability of such labels in an industrial setting depends on the clarity of the activity definitions and the consistency between observers [19]. A second level concerns spatial and movement-related information. When video is combined with detection or tracking methods, the operator's position can be followed over time. With RGB-D cameras, the visual image can be associated with depth measurements, so movement is represented not only as image coordinates but also as approximate spatial information relative to the camera. RGB-D cameras have been used to track progress in manual assembly work [20], and depth has been shown to provide spatial cues complementary to color images in action recognition more broadly [21]. From these signals, it is possible to derive process indicators such as time spent in different work areas, movement between zones, and transitions across the workspace.

A third level concerns visible objects, tools, and components. Object detection and vision-language models can be used to identify whether specific items are visible in a frame or sequence, linking the visual scene to process-relevant entities such as tools, machine components, or workpieces. Recent vision-language models have shown the ability to connect visual inputs with textual prompts for tasks such as image understanding, localization, and visual question answering [22]. Visibility in the image, however, does not necessarily imply that an item is being used, correctly mounted, or relevant to the current decision; object-level information, therefore, needs to be interpreted together with process knowledge. A fourth level concerns latent visual–temporal representations. Instead of manually defining each phase or object, video segments are transformed into embedding vectors using pretrained video models. These representations do not directly provide human-readable labels, but they enable comparison of segments, identification of similarity relationships, and exploration of recurring patterns with limited manual annotation [23, 24]. Recent work in industrial video analysis has shown that learned video representations can support temporal action segmentation while reducing dependence on manual annotation [17], making this level particularly relevant in exploratory settings where a complete labeling scheme is not yet available.

These four levels are complementary rather than mutually exclusive. Manual annotation provides interpretable phase labels, tracking, and depth extraction, which provide spatial movement signals. Object-level analysis provides semantic visibility information, and embeddings provide a compact representation for similarity-based exploration. At the same time, video-derived information remains limited to what is visible from the chosen camera perspectives, so internal machine states, parameter values, quality outcomes, and operator reasoning require complementary, non-visual data sources to be contextualized [25].

2.3 Camera-Based Data Collection

The previous section stated why video is a useful source of observational data for manual industrial work. This section turns to the hardware side of that data collection. It reviews the camera technologies used to capture such recordings, focusing on the two camera types deployed in this study. Camera-based data collection is increasingly used in industrial settings as a non-intrusive way to capture operator behavior and process dynamics at a level of detail that aggregated system metrics do not provide [18]. Cameras deployed on or near production lines have been used to detect operator presence, track movement within the workspace, and structure the observed activity into discrete phases or events [20, 18].

RGB-D cameras, which record both a conventional color image and a per-pixel depth map (a distance reading for each pixel), have been used in several industrial monitoring studies because the depth channel enables spatial measurements without adding separate distance sensors. In one such study, a low-cost RGB-D camera is used to track the real-time progress of a manual assembly sequence, arguing that such data sources can support a shift towards more flexible, data-driven manufacturing [20]. Beyond single-sensor setups, multi-camera configurations have been proposed to overcome the field-of-view limitations of any one sensor [26]. By combining a thermal camera and a depth camera for human detection and activity recognition in a manufacturing setup, this work illustrates, more generally, how complementary camera modalities can be integrated to capture what no single sensor can observe alone.

Camera-based observation also introduces methodological constraints. A study of observer reliability in industrial activity analysis based on video recordings, conducted in a truck-engine assembly context, finds that inter-observer agreement depends on the clarity of the activity definitions and on the constraints imposed by camera placement and image resolution [19]. In other words, the reliability of any video-derived activity label is bounded by what the camera can actually resolve. For this reason, the present study uses two cameras with complementary roles: a wide-angle RGB-D camera that captures the full work area and depth information, and a close-up action camera that captures fine-grained manual activity.

2.3.1 Intel RealSense Depth Camera (D455f)

The Intel RealSense D455f is a stereo RGB-D camera designed for capturing synchronized color and depth data in a compact sensor package. According to the RealSense D455f product specifications, the camera is designed for indoor/outdoor use and provides an ideal depth range of 0.6 m to 6 m, with a depth field of view of 87° × 58° (H×V) and depth frame rates up to 90 fps [27]. The D455f includes an IR-pass filter intended to reduce repetitive-pattern artifacts and false detections caused by reflections, which is beneficial in reflective industrial environments [27].

Independent evaluations support that the D455 depth measurements are sufficiently reliable for indoor applications within this range. In particular, the D455 was evalu-

ated under indoor conditions and quantified how measurement distance and surface inclination affect depth accuracy and noise, showing stable performance in representative indoor scenarios [28]. Complementary metrological characterization work also reports the D455's wide field of view and long ideal range within the RealSense D400 family, supporting its suitability for general RGB-D data acquisition in close-to-medium distances [29].

The camera records RGB and depth streams in the RealSense `.bag` format, preserving per-frame timestamps and camera metadata for synchronized downstream processing; similar ROS/RealSense recording workflows are commonly used in D455 evaluation studies [28].

2.3.2 GoPro as a Complementary Camera (HERO13)

While a wide-angle RGB-D camera provides a broad spatial view and depth information, a distant mounting position limits visibility of fine-grained manual actions, such as fingertip adjustments and interactions with hand tools. To address this limitation, an action camera can be used as a complementary recording device, positioned close to the work area to provide a detailed view of the operator's hands.

The GoPro HERO13 supports high-resolution recording at high frame rates, which is beneficial for reviewing fast hand motions frame-by-frame [30]. It is also designed for rugged environments and is waterproof to 10 m (33 ft) without additional housing, with a water-repelling lens cover that helps reduce water-related image artifacts [30]. This is particularly relevant in the present study because the grinding machine environment is wet and exposed to coolant spray. More generally, action sport cameras are viable video sensors in underwater settings when used with waterproof housings and appropriate acquisition procedures, supporting their suitability for wet environments [31].

A practical trade-off of action-camera footage is that wide-angle optics increase coverage in constrained spaces but also introduce substantial lens distortion. Such distortion can affect the geometric interpretability of the image unless calibration and correction procedures are applied [32]. In this thesis, the GoPro stream is therefore treated primarily as a qualitative, close-up observational layer for resolving fine actions and interactions. In contrast, the RealSense stream provides stable wide-area context and depth information.

Additionally, the GoPro is positioned to capture the operator's interactions within the machine. The GoPro recording is not electronically synchronized with the RealSense stream. Instead, the two recordings are temporally aligned during post-processing using a shared visual event at the start of each session. This complementary viewpoint supports manual annotation and enables later inspection of reset actions that are difficult to resolve from the wide-angle RGB-D view alone.

2.4 Computer Vision for Operator Tracking

2.4.1 Object Detection and Person Tracking

The application of object detection models such as YOLO to industrial video analysis has been demonstrated in recent work on workplace behavior recognition. [33] proposed Unsafe-Net, a hybrid deep learning pipeline combining YOLO v4 for frame-level object detection with a Convolutional LSTM (ConvLSTM) for temporal classification of safe and unsafe worker behaviors in a factory environment. In their approach, YOLO v4 first identifies frames containing relevant persons and actions, then passes the selected sequences to a ConvLSTM to classify behavior over time. Trained and evaluated on 39 days of real factory footage, the system achieved high classification accuracy in real time. This two-stage architecture, combining frame-level detection with temporal sequence modeling, illustrates a pattern relevant to the present study. YOLO-based person detection serves as a first step to locate and track the operator within the workspace before extracting spatial and movement-based features for further analysis.

2.4.2 Depth Estimation and Spatial Measurement

Depth cameras extend conventional RGB video by adding spatial information to the visual scene. In RGB-D systems, each color frame is paired with a depth map, where each pixel contains an estimated distance to the camera. This enables moving beyond visual appearance alone and obtaining approximate spatial measurements of people, objects, or surfaces in the recorded environment [20, 21].

In industrial contexts, RGB-D data can be useful because many activities involve spatial relationships between operators, tools, components, and machines. Previous studies have used RGB-D cameras to monitor manual assembly progress and human activity, illustrating how depth information can complement conventional video when analyzing physical work environments [26]. Compared with RGB video alone, the depth channel provides an additional source of information about distance and position, which can support more structured analysis of recorded activity.

However, depth measurements are not error-free. Individual depth pixels may be missing, noisy, or unstable due to sensor limitations, occlusions, reflective surfaces, surface angle, or distance from the camera. Metrological evaluations of Intel RealSense sensors show that measurement accuracy and noise are affected by factors such as distance, surface inclination, and sensor configuration [29]. For this reason, depth values are often interpreted with caution and may need to be processed or aggregated over small image regions rather than relying on single pixels.

2.5 Video Embeddings for Activity Representation

The challenge of limited annotated data is a recurring constraint in industrial video analysis. Vybornova et al. [17] highlight that collecting labeled datasets for specialized manufacturing tasks is time-consuming and that rare events may be systematically underrepresented, making it difficult to train supervised models effectively. Their work motivates approaches that reduce reliance on manual annotation, such as self-supervised pre-training on unlabeled video data, followed by fine-tuning on a small labeled subset. This is consistent with the embedding-based workflow explored in the present study, in which pretrained models serve as feature extractors to generate compact representations of video segments without requiring exhaustive manual labeling of the full recording.

Video recordings provide rich observational data, but the high dimensionality of raw video makes it difficult to analyze directly using computational methods. A common approach in modern machine learning is therefore to transform segments of video into numerical representations, often called embeddings. An embedding represents a piece of data as a vector of numerical values in a high-dimensional space, where semantically similar inputs are positioned closer to each other [34]. By converting complex inputs such as images, audio, or video into vector representations, it becomes possible to perform tasks such as similarity search, clustering, and pattern discovery.

In video analysis, this typically involves dividing a video into short temporal windows and extracting an embedding vector for each segment using a pretrained model. These embeddings can then be analyzed using similarity metrics, dimensionality reduction, or clustering-based exploration. This is particularly relevant when exhaustive manual annotation is impractical, as learned representations can provide an initial structure for exploring large collections of video segments before a complete labeling scheme has been developed [34, 17].

Recent developments in deep learning have made it possible to extract video embeddings using models originally trained for large-scale action recognition tasks. Examples include convolutional neural networks and transformer-based architectures trained on datasets such as Kinetics [35]. These models learn representations that capture both spatial information from individual frames and temporal patterns across sequences of frames. As a result, embeddings derived from these models can capture recurring actions, interactions with objects, and transitions between activity phases.

In the context of industrial processes such as machine resetting, embedding-based representations could enable the discovery of recurring micro-events or process phases in recorded video. For example, short video segments showing actions such as retrieving tools, performing mechanical adjustments, or verifying machine output could cluster in the embedding space if similar visual patterns recur. This could allow large collections of reset recordings to be explored and structured with lim-

ited manual annotation effort. An embedding-first workflow, therefore, represents one potential strategy for transforming raw video recordings into structured process information that may support the development of AI-based operator assistance systems.

However, embedding-based approaches typically require additional computational infrastructure and careful parameter selection, including window length, frame sampling, and model choice. Such workflows often involve dividing videos into overlapping temporal windows, generating embeddings using pretrained models, and storing them in a vector index for similarity search and clustering. While this thesis does not aim to develop a complete embedding pipeline, the concept is relevant as a potential method for structuring video data and enabling scalable analysis of resetting activities.

The potential of video embeddings to capture meaningful activity information has also been demonstrated in cross-modal contexts. Tong et al. [36] proposed a zero-shot learning framework for IMU-based activity recognition, in which video embeddings extracted from a pretrained I3D model served as a semantic space bridging seen and unseen activity classes. Their results showed that video-derived embeddings consistently outperformed text-based semantic spaces, supporting the notion that pretrained video models encode motion-relevant features transferable across domains. However, their application differs from the present study, where embeddings are used directly to explore and cluster industrial reset activities rather than to bridge video and sensor modalities. Their findings provide supporting evidence that pretrained spatiotemporal models produce representations sufficiently rich to distinguish activity classes, even in domains outside their original training purposes.

2.5.1 R3D-18 Model

This thesis uses R3D-18 as a baseline model for extracting video embeddings from short reset sequences. R3D-18 is a three-dimensional convolutional neural network designed for video analysis, where the model processes several frames together rather than treating each frame as a separate image. This allows the model to represent both visual appearance and motion over time, which is important when analyzing activities rather than still images [23, 24].

The model builds on the ResNet architecture, where residual connections enable the training of deeper neural networks more effectively [37]. In R3D-18, this idea is extended to video by applying operations across both spatial and temporal dimensions. In practice, a short clip of frames is passed through the model, which converts it into a compact numerical representation.

In this study, R3D-18 is used as a feature extractor rather than as an action classifier. The last layer of the model is removed, leaving an output of only a 512-dimensional feature vector, also known as an embedding. Each embedding, therefore, represents one short temporal window of the reset recording. These vectors can then be compared by looking at similarity measures or visualized via dimensionality reduction to determine whether different parts of the reset process are represented as similar

or distinct.

R3D-18 was selected because pretrained versions are available, computationally manageable, and suitable for exploratory analysis when limited labeled data are available. The model is commonly pretrained on large action-recognition datasets such as Kinetics-400, which means that it has learned general visual and motion patterns from many types of human activities [38]. This makes it useful as a starting point for investigating whether reset videos contain structure that can be captured without first creating a large annotated dataset.

However, R3D-18 is not trained specifically for industrial resetting or grinding-machine environments. The embeddings may therefore capture general scene characteristics, such as camera perspective, background, lighting, and machine layout, in addition to process-relevant operator actions. This is an important limitation when applying pretrained video models in a specialized industrial setting. Recent work on industrial video analysis suggests that domain-specific self-supervised pretraining can improve temporal action segmentation in manufacturing videos with limited labels [17]. For this reason, R3D-18 should be understood as a practical baseline in this thesis, rather than as the final model architecture for a future operator assistance system.

2.5.2 Similarity and Dimensionality Reduction

Embeddings are useful because they represent complex inputs as numerical vectors, where relationships between vectors can be used to compare inputs and identify structure in the data [34]. In video analysis, this allows comparison of temporal windows without first assigning a complete set of manual labels.

A common way to compare embedding vectors is cosine similarity, which measures the directional similarity between two vectors. Rather than comparing absolute magnitude, cosine similarity compares the orientation of vectors in the embedding space. This makes it suitable when the analytical interest is whether two segments are represented as similar in terms of visual or temporal content. In activity recognition research, video embeddings have been used as representation spaces for comparing activity classes and transferring information between related modalities, illustrating that pretrained video representations can contain motion-relevant structure beyond the original training task [36].

High-dimensional embeddings can be difficult to interpret directly. Dimensionality reduction is therefore often used to inspect whether the embedding space contains dominant patterns. Principal Component Analysis (PCA) is a widely acknowledged technique for simplifying data. PCA converts the original variables into a new set of independent components, ordered by the variance they account for. This process allows us to represent a high-dimensional dataset with fewer dimensions while keeping most of the original information. [39]. This makes PCA useful both for examining how variation is distributed across an embedding space and for producing lower-dimensional projections for exploratory visualization.

However, PCA projections should be interpreted with caution. A two-dimensional projection does not contain all information present in the original high-dimensional embedding space, and PCA does not assign semantic meaning to the components by itself. Instead, interpretation requires comparison with labels, timestamps, or the original video material. In exploratory industrial video analysis, similarity measures and dimensionality reduction should therefore be understood as tools for inspecting the data's structure rather than as final classification or segmentation methods.

2.6 Vision-Language Models

Vision-Language Models (VLMs) are multimodal models that integrate visual and textual information within a single framework, enabling tasks such as captioning, visual question answering, retrieval, and grounded visual reasoning [40]. Unlike a conventional large language model (LLM), which processes only text, a VLM can also accept images (and in some cases video frames) by converting visual input into representations that can be combined with text and processed jointly.

In this thesis, VLMs are relevant because they enable natural-language queries over recorded video frames to extract structured information, such as whether specific tools or components are visible. The model used in the experiments is Qwen3-VL [22], the most recent vision-language model in the Qwen-VL family available at the time of the experiments. The remainder of this section summarizes the typical architecture of a modern VLM, using the Qwen-VL family as the running example.

2.6.1 Architecture Overview

Most modern VLMs follow a simple pipeline, illustrated in Figure 2.3: an image is encoded into visual features, those features are converted into a set of visual tokens compatible with a language model, and an LLM backbone generates an output conditioned on both the visual tokens and the text prompt [40]. A common way to describe this is as three components: a visual encoder, a vision-language connector (adapter/projector), and a language model backbone [22].

2.6.1.1 Visual Encoder

The visual encoder transforms the input image into a sequence of feature vectors (embeddings) that represent the scene [22]. In many current VLMs, this encoder is transformer-based and produces patch-level features that can later be fused with language tokens. A limitation of fixed-resolution vision encoders is that small details can be lost during resizing. Newer designs, therefore, support more flexible resolution handling, where the number of visual tokens can scale with the input to preserve detail when needed [41].

2.6.1.2 Vision-Language Connector

The vision encoder can produce many visual features, and passing them all into an LLM can be computationally expensive. A connector module maps the visual

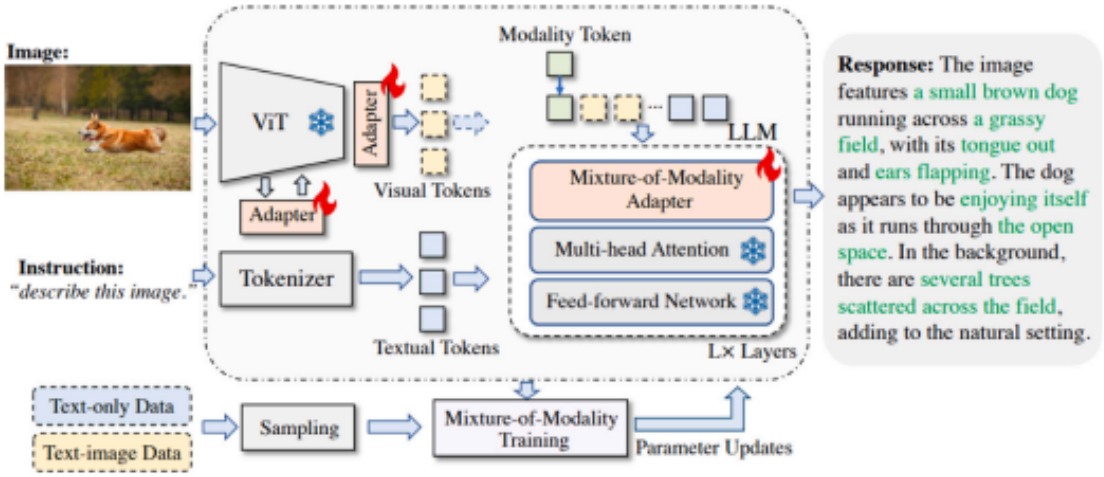


Figure 2.3: Example VLM pipeline. Adapted from [40].

features into the language model's embedding space and typically compresses them into fewer visual tokens [42]. Moreover, a common connector design uses cross-attention with a small set of learnable query vectors to summarize the visual feature sequence into a compact token set [22].

In Qwen-VL, for example, this produces a fixed-length sequence of visual tokens that is inserted alongside the text tokens before the language model [22]. Later models allow more flexible token counts, consistent with dynamic-resolution processing [41].

2.6.1.3 Language Model Backbone

The language model backbone receives the combined sequence of visual tokens and text tokens and performs the main reasoning and generation [40]. Because both modalities are represented as tokens in a single sequence, standard attention mechanisms can mix information across image content, and the prompt [22]. The output is typically generated as text (e.g., an answer or a structured description), using the same general decoding behavior as in language models [40].

Overall capability depends strongly on the language model backbone, whereas the vision encoder and connector largely determine how much visual detail is preserved and how efficiently it is processed [42].

2.7 Multimodal Data for Manufacturing Assistance

Manufacturing processes increasingly generate data from multiple sources, including machine logs, sensor signals, images, video streams, production records, and human input. These sources do not describe the same aspects of a process. Video can capture visible operator actions, movement, tool use, and interaction with the work environment, while machine-generated data can capture internal states, parameter values, alarms, timestamps, and process events. Expert knowledge and interviews

can further explain why particular actions, parameters, or machine states are important.

Multimodal data integration is relevant in manufacturing because different data sources can provide complementary views of the same process. In activity recognition, multi-sensor fusion is commonly motivated by the idea that information missing or uncertain in one source may be available in another [25]. In industrial systems, this principle extends beyond physical sensors to combinations of visual data, machine-state data, event logs, interface data, and human expertise.

2.7.1 Complementary Data Sources in Manufacturing

Machine logs and sensor data provide access to process information that is not directly observable from external video. Examples include machine states, parameter values, actuator positions, alarms, and event timestamps. Such data can describe what is happening inside the control system. Recent work on industrial data fusion highlights that manufacturing environments can contain several complementary data streams, including visual data, sensor signals, and machine-state information, which may be combined to support monitoring and quality-related analysis [43, 44].

This complementarity is important in reset processes because many important decisions depend on information that is not visible from the outside. A camera may show an operator performing an adjustment, starting a cycle, or inspecting a part. However, it does not directly reveal the active recipe, compensation values, measurement results, or internal machine state. Consequently, machine data may show that a parameter changed or that a cycle was executed, but it does not explain the physical action, interpretation, or operator reasoning behind the event.

Industrial communication standards such as OPC-UA provide structured access to machine data for use in industrial automation and machine learning applications. Recent reviews indicate growing interest in integrating OPC-UA-based architectures with machine learning for applications such as predictive maintenance, process monitoring, and quality control [45].

2.7.2 Challenges in Multimodal Data Integration

Combining data sources into a single representation is technically and analytically challenging, and designing a complete multimodal system involves choices that lie beyond the scope of this thesis. Fusion can in principle be performed at different levels (early, intermediate, or late), depending on whether data are combined before feature extraction, within learned representations, or after separate model outputs are produced [43]. This thesis does not implement any of these fusion strategies. Instead, it focuses on the two preconditions that any of them would require. Identifying which sources provide complementary information and aligning those sources on a shared process timeline.

Temporal alignment is the more fundamental of the two. Video streams are continuous, whereas machine logs, alarms, parameter snapshots, and production events

may be event-based or sampled at different frequencies. To be analyzed together, these sources must be referred to the same process timeline. Without reliable synchronization, it cannot be established whether an observed operator action, an HMI interaction, a parameter change, or a machine event refers to the same moment.

A second challenge is data selection. Manufacturing systems can expose a large number of variables, but only a subset is relevant to a given task. [44] notes that modern manufacturing environments generate high-dimensional, complementary data streams, but that fusing and interpreting them requires deciding in advance which signals are meaningful. This thesis addresses this challenge only at the level of identifying and structuring candidate sources, such as the internal parameters captured through the OPC interface. Their joint modeling is treated as a direction for future work.

Finally, multimodal integration does not automatically produce process understanding. Even when several data sources are available, the relationship among observable actions, machine states, and process outcomes must still be interpreted in the context of the process. This is why multimodal manufacturing assistance requires not only data access but also careful synchronization, variable selection, validation, and domain knowledge.

3

Methods

This chapter describes the study's methodological approach. The work is designed as an exploratory single-case study conducted at SKF's production facility in Gothenburg, focusing on the SGP 320 grinding machine. The gathered material consists of video recordings of reset activities performed by machine operators. Also, semi-structured interviews with machine operators, machine experts, and representatives from the machine manufacturer. In addition, the official machine manual is used as a formal reference for procedures and terminology.

The chapter starts by providing the research design and the strategy for identifying relevant literature. It then describes the data-collection methods, followed by the technical experiments developed from the recorded material. These include a pilot study on manual video annotation, operator tracking, and depth extraction from RGB–D data, video embedding experiments, and an evaluation of a vision-language model. The chapter concludes with complementary data collection from machine systems and interfaces, along with the ethical considerations that guided the study.

3.1 Research Design

In this thesis, case studies are used as a research strategy to gain an in-depth understanding of a contemporary phenomenon within its real-world context. This is particularly important when the boundary between the subject and its context is unclear [46]. Case studies are well-suited to investigating complex situations because the subject and its context are so closely linked that the two are difficult to study in isolation.

The study is exploratory in orientation because limited structured knowledge exists about the resetting process at the level of detail needed to inform a future multimodal AI-based operator-assistance system. Rather than testing predefined hypotheses, the work investigates what process-relevant information about operator actions, sequences, and decision points can be derived from video, and how that information can complement other data sources available around the machine. The empirical material combines several complementary sources. Video recordings of reset events form the primary observational data, supported by semi-structured interviews, the official machine manual, and complementary data from the machine and its operator interface. Using several sources in parallel is intended to enable

triangulation. For example, using multiple data sources or methods — to provide a clearer and more reliable understanding of the topic being studied [47].

The study does not aim to produce a deployable AI system or to establish statistically generalizable results. The contribution is analytical and case-specific: the findings are tied to the SGP 320 machine and the resetting context at SKF in Gothenburg, and any broader implications are treated as analytical insights and directions for future work rather than as established conclusions.

3.2 Literature Search

A targeted literature search was carried out to establish the theoretical foundation of the study and to identify relevant prior work on machine resetting in high-mix manufacturing, video-based observation of manual industrial work, camera-based data collection, computer-vision methods for operator tracking, video embeddings for activity representation, vision-language models, and multimodal data for manufacturing assistance. Scopus was used as the primary database, complemented by Google Scholar and the Chalmers Library catalog to broaden coverage of peer-reviewed sources. Search strings combined keywords related to machine resetting, changeover, video-based observation, operator tracking, activity recognition, video embeddings, vision-language models, and multimodal AI assistance, using Boolean operators (AND, OR) to balance sensitivity and precision in line with recommendations for systematic literature searches [48].

The set of search terms was refined iteratively: keywords were expanded and rescoped as the understanding of the research topic developed during the review process [49]. Recent peer-reviewed journal and conference papers were prioritized, with older seminal works retained where they introduced foundational methodological concepts (for example, the SMED methodology and the shoe centerless grinding process).

3.3 Data Collection

3.3.1 Video Recordings of Reset Events

Video recordings of reset events were collected at SKF's laboratory environment using two mounted cameras operating in parallel, complemented by a screen recording of the operator's interaction with the machine interface. The Intel RealSense D455f provided a fixed, wide-angle view that captured the full width of the work area, including both the machine and the adjacent worktable, with synchronized RGB and depth streams. A GoPro action camera was positioned at approximately operator height to capture close-up detail of hand movements and tool handling, and was mounted inside the machine. In addition, the Siemens Sinumerik HMI was screen-recorded through a VNC viewer connected to the machine, providing a direct record of menu navigation, parameter changes, and on-screen feedback during the reset. The roles and technical specifications of the two cameras are described in Section

2.3, and the VNC-based HMI capture is treated as a complementary, non-visual record of the operator's interaction with the control system.

Because operator availability was limited and access to the production area where the bearings are manufactured was restricted, the reset events were recorded in SKF's laboratory environment rather than during live production. This has several consequences for how the recordings should be interpreted. First, the range of recorded events is narrower than the variation observed in production. The sessions reflect the variants, tooling, and starting conditions that could be arranged in the laboratory, rather than the full range of products, machine states, and shift conditions encountered on the production line. Second, the recordings were made without the time pressure, parallel tasks, and interruptions that characterize normal production, and the operator was aware that they were being filmed. Both factors may lead to reset work being performed more deliberately or in a different order than it would be under production conditions. Third, the laboratory machine may differ from a production machine in its wear state, calibration history, and available tooling, which can influence both the steps required and the time they take. For these reasons, the recorded behavior is treated as illustrative of how a reset can unfold rather than as a representative sample of production resets.

Each recording session began before the operator started the reset and continued until the reset was completed or the recording had to be stopped for practical reasons. As a result, not all sessions cover the reset from start to finish, and some recordings capture only parts of the overall process. Before each session, the setup was verified to confirm that both cameras had an adequate field of view, that the RealSense was logging to a .bag file, and that the VNC connection to the HMI was active and recording. The three streams were not electronically synchronized; temporal alignment was performed in post-processing. The two cameras were aligned using a shared visual event at the start of each session. In contrast, the HMI screen recording was aligned to the camera footage using the operator's first interaction with the HMI as a common reference, since the HMI display remained static until the operator engaged with it.

3.3.2 The Machine Manual

Alongside the video recordings and interviews, the official SGP 320 operator's manual was used as a formal reference for the procedures investigated in this thesis. The manual, *"Instruktionsbok SGP 320"*, is published by UVA Lidköping AB and delivered together with the machine at SKF. It describes the machine in its intended operating state and is therefore treated here as the prescribed counterpart to the recorded and reported reset work.

The document covers machine specifications, the control system and HMI, installation, safety, production cycles, changeover, grinding-cycle optimization, and maintenance. The chapter most relevant to this study is Chapter 7, *"Omriggning"*, which describes the full reset sequence as it is designed to be carried out. It covers the components exchanged between product variants, the required tools, the disassem-

bly and assembly order, the calibration of the measurement finger, the grinding of the magnetic drivehead, the dressing of the grinding wheel and the associated dressing cycle, and the loading of the workpiece recipe. Chapters on the control system, HMI menus, and grinding-cycle optimization are used as references when interpreting the operator's actions on the machine panel.

The manual serves three roles in this study. It provides the procedural backbone for the interview guide, supplies a shared vocabulary for describing machine components and actions in the video annotations and interview transcripts, and serves as the reference against which any deviations between prescribed and observed reset work can be identified in the later analysis.

3.3.3 Interviews with Machine Experts

To complement the video observations and surface aspects of the resetting process that are difficult to observe directly, two semi-structured expert interviews were conducted. Interviews are a commonly used qualitative method for obtaining in-depth insights into participants' experiences, interpretations, and practices [50]. The first interview was conducted with an SGP 320 machine operator at SKF and captured the process as experienced in the production environment. The second was held remotely with three representatives from UVA Lidköping, the manufacturer of the SGP 320, and captured the machine builder's perspective, drawing in particular on one participant with around a decade of experience in tool design, machine development, commissioning, and customer support. Sessions lasted approximately 30-60 minutes and were conducted either on-site at SKF or via Microsoft Teams.

A single semi-structured interview guide was used for both sessions and is available in appendix A. Semi-structured interviews allow for a consistent set of topics while still enabling flexibility to explore context-specific details during the conversation [50]. The guide was structured in four parts: a free-text overview of the process (Part A), critical phases and experience-based decisions during the reset (Part B), camera placement and what video should capture (Part C), and sources of variation between operators (Part D). The guide was anchored in the machine manual, *"Instruktionsbok SGP 320"*, and mapped to RQ1 and RQ2 so that each research question was addressed during the session. For the UVA interview, individual questions were adapted in real time to fit the manufacturer's perspective; for example, questions about operator behavior during production were reframed to focus on what UVA observes during customer training and remote support.

The interviews were recorded separately as audio files, then transcribed locally using OpenAI's Whisper automatic speech recognition model via the whisper Python package. The transcription was performed using a short Python script executed in Visual Studio Code, with the medium model variant and the language parameter set to Swedish. The implementation details and code used for transcription are provided in appendix B.1.

The resulting transcripts were reviewed against the original audio, and corrections were made where the model output was unclear or contained domain-specific ter-

minology. The transcripts were then stored as plain-text files to serve as the basis for analysis. Moreover, running the transcription locally was preferred over a cloud-based service in order to maintain control over the audio recordings and avoid transferring potentially sensitive data to external systems.

The transcripts were analyzed through thematic coding of the interview content, following the procedural phases [51]. Familiarisation with each transcript through repeated reading, initial coding of segments relevant to the resetting process, grouping of related codes into candidate themes, and review of those themes against the transcripts before naming them in relation to the research questions. Given that the analysis draws on only two interviews, it is not presented as a full reflexive thematic analysis, which typically rests on a larger body of accounts. Instead, the coding is used to surface and organize recurring topics that help interpret the material. Therefore, it should be read as structured expert input that contextualizes the observational findings, rather than as generalizable claims about reset practice.

Together, the two interviews provide complementary perspectives: the SKF interview describes how the reset is experienced inside production, while the UVA interview describes how the machine is intended to be reset and the constraints that shape what can and cannot be observed in the video. This contrast between intended and experienced practice supports the interpretation of the video material in the later sections of this chapter.

3.4 Video Annotation Pilot

The purpose of this pilot was to evaluate whether manual annotation could serve as a viable approach for structuring video data into process-relevant phases, and to assess its practical limitations in terms of effort and scalability. It should be noted that the video used for this purpose was not recorded in an industrial setting but designed to mimic basic elements of resetting activities. Therefore, it is not intended to represent real process behavior, but rather to illustrate the annotation approach.

The BORIS (Behavioral Observation Research Interactive Software) tool was used to annotate and segment a video sequence manually. BORIS is behavioral coding software that allows users to define action categories or phases and assign each category to a keyboard key. During video playback, the researcher can mark the start and end of specific activities by pressing the corresponding keys. This process generates a timestamped table of coded events that describes when different activities occur within the video.

For the pilot test, a short video was recorded showing simple actions intended to simulate basic elements of resetting activities, such as searching for an object, grasping a tool, and making adjustments. The video was recorded with a single camera at a relatively wide, high angle to capture as much of the activity area as possible. While small details were occasionally occluded, the general phases of activity remained observable. This setup was intended to mimic some of the practical constraints that may occur in real industrial recordings.

The video was manually segmented into phases. The start and end of each activity were determined based on the researcher's interpretation of when a specific action began and ended. Using the predefined keyboard mappings in BORIS, a new phase was recorded upon the end of the previous phase, creating a sequence of timestamped actions.

The purpose of this pilot was not to establish a finalized annotation procedure, but rather to test whether video recordings could be structured into observable phases and to illustrate the level of manual effort required when segmenting video data in this way. The test was a proof-of-concept, demonstrating that meaningful activity phases can be extracted from video recordings while also highlighting the labor-intensive nature of fully manual annotation.

3.5 Video Data Processing

To enable systematic analysis of the recorded reset operations, a data processing pipeline was developed to transform the raw video recordings into structured datasets suitable for further analysis. The recordings used in this stage of the study consist of videos of an operator performing actual resetting activities at the machine. These recordings were captured using the Intel RealSense camera described in Section 3.3.

While the earlier video annotation pilot relied on a demonstration video recorded by the researcher to explore the structure of the resetting process, the recordings processed in this stage represent real operational activity. This shift from exploratory pilot material to actual machine recordings enables the evaluation of video-based analysis methods in a realistic setting.

The RealSense recordings contain synchronized RGB and depth streams, stored in .bag format, along with associated timestamps and camera metadata. Although this format is well-suited for recording and playback, direct analysis of .bag files is less convenient for frame-level processing and experimentation with computer vision methods.

Therefore, a custom data processing pipeline was implemented in Python to extract relevant segments from the recordings and convert them into structured datasets comprising RGB and depth frames, along with associated metadata. The pipeline enables selecting specific temporal intervals in the recording and ensures that the RGB and depth streams remain synchronized throughout the extraction process.

The resulting dataset serves as the basis for subsequent experiments on operator tracking and depth-based feature extraction. More generally, this transformation of video recordings into structured frame-level data is an important step towards computational analysis of operator behavior. By representing the video as a sequence of temporally ordered observations with associated spatial measurements, the pipeline creates a dataset that supports subsequent techniques such as feature extraction, machine learning, and video embedding.

It should also be noted that the resetting sequence was recorded simultaneously using an additional camera system. A GoPro camera was positioned close to the machine to capture detailed views of the operator's hand movements and interactions with the machine interface. While the present processing pipeline focuses on the RealSense recordings, these additional recordings provide complementary perspectives that may support future analysis of fine-grained operator actions.

The processing pipeline has two main steps. First, relevant time intervals are extracted from the recorded video streams and exported as synchronized RGB and depth frames together with frame-level metadata. Second, an experimental tracking method is applied to follow the operator within the video and extract depth measurements corresponding to the operator's spatial position relative to the camera.

3.5.1 Segment Extraction from RealSense Recordings

The raw video recordings captured by the RealSense camera were stored in ".bag" format. Each recording contains synchronized RGB and depth streams together with timestamps and camera metadata. Although this format enables accurate playback of the recorded scene, it is not well-suited for detailed frame-level analysis.

To address this, a Python-based extraction script was developed to convert selected time intervals from the recordings into structured datasets. The script replays the .bag recording using the RealSense software development kit, and exports synchronized RGB and depth frames for user-defined temporal intervals. The start and end times of the desired segment can be specified in seconds, enabling flexible selection of relevant portions of the recording.

3.5.2 RGB and Depth Frame Synchronization

For each extracted segment, a dedicated directory is automatically generated, containing three main components: a folder of RGB frames, a folder of depth frames, and a metadata file describing the frames in the segment. The RGB and depth frames are stored as image files while preserving their original temporal order.

A metadata file is generated for each segment. This file records information such as frame index, timestamp, elapsed time within the recording, and file paths to the corresponding RGB and depth images. The metadata structure ensures that each frame can be uniquely identified and linked to both image streams.

This structured representation allows the recorded video to be treated as a dataset rather than a single continuous recording. It enables efficient experimentation with different processing methods while preserving synchronization between RGB and depth data.

3.6 Operator Tracking and Depth Extraction

An operator-tracking and depth-extraction pipeline was developed to derive spatial information about the operator's position over time. The detection model used was YOLOv8n, accessed via the Ultralytics Python package, and it was not fine-tuned specifically for the recordings. The purpose of this analysis was to investigate whether video recordings could be transformed into structured signals that reflect operator movement within the workspace, and to explore the feasibility of extracting higher-level process indicators, such as movement patterns and transitions between different work areas.

3.6.1 Person Detection and Tracking

To localize the operator in each frame, a person detection approach was applied using a pre-trained YOLO model. The model was used to identify bounding boxes corresponding to persons in the RGB frames. In recordings with multiple individuals present, the operator of interest was initially identified manually in the first frame and subsequently tracked using spatial continuity across frames.

The tracking process was based on associating detections across consecutive frames based on image position, bounding-box size, and estimated depth. This approach enabled the system to maintain a consistent representation of the operator's position over time. Initial experiments using a standalone visual tracking approach revealed limitations in maintaining stable tracking under more dynamic conditions, which motivated the use of a detection-based approach to improve robustness by continuously re-identifying the operator.

3.6.2 Depth-Based Position Estimation

For each detected operator position, depth information was extracted from the corresponding aligned depth frame. The operator's spatial position was represented by the detected bounding box's image coordinates and the associated depth value.

To obtain a stable depth estimate, the depth value was not taken from a single pixel but rather extracted from a small patch centered on the detected bounding box. A patch radius of three pixels was used, resulting in a 7×7 pixel neighborhood. Invalid depth values were excluded from the calculation. This approach reduces the influence of missing or noisy depth pixels and provides a more robust estimate of the operator's distance from the camera.

For each processed frame, the pipeline produced a set of structured data, including timestamp, image coordinates, and estimated depth. These values form a time series of the operator's relative position in the workspace.

3.6.3 Signal Processing

The extracted position and depth signals were smoothed before the zone-based analysis to reduce short-term noise and isolated tracking errors. Only frames in which the operator was detected and the required coordinate and depth values were available were included in this step.

A two-stage rolling filter was applied to the horizontal coordinate, vertical coordinate, and patch-based median depth signal. First, a centered rolling median filter with a window length of 11 frames was used to reduce isolated spikes. This was followed by a centered rolling mean filter with a window length of 7 frames to smooth the remaining fluctuations. The smoothed signal was stored separately from the raw tracking output and used as input for the zone-based movement analysis.

3.6.4 Zone-Based Movement Analysis

After smoothing the tracking signal, the operator's position was related to three predefined workspace zones: the machine area, the workstation area, and the interface area. These zones represented the main locations in the RealSense view where reset activities were performed. The zone layout used for this analysis is shown in Figure 3.1.

Each valid frame was assigned to one of the zones based on the operator's estimated position. This produced a time-ordered zone sequence showing how the operator moved between the main work areas during the recorded segment. A transition was registered when the assigned zone changed from one frame to the next.

The zone sequence was then used to calculate simple movement indicators, including time spent in each area, number of visits, and transitions between zones. These indicators are used in the results section to examine whether tracking and depth data can provide spatial information about the reset process.

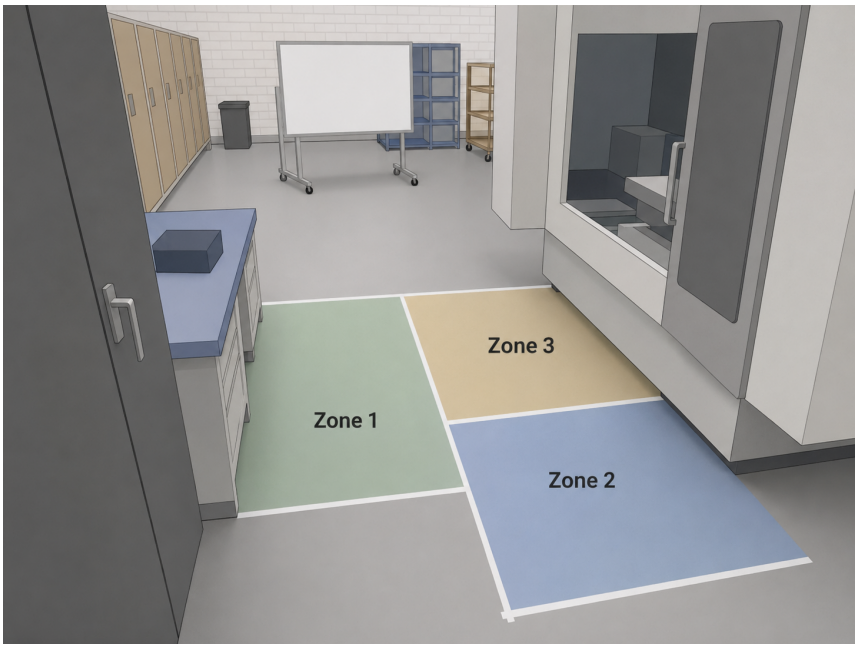


Figure 3.1: Workspace zone layout used in the zone-based movement analysis, including machine area, workstation area, and interface area. Adapted with ChatGPT (GPT-5.5), OpenAI, 2026.

3.7 Video Embedding Experiments

The embedding experiments build directly on the structured frame-level dataset generated in Section 3.5. In that step, the raw RealSense recordings were converted into synchronized RGB frames with associated timestamps and metadata. These extracted RGB frames serve as input to the embedding pipeline, so no additional processing of the original .bag files is required at this stage.

The purpose of the embedding experiments is to transform short temporal segments of the video into numerical representations that capture visual and temporal characteristics of the resetting process. These representations are subsequently analyzed to investigate whether they reflect temporal continuity, operator movement, and spatial structure in the workspace.

The embedding workflow consisted of the stages shown in Table 3.1.

Table 3.1: Overview of the embedding workflow.

Stage	Description
Temporal windowing	The extracted RGB frames were grouped into overlapping time windows representing short segments of the resetting process.
Embedding generation	Each temporal window was processed with a pretrained R3D-18 model to generate a fixed-length numerical representation.
Embedding analysis	The resulting embeddings were analyzed using cosine similarity and principal component analysis to examine temporal continuity and internal structure.

3.7.1 Segment Preparation and Windowing

To enable temporal analysis, the extracted RGB frames were grouped into overlapping time windows. Each window represented a short segment of the resetting process and served as the basic unit for embedding generation.

Windows were defined with a fixed temporal duration of 2 seconds and a stride of 1 second, resulting in 50 % overlap between segments. For each window, 16 frames were sampled uniformly across the time interval. This ensured that each embedding captured both spatial information from individual frames and temporal variation within the segment.

The use of overlapping windows allowed the representation to evolve smoothly over time, which was important for analyzing temporal continuity and detecting transitions between different process states.

Each window was associated with metadata inherited from the frame-extraction process, including start and end times and frame indices. This enabled direct alignment between embeddings and the original video when interpreting the results.

3.7.2 Embedding Generation

For each temporal window, a numerical embedding was generated using a pretrained R3D-18 model from the Torchvision library. The model was set up using weights from the Kinetics-400 dataset and acted as a feature extractor.

To obtain embeddings, the model's final classification layer was removed, and the penultimate layer's output was used as a fixed-length feature representation. This resulted in a 512-dimensional embedding vector for each window.

The model's input was a tensor of 16 RGB frames per window. Before inference, frames were resized and normalized to the model's expected input format. Each window was therefore represented as a compact numerical vector summarizing its visual and temporal content.

The resulting embeddings were stored with their associated metadata, forming a time-ordered sequence of vectors that represent the resetting process.

3.7.3 Analysis Approach

After embedding generation, a set of analysis techniques was applied to investigate the embedding's structure and behavior.

First, cosine similarity was computed between embeddings to compare temporal segments of the video. Two types of comparisons were performed: similarity between consecutive windows and similarity between randomly selected window pairs. This analysis was used to evaluate whether temporally adjacent segments were represented as more similar than unrelated segments.

Second, PCA was applied to the embedding vectors to examine how variance was distributed across dimensions and to reduce the dimensionality of the representation. PCA was used both to analyze the structure of the embedding space and to visualize the data in lower-dimensional projections.

Third, manually labeled subsets of the data were used to investigate whether the embedding space captured specific process characteristics. Windows were labeled based on operator activity, that is, movement versus stationary states, and spatial zones within the workspace. These labeled embeddings were projected into PCA space to evaluate whether distinct patterns or clusters emerge.

Finally, the temporal progression of embeddings was analyzed by examining their trajectories in reduced-dimensional space. This enabled an investigation of whether the representation evolved continuously and reflected the underlying process dynamics.

3.8 Vision-Language Model Experiments

In parallel with the embedding analysis, a separate experiment was conducted to investigate whether a pretrained VLM could extract structured, per-frame information about the visibility of the machine operator, specific tools, and machine components during the dismantling step in the resetting process. The aim was to examine how much process-relevant detail can be recovered from video using a general-purpose model without any task-specific training.

3.8.1 Experimental Design

The experiment was framed as a feasibility evaluation of per-frame visibility detection, not as a complete replacement for manual annotation. For each sampled frame, the VLM was prompted to determine whether each item in a predefined vocabulary of tools and components was visible in the image and to answer yes or no. The model's per-frame binary outputs were then compared with a manually annotated

ground truth and a visibility log for the same recording, as described in Section 3.8.5.

The detection target was visibility rather than action. A frame was treated as a positive case for an item if that item was present in the camera view at that moment, regardless of whether the operator was actively using or interacting with it. This separates the experiment from the action-level annotation used elsewhere in the thesis and keeps the evaluation aligned with what a vision-only model can, in principle, observe.

The prompt vocabulary contained 17 items: 1 operator label, 4 tools, and 12 components, of which 12 are observed in the ground-truth log and 5 serve as negative controls. Negative controls are tools and components named in the prompt but never visible in the recording. They make it possible to measure the model's false-positive rate on categories that are never correct to assert, which is treated separately from per-item performance on observed items. The vocabulary itself, including the merging of physically rigid pairs into single labels and the exclusion of small repetitive parts, is described in Section 3.8.4.

Model performance was assessed at three levels: per item, across item groups, and over time. For each item, every frame was compared with the visibility log and classified as a true positive, false positive, true negative, or false negative. Precision, recall, and F1-score were then computed using the standard definitions for binary classification, where precision corresponds to the positive predictive value, recall corresponds to sensitivity, and the F1-score is defined as the harmonic mean of precision and recall [52].

For items with no positive ground-truth frames (the negative controls), the F1-score is undefined, and the false-positive rate was reported instead. At the group level, per-item scores were aggregated for the operator, the observed tool, and the observed components using macro-F1 (the unweighted mean of per-item F1) and support-weighted F1 (each item weighted by its number of positive ground-truth frames). To avoid dominance by any single item, the component group was also evaluated with one item removed.

At the temporal level, per-frame outcomes were mapped across the recording as a per-item, per-second heatmap. This representation enables inspection of how true positives, false positives, and false negatives are distributed over time, rather than relying solely on aggregated metrics.

The scope of the evaluation was kept narrow on purpose. The thesis measures per-frame detection accuracy and reports the patterns described, not the performance of an end-to-end automated visibility-logging system. Turning per-frame detections into clean temporal intervals, as a human annotator would produce, requires additional post-processing.

The experiment was conducted on a single dismantling recording of approximately 8 minutes and 32 seconds, sampled at 1 frame per second, yielding 513 frames. This

is consistent with the single-case study design presented in Section 3.1 and with the delimitations set out in Section 1.4.2. The findings are not generalized beyond the specific machine, operator, and recording.

3.8.2 Model Selection and Deployment

The selected model was Qwen3-VL 8B, a vision-language model from the Qwen family reviewed in Section 2.6. Three considerations motivated this choice. The architecture and training approach of the Qwen-VL family are documented in the public literature [22, 41], which places the method within the family of recent open-weight VLMs covered earlier in the thesis. The 8-billion-parameter count was the largest model that could be run locally on the available hardware within an acceptable inference time, thereby avoiding the data transfer and confidentiality concerns associated with cloud inference of recordings from an industrial facility. The model weights are also openly available, which supports reproducibility.

The model was deployed locally using the Ollama runtime on a Windows 11 Enterprise workstation equipped with an NVIDIA RTX PRO 2000 Blackwell GPU (8 GB VRAM), an Intel Core Ultra 7 265H CPU (16 cores), and 64 GB of system memory. The qwen3-vl:8b model was used in its default 4-bit quantized configuration to balance performance and resource usage.

The deployment was carried out locally rather than using a cloud-based service to ensure that the SKF recordings remained within the production environment, in line with the ethical considerations described in Section 3.10.

3.8.3 Ground Truth and Visibility Log

The ground-truth reference for the evaluation is a manually annotated visibility log that records which tools and machine components are visible, and whether the machine operator is in the camera frame, throughout the recording. The log is structured as a sequence of intervals. Each row captures a span of time during which the set of visible items does not change, and a new row begins whenever a tool appears or disappears, a component enters or leaves the frame (for example, by being dismounted and placed out of view), or the operator enters or leaves the frame. The endpoints of consecutive intervals are contiguous, so the log covers the full 08:32 duration without gaps.

3.8.4 Prompt Design and Script

The prompt was designed so that the model's output could be evaluated objectively against a discrete ground-truth label, rather than producing free-form descriptions that would require subjective interpretation. A structured checklist format was used, with each item answered yes or no based on its visibility in the current frame. Free-text output would introduce parsing ambiguity and make comparison against a binary ground-truth label unreliable, whereas a binary output format avoids this.

The instructions specified that the model should avoid guessing and respond with "no" whenever it was uncertain, in order to minimize the baseline false-positive rate. Additionally, mounting bolts were excluded from the detection targets due to their small size, high quantity, and consistent presence, which would not provide meaningful process information. The complete text of the prompt, along with the Python script utilized to sample the recording into 513 frames and transmit them to the model, can be found in Appendix B.5.

3.8.5 Frame Sampling

Frames were sampled at one frame per second from the full-length recording, yielding 513 frames covering the interval 00:00 to 08:32. This rate was chosen as a compromise between temporal resolution and computational cost. Frames were extracted with OpenCV by seeking each target timestamp and reading a single frame on demand, then saved as base64-encoded JPEG files and passed to the model along with the textual prompt. Because the video was interpreted at a fixed interval every second, memory usage remained constant.

3.9 Complementary Machine Data Collection

In addition to the video recordings, exploratory machine-side data were collected to examine how internal machine-state data could complement video-derived information in a future multimodal system. The aim of this part of the data collection was not to integrate machine data with the video analysis within the scope of this thesis, but to examine what types of complementary data could be retrieved and structured for later use. The VNC-based HMI screen recording introduced during the video recording setup is described in Section 3.3.1 and treated there as one of the recorded streams rather than as a separate machine data source.

3.9.1 OPC-Based Parameter Snapshot

To explore whether internal machine-state data could be retrieved and used as a complement to video observations, a client was configured to access the machine's OPC interface and collect selected parameter values during a reset. In discussions with operators and technical personnel, approximately 60 internal parameters were selected based on their expected relevance to the reset process. These were parameters expected to change during resetting or to reflect important aspects of the machine state.

Trigger scripts were configured to export the current values of the selected parameters whenever a predefined event occurred. Three trigger types were used: a timer-based trigger that produced snapshots at regular intervals, a piece-counter trigger that produced a snapshot whenever a ring had been ground, and compensation-related triggers that produced a snapshot whenever one of the selected compensation parameters changed.

Each trigger event produced a JSON file containing the values of the selected parameters at that time, along with the event timestamp. The resulting files were parsed and loaded into database tables, yielding a dataset of parameter values that can be queried by event type and by time. Within the scope of this thesis, the dataset was collected and structured, but not integrated with the video analysis. It is included here as an indication of what a future multimodal integration could draw on.

3.10 Ethical Considerations

The empirical material in this thesis consists of workplace video, semi-structured interviews, and machine-side data, all of which involve identifiable individuals or proprietary process information. The study therefore complies with the General Data Protection Regulation (GDPR) [53] and SKF's internal data protection regulations. Video of operators constitutes personal data and was recorded only after participants were informed of the study's purpose, the cameras in use, and their right to withdraw. All recordings were made under controlled conditions in SKF's laboratory environment rather than during live production.

A central concern in workplace video analysis is the limited scope of the recordings, even though they are intended to support process understanding and the design of future operator assistance. The same material could, in principle, be reused for individual performance surveillance or disciplinary action. To mitigate this, the recordings, OPC parameter snapshots, and HMI screen captures were used only for the research questions defined in this thesis, were not linked to operator identity, shift, or performance metrics, and were not shared with line management for evaluation purposes. All empirical data were stored within SKF-controlled infrastructure, with access limited to the thesis authors and the SKF supervisor, and all model inference was performed locally. Interview audio was transcribed locally using Whisper rather than through a cloud-based service, and direct quotations are reported without names or other identifying details. Confidential parameter values and machine configurations have been omitted from the reporting.

4

Results

4.1 Manual Annotation

As introduced in Section 3.4, the manual annotation pilot was conducted on a short demonstration video rather than on a full industrial reset recording. The result should therefore be interpreted as a proof of concept for phase-level structuring, corresponding to the most explicit analytical level described in Section 2.2.3

The manual annotation pilot demonstrated that video can be segmented into distinct, interpretable activity phases. Using BORIS, the sequence was segmented into predefined categories representing different types of actions, including mechanical operations, adjustments, verification, and periods of inactivity. Table 4.1 presents the resulting distribution of annotated phases. The specific time distribution is not intended to represent an actual reset process; rather, it shows that a continuous video sequence can be decomposed into discrete, temporally bounded activities with measurable durations.

Table 4.1: Summary of manually annotated phases in the BORIS pilot video

Phase	**Time total (s)**	**Share of total video (%)**
Setup start	25.057	8.38
Retrieve/search	13.413	4.49
Mechanical action	31.030	10.38
Interface settings	31.781	10.63
Adjustment	52.767	17.65
Verification check	61.159	20.45
Waiting	72.355	24.20
Closeout	11.428	3.82
Total	**298.990**	**100.00**

The results show that phase transitions can be identified from video, even with a single camera and a relatively wide field of view. Although some fine-grained details were occasionally occluded, the overall structure of the activity sequence remained observable.

At the same time, the pilot highlights an important limitation of manual annotation. The segmentation process relies on the researcher's interpretation of when an ac-

tivity begins and ends, introducing subjectivity and requiring continuous attention during video playback. Even for a short sequence of approximately five minutes, the annotation process was time-consuming, indicating that fully manual annotation would not scale efficiently to larger reset-video datasets.

Overall, the pilot demonstrates that video recordings can be transformed into structured phase-level representations. However, manual annotation is also best suited as an exploratory tool for defining categories and reference labels. This motivates the subsequent analyses, which examine whether more automated methods can yield additional video-derived process information.

4.2 Operator Tracking and Depth Analysis

Unlike the manual annotation pilot presented in Section 4.1, the operator tracking and depth analysis was applied to a RealSense recording of an actual reset activity. The objective of this analysis was to analyze whether video and depth data could be transformed into spatial process information about the operator's movement within the workspace. The main result is that the tracking and depth pipeline produced a time-based position signal that could be further converted into zone-level indicators, including time spent in different work areas and transitions between zones. This complements the phase-level information from the manual annotation pilot by showing that video can also provide spatial information relevant to RQ1.

4.2.1 Continuous Operator Position Signal

The combined use of visual tracking and depth data enabled the extraction of a continuous signal representing the operator's distance from the camera over time. By associating the operator's tracked position in each frame with its corresponding depth measurement, a time series of relative spatial positions was obtained.

Initial attempts to extract depth values directly from a single pixel at the center of the tracked region resulted in unstable measurements, primarily due to missing depth values and sensor noise. This issue was mitigated by computing the median depth within a small neighborhood surrounding the reference point, while excluding invalid (zero) values. This approach significantly improved signal stability and reduced the impact of noise on depth data.

The resulting signal provides a continuous representation of operator movement along the depth dimension, enabling the identification of relative position changes throughout the recorded sequence. Although the signal does not capture full three-dimensional motion, it provides a simplified but informative measure of spatial behavior over time.

4.2.2 Zone-Based Movement Patterns

Based on the extracted position signal, the workspace was divided into predefined zones representing different areas of the work environment. The continuous track-

ing output was then converted to a discrete zone sequence, with each valid frame assigned to a zone based on the estimated operator position.

Table 4.2 summarizes the zone-based movement output from the analyzed segment. The result shows that the pipeline quantified how the operator moved between different parts of the workspace. In the analyzed sequence, Zone 3 accounted for the most valid zone time, with 17.07 seconds distributed across two visits, followed by Zone 1 with 15.23 seconds across two visits. Zone 2 was visited once and accounted for 7.43 seconds of the valid zone time.

Table 4.2: Zone-based movement summary from the YOLO tracking experiment

Zone	**Time spent (s)**	**Visits (n)**	**Avg. visit duration (s)**
Machine area	15.23	2	7.62
Workstation area	7.43	1	7.43
Interface area	17.07	2	8.54
Total	**39.73**	**5**	–

In addition to time allocation, the analysis identified a sequence of transitions between zones. Four transitions were detected in the analyzed segment: Zone 1 to Zone 2 at 8.97 seconds, Zone 2 to Zone 3 at 16.70 seconds, Zone 3 to Zone 1 at 23.80 seconds, and Zone 1 to Zone 3 at 30.73 seconds. This demonstrates that the tracking output can be used not only to estimate the operator's location, but also to represent the movement pattern as a sequence of workspace transitions.

The result should be interpreted as a feasibility demonstration rather than a general characterization of reset behavior. The analyzed segment is short, and the zone definitions are specific to the camera placement and workspace layout. Nevertheless, the output shows that video-derived tracking data can be transformed into structured process indicators, such as time spent in work areas, the number of visits, and transition points.

4.3 Embedding-based Analysis

The embedding results show that the R3D-18 representations captured structured variation in the reset recordings rather than forming an undifferentiated set of vectors. This structure was related to operator movement, temporal continuity, and workspace zones, and also reflected general scene context, such as the shared machine environment, camera perspective, and background. A first step in the analysis was therefore to examine how this variation was distributed across the embedding space using Principal Component Analysis (PCA).

4.3.1 Embedding Space Structure

The RealSense-based embeddings showed visible variation in the reduced PCA space, indicating that the windows were not represented as a homogeneous cloud. Earlier projections also showed that the first two principal components did not capture all relevant structure, as additional separation became visible when higher-order components were examined. This was particularly apparent in the zone-based analysis, where some workspace-related differences were clearer in component combinations beyond PC1 and PC2.

The GoPro-based embeddings showed a similar pattern of concentrated variance. As shown in Figure 4.1, a relatively small number of principal components captured a large share of the total variance. Table 4.3 shows the number of components required to explain selected variance thresholds for both datasets. For the RealSense embeddings, 16 components explained 80% of the variance, 20 components explained 85%, and 28 components explained 90%. For the GoPro embeddings, the corresponding values were 18, 27, and 42 components.

These results show that the embedding space was structured and compressible rather than evenly distributed across all 512 dimensions. This does not imply that a two-dimensional PCA projection can fully capture the process. However, it indicates that the embeddings contain dominant patterns of variation that can be further analyzed. The following sections, therefore, examine whether this structure relates to temporal continuity, operator movement, and spatial context.

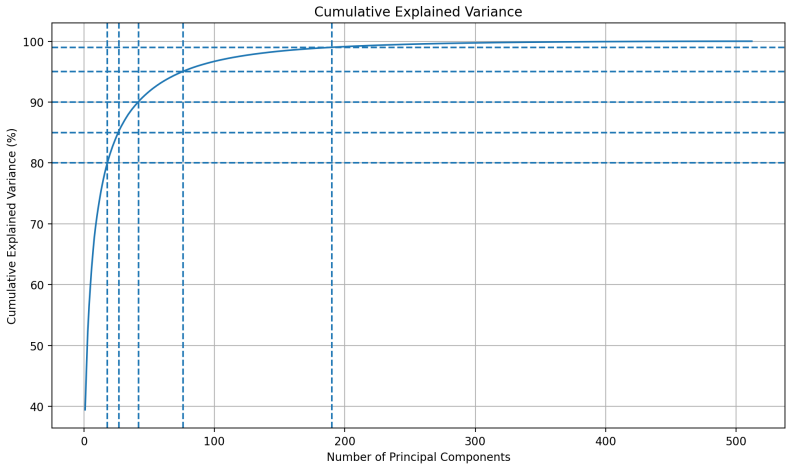


Figure 4.1: Cumulative explained variance for the principal components of the GoPro-based window embeddings, showing that a relatively small number of components captures a large share of the total variance.

Table 4.3: Number of principal components required to explain selected levels of cumulative variance for the two embedding datasets.

Embedding dataset	80% variance	85% variance	90% variance
RealSense (wide-angle)	16 PCs	20 PCs	28 PCs
GoPro (close-up)	18 PCs	27 PCs	42 PCs

4.3.2 Similarity and Temporal Behavior

A central question in the embedding analysis was whether the extracted representations reflected the temporal progression of the resetting process or primarily captured static visual similarities. To examine this, cosine similarity was computed between embeddings from different time windows. Two types of comparisons were used: similarity between consecutive windows and similarity between randomly selected window pairs.

For the RealSense-based recordings, similarity between consecutive windows was analyzed as a function of time. The overall similarity values were generally high, which is expected given that the same environment and operator are present throughout the recording. However, the similarity curve contained localized drops. Compared with the underlying video, these drops corresponded to moments when the operator moved between positions or transitioned between tasks. This shows that relative changes in embedding similarity were associated with observable changes in the recorded process, even though the absolute similarity values remained high, as shown in Figure 4.2.

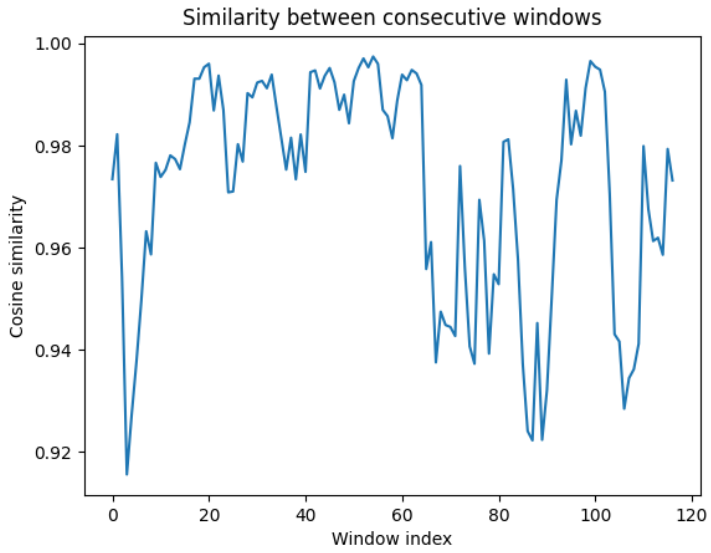
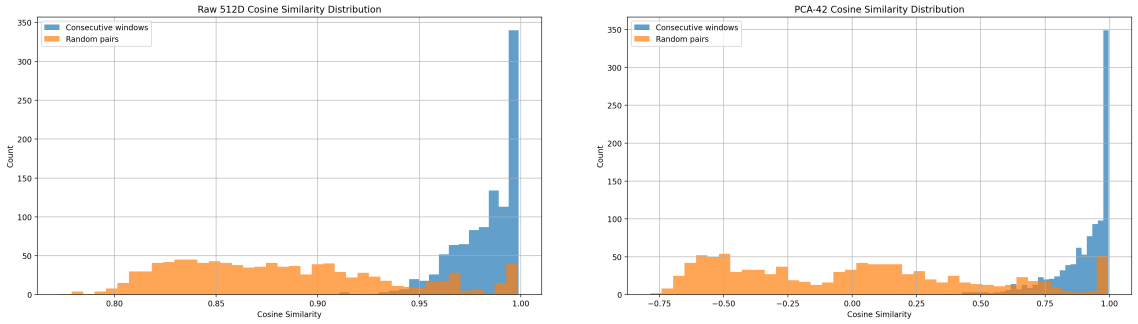


Figure 4.2: Consecutive-window cosine similarity over time for the RealSense-based embeddings.

The GoPro-based analysis extended this by comparing consecutive-window similarity with similarity between randomly sampled window pairs. Consecutive windows consistently showed high similarity values, reflecting the temporal continuity of the recording. Random window pairs showed a wider distribution of similarity values, indicating that windows from different parts of the process were represented as less similar than neighboring windows.

The comparison was performed both in the original 512-dimensional embedding space and after PCA dimensionality reduction. In the full embedding space, similarity values were generally high for both consecutive and random window pairs, resulting in substantial overlap between the distributions. This indicates that the raw embedding space was strongly influenced by shared visual context, such as the same machine, camera view, and operator.

After PCA reduction, the separation between consecutive and random window pairs became clearer. Consecutive windows remained concentrated at high similarity values, while random window pairs spread across a wider range, including substantially lower values. Figure 4.3 shows the differences between the GoPro embeddings in the original 512-dimensional space and in the PCA-reduced representation that retains approximately 90% of the total variance. Figure 4.4 further summarizes the mean cosine similarity for consecutive and random pairs across the different representation spaces.



(a) Full embedding space (512 dimensions, no PCA reduction)

(b) PCA-reduced space (42 components, ∼90% variance)

Figure 4.3: Cosine similarity distributions for consecutive and random window pairs in the GoPro video.

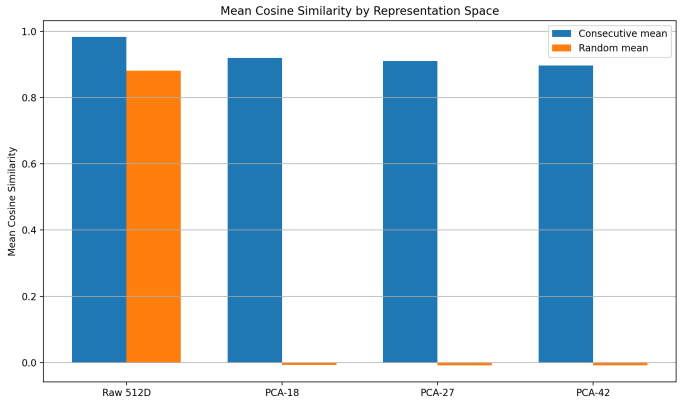


Figure 4.4: Mean cosine similarity for consecutive and random window pairs across different representation spaces.

Overall, the similarity analysis shows that the embeddings preserve temporal continuity while still allowing differentiation between non-adjacent parts of the process, particularly after PCA-based dimensionality reduction. Additional similarity distributions for PCA-reduced representations using 18 and 27 components are provided in Appendix C.1.

4.3.3 Labeled Projections of the Embedding Space

While the previous section examined similarity relationships between windows, it did not show which process characteristics the embedding space might reflect. To investigate this, manually labeled subsets of the data were projected into PCA space. The labels were used only to interpret the embedding projections, not to train the embedding model.

A subset of windows from the RealSense recording was manually labeled to indicate whether the operator was actively moving or stationary. The labeled windows were then projected onto principal components.

As shown in Figure 4.5, movement and stationary windows formed two visually distinct groups in the PC1-PC2 projection. This shows that differences related to operator motion were reflected in the embedding space.

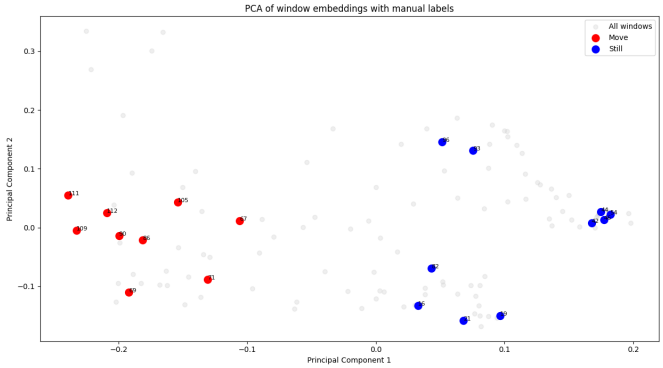


Figure 4.5: PCA projection (PC1 vs PC2) of windows labeled as "Move" and "Still".

To further examine whether the embedding representation reflected spatial context within the resetting process, another subset of windows was manually labeled according to predefined workspace zones. Three zones were defined based on the machine's physical layout and the surrounding work area, along with transition periods that represent movement between them.

An initial projection using PC1 and PC2 did not show a clear separation between the zones. However, when alternative component combinations were examined, stronger separation was observed. In particular, the projection using PC1 and PC3 showed clearer grouping of the labeled zones, with transition windows positioned between them, as shown in Figure 4.6.

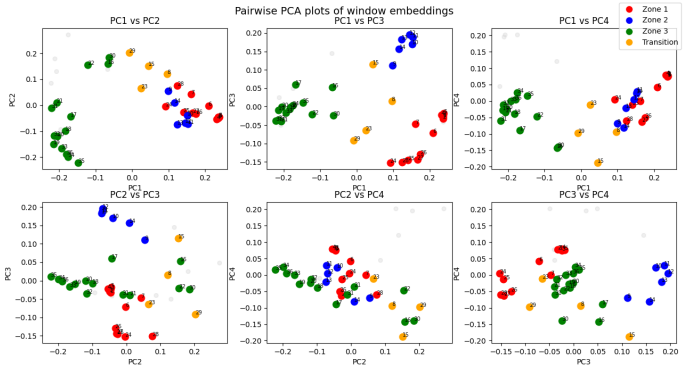


Figure 4.6: PCA projections of RealSense embeddings labeled by workspace zone and transition periods.

The embedding structure was also examined using the GoPro recording, which captures a close-up view of the operator's interaction with the machine. A PCA projection of the embeddings was generated, with windows colored according to their temporal order.

As shown in Figure 4.7, the GoPro embeddings were not randomly distributed in the PCA projection. Instead, temporally adjacent windows tended to be located

close together, forming a visible trajectory through the embedding space.

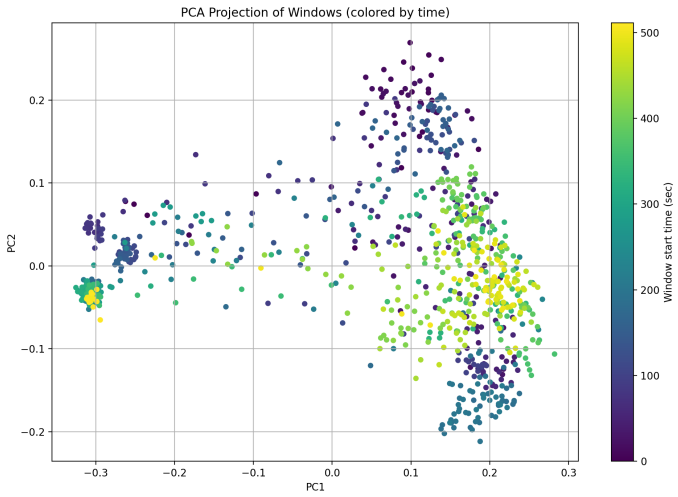


Figure 4.7: PCA projection (PC1 vs PC2) of GoPro embeddings colored by time.

Together, these labeled projections show that the embedding space contains visible structure related to movement state, workspace zone, and temporal progression. The combined interpretation of these patterns is presented in the following subsection.

4.3.4 What the Embedding Representation Captures

The results from the earlier sections show a consistent view of what aspects of the resetting process are represented in the embedding space. By combining the findings from the PCA analysis, similarity distributions, and manually labeled projections, it is possible to describe the type of information captured and its structure.

First, across both camera perspectives, the embedding representation yields clear, consistent results that capture differences in operator motion. The separation between moving and stationary segments observed in the PCA projections demonstrates that motion-related variation constitutes a dominant component of the embedding space. This is further supported by the similarity analysis, which shows that consecutive windows remain highly similar, while transitions between different states result in noticeable changes in similarity.

Second, the embedding space preserves temporal structure. The similarity distributions show that consecutive windows consistently exhibit higher cosine similarity compared to randomly sampled window pairs. This indicates that the representation evolves smoothly over time and reflects the continuity of the underlying process. At the same time, lower similarity values across non-consecutive windows demonstrate that the embedding space is sensitive to changes in process state.

Third, the results show that spatial information related to the operator's position within the workspace is encoded in the embedding space. The analysis based on manually labeled zones demonstrates that distinct spatial regions can be separated

when considering appropriate principal components. In particular, the comparison across multiple component pairs revealed that zone-related structure was not fully visible in the first two principal components but became clearly distinguishable when higher-order components were included.

Taken together, these results indicate that the embedding representation captures structured variation in movement, temporal progression, and spatial context during the resetting process. At the same time, the analysis suggests that this structure is distributed across multiple dimensions, meaning that different aspects of the process become visible depending on how the embedding space is analyzed.

4.4 Vision-Language Model visibility detection

The Qwen3-VL 8B model was prompted to perform per-frame binary visibility detection on 513 frames of the SGP 320 dismantling recording. For each frame, the model produced a yes/no decision for the 17 items in the prompt vocabulary: 1 operator label, 4 tools, and 12 components. Of these, 12 are observed in the ground-truth annotation, and 5 act as negative controls named in the prompt but never visible in the recording.

4.4.1 Per-item performance

Table 4.4 reports the confusion-matrix counts and classification metrics for every item in the vocabulary. Rows are sorted by F1 in descending order. The five negative controls appear at the bottom because F1 is undefined when no positive ground-truth frames exist. The per-item performance falls into three findings, described below.

Table 4.4: Per-item classification performance of Qwen3-VL 8B on the SGP 320 dismantling video.

Item	TP	FP	TN	FN	Support	Precision	Recall	F1
Grinding wheel	500	0	0	13	513	1.00	0.97	0.99
Operator	326	96	84	7	333	0.77	0.98	0.86
Guard plate	154	148	94	117	271	0.51	0.57	0.54
Hex key	102	1	203	207	309	0.99	0.33	0.50
Inlet chute	17	64	380	52	69	0.21	0.25	0.23
Bearing rings	2	1	494	16	18	0.67	0.11	0.19
Guide rail	4	11	401	97	101	0.27	0.04	0.07
Magnetic drivehead	0	0	223	290	290	–	0.00	0.00
Coolant nozzle	0	0	452	61	61	–	0.00	0.00
Feeder/ejector	0	0	406	107	107	–	0.00	0.00
Outlet chute	0	0	426	87	87	–	0.00	0.00
Microcentric bracket	0	0	369	144	144	–	0.00	0.00
Torque wrench	0	7	506	0	0	–	–	–
Pneumatic screwdriver	0	0	513	0	0	–	–	–
Fixed spanner	0	7	506	0	0	–	–	–
Measuring finger	0	3	510	0	0	–	–	–
Diamond roller	0	1	512	0	0	–	–	–

The first finding contains the two highest-scoring items. The grinding wheel F1 = 0.99 is visible in every frame and is correctly identified in 500 of 513 frames. The operator label achieves a high recall of 0.98 but a markedly lower precision of 0.77. The model returns operator in frame: yes for 96 of the 180 frames in which the operator is absent, a false-positive rate of 53.3 %. The model, therefore, shows a strong positive bias toward operator presence and is unreliable as a negation signal.

The second finding contains items identified with a partial but non-trivial signal. The hex key, despite being the only observed tool, produces the most informative pattern in the experiment: precision is near-perfect, 0.99, while recall is poor, 0.33. When the model predicts the hex key is visible, it is correct in 102 of 103 cases. When it predicts the hex key is not visible, it is wrong in 207 of 310 cases. This combination of high precision and low recall is consistent with a visibility effect rather than a failure to recognize the tool. The guard plate achieves moderate scores in both directions (precision 0.51, recall 0.57), while the inlet chute and bearing rings register only weakly, and the guide rail has a near-zero recall of 0.04 with low precision.

The third finding contains five components that the model never predicts as visible across the 513 frames, despite a combined ground-truth support of 689 positive

frames. The magnetic drivehead (290 ground-truth positives), microcentric bracket (144), feeder/ejector (107), outlet chute (87), and coolant nozzle (61) receive no positive predictions at all. Naming these components in the prompt was not sufficient for the model to ground them visually.

4.4.2 Negative controls and hallucination

Table 4.5 reports false-positive rates for the five negative-control items, which are never visible in the recording. Three of the five produced at least one false positive. The torque wrench and fixed spanner each registered 7 across 513 frames (FPR = 1.36 %), the measuring finger 3, and the diamond roller 1, and the pneumatic screwdriver produced none. The aggregate false-positive rate across the five controls was 0.7 % of frame-item cells, which is low. The model rarely asserts an item that is never present, so hallucination is not a major failure mode in the negative controls.

Table 4.5: False-positive rate (FPR = FP / (FP + TN)) for negative-control items that are named in the prompt but never visible in the video.

Item	FP	TN	FPR
Torque wrench	7	506	1.36 %
Pneumatic screwdriver	0	513	0.00 %
Fixed spanner	7	506	1.36 %
Measuring finger	3	510	0.58 %
Diamond roller	1	512	0.19 %

4.4.3 Group aggregates

Table 4.6 reports group-level F1 scores aggregated across the operator, tool, and component groups. The components row is reported both with and without the grinding wheel, which is visible in every frame and would otherwise dominate any aggregate it appears in. With the grinding wheel included, the macro-F1 for components is 0.20, and without it, the macro-F1 drops to 0.11. The latter is the more representative characterization of the model's component-recognition capability on this machine.

Table 4.6: Group-level F1 scores: Macro-F1 (unweighted mean) and support-weighted F1. Components row shown with/without the frame-dominating grinding wheel.

Group	Items	With support	Macro-F1	Weighted F1
Operator	1	1	0.864	0.864
Tools (observed)	1	1	0.495	0.495
Components (all observed)	10	10	0.201	0.408
Components (excl. grinding wheel)	9	9	0.114	0.150

4.4.4 Temporal failure analysis

Figure 4.8 visualizes the per-item, per-frame classification outcome across the full 08:32 recording. Three temporal patterns emerge that are not visible in the aggregate metrics.

Figure 4.8: Per-item temporal classification of Qwen3-VL 8B output against ground truth.

First, the hex key row shows two clearly distinct phases. From frame 0 to approximately frame 70, when the operator holds the hex key in clear view, the model reliably predicts it. From frame 70 onward, despite the operator continuing to use the hex key. Throughout most of the recording, the model registers almost exclu-

sively false negatives. A second region of partial recovery appears around frames 370-405.

Second, the four components with zero recall (magnetic drivehead, microcentric bracket, feeder/ejector, outlet chute) appear as continuous false-negative bands across their entire ground-truth visibility windows. The magnetic drivehead, in particular, spans roughly frame 213 to frame 500, nearly half the recording, as a single uninterrupted block. No isolated true positives appear within these regions, ruling out the explanation that the model occasionally identifies the part.

Third, the negative-control false positives are not uniformly distributed in time. Torque wrench false positives cluster around frame 30, frames 130-140, and frame 425. Fixed-spanner false positives cluster around frames 5-12 and frames 390-402. These clusters coincide with phases in which the operator is using the hex key, suggesting that the hallucinations are not random noise but reflect the model's uncertainty about which named tool the visible object corresponds to.

4.5 Complementary Data Source Analysis

The preceding analyses show that video can provide external observational information about the reset process, including visible actions, movement patterns, workspace transitions, temporal structure, and selected object or component visibility. However, the results also show that video-derived information is bounded by camera perspective, visibility, model capability, and the degree of interpretation required.

4.5.1 Complementary Machine and Interface Data

The complementary data sources described in Sections 3.3.1 and 3.9.1 provide examples of information that can contextualize the video-based results. The VNC-based HMI recording provides direct visibility into the machine interface, including menu navigation, displayed values, and digital interactions with the control system. This complements the external video by showing what the operator is viewing or changing as they interact with the panel.

The OPC-based parameter snapshots add selected internal machine-state information at trigger events. These snapshots provide access to values that are not visible in the camera recordings, such as selected compensation-related changes, timer-based machine-state snapshots, and piece-counter events. Compared with video, this information describes internal states and event markers rather than visible operator behavior.

The complementary sources, therefore, describe different aspects of the same reset process. A video can show the visible actions surrounding an adjustment, test run, or inspection, while HMI and machine-side data can provide interface and internal-state context for those actions. However, these sources were not analyzed as a fully integrated multimodal dataset in this thesis. Their contribution is therefore to illustrate what additional context may be needed to interpret video-derived information,

rather than to validate a complete multimodal analysis.

Table 4.7 summarizes this relationship by mapping selected information needs to the contribution of video, the contribution of complementary data sources, and the remaining limitations. The overall pattern is that video contributes to the observable process layer, while complementary sources are needed for interface content, internal machine state, expert meaning, and output-related context.

Table 4.7: Complementarity between video-derived information and other data sources

Information need	Video contribution	Complementary source	Remaining limitation
Activity phases	Phase structure from annotation and embeddings.	Expert input and manuals define meaningful phases.	Boundaries remain interpretive.
Operator movement	Position signals and zone transitions.	Machine events provide timing context.	Movement does not explain intention.
Tool/component visibility	Some visible tools and components can be detected.	Manuals and experts define relevant components.	Domain-specific items remain unreliable.
HMI interaction	Shows physical interaction with the panel.	VNC recording captures menus and displayed values.	Requires alignment with physical actions.
Machine state	Shows visible adjustment activity.	OPC snapshots provide selected internal values.	Many parameters remain unobserved.
Output-related events	Shows test grinding or inspection activity.	Piece-counter snapshots mark produced rings.	Quality relationship not validated.
Operator reasoning	Shows hesitation or repeated actions.	Interviews explain tacit decisions.	Reasoning cannot be inferred directly.

4.6 Interview Findings

Two semi-structured expert interviews complemented the video-based analysis. The operator interview was conducted with an SKF operator working on the SGP 320, who is experienced but has limited reset experience with this specific machine. The machine manufacturer interview was conducted with three representatives from UVA Lidköping, the manufacturer of the SGP 320, with one participant leading the discussion, and was captured via a Teams call. The material is organized below into three topic areas relevant to the research questions: drivers of variation in reset time, experience-dependent decisions, and the boundaries of camera-based observation. Because the analysis rests on only two interviews, they are presented as structured expert input to help interpret the video findings, rather than as general-

izable themes.

4.6.1 Drivers of reset time variation

The machine manufacturer's interview identified dressing a new grinding wheel as the largest single source of variation in reset time. Reported durations ranged from 5 to 25 or 30 minutes, depending on the size of the dressing tool and the wheel's initial out-of-roundness. The manufacturer described this step as "*the biggest thief of all*" in the total reset time. On the SGP 320, the wheel diameter requires more material removal to achieve consistent contact across the wheel face than on smaller machines in the same family.

A second contributor identified in the machine manufacturer interview was the drivehead grinding. The face tolerance is on the order of one micrometer, and the drivehead must be ground in the machine because its critical reference is the machine's own spindle. Attempts to prepare the drivehead externally were reported as unsuccessful. Design mitigations, such as cylindrical locating pins, reduce variability when the same tooling is reused on the same machine. However, this benefit is lost when the tooling is rotated between machines.

The operator interview identified the test grinding phase as a further source of variation. The operator estimated that an experienced operator may need 1 to 2 parts to achieve acceptable output, while a less experienced operator may need 8 to 15 parts or more. The operator attributed differences between faster and slower operators primarily to routine rather than to step complexity, stating that "*it is more the routine that makes him faster, rather than some parts being more or less complex.*"

The operator interview also described tool organization as a recurring source of lost time, citing situations in which a component is mounted before another that should have been installed first, necessitating partial disassembly to correct the issue. The machine manufacturer interview noted that the intended workflow assumes replacement tooling is pre-set in a separate preparation station, but that customers often operate with only one set of tooling, which prevents parallel preparation.

4.6.2 Experience-dependent decisions

The operator interview described the auditory assessment of the grinding process as a tacit judgment that is not captured by the machine instrumentation. "*Sometimes you hear that it does not sound right when it is grinding, so you stop because you realize it is not grinding well. And you cannot get that from any graph. You hear it and think it does not sound right.*" The operator distinguished this from data the machine already records, such as vibration and force measurements, noting that these signals exist but do not directly indicate whether the grinding is producing acceptable output.

The machine manufacturer's interview described the reading of the acoustic emission pattern during wheel dressing as an experience-based judgment. The SGP 320

is equipped with acoustic emission monitoring as standard, but interpreting the pattern is left to the operator, who visually assesses the contact graph displayed during dressing. The manufacturer noted that a newer, exact address function is intended to automate this judgment by comparing measured patterns against a reference template. However, stated that this function is not yet implemented in practice and that the assessment therefore remains manual.

The operator interview described the inspection of components during disassembly as another step that depends on experience, particularly regarding the drivehead. Damage may not be visible before the changeover begins. The operator stated that "*for each part you remove, you see how that part is doing*" and noted that there is a risk of remounting a damaged drivehead without spotting the problem if the operator lacks experience.

4.6.3 Boundaries of camera-based observation

The machine manufacturer interview drew on UVA's own prior experience with camera-based monitoring during customer support and commissioning. On capturing the machine HMI display. The manufacturer stated: "*What we have not succeeded with is getting a camera to capture the machine's screen properly. So we have used VNC software where we duplicate the whole screen.*" This approach corresponds to the VNC-based HMI capture described in Section 3.3.1.

The machine manufacturer's interview also described previous attempts to mount a GoPro or similar camera inside an SGP machine. Coolant and water were reported to make it hard to see anything from inside the machine, and tight spacing around the splash guards and the grinding wheel limited possible camera positions. Latency in the video feedback prevented real-time use during commissioning. A camera positioned in a window of a larger machine was reported to be visually appealing but to provide no useful process information due to its distance from the grinding zone and the coolant's obscuring effect.

The operator interview reinforced the practical limits of fixed-camera observation, stating that a single camera position cannot capture all relevant steps and that the camera would need to be moved between work areas. The operator also noted that the machine already provides much of the data that would otherwise need to be inferred visually.

The machine manufacturer interview identified phases where camera-based observation can usefully capture process information. For the phases when the machine door is open, a camera placed on the operator side was described as suitable for capturing tool contact surfaces, reference surface checks, guide and loading plate adjustments, and the use of lifting equipment during wheel changes.

5

Discussion

5.1 Answering the Research Questions

5.1.1 RQ1: Process-Relevant Information from Video

The findings indicate that video-derived information should not be treated as a single, uniform form of process data. Rather, the study shows that video recordings can be transformed into several levels of process-relevant information depending on the analytical approach used. Manual annotation provides explicit phase-level information by segmenting video into interpretable activities and durations. Tracking and depth extraction provide spatial information by representing the operator's position and movement within the workspace. Embedding-based analysis provides a latent visual-temporal representation of short video windows, capturing structure related to temporal continuity, movement, and spatial context. Vision-language models add a semantic layer by attempting to identify visible tools and components. This means that the answer to RQ1 depends not only on the video itself, but also on the analytical level at which the video is processed and interpreted.

At the most explicit level, manual annotation showed that video can be transformed into a structured phase-level representation of work. This is valuable because it produces directly interpretable information, allowing activity categories to be defined, phase boundaries to be marked, and the duration of each phase to be calculated. Manual annotation, therefore, provides the clearest link between raw video and a human-readable description of the process. At the same time, the pilot also illustrates the main limitation of this approach. Even a short video sequence required continuous interpretation by the annotator, and the resulting phase boundaries depended on human judgment. Manual annotation is therefore most useful as an exploratory tool for developing phase categories and reference labels, rather than as a scalable method for analyzing larger collections of reset recordings.

Nevertheless, manual annotation remains valuable in an exploratory setting. When predefined activity phases are not yet clearly established, investing time in manual annotation can help identify relevant categories, define phase boundaries, and create reference examples. These reference labels can then support the validation and interpretation of more automated approaches. This is particularly important for methods such as embeddings, where the output is not directly human-readable and

must be interpreted by relating patterns in the representation back to observable process activities.

The operator tracking and depth analysis showed that video can also provide spatial process information. By combining person detection with RealSense depth data, the operator's position could be represented as a time-based signal and transformed into zone-level indicators, such as time spent in different work areas and transitions between zones. This type of information is process-relevant because reset work involves movement between the machine, workstation, interface, and surrounding workspace, rather than only a sequence of hand actions.

The level of detail provided by this approach is mainly spatial and temporal. It can identify where the operator is located, when the operator changes position, how long the operator remains in predefined areas, and when transitions between areas occur. However, it does not explain the purpose or meaning of the activity within each zone. For example, a transition from the machine area to the interface area may indicate parameter adjustment, verification, or troubleshooting, but the tracking signal alone cannot distinguish among these. Tracking-based information is, therefore, useful as a structured movement layer. However, it must be interpreted in conjunction with video inspection, HMI data, machine parameters, and expert knowledge to understand the underlying task.

The embedding-based analysis provided a more abstract level of process-relevant information than manual annotation and tracking. Rather than extracting predefined phases or explicit spatial coordinates, the video was represented as a sequence of numerical vectors corresponding to short temporal windows. The results showed that these vectors contained structured variation. PCA indicated that a relatively small number of components captured much of the variance in the data, while the similarity analysis showed that temporally adjacent windows were generally represented as more similar than randomly selected windows. The manually labeled projections further showed that parts of the embedding space corresponded to movement state, workspace zone, and temporal progression. This means that embeddings can capture recurring structure in the reset recording without requiring every segment to be manually annotated in advance.

At the same time, the level of detail provided by embeddings is indirect. The embedding space does not directly state what the operator is doing, which process phase a segment belongs to, or whether an activity has been performed correctly. These meanings have to be inferred by relating the numerical representation to timestamps, sparse labels, and the original video. Embeddings are therefore valuable as an exploratory layer for identifying patterns, similarities, and candidate transitions, but they do not replace human interpretation or validated process labels. In relation to RQ1, embeddings indicate that a video can provide structured latent information about the reset process, but not a complete semantic description.

A further limitation is that the embeddings were generated using a pretrained model that was not specifically trained for this industrial setting. In the reset recordings, many video windows share the same machine, camera perspective, operator, and

background. The relative visual differences between windows are therefore often subtle compared with differences between videos from completely different environments. As a result, the embedding space may capture both shared contextual features and process-relevant differences. PCA helped to reduce this effect by emphasizing dominant patterns of variation and making the embedding space more interpretable. However, PCA also introduces an additional interpretive step. A reduced projection does not preserve all information from the original 512-dimensional representation, and two-dimensional plots only show selected component combinations. As a result, PCA is useful for examining expected patterns in the data, but more subtle or unexpected patterns may be overlooked if they are not visible in the selected component projections.

The VLM added a further level of analysis by turning each frame's visual content into short yes/no answers, indicating whether the operator, tools, and components were visible. Its main advantage over the other levels is that the output is directly readable. However, the results show that the level of detail it can deliver depends strongly on whether the named item matches a visual concept that the underlying model has been trained to recognize. Common items, such as the operator and the grinding wheel, were reliably detected. In contrast, domain-specific components, such as the magnetic drivehead and microcentric bracket, were never predicted to be visible despite being present for substantial parts of the recording. The negative-control items further showed that the model occasionally hallucinates named tools when uncertain. For RQ1, this means that a general-purpose VLM can confirm the presence of common, visually distinctive items but, in its current form, cannot reliably produce per-frame visibility information for the domain-specific components that define the reset process.

The level of detail that can be extracted from video depends on the observability of the activity and the camera perspective used. The RealSense recording was useful for capturing the wider workspace and, therefore, supported analysis of operator movement and position. The GoPro recording provided a closer view of interactions inside the machine and was therefore better suited for inspecting fine-grained manual activity. However, neither perspective provided complete access to all relevant process information. External video could show the operator moving to the machine interface or performing an adjustment. However, it could not reliably show the exact HMI content, active parameter values, measurement results, or the reasoning behind the action. This means that the level of detail is not determined solely by the video recording, but by the combination of camera placement, visibility, resolution, occlusion, and the analytical method applied to the recording. An important implication is that the most detailed representation is not always the most useful representation. Manual annotation provides rich, interpretable detail but is difficult to scale, while tracking and embeddings provide less explicit information but may be more suitable for analyzing larger amounts of video.

Overall, the findings show that video is most valuable as an external source of observational data. It can make the reset process visible as a sequence of actions, movements, transitions, and recurring patterns. The results also show that video-

derived information should not be interpreted as a complete representation of the reset process. Important aspects of resetting remain partly or entirely outside what external video can capture. The answer to RQ1 is therefore that video can provide several forms of process-relevant information, but primarily about the observable execution of the reset rather than the full technical and cognitive logic behind it. For a future AI-based operator assistance system, this means that video is most relevant as an input for recognizing observable process context, such as activity progression, movement patterns, repeated actions, and visible interactions. However, it would need to be combined with other data sources before it could provide reliable guidance.

5.1.2 RQ2: Contextualizing Video for AI-Based Operator Support

RQ2 concerns the extent to which video-derived information can support a future AI-based operator assistance system and the complementary data sources needed to contextualize it. Building on the answer to RQ1, the main contribution of video is that it provides an external observational layer of the reset process. Machine-generated data can describe internal states, events, parameters, or outcomes. A video can show how the reset is physically carried out, how the operator moves, interacts with the machine, handles tools, and repeats or transitions between activities. Video, therefore, does not replace existing data sources, but adds process context that is difficult to obtain from other sources alone.

The complementary sources explored in this study should be understood as examples rather than as an exhaustive set of data needed for a future assistance system. The HMI screen recording, OPC-based parameter snapshots, interviews, and internal documentation each addressed specific limitations of video that are either difficult or impossible to interpret. For example, video data can show that an action is being performed, while machine-state data may show which internal values or events were present at the same point in time. When combined, they can provide a much more concrete understanding of the process. Other sources may also be relevant depending on the intended use of the system. These could include alarm logs, acoustic emission signals, quality measurements, maintenance records, tool-preparation data, or operator notes. The broader point is that a single data type cannot fully represent the reset process. A future AI-based assistance system would need to combine sources that describe different aspects of the same process, including what the operator does, the machine state, what the interface displays, and why a particular action or parameter change matters.

A central implication from the results is that multimodal support should not treat each data source as a separate analytical output. The value of combining data sources lies in aligning them around shared reset events or time windows. In the context of this study, the most relevant unit of analysis appears to be the moment or short sequence in which an operator action, an HMI interaction, and a machine-state change occur in relation to the same reset task. For example, a video may show the operator performing an adjustment, the HMI recording may show which menu or

parameter page was active, and parameter snapshots may show the selected internal values at or near the same point in time. None of these sources alone explains the full event, but together they can connect the visible action, the digital interaction, and the internal machine state. This requires decisions about what the common unit of analysis should be, such as a timestamp, a video window, a process phase, or a reset event, and clear rules for linking data points that do not occur at the same time or describe the process at the same level of detail. A practical way to support this type of event-level alignment is to store the different data streams in a time-indexed structure, such as a time-series database, where video windows, HMI states, parameter snapshots, and machine events can be linked via shared timestamps.

The exploratory complementary data collection illustrates how such an event-level representation could begin to be constructed. The VNC-based HMI recording addressed a limitation of external video by capturing interface navigation and displaying values directly. The OPC-based parameter snapshots added selected internal machine-state values at trigger events, including timer intervals, compensation changes, and piece-counter updates. Interviews and internal documentation provided an additional interpretive layer, clarifying which actions, components, and parameter areas are meaningful in the reset process. These sources, therefore, do not duplicate the video-derived information but add different types of context that help explain what the visible actions may correspond to. However, the present study did not have the scope to evaluate how these sources perform when analyzed together as a fully integrated multimodal dataset. The contribution is therefore limited to identifying and structuring complementary data layers rather than validating their combined explanatory power.

The answer to RQ2 is therefore that video-derived information can support a future AI-based operator assistance system by providing observable process context. However, it is not sufficient on its own. Its greatest value is not as an isolated data stream, but as one modality in a shared representation of the reset process. This thesis does not demonstrate a complete multimodal AI-based assistance system. Moreover, it identifies how such a system could be grounded: by aligning external video, HMI activity, internal machine data, and expert knowledge around common reset events. This provides a more realistic basis for future operator assistance than relying on a single data source.

5.2 Practical and Methodological Implications

The findings have several practical implications for the continued development of video-based and multimodal operator support at SKF. Since this thesis is exploratory, the implications should not be understood as final implementation guidelines, but as design considerations for future data collection, model development, and system evaluation.

5.2.1 Interpretability and Scalability

A key implication of the findings is that video-based process analysis involves a trade-off between interpretability and scalability. Highly interpretable representations require human judgment, since process meaning must be defined, labeled, and validated by people with domain knowledge. More automated representations can support analysis of larger amounts of video, but their outputs are less transparent and require validation before they can be trusted as process knowledge.

For future operator support, there is therefore no single optimal level of analysis. Detailed human interpretation is useful for defining meaningful activities, components, and decision points. While scalable computational methods can help identify recurring patterns, similar situations, or candidate deviations across larger datasets. For SKF, a practical development path is to combine expert input and selected manual labels that provide reference points, with automated methods for searching, comparing, and structuring reset recordings. Automation should therefore be viewed as a way to reduce and guide human interpretation, rather than replace it entirely.

5.2.2 Domain Specificity

A second implication concerns the domain specificity of the models used for video analysis. The experiments in this thesis were conducted with models that could be run locally, which was necessary due to practical and confidentiality constraints. However, this also limited the available model capacity compared with larger externally hosted models. More capable models may offer stronger recognition and reasoning performance, but their use in an industrial setting must be weighed against data security, cost, and confidentiality requirements.

The findings also show that general model capability does not automatically translate into industrial process understanding. Models trained on broad, general datasets may recognize common visual concepts, but they cannot be assumed to understand specialized machine components, subtle reset actions, or process-specific states. For future operator support, this means models should be evaluated in the specific industrial context where they will be used, rather than selected solely on general benchmark performance.

A practical implication is that future development should include domain-specific validation and, where necessary, adaptation. This could involve collecting additional reset recordings, creating targeted labels for important tools and components, fine-tuning models on industrial material, or combining model outputs with expert-defined process knowledge. Without such adaptation, there is a risk that the system appears technically capable while still missing the details that matter most in the reset process.

5.2.3 Observability

A final implication concerns observability. The specific information should guide camera placement rather than by a general idea of a "good" camera angle. Different perspectives make different aspects of the reset process visible, and no single view can capture all relevant information. This means that future video collection should begin by defining what the system or analyst needs to observe, such as workspace movement, tool use, component handling, or fine manual adjustments.

For SKF, the practical implication is that video should be treated as a selective observational layer rather than a complete data source. When the relevant information is visual, camera placement should be designed around that specific process question. When the information is displayed on the HMI, stored internally in the machine, or based on operator judgment, external video is unlikely to be sufficient. Direct interface capture, machine data, or expert input should complement it.

5.3 Methodological Limitations and Trustworthiness

Several limitations affect how the findings should be interpreted. First, the study is based on a single industrial case: the SGP 320 resetting process at SKF. Operator availability, access restrictions, and practical constraints around recording reset events limited the empirical material. The findings should therefore be understood as context-specific analytical insights rather than statistically generalizable results.

Second, the empirical basis was limited. Only a small number of reset recordings were collected, and only two expert interviews were conducted: one with an SKF operator and one with representatives from the machine manufacturer. These interviews provided valuable process understanding, but they cannot represent the full range of operator experience, reset variation, or machine-use practices.

Third, several parts of the analysis relied on the researcher's interpretation. This includes the manual annotation pilot, workspace zone definitions, labeled embedding projections, VLM visibility log, and thematic analysis of interviews. These interpretations were grounded in the video material, machine manual, and expert input, but no formal inter-rater reliability assessment was conducted. The labels and boundaries should therefore be treated as exploratory rather than an objectively validated ground truth.

Fourth, the technical experiments were focused on feasibility. The tracking analysis was sensitive to camera placement, occlusion, multiple people in the scene, and depth-sensor noise. The embedding analysis used a pretrained model not specifically trained on industrial resetting, meaning that the representations may capture shared scene context as well as process-relevant variation. The VLM experiment was conducted on a single dismantling video with one visibility log, so the results describe model behavior in this specific setting rather than general VLM performance.

Finally, the study did not validate a complete multimodal assistance system. HMI screen recordings and OPC-based parameter snapshots were collected and structured to identify complementary data that could contextualize video-derived information. However, they were not analyzed alongside the video as a synchronized multimodal dataset. The thesis can therefore support conclusions about feasible information types, practical constraints, and complementary data needs. It cannot be concluded that video-based or multimodal support would reduce reset time, improve quality, or provide reliable real-time operator guidance without further validation.

5.4 Contribution to SKF

This thesis contributes to SKF by providing both empirical material and practical groundwork for continued investigation of video as a data source in machine resetting. The collected material includes recordings of full reset events, expert interview material, and structured outputs from the video-based analyses. In addition, the thesis work initiated the retrieval of internal machine parameters via OPC-based trigger scripts and developed a pipeline to sort the extracted data into a database structure. Together, these contributions provide SKF with a starting point for continuing the work beyond the thesis, both by reusing the collected material and by further developing the data-processing workflows established during the project. The workflows are not production-ready systems, but they demonstrate how raw recordings and machine-side data can be transformed into formats that are more suitable for analysis, comparison, and future integration.

The thesis also contributes to a clearer understanding of the role video can play in a future AI-based operator assistance system. The findings show that video is useful for capturing observable process context, such as operator actions, movement, workspace transitions, and visible interaction with the machine. At the same time, the study clarifies that video alone is insufficient for reliable operator guidance, as internal machine states, parameter values, quality outcomes, HMI content, and operator reasoning require complementary data sources. For SKF, the main contribution is therefore not a finished AI solution, but a structured foundation for deciding what data should be collected, how it can be organized, and how video may fit into a future multimodal support system.

5.5 Future Research

Future research should focus on moving from exploratory feasibility tests toward a more systematic data foundation for AI-based operator support. A first step is to expand the empirical dataset by recording a larger number of complete reset events across different operators, product variants, and production conditions. This would make it possible to distinguish patterns specific to a single recording or operator from those that recur consistently across reset occasions.

A central next step is to develop a synchronized multimodal reset dataset. Observational video data, along with other data sources such as internal machine parame-

ters, machine events, and quality-related outcomes, should be aligned on a common timeline. This would allow future analyses to connect visible operator actions with interface interaction and internal machine-state changes, rather than treating each source as a separate dataset. Such a dataset would also provide a stronger basis for evaluating whether video-derived information can support operator guidance in practice.

Future work should also strengthen the annotation and validation process. A larger set of reset recordings should be annotated using a clearer coding scheme, ideally by multiple annotators and with expert review. This would enable evaluation of inter-rater agreement and the creation of more reliable reference labels for training or validating automated methods. In particular, annotation should focus on reset phases, tool and component interaction, HMI actions, and iterative adjustment or verification loops.

Finally, an important direction is to evaluate models better suited to industrial reset work. The present thesis used R3D-18 as a baseline representation model, but future work could investigate self-supervised transformer-based models trained on larger collections of unlabeled industrial process videos. For example, Vybornova et al. [17] demonstrated that a Vision Transformer pretrained on unlabeled manufacturing videos using a V-JEPA self-supervised approach achieved strong temporal action segmentation performance with limited annotations. Given the similarity between multi-step maintenance procedures and the reset process studied here, such approaches may offer a more powerful alternative for learning reset-specific activity representations while reducing the amount of manual annotation required.

6

Conclusion

This thesis investigated whether video recordings can serve as a useful source of observational data for the resetting process of the SGP 320 grinding machine at SKF, and how such data could support the future development of a multimodal, AI-based operator assistance system.

For the first research question, the results show that video captures what the operator actually does during a reset. It shows actions, movements, and the process's development over time. The study also demonstrates that this information can be structured at different levels. For example, the process can be divided into phases, operator movement can be tracked to show where work takes place, and patterns over time can be identified using learned representations. At the same time, video has clear limits. It does not capture internal machine states, parameter values, or the reasoning behind operator decisions. It also struggles to identify machine-specific components in detail when using general AI models. This means that the video is reliable for understanding the reset's execution, but not sufficient to explain the full process on its own.

For the second research question, the findings show that video should be seen as one part of a broader system rather than a complete solution. Video can describe what is happening physically, such as actions and interactions, but it needs to be combined with other data sources to provide context. Information from the machine interface, internal parameters, and system events is necessary to understand what the machine is doing and why certain decisions are made. Without this additional information, video data alone cannot support reliable decision-making or guidance.

From a practical standpoint, the study highlights several important considerations. Camera placement is critical, since what can be analyzed depends directly on what is visible. It is also beneficial to focus on simple and robust signals early on, such as movement between work areas or clear interaction steps. These are easier to extract and validate compared to detailed object recognition.

The study has limitations. It is based on a single machine and a limited number of recordings, and the data was collected under controlled conditions. As a result, the results are specific to the case studied and cannot be generalized without further validation. Future work should include more recordings under real-world production conditions, better synchronization across data sources, and models adapted to

industrial environments.

In conclusion, video is a useful tool for capturing how the resetting process is carried out in practice, especially for actions, movements, and variations. Its main strength is in how well it works with a combined data setup. When used together with machine and interface data, it can contribute to a more complete understanding of the process and support the development of future operator assistance systems.

Bibliography

[1] G. Da Silveira, D. Borenstein, and F. S. Fogliatto, "Mass customization: Literature review and research directions," *International Journal of Production Economics*, vol. 72, no. 1, pp. 1–13, 2001. doi:10.1016/S0925-5273(00)00079-7.

[2] Z. L. Gan, S. N. Musa, and H. J. Yap, "A Review of the High-Mix, Low-Volume Manufacturing Industry," *Applied Sciences*, vol. 13, no. 3, p. 1687, 2023. doi:10.3390/app13031687.

[3] C. Low, T.-H. Wu, and C.-M. Hsu, "Mathematical modelling of multi-objective job shop scheduling with dependent setups and re-entrant operations," *The International Journal of Advanced Manufacturing Technology*, vol. 27, pp. 181–189, 2005. doi:10.1007/s00170-004-2137-0.

[4] T. Ruppert, R. Csalodi, and J. Abonyi, "Estimation of machine setup and changeover times by survival analysis," *Computers & Industrial Engineering*, vol. 153, p. 107026, 2021. doi:10.1016/j.cie.2020.107026.

[5] L. Tang and G. Wang, "Decision support system for the batching problems of steelmaking and continuous-casting production," *Omega*, vol. 36, no. 6, pp. 976–991, 2008. doi:10.1016/j.omega.2007.11.002.

[6] T. Engström and P. Medbo, "Data collection and analysis of manual work using video recording and personal computer techniques," *International Journal of Industrial Ergonomics*, vol. 19, no. 4, pp. 291–298, 1997. doi:10.1016/S0169-8141(96)00038-8.

[7] B. B. Van Acker, D. D. Parmentier, P. D. Conradie, S. Van Hove, A. Biondi, K. Bombeke, P. Vlerick, and J. Saldien, "Development and validation of a behavioural video coding scheme for detecting mental workload in manual assembly," *Ergonomics*, vol. 64, no. 1, pp. 78–102, 2021. doi:10.1080/00140139.2020.1811400.

[8] M.-A. Zamora-Hernández, J. A. Castro-Vargas, J. Azorin-Lopez, and J. Garcia-Rodriguez, "Deep learning-based visual control assistant for assembly in Industry 4.0," *Computers in Industry*, vol. 131, p. 103485, 2021. doi:10.1016/j.compind.2021.103485.

[9] AVIX Suite, "Avix method: Method and time study software for process doc-

umentation and production planning," n.d.

[10] F. Hashimoto, "Fundamentals of shoe centerless grinding." AFT Lecture Series, Advanced Finishing Technology Ltd., 2025.

[11] S. Shingo, *A Revolution in Manufacturing: The SMED System*. Stamford, CT, USA: Productivity Press, 1985.

[12] D. Guzel and A. Shahbazpour Asiabi, "Improvement setup time by using smed and 5s (an application in sme)," *International Journal of Scientific & Technology Research*, vol. 9, no. 1, pp. 3727–3732, 2020.

[13] B. Zhang, Z. Gan, Y. Yang, and T. D. Howes, "Workholding stability in shoe centerless grinding," *Journal of Manufacturing Science and Engineering*, vol. 121, no. 1, pp. 41–48, 1999. doi:10.1115/1.2830573.

[14] F. Hashimoto, "Generalization of geometrical rounding mechanism in regenerative centerless supports," *Processes*, vol. 13, no. 4, p. 1103, 2025. doi:10.3390/pr13041103.

[15] UVA Lidköping AB, "LIDKÖPING High Volume SGP 320 – External." `https://www.uvalidkoping.com/machine/lidkoping-sgp-320/`, 2024. Accessed: 2026-05-05.

[16] Y. Sheng, G. Zhang, Y. Zhang, M. Luo, Y. Pang, and Q. Wang, "A multimodal data sensing and feature learning-based self-adaptive hybrid approach for machining quality prediction," *Advanced Engineering Informatics*, vol. 59, p. 102324, 2024. doi:10.1016/j.aei.2023.102324.

[17] Y. Vybornova, M. Aleshin, S. Illarionova, I. Novikov, D. Shadrin, A. Nikonorov, and E. Burnaev, "Self-supervised learning for temporal action segmentation in industrial and manufacturing videos," *IEEE Access*, vol. 13, pp. 39650–39665, 2025. doi:10.1109/ACCESS.2025.3545768.

[18] N. Hakam, K. Benfriha, V. Meyrueis, and C. Liotard, "Advanced monitoring of manufacturing process through video analytics," *Sensors*, vol. 24, no. 3, p. 687, 2024. doi:10.3390/s24030687.

[19] K. Kazmierczak, S. E. Mathiassen, and W. P. Neumann, "Observer reliability of industrial activity analysis based on video recordings," *International Journal of Industrial Ergonomics*, vol. 36, no. 3, pp. 275–282, 2006. doi:10.1016/j.ergon.2005.11.004.

[20] J. Oyekan, A. Fischer, W. Hutabarat, and C. Turner, "Utilising low cost RGB-D cameras to track the real-time progress of a manual assembly sequence," *Assembly Automation*, vol. 40, no. 5, pp. 661–670, 2020. doi:10.1108/AA-01-2019-0025.

[21] M. B. Shaikh and D. Chai, "RGB-D data-based action recognition: A review," *Sensors*, vol. 21, no. 12, p. 4246, 2021. doi:10.3390/s21124246.

[22] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou, "Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond." arXiv:2308.12966, 2023. [Online]. Available: `https://arxiv.org/abs/2308.12966`.

[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. doi:10.1109/ICCV.2015.510.

[24] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. doi:10.1109/CVPR.2018.00685.

[25] A. A. Aguileta, R. F. Brena, O. Mayora, E. Molino-Minero-Re, and L. A. Trejo, "Multi-sensor fusion for activity recognition: A survey," *Sensors*, vol. 19, no. 17, p. 3808, 2019. doi:10.3390/s19173808.

[26] J. Berger and S. Lu, "A multi-camera system for human detection and activity recognition," *Procedia CIRP*, vol. 107, pp. 1305–1310, 2022. doi:10.1016/j.procir.2022.05.148.

[27] Intel Corporation, "Intel® RealSense™ Depth Camera D455 – product specifications." `https://www.realsenseai.com/products/real-sense-depth-camera-d455f/`, 2023. Accessed: 2026-03-26.

[28] P. Hübner, J. Hou, and D. Iwaszczuk, "Evaluation of Intel RealSense D455 camera depth estimation for indoor SLAM applications," in *Archives of the Photogrammetry, Cartography and Remote Sensing*, vol. XLVIII-1-W3-2023, pp. 97–104, ISPRS, 2023. doi:10.5194/isprs-archives-XLVIII-1-W3-2023-97-2023.

[29] M. Servi, E. Mussi, A. Profili, R. Furferi, Y. Volpe, and L. Governi, "Metrological characterization and comparison of D415, D455, L515 RealSense devices in the close range," *Sensors*, vol. 21, no. 8, p. 2770, 2021. doi:10.3390/s21082770.

[30] GoPro, Inc., "HERO cameras – technical specifications." `https://gopro.com/en/us/shop/cameras/learn/hero13black/CHDHX-131-master.html`, 2024. Accessed: 2026-03-26.

[31] G. R. D. Bernardina, P. Cerveri, R. M. L. Barros, J. C. B. Marins, and A. P. Silvatti, "Action sport cameras as an instrument to perform a 3d underwater motion analysis," *PLOS ONE*, vol. 11, no. 8, p. e0160490, 2016. doi:10.1371/journal.pone.0160490.

[32] H. Hastedt, T. Ekkel, and T. Luhmann, "Evaluation of the quality of action cameras with wide-angle lenses in uav photogrammetry," in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sci-*

ences, vol. XLI-B1, pp. 851–859, 2016. doi:10.5194/isprs-archives-XLI-B1-851-2016.

[33] O. Önal and E. Dandıl, "Unsafe-Net: YOLO v4 and ConvLSTM based computer vision system for real-time detection of unsafe behaviours in workplace," *Multimedia Tools and Applications*, vol. 84, pp. 34967–34993, 2025. doi:10.1007/s11042-024-19276-8.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[35] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. doi:10.1109/CVPR.2017.502.

[36] C. Tong, J. Ge, and N. D. Lane, "Zero-shot learning for IMU-based activity recognition using video embeddings," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 4, pp. 1–23, 2021. doi:10.1145/3494995.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi:10.1109/CVPR.2016.90.

[38] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017. doi:10.48550/arXiv.1705.06950.

[39] I. T. Jolliffe, *Principal Component Analysis*. Springer Series in Statistics, New York, NY, USA: Springer, 2 ed., 2002. doi:10.1007/b98835.

[40] S. Danish, A. Sadeghi-Niaraki, S. U. Khan, L. M. Dang, L. Tightiz, and H. Moon, "A comprehensive survey of vision–language models: Pretrained models, fine-tuning, prompt engineering, adapters, and benchmark datasets," *Information Fusion*, vol. 126, p. 103623, 2026. doi:10.1016/j.inffus.2025.103623.

[41] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, Y. Fan, K. Dang, M. Du, X. Ren, R. Men, D. Liu, C. Zhou, J. Zhou, and J. Lin, "Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution." arXiv:2409.12191, 2024. [Online]. Available: `https://arxiv.org/abs/2409.12191`.

[42] G. Shinde, A. Ravi, E. Dey, S. Sakib, M. Rampure, and N. Roy, "A survey on efficient vision-language models," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 15, p. e70036, 2025. doi:10.1002/widm.70036.

[43] N. Gaw, S. Yousefi, and M. Reisi Gahrooei, "Multimodal data fusion for systems improvement: A review," *IISE Transactions*, vol. 54, no. 11, pp. 1098–1116, 2022. doi:10.1080/24725854.2021.1987593.

[44] M. McKinney, A. Garland, D. Cillessen, J. Adamczyk, D. Bolintineanu, M. Heiden, E. Fowler, and B. L. Boyce, "Unsupervised multimodal fusion of in-process sensor data for advanced manufacturing process monitoring," *arXiv preprint arXiv:2410.22558*, 2024. doi:10.48550/arXiv.2410.22558.

[45] H. O. Velesaca, J. A. Holgado-Terriza, and J. M. Gutierrez-Guerrero, "Industrial process automation through machine learning and OPC-UA: A systematic literature review," *Electronics*, vol. 14, no. 18, p. 3749, 2025. doi:10.3390/electronics14183749.

[46] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012. doi:10.1007/978-3-642-29044-2.

[47] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009. doi:10.1007/s10664-008-9102-8.

[48] O. Dieste, A. Grimán, and N. Juristo, "Developing search strategies for detecting relevant experiments," *Empirical Software Engineering*, vol. 14, pp. 513–539, 2009. doi:10.1007/s10664-008-9091-7.

[49] P. Vakkari, M. Pennanen, and S. Serola, "Changes of search terms and tactics while writing a research proposal: A longitudinal case study," *Information Processing & Management*, vol. 39, no. 3, pp. 445–463, 2003. doi:10.1016/S0306-4573(02)00031-6.

[50] M. Denscombe, *The Good Research Guide*. Maidenhead, UK: Open University Press, 5th ed., 2014. PDF edition.

[51] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006. doi:10.1191/1478088706qp063oa.

[52] K. Takahashi, K. Yamamoto, A. Kuchiba, and T. Koyama, "Confidence interval for micro-averaged f1 and macro-averaged f1 scores," *Applied Intelligence*, vol. 52, pp. 4961–4972, 2022. doi:10.1007/s10489-021-02635-5.

[53] European Parliament and Council of the European Union, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation)." Official Journal of the European Union, L 119, 2016. Accessed: 2026-02-06.

A

Interview Guide

Interview Guide – SGP 320 Reset Process

Master's thesis SKF / Chalmers University of Technology

Based on the operator's manual

This appendix reproduces the semi-structured interview guide used in the two expert interviews described in Section 3.4 of the thesis. The guide was originally written and conducted in Swedish. The questions are translated here for accessibility. For the interview with representatives from UVA Lidköping (the machine manufacturer), individual questions were adapted in real time to fit the manufacturer's perspective. The structure, time disposition, and themes were identical for both sessions.

Time disposition (≈ 30 minutes total)

Part	Focus	Time	RQ
Intro	Background and purpose of the interview	2 min	–
Part A	Process overview	4 min	RQ1
Part B	Critical phases and experience-based decisions	12 min	RQ1
Part C	What the video should capture and camera placement	6 min	RQ1, RQ2
Part D	Sources of variation	4 min	RQ1
Closing	Open reflections and thank-you	2 min	–

> *Strategic reminder: Let the operator describe the process freely first. Then compare against the manual and ask about deviations. You are not studying the machine. You are studying the expert's interaction with it.*

Intro – Introduction to the Operator (2 min)

Read this out loud:

> *"We are preparing to film resets on the SGP 320. We are particularly interested in what is not directly visible on video. Moreover, things that draw on your experience. What you tell us helps us place the cameras correctly and interpret what we see. The interview takes about 30 minutes and is recorded if you consent."*

☐ Confirm consent to recording.

☐ Remind the participant that there are no right or wrong answers. It is practical knowledge we are after.

☐ Ask about experience level: How long have you worked with the SGP 320?

Part A – Process Overview

RQ1 · 4 min

→ Briefly describe how a reset proceeds, from the moment you stop the machine until you start production again.

- What are the main phases for you?
- What do you do first, and what do you do last?

→ What types of resets occur, and what distinguishes them in practice?

- What is included in a mechanical reset compared with a variant reset? Which steps are added or omitted?
- In which situations do you (or someone else) decide which type of reset is to be performed?

Part B – Critical Phases and Experience-Based Decisions

RQ1 · 12 min

B1 · Disassembly

→ Are there steps in the disassembly where you make an active decision rather than follow a fixed order?

- What guides the order you choose?

→ What do you inspect particularly carefully during disassembly, and what are you looking for?

- Does what you see here affect what you do during subsequent assembly?
- What would it look like on video if something was not as it should be? Is it visible from the outside?

B2 · Assembly

→ How do you know the components are correctly mounted after assembly?

- Is there a specific step you always double-check?

→ Which assembly step do you experience as the most critical — and what happens in production if it is not done correctly?

- What would it look like on video if a component were mounted incorrectly — is that visible from the outside, or only noticed later?

B3 · Grinding the Drive Plate and Setting the Axial Position

→ How do you decide when the drive plate has been ground sufficiently?

- Do you base it on the Tesa probe, sound, feel, or a combination?
- How many grinding passes are typically needed?

→ How do you check and set the axial position?

- Do you trust the Tesa probe reading entirely, or is there also a feel that guides you?
- What does an outside observer see? Is it visible in your movements or behavior when you are satisfied with the result?

B4 · Adjusting the Measuring Finger

→ How do you know the measuring finger is correctly seated and that the calibration is accurate?

- Is it visible on the display, or do you feel/hear it?
- What do you do if it does not rest cleanly?

→ Do you always use a reference ring during calibration, or do you sometimes

do it without?

- What is an acceptable deviation after calibration, for you?

B5 · Learning the Dressing Position and Performing the Dressing Cycle

→ How do you know that the grinding wheel has made correct contact with the diamond roll?

- Do you base it on display values, sound, vibration, or feel?
- What happens if you feed too hard or miss the contact?

→ How do you decide whether the dressing cycle needs to be repeated?

- What do you look or listen for, and how many repetitions are normal?

Part C – What the Video Should Capture (Camera Placement)

RQ1, RQ2 · 6 min

→ If we could only film three things during an entire reset, what would have to be included?

- In which parts of the reset can video recording, together with data collection, make the biggest difference?

→ Are there moments where you look at something other than the screen while you interact with the machine?

- Listening, feeling with your hand, looking inside the machine?
- Which decisions do you make based on sound or feel rather than on the display?

→ Are there steps inside the machine that are crucial to see in order to understand what is happening, and that a camera could realistically capture?

- Are there steps that could easily be misinterpreted if seen without explanation?

→ Are there moments that are important but that do not show up on video at all?

- E.g., decisions based on feel, experience, sound, or hand pressure.

→ What do the machine's own logs, display values, or MARPOSS measurements *not* capture? What would only a camera be able to see?

- Are there things you do that show neither in the data nor on a screen, but that a camera in the right position could capture?
- And conversely, are there things the machine logs that you still need to see with your own eyes in order to trust?

Part D – Sources of Variation

RQ1 · 4 min

→ Which phase of the reset takes the most time, in your view? Is there a point where time is typically wasted and where it could have been done more efficiently?

→ Which phase varies most across reset occasions?

- Is it in the mechanical assembly, the calibration, the grinding parameters, or the test grinding?
- Are there visible signs early in the process that indicate a reset will be a difficult one?

→ Which parameters do you adjust "by feel" rather than directly from tabulated values?

- E.g, feed rate, dressing depth, spark-out time, compensation?
- Are there parameters you always fine-tune after a recipe transfer from another machine?

Closing – Open Reflections (2 min)

→ Is there something you do automatically during a reset that a new operator would not understand and that we have not asked about?

→ Are there moments where you deviate from what the instructions describe, and why?

> *Thank the participant. Remind them that you may reach out with follow-up questions. Confirm that the recording is stopped.*

B

Python Scripts

This appendix provides the Python scripts developed and used in this thesis. Section B.1 contains the audio transcription script. Section B.2 contains the RealSense .bag extraction script. Section B.3 describes the operator-tracking and depth-extraction pipeline. Section B.4 contains the video embedding generation script using R3D-18. Section B.5 contains the VLM prompt text and the frame-sampling and inference script.

All scripts were executed locally on the workstation described in Section 3.8.3 and were not deployed to any external service. Code listings have been edited for readability: file paths, credentials, and internal SKF identifiers have been replaced with placeholders.

B.1 Whisper Transcription Script

The following script was used to transcribe the interview audio files locally using OpenAI's Whisper model. The medium model variant was used with the language parameter set to Swedish.

```
""" Transcribes an interview audio file to UTF-8 text using Whisper
    , run locally to keep audio within the SKF environment"""

import whisper

# Medium variant: balances accuracy and CPU runtime.
model = whisper.load_model("medium")

# Set language explicitly
result = model.transcribe("<interview_audio>.m4a", language="sv")

# Writing the text
print(result["text"])

# UTF-8 preserves å, ä, ö.
with open("<transcript>.txt", "w", encoding="utf-8") as f:
    f.write(result["text"])

print("\nDone! Transcription saved to <transcript>.txt")
```

Listing B.1: Local audio transcription using Whisper.

B.2 RealSense .bag Segment Extraction

The script below converts selected temporal intervals from a RealSense .bag recording into synchronized RGB and depth frames together with frame-level metadata, as described in Section 3.6.1.

```python
import pyrealsense2 as rs
import numpy as np
import cv2
import os
import csv

# ----------------------------
# USER SETTINGS
# ----------------------------
Datafile =  #Choose your .bag file
BAG_FILE = os.path.join("data", Datafile)

START_SEC = #In seconds
END_SEC = #In seconds: "None" means entire video

SEGMENT_ID = f"seg_{START_SEC}_{END_SEC}"
if END_SEC is None:
    OUTPUT_FOLDER = f"{Datafile}_end"
else:
    OUTPUT_FOLDER = f"{Datafile}_{START_SEC}_{END_SEC}"
# ----------------------------
# SETUP
# ----------------------------
os.makedirs(os.path.join(OUTPUT_FOLDER, "color"), exist_ok=True)
os.makedirs(os.path.join(OUTPUT_FOLDER, "depth"), exist_ok=True)

pipeline = rs.pipeline()
config = rs.config()

rs.config.enable_device_from_file(config, BAG_FILE, repeat_playback=False)

align = rs.align(rs.stream.color)

print("Starting pipeline...")
profile = pipeline.start(config)

playback = profile.get_device().as_playback()
playback.set_real_time(False)

device = profile.get_device()
depth_sensor = device.first_depth_sensor()
depth_scale = depth_sensor.get_depth_scale()

print(f"Depth scale: {depth_scale}")

with open(os.path.join(OUTPUT_FOLDER, "segment_info.txt"), "w", encoding="utf-8") as f:
    f.write(f"source_bag={BAG_FILE}\n")
```

```
    f.write(f"start_sec={START_SEC}\n")
    f.write(f"end_sec={END_SEC}\n")
    f.write(f"depth_scale={depth_scale}\n")

frame_count = 0
global_frame_count = 0
recording_start_timestamp = None

frames_csv_path = os.path.join(OUTPUT_FOLDER, f"frames_{Datafile}.csv")

with open(frames_csv_path, "w", newline="", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=";")

    csv_writer.writerow([
        "segment_id",
        "frame_id",
        "global_frame_id",
        "timestamp_ms",
        "elapsed_ms",
        "elapsed_sec",
        "color_path",
        "depth_path"
    ])

    try:
        while True:
            try:
                frames = pipeline.wait_for_frames()
                aligned_frames = align.process(frames)
            except RuntimeError:
                print("Reached end of file.")
                break

            depth_frame = aligned_frames.get_depth_frame()
            color_frame = aligned_frames.get_color_frame()

            if not depth_frame or not color_frame:
                continue

            timestamp_ms = frames.get_timestamp()

            if recording_start_timestamp is None:
                recording_start_timestamp = timestamp_ms

            elapsed_ms = timestamp_ms - recording_start_timestamp
            elapsed_sec = elapsed_ms / 1000.0

            # Skip frames before chosen start time
            if elapsed_sec < START_SEC:
                global_frame_count += 1
                continue

            # Stop after chosen end time
            if END_SEC is not None and elapsed_sec > END_SEC:
```

```
                break

            depth_image = np.asanyarray(depth_frame.get_data())
            color_image = np.asanyarray(color_frame.get_data())

            color_rel_path = f"color/frame_{frame_count:05d}.png"
            depth_rel_path = f"depth/frame_{frame_count:05d}.png"

            color_abs_path = os.path.join(OUTPUT_FOLDER,
    color_rel_path)
            depth_abs_path = os.path.join(OUTPUT_FOLDER,
    depth_rel_path)

            cv2.imwrite(color_abs_path, color_image)
            cv2.imwrite(depth_abs_path, depth_image)

            csv_writer.writerow([
                SEGMENT_ID,
                frame_count,
                global_frame_count,
                round(timestamp_ms, 3),
                round(elapsed_ms, 3),
                round(elapsed_sec, 3),
                color_rel_path,
                depth_rel_path
            ])

            frame_count += 1
            global_frame_count += 1

            if frame_count % 30 == 0:
                print(f"Saved {frame_count} frames in selected
    interval")

    finally:
        pipeline.stop()

print(f"\nFinished. Saved {frame_count} frames from {START_SEC}s to
    {END_SEC}s.")
print(f"Output folder: {OUTPUT_FOLDER}")
print(f"Frame metadata saved to: {frames_csv_path}")
```

Listing B.2: Frame creation from selected .bag intervals.

B.3 Operator Tracking and Depth Extraction

The script below performs person detection with YOLOv8n, associates detections across consecutive frames using spatial continuity, and extracts a depth estimate from a 7×7 pixel patch at the centre of the operator's bounding box. The output is a time series of position and depth values, as described in Section 3.7.

```
import os
import csv
import cv2
```

```
import numpy as np
from ultralytics import YOLO

# ----------------------------------------------------
# USER SETTINGS
# ----------------------------------------------------
SEGMENT_FOLDER = "" # Change this to your segment folder where the frames are located
COLOR_FOLDER = os.path.join(SEGMENT_FOLDER, "color")
DEPTH_FOLDER = os.path.join(SEGMENT_FOLDER, "depth")

OUTPUT_CSV = os.path.join(SEGMENT_FOLDER, "tracking_yolo.csv")
OUTPUT_VIDEO = os.path.join(SEGMENT_FOLDER, "tracking_yolo_preview.mp4")
FIRST_FRAME_PREVIEW = os.path.join(SEGMENT_FOLDER, "yolo_first_frame_choices.jpg")

MODEL_NAME = "yolov8n.pt"
CONF_THRESHOLD = 0.50
MIN_BBOX_AREA = 5000  # Filter out very tiny detections
PATCH_RADIUS = 3      # 7x7 patch for depth median
FPS = 30
REACQUIRE_RESET_FRAMES = 1   # 0.5 sec at 30 FPS

# Matching weights
W_POSITION = 1.0
W_AREA = 0.35
W_DEPTH = 0.35


# ----------------------------------------------------
# HELPERS
# ----------------------------------------------------

def find_first_frame_with_person(model, color_files, depth_files):
    frame_count = min(len(color_files), len(depth_files))

    for i in range(frame_count):
        color_path = os.path.join(COLOR_FOLDER, color_files[i])
        depth_path = os.path.join(DEPTH_FOLDER, depth_files[i])

        color_frame = cv2.imread(color_path)
        depth_frame = cv2.imread(depth_path, cv2.IMREAD_UNCHANGED)
        depth_frame = ensure_single_channel_depth(depth_frame)

        if color_frame is None or depth_frame is None:
            continue

        detections = run_yolo_person_detection(
            model,
            color_frame,
            conf_threshold=CONF_THRESHOLD,
            min_bbox_area=MIN_BBOX_AREA
        )

        if detections:
```

```
            return i, color_frame, depth_frame, detections

    return None, None, None, []

def get_frame_files(folder):
    files = [f for f in os.listdir(folder) if f.lower().endswith(".png")]
    files.sort()
    return files


def ensure_single_channel_depth(depth_frame):
    if depth_frame is None:
        return None

    if len(depth_frame.shape) == 2:
        return depth_frame

    if len(depth_frame.shape) == 3:
        return depth_frame[:, :, 0]

    return depth_frame


def get_patch_median_depth(depth_frame, center_x, center_y, patch_radius=3):
    depth_frame = ensure_single_channel_depth(depth_frame)

    h, w = depth_frame.shape[:2]

    x1 = max(0, center_x - patch_radius)
    x2 = min(w, center_x + patch_radius + 1)
    y1 = max(0, center_y - patch_radius)
    y2 = min(h, center_y + patch_radius + 1)

    patch = depth_frame[y1:y2, x1:x2]
    valid_values = patch[patch > 0]

    if valid_values.size == 0:
        return 0, 0

    median_depth = int(np.median(valid_values))
    valid_count = int(valid_values.size)
    return median_depth, valid_count


def get_patch_median_depth_at_point(depth_frame, cx, cy, patch_radius=3):
    depth_frame = ensure_single_channel_depth(depth_frame)

    if depth_frame is None:
        return 0, 0

    h, w = depth_frame.shape[:2]

    cx = max(0, min(cx, w - 1))
```

```
    cy = max(0, min(cy, h - 1))

    x1 = max(0, cx - patch_radius)
    x2 = min(w, cx + patch_radius + 1)
    y1 = max(0, cy - patch_radius)
    y2 = min(h, cy + patch_radius + 1)

    patch = depth_frame[y1:y2, x1:x2]
    valid_values = patch[patch > 0]

    if valid_values.size == 0:
        return 0, 0

    median_depth = int(np.median(valid_values))
    valid_count = int(valid_values.size)
    return median_depth, valid_count


def run_yolo_person_detection(model, frame, conf_threshold=0.5,
    min_bbox_area=5000):
    results = model(frame, verbose=False)
    detections = []

    for result in results:
        for box in result.boxes:
            cls_id = int(box.cls[0].item())
            conf = float(box.conf[0].item())

            if cls_id != 0:
                continue
            if conf < conf_threshold:
                continue

            x1, y1, x2, y2 = map(int, box.xyxy[0].tolist())
            w = x2 - x1
            h = y2 - y1
            area = w * h

            if area < min_bbox_area:
                continue

            cx = x1 + w // 2
            cy = y1 + h // 2

            detections.append({
                "x": x1,
                "y": y1,
                "w": w,
                "h": h,
                "x2": x2,
                "y2": y2,
                "cx": cx,
                "cy": cy,
                "area": area,
                "conf": conf
            })
```

```
165
166     return detections
167
168
169 def annotate_first_frame_choices(frame, detections, output_path):
170     preview = frame.copy()
171
172     for idx, det in enumerate(detections, start=1):
173         x, y, w, h = det["x"], det["y"], det["w"], det["h"]
174         conf = det["conf"]
175
176         cv2.rectangle(preview, (x, y), (x + w, y + h), (0, 255, 0),
    2)
177         cv2.putText(
178             preview,
179             f"ID {idx} conf={conf:.2f}",
180             (x, max(30, y - 10)),
181             cv2.FONT_HERSHEY_SIMPLEX,
182             0.8,
183             (0, 255, 0),
184             2
185         )
186
187     cv2.imwrite(output_path, preview)
188
189
190 def compute_match_cost(candidate, prev_target):
191     cand_x = candidate.get("torso_cx", candidate["cx"])
192     cand_y = candidate.get("torso_cy", candidate["cy"])
193     prev_x = prev_target.get("torso_cx", prev_target["cx"])
194     prev_y = prev_target.get("torso_cy", prev_target["cy"])
195
196     pos_dist = np.hypot(cand_x - prev_x, cand_y - prev_y)
197     pos_cost = pos_dist / 100.0
198
199     prev_area = max(prev_target["area"], 1)
200     area_ratio = abs(candidate["area"] - prev_area) / prev_area
201
202     prev_depth = prev_target.get("depth_patch_median", 0)
203     cand_depth = candidate.get("depth_patch_median", 0)
204
205     if prev_depth > 0 and cand_depth > 0:
206         depth_cost = abs(cand_depth - prev_depth) / max(prev_depth,
    1)
207     else:
208         depth_cost = 1.0
209
210     total_cost = (
211         W_POSITION * pos_cost +
212         W_AREA * area_ratio +
213         W_DEPTH * depth_cost
214     )
215
216     return total_cost
217
218
```

```
219 def get_torso_region(det):
220     x, y, w, h = det["x"], det["y"], det["w"], det["h"]
221 
222     # Keep the currently accepted torso region for now
223     torso_x1 = int(x + 0.25 * w)
224     torso_x2 = int(x + 0.75 * w)
225     torso_y1 = int(y + 0.15 * h)
226     torso_y2 = int(y + 0.45 * h)
227 
228     torso_cx = (torso_x1 + torso_x2) // 2
229     torso_cy = (torso_y1 + torso_y2) // 2
230 
231     return torso_x1, torso_y1, torso_x2, torso_y2, torso_cx,
    torso_cy
232 
233 
234 def get_torso_median_depth(depth_frame, det):
235     depth_frame = ensure_single_channel_depth(depth_frame)
236 
237     torso_x1, torso_y1, torso_x2, torso_y2, torso_cx, torso_cy =
    get_torso_region(det)
238 
239     h, w = depth_frame.shape[:2]
240 
241     torso_x1 = max(0, min(torso_x1, w - 1))
242     torso_x2 = max(0, min(torso_x2, w))
243     torso_y1 = max(0, min(torso_y1, h - 1))
244     torso_y2 = max(0, min(torso_y2, h))
245 
246     patch = depth_frame[torso_y1:torso_y2, torso_x1:torso_x2]
247     valid_values = patch[patch > 0]
248 
249     if valid_values.size == 0:
250         return 0, 0, torso_cx, torso_cy, torso_x1, torso_y1,
    torso_x2, torso_y2
251 
252     median_depth = int(np.median(valid_values))
253     valid_count = int(valid_values.size)
254 
255     return median_depth, valid_count, torso_cx, torso_cy, torso_x1,
    torso_y1, torso_x2, torso_y2
256 
257 
258 def clamp(value, min_value, max_value):
259     return max(min_value, min(value, max_value))
260 
261 
262 def stabilize_anchor(measured_cx, measured_cy, prev_cx, prev_cy,
263                      max_dx=20, max_dy=15, alpha=0.25):
264     clamped_cx = clamp(measured_cx, prev_cx - max_dx, prev_cx +
    max_dx)
265     clamped_cy = clamp(measured_cy, prev_cy - max_dy, prev_cy +
    max_dy)
266 
267     smooth_cx = int(round(alpha * clamped_cx + (1 - alpha) *
    prev_cx))
```

```
268     smooth_cy = int(round(alpha * clamped_cy + (1 - alpha) *
    prev_cy))
269 
270     return smooth_cx, smooth_cy
271 
272 
273 # ------------------------------------------------
274 # MAIN
275 # ------------------------------------------------
276 def main():
277     color_files = get_frame_files(COLOR_FOLDER)
278     depth_files = get_frame_files(DEPTH_FOLDER)
279 
280     if not color_files or not depth_files:
281         print("No color or depth frames found.")
282         return
283 
284     frame_count = min(len(color_files), len(depth_files))
285     print(f"Using {frame_count} frame pairs.")
286 
287     print("Loading YOLO model...")
288     model = YOLO(MODEL_NAME)
289 
290     first_detected_frame_idx, first_color, first_depth,
    first_detections = find_first_frame_with_person(
291     model,
292     color_files,
293     depth_files)
294 
295     if first_detected_frame_idx is None:
296         print("No valid person detections found in any frame.")
297         return
298 
299     print(f"First valid person detection found in frame {
    first_detected_frame_idx}.")
300 
301     for det in first_detections:
302         depth_torso_median, valid_depth_count, torso_cx, torso_cy,
    torso_x1, torso_y1, torso_x2, torso_y2 = get_torso_median_depth(
303             first_depth, det
304         )
305 
306         torso_cx = max(0, min(torso_cx, first_depth.shape[1] - 1))
307         torso_cy = max(0, min(torso_cy, first_depth.shape[0] - 1))
308 
309         det["torso_cx"] = torso_cx
310         det["torso_cy"] = torso_cy
311         det["torso_x1"] = torso_x1
312         det["torso_y1"] = torso_y1
313         det["torso_x2"] = torso_x2
314         det["torso_y2"] = torso_y2
315         det["depth_center_raw"] = int(first_depth[torso_cy,
    torso_cx])
316         det["depth_patch_median"] = depth_torso_median
317         det["valid_depth_count"] = valid_depth_count
318 
```

```
319     annotate_first_frame_choices(first_color, first_detections,
    FIRST_FRAME_PREVIEW)
320 
321     print("\nDetected persons in first frame:")
322     for idx, det in enumerate(first_detections, start=1):
323         print(
324             f"ID {idx}: conf={det['conf']:.2f}, "
325             f"center=({det['cx']},{det['cy']}), "
326             f"area={det['area']}, "
327             f"depth={det['depth_patch_median']}"
328         )
329 
330     print(f"\nPreview saved to: {FIRST_FRAME_PREVIEW}")
331     selected_id = int(input("Enter target person ID from first
    frame: ").strip())
332 
333     if selected_id < 1 or selected_id > len(first_detections):
334         print("Invalid selection.")
335         return
336 
337     prev_target = first_detections[selected_id - 1].copy()
338     missing_streak = 0
339 
340     h, w = first_color.shape[:2]
341     fourcc = cv2.VideoWriter_fourcc(*"mp4v")
342     video_writer = cv2.VideoWriter(OUTPUT_VIDEO, fourcc, FPS, (w, h
    ))
343 
344     with open(OUTPUT_CSV, "w", newline="", encoding="utf-8") as f:
345         writer = csv.writer(f, delimiter=";")
346         writer.writerow([
347             "frame_id",
348             "target_found",
349             "det_conf",
350             "bbox_x",
351             "bbox_y",
352             "bbox_w",
353             "bbox_h",
354             "center_x",
355             "center_y",
356             "depth_center_raw",
357             "depth_patch_median",
358             "valid_depth_count"
359         ])
360 
361         for i in range(frame_count):
362             color_path = os.path.join(COLOR_FOLDER, color_files[i])
363             depth_path = os.path.join(DEPTH_FOLDER, depth_files[i])
364 
365             color_frame = cv2.imread(color_path)
366             depth_frame = cv2.imread(depth_path, cv2.
    IMREAD_UNCHANGED)
367             depth_frame = ensure_single_channel_depth(depth_frame)
368 
369             if color_frame is None or depth_frame is None:
370                 print(f"Skipping frame {i}: could not load frame
```

```
    pair.")
371                 continue
372
373             detections = run_yolo_person_detection(
374                 model,
375                 color_frame,
376                 conf_threshold=CONF_THRESHOLD,
377                 min_bbox_area=MIN_BBOX_AREA
378             )
379
380             for det in detections:
381                 depth_torso_median, valid_depth_count, torso_cx,
    torso_cy, torso_x1, torso_y1, torso_x2, torso_y2 =
    get_torso_median_depth(
382                     depth_frame, det
383                 )
384
385                 torso_cx = max(0, min(torso_cx, depth_frame.shape
    [1] - 1))
386                 torso_cy = max(0, min(torso_cy, depth_frame.shape
    [0] - 1))
387
388                 det["torso_cx"] = torso_cx
389                 det["torso_cy"] = torso_cy
390                 det["torso_x1"] = torso_x1
391                 det["torso_y1"] = torso_y1
392                 det["torso_x2"] = torso_x2
393                 det["torso_y2"] = torso_y2
394                 det["depth_center_raw"] = int(depth_frame[torso_cy,
     torso_cx])
395                 det["depth_patch_median"] = depth_torso_median
396                 det["valid_depth_count"] = valid_depth_count
397
398             target_found = 0
399             chosen = None
400
401             if detections:
402                 best_cost = float("inf")
403
404                 for det in detections:
405                     cost = compute_match_cost(det, prev_target)
406                     if cost < best_cost:
407                         best_cost = cost
408                         chosen = det
409
410                 if chosen is not None and best_cost < 8.0:
411                     target_found = 1
412                 else:
413                     chosen = None
414
415             if chosen is not None:
416                 reacquired_after_gap = missing_streak >=
    REACQUIRE_RESET_FRAMES
417
418                 if reacquired_after_gap:
419                     stable_cx = chosen["torso_cx"]
```

```
420                     stable_cy = chosen["torso_cy"]
421                 else:
422                     if "torso_cx" in prev_target and "torso_cy" in
    prev_target:
423                         stable_cx, stable_cy = stabilize_anchor(
424                             chosen["torso_cx"],
425                             chosen["torso_cy"],
426                             prev_target["torso_cx"],
427                             prev_target["torso_cy"],
428                             max_dx=20,
429                             max_dy=15,
430                             alpha=0.25
431                         )
432                     else:
433                         stable_cx = chosen["torso_cx"]
434                         stable_cy = chosen["torso_cy"]
435
436                 chosen["torso_cx_raw"] = chosen["torso_cx"]
437                 chosen["torso_cy_raw"] = chosen["torso_cy"]
438
439                 safe_cx = max(0, min(stable_cx, depth_frame.shape
    [1] - 1))
440                 safe_cy = max(0, min(stable_cy, depth_frame.shape
    [0] - 1))
441
442                 chosen["torso_cx"] = safe_cx
443                 chosen["torso_cy"] = safe_cy
444
445                 depth_patch_median, valid_depth_count =
    get_patch_median_depth_at_point(
446                     depth_frame,
447                     chosen["torso_cx"],
448                     chosen["torso_cy"],
449                     PATCH_RADIUS
450                 )
451
452                 chosen["depth_patch_median"] = depth_patch_median
453                 chosen["valid_depth_count"] = valid_depth_count
454                 chosen["depth_center_raw"] = int(depth_frame[chosen
    ["torso_cy"], chosen["torso_cx"]])
455
456                 x, y, bw, bh = chosen["x"], chosen["y"], chosen["w"
    ], chosen["h"]
457                 torso_cx, torso_cy = chosen["torso_cx"], chosen["
    torso_cy"]
458                 torso_x1, torso_y1 = chosen["torso_x1"], chosen["
    torso_y1"]
459                 torso_x2, torso_y2 = chosen["torso_x2"], chosen["
    torso_y2"]
460
461                 conf = chosen["conf"]
462                 depth_center_raw = chosen["depth_center_raw"]
463                 depth_patch_median = chosen["depth_patch_median"]
464                 valid_depth_count = chosen["valid_depth_count"]
465
466                 prev_target = chosen.copy()
```

```
467                 missing_streak = 0
468
469                 cv2.rectangle(color_frame, (x, y), (x + bw, y + bh)
    , (0, 255, 0), 3)
470                 cv2.rectangle(color_frame, (torso_x1, torso_y1), (
    torso_x2, torso_y2), (255, 0, 0), 2)
471                 cv2.circle(color_frame, (torso_cx, torso_cy), 5,
    (0, 0, 255), -1)
472
473                 if reacquired_after_gap:
474                     cv2.putText(
475                         color_frame,
476                         "REACQUIRED",
477                         (30, 80),
478                         cv2.FONT_HERSHEY_SIMPLEX,
479                         0.8,
480                         (0, 255, 255),
481                         2
482                     )
483
484                 cv2.putText(
485                     color_frame,
486                     f"TARGET conf={conf:.2f} depth={
    depth_patch_median}",
487                     (x, max(30, y - 10)),
488                     cv2.FONT_HERSHEY_SIMPLEX,
489                     0.75,
490                     (0, 255, 0),
491                     2
492                 )
493
494                 writer.writerow([
495                     i,
496                     1,
497                     round(conf, 3),
498                     x,
499                     y,
500                     bw,
501                     bh,
502                     torso_cx,
503                     torso_cy,
504                     depth_center_raw,
505                     depth_patch_median,
506                     valid_depth_count
507                 ])
508             else:
509                 missing_streak += 1
510
511                 cv2.putText(
512                     color_frame,
513                     "TARGET NOT FOUND",
514                     (30, 40),
515                     cv2.FONT_HERSHEY_SIMPLEX,
516                     1.0,
517                     (0, 0, 255),
518                     2
```

```
                )

                writer.writerow([
                    i,
                    0,
                    "",
                    "",
                    "",
                    "",
                    "",
                    "",
                    "",
                    "",
                    ""
                ])

            video_writer.write(color_frame)

            if i % 30 == 0:
                print(f"Processed frame {i}/{frame_count}")

    video_writer.release()
    print("\nDone.")
    print(f"Tracking CSV saved to: {OUTPUT_CSV}")
    print(f"Preview video saved to: {OUTPUT_VIDEO}")


if __name__ == "__main__":
    main()
```

Listing B.3: Person detection with YOLOv8n.

B.4 Video Embedding Generation (R3D-18)

The following script generates 512-dimensional embeddings for overlapping 2-second windows of the extracted RGB frames using a pretrained R3D-18 model from Torchvision, as described in Section 3.8.2. The model's final classification layer is removed, and the penultimate layer's output is used as a fixed-length feature representation.

```
import os
import csv
import cv2
import numpy as np
import pyrealsense2 as rs
import torch
import torch.nn as nn
from torchvision.models.video import r3d_18, R3D_18_Weights

# ------------------------------
# USER SETTINGS
# ------------------------------
Datafile =  # Choose your .bag file
BAG_FILE = os.path.join("data", Datafile)
```

```
15
16 START_SEC = #In seconds
17 END_SEC = #In seconds
18
19 WINDOW_SEC = 2.0
20 STRIDE_SEC = 1.0
21 NUM_SAMPLED_FRAMES = 16
22
23 OUTPUT_FOLDER = "embedding_output"
24 RUN_NAME = "File_Name"
25
26 RUN_FOLDER = os.path.join(OUTPUT_FOLDER, RUN_NAME)
27 os.makedirs(RUN_FOLDER, exist_ok=True)
28
29 EMBEDDINGS_PATH = os.path.join(RUN_FOLDER, "window_embeddings.npy")
30 WINDOW_METADATA_PATH = os.path.join(RUN_FOLDER, "window_metadata.
       csv")
31
32 # ------------------------------
33 # SETUP
34 # ------------------------------
35 pipeline = rs.pipeline()
36 config = rs.config()
37
38 rs.config.enable_device_from_file(config, BAG_FILE, repeat_playback
       =False)
39
40 align = rs.align(rs.stream.color)
41
42 print("Starting pipeline...")
43 profile = pipeline.start(config)
44
45 playback = profile.get_device().as_playback()
46 playback.set_real_time(False)
47
48 device = profile.get_device()
49 depth_sensor = device.first_depth_sensor()
50 depth_scale = depth_sensor.get_depth_scale()
51
52 print(f"Depth scale: {depth_scale}")
53
54 frame_count = 0
55 global_frame_count = 0
56 recording_start_timestamp = None
57
58 all_color_frames = []
59 all_elapsed_sec = []
60
61 try:
62     while True:
63         try:
64             frames = pipeline.wait_for_frames()
65             aligned_frames = align.process(frames)
66         except RuntimeError:
67             print("Reached end of file.")
68             break
```

```
69
70          depth_frame = aligned_frames.get_depth_frame()
71          color_frame = aligned_frames.get_color_frame()
72
73          if not depth_frame or not color_frame:
74              continue
75
76          timestamp_ms = frames.get_timestamp()
77
78          if recording_start_timestamp is None:
79              recording_start_timestamp = timestamp_ms
80
81          elapsed_ms = timestamp_ms - recording_start_timestamp
82          elapsed_sec = elapsed_ms / 1000.0
83
84          # Skip frames before chosen start time
85          if elapsed_sec < START_SEC:
86              global_frame_count += 1
87              continue
88
89          # Stop after chosen end time
90          if END_SEC is not None and elapsed_sec > END_SEC:
91              break
92
93          color_image = np.asanyarray(color_frame.get_data())
94
95          # Store in memory for embedding
96          all_color_frames.append(color_image.copy())
97          all_elapsed_sec.append(elapsed_sec)
98
99          frame_count += 1
100         global_frame_count += 1
101
102         if frame_count % 30 == 0:
103             print(f"Collected {frame_count} frames in selected
    interval")
104
105 finally:
106     pipeline.stop()
107
108 print(f"\nFinished collecting {frame_count} frames from {START_SEC}
    s to {END_SEC}s.")
109
110 # ------------------------------
111 # MODEL SETUP
112 # ------------------------------
113 print("\nLoading R3D-18 model...")
114 weights = R3D_18_Weights.DEFAULT
115 model = r3d_18(weights=weights)
116 model.fc = nn.Identity()
117 model.eval()
118
119 preprocess = weights.transforms()
120 print("Model ready.")
121
122 if len(all_color_frames) == 0:
```

```
    raise ValueError("No color frames were collected, cannot create embeddings.")

collected_duration = all_elapsed_sec[-1] - all_elapsed_sec[0]
print(f"Collected duration: {collected_duration:.2f}s")

if collected_duration < WINDOW_SEC:
    raise ValueError(
        f"Collected interval is shorter than one window. "
        f"Duration={collected_duration:.2f}s, WINDOW_SEC={WINDOW_SEC:.2f}s"
    )

# ------------------------------
# HELPER FUNCTION
# ------------------------------
def build_embedding_from_frames(frames_bgr):
    if len(frames_bgr) < NUM_SAMPLED_FRAMES:
        raise ValueError(
            f"Need at least {NUM_SAMPLED_FRAMES} frames in a window, got {len(frames_bgr)}"
        )

    sample_indices = np.linspace(
        0,
        len(frames_bgr) - 1,
        NUM_SAMPLED_FRAMES,
        dtype=int
    )

    sampled_frames = [frames_bgr[i] for i in sample_indices]

    processed_frames = []
    for frame in sampled_frames:
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_tensor = torch.from_numpy(frame_rgb).permute(2, 0, 1).float() / 255.0
        processed_frames.append(frame_tensor)

    # [T, C, H, W]
    video_tensor = torch.stack(processed_frames, dim=0)

    # Apply torchvision preprocessing
    video_tensor = preprocess(video_tensor)

    if len(video_tensor.shape) != 4:
        raise ValueError(f"Unexpected tensor shape after preprocess: {video_tensor.shape}")

    if video_tensor.shape[0] == NUM_SAMPLED_FRAMES:
        # [T, C, H, W] -> [C, T, H, W]
        video_tensor = video_tensor.permute(1, 0, 2, 3)
    elif video_tensor.shape[1] == NUM_SAMPLED_FRAMES:
        # already [C, T, H, W]
        pass
    else:
```

```
174         raise ValueError(f"Could not identify time dimension in
    shape: {video_tensor.shape}")
175
176     # [B, C, T, H, W]
177     video_tensor = video_tensor.unsqueeze(0)
178
179     with torch.no_grad():
180         embedding = model(video_tensor)
181
182     embedding_np = embedding.squeeze(0).cpu().numpy()
183
184     # Normalize for cosine similarity later
185     norm = np.linalg.norm(embedding_np)
186     if norm > 0:
187         embedding_np = embedding_np / norm
188
189     return embedding_np, sample_indices
190
191 # ------------------------------
192 # CREATE WINDOWS
193 # ------------------------------
194 window_starts = np.arange(
195     all_elapsed_sec[0],
196     all_elapsed_sec[-1] - WINDOW_SEC + 1e-9,
197     STRIDE_SEC
198 )
199
200 print(f"Creating {len(window_starts)} windows...")
201
202 all_embeddings = []
203 window_rows = []
204 skipped_windows = 0
205
206 for window_id, window_start in enumerate(window_starts):
207     window_end = window_start + WINDOW_SEC
208
209     frame_indices = [
210         i for i, t in enumerate(all_elapsed_sec)
211         if window_start <= t < window_end
212     ]
213
214     if len(frame_indices) < NUM_SAMPLED_FRAMES:
215         skipped_windows += 1
216         continue
217
218     window_frames = [all_color_frames[i] for i in frame_indices]
219     window_times = [all_elapsed_sec[i] for i in frame_indices]
220
221     try:
222         embedding_np, sample_indices = build_embedding_from_frames(
    window_frames)
223     except Exception as e:
224         skipped_windows += 1
225         print(f"Skipping window {window_id} ({window_start:.2f}-{
    window_end:.2f}s): {e}")
226         continue
```

```
227
228     all_embeddings.append(embedding_np)
229
230     sampled_times = [window_times[i] for i in sample_indices]
231
232     window_rows.append([
233         window_id,
234         round(float(window_start), 3),
235         round(float(window_end), 3),
236         len(frame_indices),
237         ",".join(map(str, sample_indices.tolist())),
238         ",".join(f"{t:.3f}" for t in sampled_times)
239     ])
240
241     if len(all_embeddings) % 25 == 0:
242         print(f"Created {len(all_embeddings)} embeddings...")
243
244 if len(all_embeddings) == 0:
245     raise ValueError("No window embeddings were created.")
246
247 embedding_matrix = np.vstack(all_embeddings)
248 np.save(EMBEDDINGS_PATH, embedding_matrix)
249
250 with open(WINDOW_METADATA_PATH, "w", newline="", encoding="utf-8")
    as f:
251     writer = csv.writer(f, delimiter=";")
252     writer.writerow([
253         "window_id",
254         "window_start_sec",
255         "window_end_sec",
256         "num_frames_in_window",
257         "sampled_frame_indices_within_window",
258         "sampled_frame_times_sec"
259     ])
260     writer.writerows(window_rows)
261
262 print("\nEmbedding extraction finished.")
263 print(f"Saved embedding matrix to: {EMBEDDINGS_PATH}")
264 print(f"Saved window metadata to: {WINDOW_METADATA_PATH}")
265 print(f"Embedding matrix shape: {embedding_matrix.shape}")
266 print(f"Skipped windows: {skipped_windows}")
```

Listing B.4: Embedding generation for .bag files using R3D-18.

B.5 VLM Prompt and Frame-Sampling Script

This section contains both the full prompt text provided to the Qwen3-VL 8B model and the Python script used to sample the recording into 513 frames and send them to the model via Ollama.

```
1 import cv2
2 import ollama
3 import base64
4 import os
```

```
import time
from datetime import datetime

# ============================================================
# CONFIG â€" change MODEL_NAME between runs
# ============================================================
MODEL_NAME = "qwen3-vl:8b"   # then "qwen2.5vl:7b", then "qwen3-vl:8b"
VIDEO_PATH = "Full_demontering_oklippt.mp4"
START_SECOND = 0
END_SECOND = 513   # 08:33 covers the full 08:32 video
FRAMES_DIR = "frames"
# ============================================================

# Output filename includes model name so runs don't overwrite each other.
# Replaces ':' and '/' with '_' for filesystem safety.
safe_model_name = MODEL_NAME.replace(":", "_").replace("/", "_")
OUTPUT_FILE = f"descriptions_{safe_model_name}.txt"

PROMPT = """You are analyzing a single frame from a video of an industrial grinding machine being dismantled.

The OPERATOR is the person actively working on the machine in the foreground. There may be another person visible in the background near a green tool cart â€" ignore that person completely.

For each item below, answer only yes or no based on what is clearly visible in the frame. Do not guess. If you are unsure, answer no.

Operator in frame: [yes/no]

Tools:
- torque wrench: [yes/no]
- pneumatic screwdriver: [yes/no]
- hex key: [yes/no]
- fixed spanner: [yes/no]

Components:
- magnetic drivehead: [yes/no]
- guide rail: [yes/no]
- coolant nozzle: [yes/no]
- feeder/ejector: [yes/no]
- inlet chute: [yes/no]
- bearing rings: [yes/no]
- outlet chute: [yes/no]
- guard plate: [yes/no]
- measuring finger: [yes/no]
- grinding wheel: [yes/no]
- diamond roller: [yes/no]
- microcentric bracket: [yes/no]"""


def extract_frames_if_needed(video_path, frames_dir, start, end):
```

```
    """Extract frames once and reuse across model runs."""
    os.makedirs(frames_dir, exist_ok=True)
    expected = [os.path.join(frames_dir, f"frame_second_{s}.jpg")
                for s in range(start, end)]
    if all(os.path.exists(p) for p in expected):
        print(f"All {len(expected)} frames already extracted in {frames_dir}/, reusing.")
        return expected

    print(f"Extracting {end - start} frames to {frames_dir}/ ...")
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        raise RuntimeError(f"Could not open video: {video_path}")
    paths = []
    for second in range(start, end):
        frame_path = os.path.join(frames_dir, f"frame_second_{second}.jpg")
        if os.path.exists(frame_path):
            paths.append(frame_path)
            continue
        cap.set(cv2.CAP_PROP_POS_MSEC, second * 1000)
        success, frame = cap.read()
        if not success:
            print(f"  WARNING: could not read frame at second {second}")
            continue
        cv2.imwrite(frame_path, frame)
        paths.append(frame_path)
    cap.release()
    print(f"  Extracted {len(paths)} frames.")
    return paths


def analyze_frame(model_name, image_b64):
    """Single Ollama call with error handling. Returns (text, error_or_none)."""
    try:
        response = ollama.chat(
            model=model_name,
            keep_alive="30m",
            messages=[{
                "role": "user",
                "content": PROMPT,
                "images": [image_b64],
            }],
        )
        return response["message"]["content"].strip(), None
    except Exception as e:
        return None, str(e)


def main():
    if not os.path.exists(VIDEO_PATH):
        print(f"ERROR: video not found at {VIDEO_PATH}")
        return

```

```
    frame_paths = extract_frames_if_needed(VIDEO_PATH, FRAMES_DIR,
START_SECOND, END_SECOND)
    total = len(frame_paths)

    print(f"\n{'=' * 60}")
    print(f"Model:  {MODEL_NAME}")
    print(f"Output: {OUTPUT_FILE}")
    print(f"Frames: {total} (seconds {START_SECOND} to {END_SECOND
- 1})")
    print(f"{'=' * 60}\n")

    run_start = time.time()
    errors = 0

    with open(OUTPUT_FILE, "w", encoding="utf-8") as out:
        out.write(f"# Model: {MODEL_NAME}\n")
        out.write(f"# Video: {VIDEO_PATH}\n")
        out.write(f"# Started: {datetime.now().isoformat()}\n")
        out.write(f"# Frames: {total}\n\n")
        out.flush()

        for i, frame_path in enumerate(frame_paths):
            second = int(os.path.basename(frame_path)
                         .replace("frame_second_", "")
                         .replace(".jpg", ""))

            with open(frame_path, "rb") as f:
                image_b64 = base64.b64encode(f.read()).decode("utf
-8")

            t0 = time.time()
            description, err = analyze_frame(MODEL_NAME, image_b64)
            elapsed = time.time() - t0

            if err is not None:
                errors += 1
                line = f"SECOND {second}: [ERROR] {err}"
            else:
                line = f"SECOND {second}: {description}"

            print(f"[{i + 1}/{total}] sec {second:>3} ({elapsed:.1f
}s): "
                  f"{'ERROR' if err else 'ok'}")
            out.write(line + "\n\n")
            out.flush()

    total_elapsed = time.time() - run_start
    minutes = total_elapsed / 60
    print(f"\n{'=' * 60}")
    print(f"Done. {total} frames in {minutes:.1f} min "
          f"(avg {total_elapsed / total:.1f}s/frame, {errors}
errors)")
    print(f"Output: {OUTPUT_FILE}")
    print(f"{'=' * 60}")


```

```
156 if __name__ == "__main__":
157     main()
```

Listing B.5: Frame sampling and VLM inference via Ollama.

C

Supplementary Material

This appendix collects supplementary results and reference material that support the analyses in the main chapters but are not essential for following the argument. Section C.1 reports additional similarity distributions from the embedding analysis using PCA-reduced representations. Section C.2 reproduces the SGP 320 resetting report referenced in the methodology and discussion chapters.

C.1 Additional Similarity Distributions

This appendix provides additional similarity distributions for PCA-reduced representations using 18 and 27 components. These results support the findings presented in Section 4.3.2.

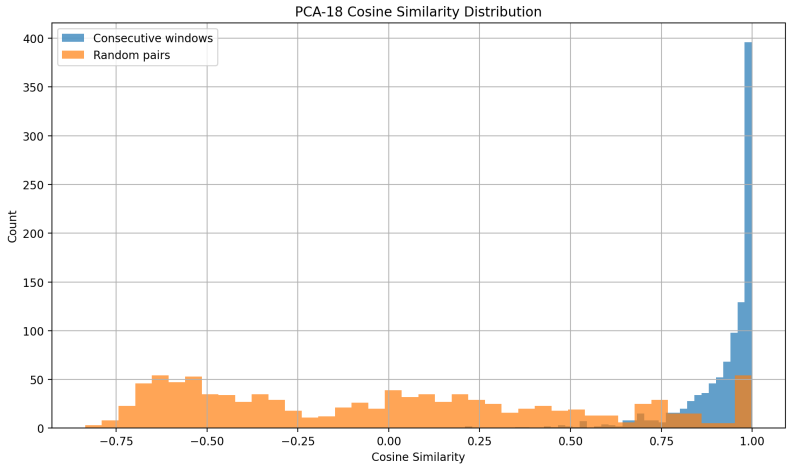


Figure C.1: Cosine similarity distribution using PCA-18 components.

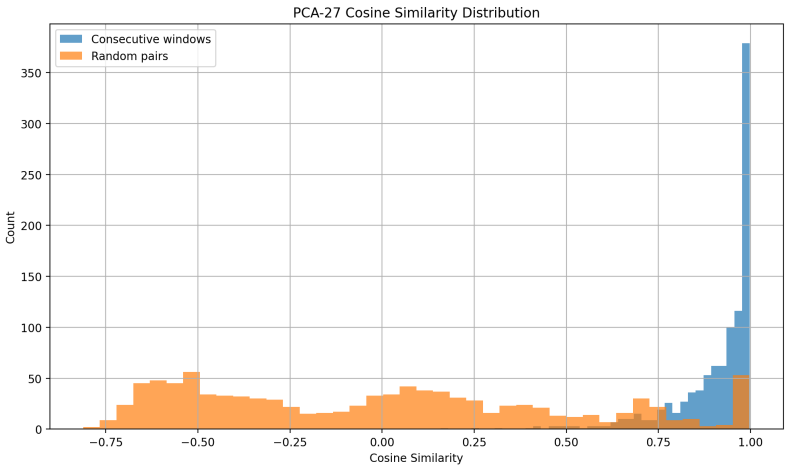


Figure C.2: Cosine similarity distribution using PCA-27 components.

C.2 SGP 320 Resetting Report

SGP 320 OMSTÄLLNINGSANALYS

MECE-strukturerad rapport för produktionsledning

Dataperiod: [redacted] till [redacted]
Genererad: [redacted]

NYCKELTAL	
Dataperiod	[redacted] till [redacted]
Vanliga omställningar	[redacted]
Variantställ	[redacted]
Total omställningstid	[redacted] timmar
Channels	DD[redacted], DD[redacted]
SGP 320 andel av omställningar	[redacted]%
SGP 320 assets i GOT	[redacted]

EXECUTIVE SUMMARY

Denna rapport analyserar omställningar för SGP 320-maskiner (channels DD[redacted], DD[redacted]) i D-Factory, GOT (Gothenburg) under perioden [redacted] till [redacted].

Omställningstyper:
- **Vanlig omställning** = Mekaniskt ställ + Justering efter ställ (sammanhängande aktivitet, ~[redacted] min)
- **Variantställ** = Fristående produktvariantbyte (~[redacted] min)

Totalt: [redacted] **vanliga omställningar** och [redacted] **variantställ**.

Huvudsakliga fynd:

- **Validerat antagande:** I [redacted]% av skift med Mekaniskt ställ förekommer även Justering - detta bekräftar att de tillhör samma arbetsmoment.
- **Alternerande produktionsmönster:** [redacted]% av alla övergångar är Vanlig↔Variant, vilket indikerar att variantbyten nästan alltid triggar en full omställning.
- **Planeringsimplikation:** Variantställ sker sällan i sekvens - optimera genom att gruppera liknande produkter.
- **Assets:** [redacted] SGP 320 assets identifierade i D-Factory (alla grinding-relaterade).

1. KONTEXT OCH PERSPEKTIV

SGP 320-maskinernas omställningar i relation till all data.

Kategori	Antal	Timmar	Andel
All data (totalt)	[redacted]	[redacted]	[redacted]%
- Övrig data	[redacted]	[redacted]	[redacted]%
- Alla omställningar	[redacted]	[redacted]	[redacted]%
- SGP 320 omställningar	[redacted]	[redacted]	[redacted]%

2. SGP 320 ASSET METADATA

Datakälla: SKF Assets [redacted].csv
Totalt antal assets: [redacted]
Assets i GOT (Gothenburg): [redacted]
SGP 320 assets i GOT: [redacted]

2.1Assets per Factory i GOT

Factory	Antal Assets
D-Factory	[redacted]
E-Factory	[redacted]
R-Factory	[redacted]
Övrig	[redacted]

2.2SGP 320 Assets - Detaljerad lista

Alla [redacted] SGP 320-relaterade assets är lokaliserade i **D-Factory**, area [redacted]**G**, channel **CH - DD**[redacted]. Detta bekräftar att produktionsdata i denna rapport motsvarar dessa fysiska assets.

Asset ID	Description	Asset Type	Main Machine?
Gothenburg-[redacted]	Slipning utvändig, SGP 320, ORC[redacted] Ma	Grinding	True
Gothenburg-[redacted]	Slipning utvändig, SGP 320, IRC[redacted] Ma	Grinding	True
Gothenburg-[redacted]	Slipning utvändig, SGP 320 AR IRC[redacted],	Grinding	True
Gothenburg-[redacted]	Slipning utvändig, SGP 320 AR ORC[redacted],	Grinding	True
Gothenburg-[redacted]	Slipning utvändig, SGP 320, IRC[redacted] Ma	Grinding	True

3. VALIDERING AV ANTAGANDEN

Analysen bygger på antagandet att "Mekaniskt ställ" och "Justering efter ställ" sker som en **sammanhängande aktivitet** inom samma skift - dvs de utgör tillsammans en "Vanlig omställning".

Channel	Endast Mek	Endast Just	Båda (=Vanlig)	Mek→Just %

[redacted]

DD[redacted]	[redacted]	[redacted]	[redacted]	[redacted]%
DD[redacted]	[redacted]	[redacted]	[redacted]	[redacted]%

Slutsats: [redacted]% samförekomst bekräftar antagandet - Mekaniskt och Justering hör ihop.

4. ANALYS PER CHANNEL

	DD[redacted]	DD[redacted]
Vanlig omställning (antal)	[redacted]	[redacted]
Variantställ (antal)	[redacted]	[redacted]
Vanlig - medeltid (min)	[redacted]	[redacted]
- varav Mekaniskt	[redacted]	[redacted]
- varav Justering	[redacted]	[redacted]
Variant - medeltid (min)	[redacted]	[redacted]

5. TRENDANALYS

6. HEATMAP PER KVARTAL

7. SEKVENSANALYS - PRODUKTIONSMÖNSTER

7.1Vad visar sekvensanalysen?

Sekvensanalysen undersöker **i vilken ordning** omställningar sker. När vi tittar på tusentals omställningar över tid kan vi identifiera mönster som avslöjar hur produktionen faktiskt fungerar - och var det finns potential för förbättring.

De fyra övergångstyperna:

- **Vanlig → Vanlig** (blå): Två fullständiga omställningar i rad. Indikerar produktbyte som kräver komplett omställning utan variant-mellansteg.
- **Vanlig → Variant** (orange): Efter en fullständig omställning görs ett enklare variantbyte. Typiskt: man ställer om till en produktfamilj och gör sedan variationer inom den.

- **Variant → Vanlig** (grön): Efter ett variantbyte krävs en fullständig omställning. Indikerar att man lämnar en produktfamilj.
- **Variant → Variant** (lila): Två variantbyten i rad. Sällsynt - visar att man stannar inom samma produktfamilj.

7.2Vad betyder det att orange och grön följer varandra?

Det faktum att **Vanlig→Variant** (orange) och **Variant→Vanlig** (grön) nästan alltid har samma värde avslöjar ett **alternerande produktionsmönster**:

Typisk sekvens: Vanlig → Variant → Vanlig → Variant → Vanlig...

I praktiken betyder detta:
1. Man gör en **fullständig omställning** (Vanlig) till en ny produktfamilj
2. Man gör ett eller flera **variantbyten** inom den familjen
3. När man byter produktfamilj krävs åter en **fullständig omställning Kvantifierat:** [redacted]% av alla övergångar är "alternerande" (Vanlig↔Variant).

7.1Varför är Variant→Variant (lila) så lågt?

Den låga frekvensen av **Variant→Variant** (två variantbyten i rad) indikerar att:

- Variantställ är **inte** en fristående optimeringsmöjlighet - de "hänger på" fullständiga omställningar
- Produktionsplaneringen resulterar i att man **sällan stannar länge** inom samma produktfamilj - Varje variantbyte följs nästan alltid av en fullständig omställning

Förbättringspotential: Om fler produkter inom samma familj kunde grupperas i produktionsplanen skulle andelen Variant→Variant kunna öka, vilket minskar behovet av tidskrävande fullständiga omställningar.

7.4 Sekvensstatistik per channel

Övergångstyp	DD[redacted]	DD[redacted]
Vanlig → Vanlig	[redacted] ([redacted]%)	[redacted] ([redacted]%)
Vanlig → Variant	[redacted] ([redacted]%)	[redacted] ([redacted]%)
Variant → Vanlig	[redacted] ([redacted]%)	[redacted] ([redacted]%)
Variant → Variant	[redacted] ([redacted]%)	[redacted] ([redacted]%)
Alternerande (V↔Var)	**[redacted]%**	**[redacted]%**

7.5 Blir Variantställ vanligare?

[redacted]

8. SLUTSATSER OCH IMPLIKATIONER

För produktionsledning:

- **Omställningsmönster bekräftat:** Mekaniskt + Justering hör samman ([redacted]% samförekomst). Rapportera och analysera dem som EN aktivitet.
- **Alternerande produktion:** [redacted]% av övergångarna är Vanlig↔Variant. Variantställ 'hänger på' fullständiga omställningar.
- **Gruppera produktfamiljer:** Den låga andelen Variant→Variant indikerar potential att minska omställningstid genom bättre sekvensering.
- **Assets validerade:** [redacted] fysiska assets i D-Factory matchar produktionsdata. Alla är Main Machines.
-

SAMMANFATTNI G	
Dataperiod	[redacted] till [redacted]
Vanliga omställningar	[redacted]
Variantställ	[redacted]
Alternerande mönster	[redacted]%
SGP 320 assets i GOT	[redacted]
Factory	D-Factory

9. SKF Översikt

1. Scope and Data Source

- Source file: SKF Assets [redacted].csv
- Total assets in dataset: [redacted]
- Assets identified as SGP 320: [redacted]

2. Installed Base Overview – SGP 320

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY