https://doi.org/10.2352/lim.2022.1.1.15

Augmented Reality for Automatically Generating Robust Manufacturing and Maintenance Logs

Tim J. Schoonbeek[*1], *Pierluigi Frisco*[2], *Hans Onvlee*[2], *Peter H.N. de With*[1], *Fons van der Sommen*[1]
[*] *Corresponding author: t.j.schoonbeek@tue.nl*
[1] *Video Coding Architectures lab, dept. EE, Eindhoven University of Technology, Eindhoven, The Netherlands*
[2] *ASML Research, Veldhoven, The Netherlands*

Abstract

Logs describing the execution of procedural steps during manufacturing and maintenance tasks are important for quality control and configuration management. Such logs are currently hand-written or typed during a procedure, which requires engineers to frequently step away from their work and results in difficulties for searching and optimizing logs. In this paper, we propose to automatically generate standardized, searchable logs, by visually perceiving and monitoring the progress of the procedure in real-time, and comparing this to the expected procedure. Unlike related work, we propose an approach which does not restrict the engineers to rigid, sequential sequences and instead allows them to execute procedures in a variety of different sequences where possible. The proposed framework is experimentally validated on the task of (dis)assembling a Duplo block model and operates properly when occlusions are absent.

Introduction

The COVID-19 pandemic prevented experts from traveling to local sites to provide assistance, causing an accelerated adoption of augmented reality (AR) and mixed reality (MR) devices in the semiconductor industry to support on-site engineers with remote expertise [1]. Previously stringent intellectual property (IP) protection policies were relaxed to allow MR devices, such as the HoloLens 2, into factories and even cleanrooms. Allowing such devices opens up a range of new possibilities to support on-site engineers by perceiving the engineer's workflow. We propose such a new support task, namely automatically generating logs for manufacturing and maintenance tasks, as outlined in Fig. 1. Such logs describe the actions of engineers during a particular task and are crucial for quality control and configuration management. The present state is that logs are created either by writing or typing what actions are performed. This is time-consuming and in free-form text data, requiring complex algorithms to convert technical language into standardized, searchable and optimizable logs [2, 3, 4]. An alternative option is to log actions with a start/stop button on electronic work instructions, but this requires engineers to frequently step away from their work to log an action. Simply storing a video as a log also has significant downsides. For example, storing videos from factories and cleanrooms poses serious IP risks and videos are not easily searchable without spending significant time [5, 6]. Therefore, in this work, we introduce the task of automated action logging and propose an AR framework that automatically generates objective, searchable text logs, based on real-time visual scene perception and procedural knowledge. This concept is introduced without recording any video or picture of the action.

Intelligent AR frameworks based on perceptual understanding are already able to actively assist operators in various ways, e.g. by training novice operators for assembly tasks [7, 8, 9, 10], or visualizing instructions that are hard to interpret from 2D representations [11, 12]. Whilst such approaches clearly provide

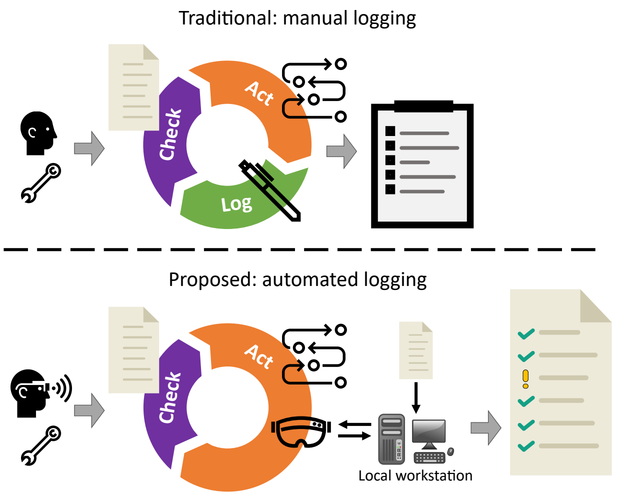


Figure 1. *Traditionally, an engineer checks the tasks to be performed, then executes those tasks and manually logs each step's completion during the procedure. We propose to automatically generate a text log of these procedural steps by perceiving the progress and comparing this to the expected progress, without enforcing a restrictive step-by-step sequence order.*

benefits, most are only intended to display instructions, without confirming whether those instructions were actually executed. Works that actually confirm whether actions are completed, verify this in a fixed and concatenated sequence, meaning that operators must perform the involved steps from first to last in a sequential order. Such rigid sequences can be beneficial for novice users, but inherently restrict the more experienced operators, since service tasks may be correctly performed in different orders. Additionally, the sequence provided in the instruction is not necessarily optimal. Therefore, we propose an intelligent framework which allows for flexible following of the procedure, while still allowing to compare observations to the expected actions based on procedural information. Analyses of the sequence in which the actions were actually executed enable for identifying ways for optimizing execution strategies.

The expected actions are pre-conditioned, which can either be extracted from existing work instructions or created specifically for a procedure. This allows the framework to perceive tasks in any order and provides a warning in the log when an engineer has potentially violated a condition. The framework can be deployed locally as well as during remote support sessions between on-site engineers and off-site experts. The objective of the framework is not to micro-manage or limit engineers, but rather to support and enable them. We demonstrate the capability of the framework on a block model assembly and disassembly task. Specifically, this work brings the following contributions.

- The task of recognizing industrial actions based on visual perception and prior knowledge on the procedure. Fur-

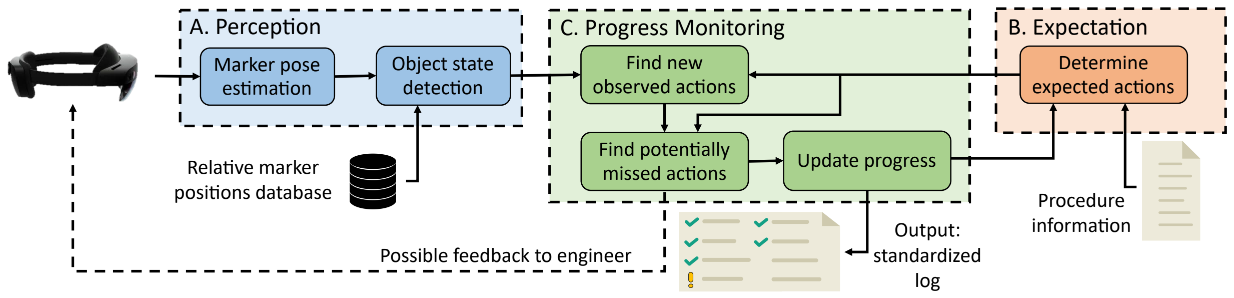


Figure 2. *Illustration of the proposed method to automatically log manufacturing and maintenance tasks. The approach consists of three main modules, namely perception, expectation, and progress monitoring. The latter automatically generates standardized, easily interpretable logs.*

thermore, we automatically generate searchable and interpretable logs based on this perception. To the best of our knowledge, this specification is novel.

- A flexible framework for automated action logging that includes flexible procedure handling and warnings in the logs for potentially missed actions.
- A validation of the approach based on a block assembly and disassembly task, demonstrating its applicability to a generic task.

Related work

The task of automatically logging human actions during manufacturing and maintenance tasks touches several research fields, namely human-object interaction recognition, object-state detection, and intelligent AR applications. This section outlines the similarities and differences of these fields compared to our newly proposed task.

The task of (egocentric) human-object interaction recognition focuses on determining human behaviour in relation to objects in their surroundings. Several datasets are available for this task, e.g. determining interactions in kitchens [13, 14] and home situations [15]. Recently, Ragusa *et al.* [16] and Sener *et al.* [17] introduced egocentric video datasets of human-object interactions in industrial-like settings. Recognizing the interactions between objects and humans can provide valuable information when trying to understand what a human is doing, but does not provide conformation of whether a task is actually completed. For instance, if a prediction is made that an operator is "placing block X", it remains unclear whether it was placed correctly and if the task was completed or stopped half-way through. However, this be determined with object-state detection.

The task of object-state detection provides the current state of an object given all of its possible states during assembly. Several possible approaches to object-state detection have been successfully implemented, using markers on each object part [9], applying region-of-interest crops based on a single marker [10], and deploying convolutional neural networks (CNNs) [18, 19, 20]. CNN-based methods operate without any markers and therefore do not require alterations to object (parts) and are robust to marker occlusions, which frequently occur when humans are interacting with such objects. However, marker-based detections do not require image datasets for training and provide a higher accuracy with significantly less computation power. Another approach to object-state detection is presented by Jones *et al.* [21], who apply a partially-observable Markov decision process to understand assembly states of Duplo blocks during assembly. While the aforementioned research all provides the state of an object during its assembly or disassembly, it does not explicitly leverage physical constraints and procedural information. As object states can look identical from several viewing angles during assembly, there is a need for a more intelligent framework, which keeps track of observed states and determines which future states are to be expected.

Gupta *et al.* [8] present such an intelligent framework, which guides a user into the correct assembly of Duplo block models with visual aids, using multiple RGB-D cameras on a fixed work-bench setup. Westerfield *et al.* [9] propose a framework using markers on each relevant component and automatically detects whether a user completed an assembly correctly. Yin *et al.* [10] propose a system which automatically determines whether an engineer's current step is completed by simultaneously recognizing hand motion and visual completeness of the object on which the engineer is working. All the previously mentioned approaches rely on a rigid, strictly sequential order, whereas we propose a new, worker-centric framework, which allows for a flexible and/or modified sequential order.

Method

We present an automatic logging system based on visual perception which allows for flexible execution of procedural steps. Fig. 2 outlines the proposed method and shows the proposed perception, progress monitoring, and expectation modules. The remainder of this section describes each module in detail.

A. Perception: object-state detection

For the perception module, we opt for object-state detection using ArUco markers [22], an approach similar to that proposed by Westerfield *et al.* [9]. The object-state detection exploits relative poses between markers attached to connecting parts. Consequently, this approach requires markers on all components and suffers when markers are (partially) occluded. However, it is computationally efficient, does not require the collection of a large image dataset or scanning of 3D objects, and does not impose a lengthy re-training of a CNN when minor changes are made to the manufacturing procedure. Our approach only requires a Database D_{rp} of relative poses between markers on correctly assembled parts.

This Database D_{rp} is created by taking a set of images $I_d = \{i_d^0, i_d^1, ..., i_d^n\}$ of the object with relevant parts properly connected, ensuring that each Marker m has at least one image where the marker is visible for each connecting part. Clearly, not all object parts have meaningful connections with all other parts, requiring a specification of meaningful connections beforehand.

Assuming a relative pose from any object a to b be $p_{a\to b}$, such that

$$p_{a\to b} = \{\mathbf{r}_{a\to b},\ \mathbf{t}_{a\to b}\} \tag{1}$$

where $\mathbf{r}_{a\to b} \in \mathbb{R}^{1x3}$ is the rotation vector and $\mathbf{t}_{a\to b} \in \mathbb{R}^{1x3}$ the translation vector from a to b. For the j^{th} image $i_d^j \in I_d$, showing correctly assembled object parts, the pose of each Marker A with respect to the camera C ($p_{A\to C}$) is calculated, using the ArUco API [22] and the camera's known intrinsic and distortion parameters. Then, the relative pose from A to each relevant other and correctly assembled Marker B in i_d^j can be determined by solving

$$p^*_{A\to B} = p_{A\to C} + p^{-1}_{B\to C} \tag{2}$$

where $p^*_{A\to B}$ is the desired relative pose from Marker A to B for correctly assembled parts, and $p^{-1}_{B\to C}$ the inverse perspective of the relative pose from Marker B to the camera. As p consists of a set of rotation and translation vectors, $p_{B\to C}$ and $p^{-1}_{A\to C}$ cannot simply be summed and, instead, two shift-and-rotation transformations are performed, which are specified by

$$\mathbf{r}^*_{A\to B} = \mathrm{R}^{-1}(\mathrm{R}(\mathbf{r}_{A\to C}) \cdot \mathrm{R}(\mathbf{r}^{-1}_{B\to C})), \tag{3}$$
$$\mathbf{t}^*_{A\to B} = \mathrm{R}(\mathbf{r}_{A\to C}) \cdot \mathbf{t}_{B\to C} + \mathbf{t}_{A\to C}, \tag{4}$$

where $\mathbf{r}^*_{A\to B}$ and $\mathbf{t}^*_{A\to B}$ are the desired rotation and translation vectors for correctly assembled parts, R denotes the Rodrigues rotation vector to a rotation-matrix transformation and R^{-1} its inverse [23, 24]. Finally, the relative poses between all markers are stored in matrices.

Having created D_{rp}, an unseen image i can be analyzed to determine the state of an object. First, the frame is converted to gray-scale and afterwards markers are detected. Then, the relative pose of each marker with respect to the camera is calculated, e.g. $p_{A\to C}$ for Marker A. Next, the relative pose of the detected marker to all relevant other markers are loaded from D_{rp}, e.g. $p^*_{A\to B}$. Using Eqs. (3) and (4), the desired pose $p^*_{B\to C}$ from all relevant markers to the camera is determined. Finally, the expected corner locations B^* of Marker B are projected using $p^*_{B\to C}$, and compared to the actual detected corners of Marker B. If the average distance per corner between B^* and B is less than τ_c pixels in the horizontal and vertical direction, the parts are deemed properly connected. If the average distance is larger than τ_d pixels, with $\tau_c > \tau_d$, they are deemed disconnected. Otherwise, the observation is considered to be ambiguous and not used. This process is repeated for all markers detected in i.

B. Expectation: extracting procedure information

As previously mentioned, this work aims to log procedural actions performed in a flexible sequence, rather than in a step-by-step pre-defined order. However, perceived object states should still be compared to an expectation of the procedure, based on current progress and procedural information. Therefore, pre-conditions are extracted from work instructions, determining which actions should be completed prior to performing each of those actions. These conditions can be based on physical limitations (one cannot "Place X on Y", if Y is yet not assembled) or procedural limitations (one should not "Place X on Y" yet, if this prevents an earlier action from being completed). For each action in a work instruction, we store its pre-conditions, whether it is connecting or disconnecting an object part, and the corresponding marker IDs of relevant (object) parts.

C. Progress monitoring

The progress-monitoring module tracks the progress of a procedure by comparing new observations from the perception module to the set of expected actions, provided by the expectation module. When an object-state change is observed that does not violate any pre-conditions, the progress is updated and a log entry is created. Such a log entry contains a description of the action, a confidence value indicating that the state change has been observed, together with its date and time. If an object-state change violates a pre-condition, the perception module either has missed a prior state change or the engineer has skipped a procedural step. In order to keep the system user-friendly, the framework assumes it has missed a prior state change and creates a log entry both for the observed action, and the action(s) that should have preceded the observed action. The latter is logged with a confidence value indicating that the change was not observed but explicitly assumed, which should be reviewed at the end of the service action. Once the engineer stops streaming video data, a standardized overview with all log entries is automatically generated, including the involved confidence levels.

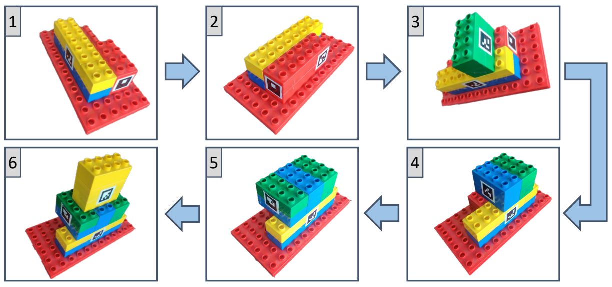


Figure 3. *Assembly of the block model used in the experiment.*

Experiments

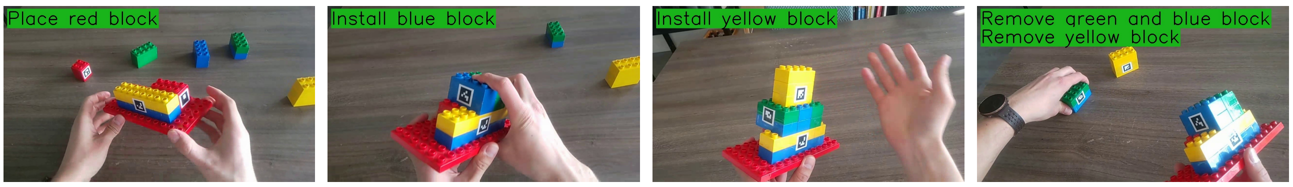
To qualitatively evaluate the performance of the proposed method, an experiment is performed on the task of assembling and disassembling a Duplo block model. This section outlines the implementation details for this experiment, the procedure, and an evaluation of the proposed method.

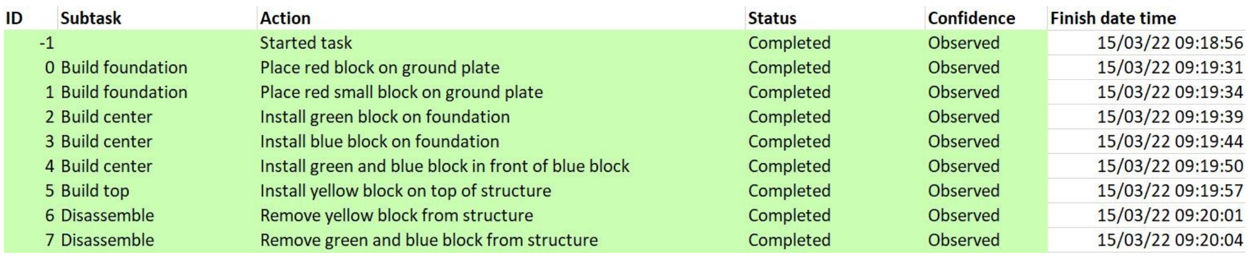
Implementation details

A HoloLens 2 is used to stream the first-person-view video to a laptop, at a resolution of 720×1280 pixels ($H \times W$) and 30 frames per second, where the live video is processed. The laptop is a T480 with an i7-8650U CPU at 1.90 GHz and 16 GB of RAM, on which the framework is executed with an average computation time of 13.0 ms per frame. To each Duplo model sub-part, a 20 × 20-mm ArUco marker from the 4x4_50 ArUco dictionary [22] is placed. For the perception module, thresholds $\tau_c = 10$ pixels and $\tau_d = 50$ pixels are chosen. Given the focal length of the HoloLens 2, these thresholds correspond to a marker at a distance of 1 m being deemed properly connected if its corners are within 1 cm of its desired position, and deemed improperly connected if its corners are not within 5 cm.

(Dis)assembly procedure

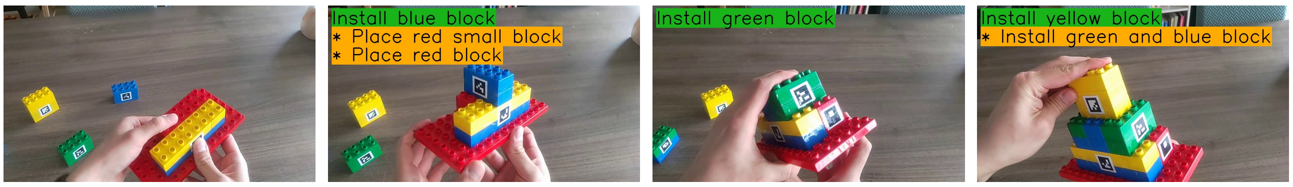
To test the proposed framework, its performance is evaluated on the task of assembling and disassembling a Duplo block model consisting of 15 blocks, divided into 7 sub-parts. One possible assembly sequence is shown in Fig. 3. Whilst the performance of the framework is demonstrated on the relatively simple task of a block model (dis)assembly, it generalizes easily to other objects, since the perception module is based solely on marker detection. Therefore, the framework can be used on a variety of different (dis)assembly tasks, even if they contain traditionally hard-to-detect components, such as objects with a high degree of symmetry or a low degree of texture. The only two requirements are that (1) the objects contain a flat surface of sufficient surface

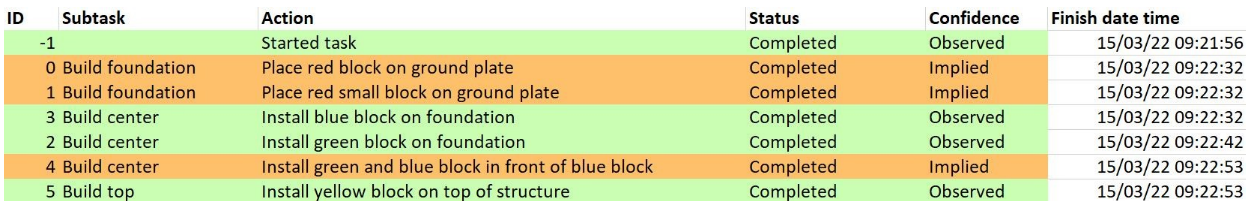




ID	Subtask	Action	Status	Confidence	Finish date time
-1		Started task	Completed	Observed	15/03/22 09:18:56
0	Build foundation	Place red block on ground plate	Completed	Observed	15/03/22 09:19:31
1	Build foundation	Place red small block on ground plate	Completed	Observed	15/03/22 09:19:34
2	Build center	Install green block on foundation	Completed	Observed	15/03/22 09:19:39
3	Build center	Install blue block on foundation	Completed	Observed	15/03/22 09:19:44
4	Build center	Install green and blue block in front of blue block	Completed	Observed	15/03/22 09:19:50
5	Build top	Install yellow block on top of structure	Completed	Observed	15/03/22 09:19:57
6	Disassemble	Remove yellow block from structure	Completed	Observed	15/03/22 09:20:01
7	Disassemble	Remove green and blue block from structure	Completed	Observed	15/03/22 09:20:04

Figure 4. *Result where all actions were correctly observed in expected order, showing examples frames of the actions (top) and the output log (bottom).*





ID	Subtask	Action	Status	Confidence	Finish date time
-1		Started task	Completed	Observed	15/03/22 09:21:56
0	Build foundation	Place red block on ground plate	Completed	Implied	15/03/22 09:22:32
1	Build foundation	Place red small block on ground plate	Completed	Implied	15/03/22 09:22:32
3	Build center	Install blue block on foundation	Completed	Observed	15/03/22 09:22:32
2	Build center	Install green block on foundation	Completed	Observed	15/03/22 09:22:42
4	Build center	Install green and blue block in front of blue block	Completed	Implied	15/03/22 09:22:53
5	Build top	Install yellow block on top of structure	Completed	Observed	15/03/22 09:22:53

Figure 5. *Results where action IDs 0, 1, and 4 were not observed (due to occlusions) but implied based on subsequent observed actions. This is explicitly highlighted in the log at the bottom of the figure. In the frames shown in the top of this figure, $*$ indicates an explicitly assumed action.*

area to place an ArUco marker and that (2) each marker has at least one relevant other marker visible during the procedure.

The Database D_{rp}, containing the relative poses of markers on correctly assembled components, is created by assembling the block model and simultaneously determining the relative poses. Therefore, it does not require images to be stored during this process. Pre-conditions for the procedure are created based on physical constraints and one procedural condition, namely to prevent a marker occlusion between Steps 4 and 5 in Fig. 3.

Evaluation

The performance of the framework is evaluated by qualitatively analyzing its performance on the aforementioned (dis)assembly operation, executed by a single operator. Fig. 4 shows the results for one execution, where every action is detected exactly in the order of which the engineer has executed the action. Fig. 4 shows the detections in real-time as well as the standardized log generated from this action. Since all actions were observed without any pre-condition being violated, no specific actions are explicitly highlighted.

Fig. 5 outlines a scenario where the operator did not show the ArUco markers to the HoloLens on several occasions, specifically the markers corresponding to action IDs 0, 1, and 4. Without detecting the markers, the action cannot be detected. Because the markers of the succeeding actions were being detected, the system successfully assumed which actions were missed by the perception module (or potentially not performed). These missed actions are explicitly highlighted in the log. As shown with these results, the performance of the framework is limited by the visibility of the markers. If the engineer does not rotate the block model with markers of newly placed components facing the camera, these components will not be registered. Marker occlusions are less problematic when engineers are working on a larger assembly that cannot be freely rotated. Additionally, instructions can be provided to inform the operators that markers should be briefly visible after actions are performed, because such visibility relieves them of the duty to later review all assumed actions.

Conclusion and future outlook

This paper has outlined the need for automatic action logging of procedural actions in manufacturing and maintenance tasks and has proposed a perception-based framework that automatically generates logs based on a head-worn AR device. With this framework, engineers no longer have to step away from their work to perform the tedious yet important task of logging performed actions. The log highlights the steps that the framework did not observe, allowing the engineer to easily review the performed procedure. The framework does not restrict engineers to rigid, step-by-step sequences. Instead, it allows engineers to execute tasks in an order they deem suitable. As the logs are stored in standardized form, they are searchable and can therefore be used to discover best practices and optimize processes.

The ideas presented in this work pave the way for research on recognizing industrial procedural actions, based on a prior knowledge of what is to be expected during procedures. The potential of recognizing such actions is not limited to automatically generating service logs, but can be extended, e.g. to provide warnings of forgotten or incorrectly executed steps in real-time, thereby reducing the number of procedural errors and potentially increasing end quality.

Acknowledgments

This work is partially executed at ASML Research and has received funding from ASML and the TKI research grant (project number TKI2112P07).

References

[1] Augmented reality to the rescue at ASML. ASML, https://www.asml.com/en/news/stories/2020/augmented-reality-to-the-rescue-coronavirus. Accessed: March 29, 2022.

[2] Juan Pablo Usuga Cadavid, Bernard Grabot, Samir Lamouri, Robert Pellerin, and Arnaud Fortin. Valuing free-form text data from maintenance logs through transfer learning with camembert. *Enterprise Information Systems*, pages 1–29, 2020.

[3] Thurston Sexton, Michael P Brundage, Michael Hoffman, and Katherine C Morris. Hybrid datafication of maintenance logs from ai-assisted human tags. In *2017 ieee international conference on big data (big data)*, pages 1769–1777. IEEE, 2017.

[4] Michael P Brundage, Thurston Sexton, Melinda Hodkiewicz, Alden Dima, and Sarah Lukens. Technical language processing: Unlocking maintenance knowledge. *Manufacturing Letters*, 27:42–46, 2021.

[5] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. Multi-modal transformer for video retrieval. In *European Conference on Computer Vision*, pages 214–229. Springer, 2020.

[6] Yang Liu, Samuel Albanie, Arsha Nagrani, and Andrew Zisserman. Use what you have: Video retrieval using representations from collaborative experts. *arXiv preprint arXiv:1907.13487*, 2019.

[7] Gabriel Evans, Jack Miller, Mariangely Iglesias Pena, Anastacia MacAllister, and Eliot Winer. Evaluating the microsoft hololens through an augmented reality assembly application. In *Degraded environments: sensing, processing, and display 2017*, volume 10197, page 101970V. International Society for Optics and Photonics, 2017.

[8] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. Duplotrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 389–402, 2012.

[9] Giles Westerfield, Antonija Mitrovic, and Mark Billinghurst. Intelligent augmented reality training for motherboard assembly. *International Journal of Artificial Intelligence in Education*, 25(1):157–172, 2015.

[10] Xuyue Yin, Xiumin Fan, Wenmin Zhu, and Rui Liu. Synchronous AR assembly assistance and monitoring system based on egocentric vision. *Assembly Automation*, 2018.

[11] Paul davies: The Boeing augmented reality kit (bark) in airplane manufacturing. The Boeing Company.

[12] Ana Malta, Mateus Mendes, and Torres Farinha. Augmented reality maintenance assistant using yolov5. *Applied Sciences*, 11(11):4758, 2021.

[13] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[14] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of the European conference on computer vision (ECCV)*, pages 619–635, 2018.

[15] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2847–2854. IEEE, 2012.

[16] Francesco Ragusa, Antonino Furnari, Salvatore Livatino, and Giovanni Maria Farinella. The meccano dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1569–1578, 2021.

[17] Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *arXiv preprint arXiv:2203.14712*, 2022.

[18] Yongzhi Su, Jason Rambach, Nareg Minaskan, Paul Lesur, Alain Pagani, and Didier Stricker. Deep multi-state object pose estimation for augmented reality assembly. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 222–227. IEEE, 2019.

[19] Bing Zhou and Sinem Güven. Fine-grained visual recognition in mobile augmented reality for technical support. *IEEE Transactions on Visualization and Computer Graphics*, 26(12):3514–3523, 2020.

[20] Hangfan Liu, Yongzhi Su, Jason Rambach, Alain Pagani, and Didier Stricker. TGA: Two-level group attention for assembly state detection. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 258–263. IEEE, 2020.

[21] Jonathan Jones, Gregory D Hager, and Sanjeev Khudanpur. Toward computer vision systems that understand real-world assembly processes. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 426–434. IEEE, 2019.

[22] Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76:38–47, 2018.

[23] Guillermo Gallego and Anthony Yezzi. A compact formula for the derivative of a 3-d rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision*, 51(3):378–384, 2015.

[24] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Author Biography

***Tim Schoonbeek** is a PhD student in the Video Coding and Architectures (VCA) research group at Eindhoven University of Technology. His research focuses on visual perception and augmented reality and is conducted in collaboration with ASML Research. He received his B.S. (2019) and M.Sc. (2021) in Automotive Technology from Eindhoven University.*

***Pierluigi Frisco** is a senior researcher at ASML. His field of expertise is Data Science and Machine Learning. He applied these techniques in a wide variety of academic and industrial problems.*

***Hans Onvlee** is senior manager research at ASML. His field is in AI and Machine Learning, including the application of this in the field of advanced system diagnostics.*

***Peter de With** is a full professor and leads the VCA group at the Eindhoven University of Technology. He has coauthored more than 70 refereed international book chapters and journal articles, over 500 conference publications and 40 international patents. He served as Technical Committee Member of the IEEE CES, ICIP, SPIE and is member of the Royal Holland Society of Academic Sciences and Humanities.*

***Fons van der Sommen** is an assistant professor at Eindhoven University of Technology and heads the Healthcare&High-tech cluster of the VCA research group. He has worked on a variety of image processing and computer vision applications, mainly in the medical domain, and strives to exploit signal processing and information theory methods to improve the robustness, efficiency and interpretability of modern-day AI architectures.*