A. Overview of Computer Vision Implementations

Table A.1: Overview of existing solutions for multi-step computer vision capabilities

Reference	**Artifact**	**Implemented/Considered Computer Vision Capabilities**								
		BS	**IC**	**OD**	**SE**	**HPE**	**OT**	**FR**	**RID**	**AR**
Lim et al. [1]	Framework	x					x			
Nazare Jr. and Schwartz [2]	Framework	x		x		x	x	x	x	x
Cooharojananone et al. [3]	Framework	x		x			x			
Lotfi et al. [4]	Framework			x			x			
Weissenfeld et al. [5]	Architecture	x		x			x			
Wang et al. [6]	Architecture	x		x			x			
Räty [7]	Review	x		x			x			x
Shidik et al. [8]	Review	x		x			x	x		x
Sikandar et al. [9]	Review	x		x						

Note: Uses the following abbreviations: Background Subtraction (BS), Image Classification (IC), Object Detection (OD), Semantic Segmentation/Instance Segmentation (SE), Human Pose Estimation (HPE), Object Tracking (OT), Face Recognition (FR), Re-identification (RID), Activity recognition (AR).

B. Approaches for Event Extraction from Unstructured Data Sources

Table B.1: Summary of approaches for event extraction from unstructured data sources

Reference	**Summary**
Pospiech et al. [10],[11]	Presents a theoretical approach that is implemented as an artifact. The artifact makes it possible to extract event data from common databases as well as unstructured documents that contain process traces.
Yang et al. [12]	Proposes an architecture for the analysis of structured and unstructured manufacturing data with process mining and presents a prototype that implements parts of the architecture.
van Eck et al. [13]	Applies process mining on sensor data from a smart baby bottle and maps sensor measurements to corresponding activities that are grouped into process instances. Analyzes the extracted event information with different process mining techniques.
Cameranesi et al. [14]	Uses a dataset that features different types of sensor data and applies process mining techniques to discover process models with predefined activities of daily living.
Raso et al. [15]	Proposes an architecture and implements a multi-sensor-based system for real-time recognition of human activities as well as for the detection of inconvenient ergonomic behavior. Does not target video data or generate an event log.
Knoch et al. [16]	Implements and evaluates a system that observes manual assembly processes and fuses video data and other sensors (e.g., ultrasonic sensor). Defines event patterns and uses an assembly workstation made from carton prototyping material for evaluation.
Knoch et al. [17]	Reduces the sensor setting presented in [16] and refines the event extraction. Also correlates events to a normative process model.
Knoch et al. [18]	Uses an assembly workstation and a single RBG camera. Tracks hand movement trajectories and uses unsupervised clustering techniques to assign them to predefined work step events.
Rebmann et al. [19]	Designs a system architecture for event log generation of manual activities based on sensor data. Incorporates image data and worker feedback to resolve ambiguities. Focuses on the process discovery use case in a real-time scenario.

C. Details on the Instantiated Software Prototype of the ViProMiRA

This section provides a detailed description on how we instantiated the prototype's components.

Use cases: We target two of the initial process mining use cases (i.e., conformance checking and process discovery), as they comprise numerous established and well-known analysis methods that facilitate the interpretation and communication of our results.

Applications: We select the Process Mining Tool ProM [20] to analyze the prototype's event log. It is the one of the most actively used tools in process mining research and is, thus, well suited for use with the selected use cases.

Video Data: Due to the availability of multiple labeled video datasets designed to train different computer vision capabilities, the prototype receives videos (e.g., .mp4 files) only as input data. Currently, no other video stream support is implemented.

Video2Frame Converter: For preprocessing, we implement FFmpeg and OpenCV's VideoCapture [21]. FFmpeg facilitates the conversion of video data, and we primarily use it to adjust the frame rate based on the user's settings. We extract frames with OpenCV's VideoCapture class [22]. These may be further processed depending on the requirements of the Information Extractor components (e.g., rescaling [23] or normalizing color dimensions).

Information Extractor Subsystem: Since we target manual activities concurrently conducted by multiple actors, we incorporate spatio-temporal activity recognition. To distinguish resources and, thus, obtain meaningful event log information, we additionally implement the Object Specifier in the form of an Object Tracker. The prototype's Object Detector and Object Tracker operate at the adjusted video frame rate and process every frame. This leads to smaller changes in motion and appearance information between two consecutive frames, which enhances the accuracy but also raises the computational complexity. To capture relevant activities, the prototype performs activity recognition at one-second intervals. Since natural human motions and activities typically last more than one second, the prototype extracts relevant information at reasonable computing costs. All instantiated components of the Information Extractor are suitable for training on a custom dataset. While training improves the accuracy, it also increases the manual effort (e.g., labels may have to be created or video data may have to be prepared). Consequently, the decision to train a component is usually a tradeoff between effort and accuracy gain and is highly dependent on the characteristics of the process at hand. Usually, pre-trained object detection and object tracking models produce sufficiently accurate results for human resources in various processes. On the contrary, the activities in different processes may vary greatly (e.g., product assembly process vs. welding process) and show a significant variation in visual appearance. Furthermore, no large datasets for manual process-related spatio-temporal activity recognition exist, and available pre-trained models usually comprise generic activity categories (e.g., "pull"). Although it might be possible to hierarchically build high-level activities based on the generic categories of the pre-trained models, activity recognition models trained on custom datasets might reveal more valuable information for specific process types (e.g., assembly processes).

Object Detector: We integrate the Detectron2 library, which comprises state-of-the-art object detection algorithms [24]. Detectron2 is publicly available and includes various pre-trained object detection models for person detection [24]. To filter out irrelevant detections, we implement the option of ignoring all bounding boxes that do not have a predefined minimum pixel height.

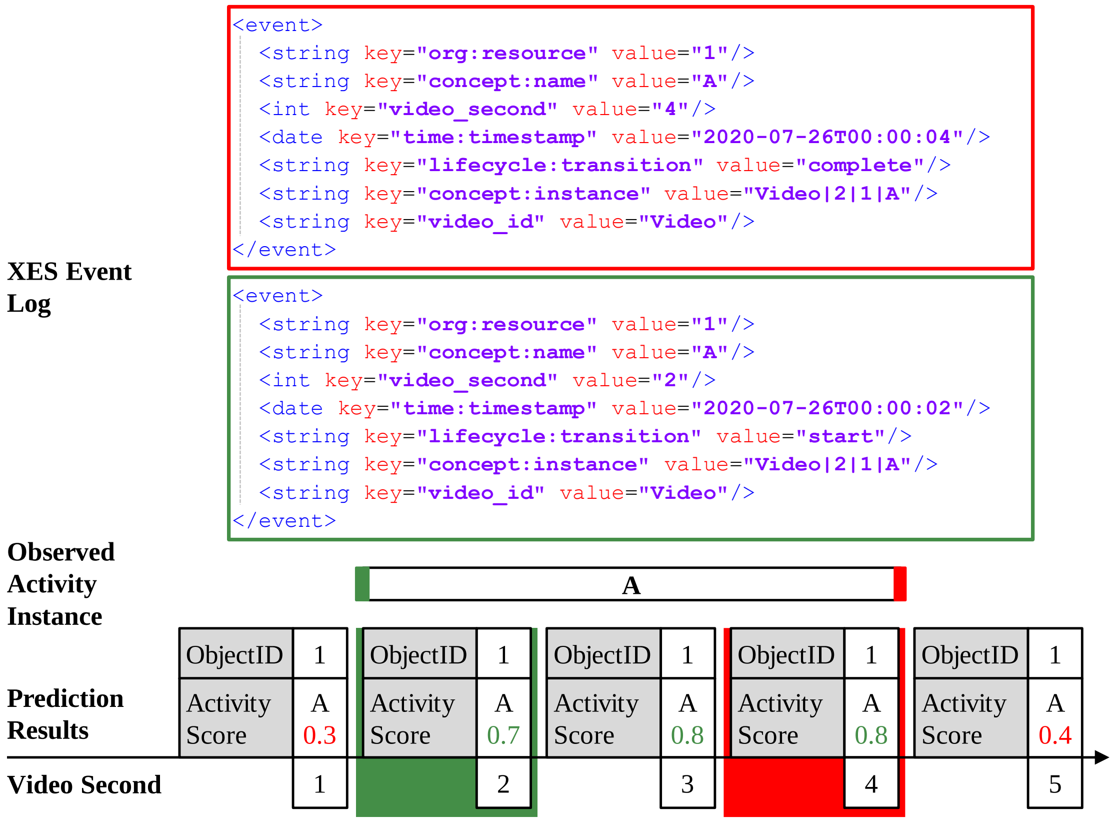
Object Specifier/Object Tracker: The prototype incorporates DeepSORT [25,26] as real-time object Tracker. DeepSORT leverages information on motion as well as appearance, which is generated using a CNN that has been trained to distinguish people on a massive person RID dataset [26]. The appearance information improves the tracking results after periods of occlusion.

Spatio-temporal Activity Recognizer: The prototype implements the publicly available SlowFast repository for supervised activity recognition [27]. SlowFast networks consist of a Slow pathway with a low frame rate but higher computational complexity and a Fast pathway with a high frame rate and lower computational complexity [23]. They achieve very high accuracy for notable benchmark datasets including video classification (e.g., Kinetics [28]), temporal activity recognition (e.g., Charades [29]), and spatio-temporal activity recognition (e.g., Atomic Visual Actions (AVA) [30]) [23]. The SlowFast architecture promises high accuracy for activity recognition, which is of the utmost importance for creating trustworthy event logs [31]. While

SlowFast's current code base supports the modification of several model settings (e.g., the number of frames for the Slow and Fast pathway or the type of model), it does not offer the possibility of specifically training a model on a custom dataset. We added this function by extending SlowFast's existing modules for the AVA dataset. The Activity Recognizer requires annotations corresponding to the format of the AVA dataset (i.e., the activity as well as the position of each actor must be provided at each full video second) [30]. To facilitate the supervised training for new custom datasets, we further developed a preprocessing functionality that automates manual preprocessing tasks (e.g., creating the folder structure, preparing annotation files, extracting frames from the original videos, or predicting person bounding boxes for validation and test data).

Event Processor Subsystem: The two targeted process mining use cases require the extraction of an event log based on the results of the Activity Recognizer. Therefore, the prototype can abstract high-level events and – on the assumption that a single activity is exclusively executed by a single resource – activity instances. Users may specify a minimum threshold (i.e., export threshold) that determines whether a predicted activity class is "observed." Only observed predictions are considered for export. Since the prototype does not dispose of any process knowledge, it cannot assign activity instances to cases. Consequently, we leave the event-case correlation open to the process mining application layer.

Event Aggregator: Figure C.1 illustrates how the prototype aggregates the low-level predictions into activity instances and how the information is stored in the XES format. In our example, the Event Aggregator receives the prediction of activity class "A" for resource "1" with alternating probabilities (i.e., scores) for five consecutive seconds. Assuming an export threshold of 50%, the prototype observes the low-level events "A" in seconds two, three, and four. It concludes that the activity instance "A" starts in the second two (green) and is completed in the second four (red). Using the standard transactional life-cycle model, the XES-conforming representation of this information results in two events (i.e., green for start and red for completion). The Event Aggregator assigns the values of the corresponding event attributes based on the information at hand. To distinguish several observed activity instances in a trace, each corresponding event pair must have the same value for the “concept:instance” attribute.



```
<event>
  <string key="org:resource" value="1"/>
  <string key="concept:name" value="A"/>
  <int key="video_second" value="4"/>
  <date key="time:timestamp" value="2020-07-26T00:00:04"/>
  <string key="lifecycle:transition" value="complete"/>
  <string key="concept:instance" value="Video|2|1|A"/>
  <string key="video_id" value="Video"/>
</event>
<event>
  <string key="org:resource" value="1"/>
  <string key="concept:name" value="A"/>
  <int key="video_second" value="2"/>
  <date key="time:timestamp" value="2020-07-26T00:00:02"/>
  <string key="lifecycle:transition" value="start"/>
  <string key="concept:instance" value="Video|2|1|A"/>
  <string key="video_id" value="Video"/>
</event>
```

Video Second	ObjectID	Activity	Score
1	1	A	0.3
2	1	A	0.7
3	1	A	0.8
4	1	A	0.8
5	1	A	0.4

Figure C.1: Event abstraction logic and event log generation (export threshold = 0.5)

Thus, the event aggregator creates a unique key in the format “video_id|video_second|org:resource|concept:name” based on the respective attribute values of the start event. This key is assigned to the “concept:instance” attribute of the start event and the completion event, which jointly represent the observed activity instance. This concept can easily be extended for scenarios with non-exclusive activity classes and predictions for several resources at the same time. Instead of focusing only on the activity class with the highest predicted probability in each second, the Event Aggregator would individually apply the procedure described above for each resource to all activity classes in each second.

Event Log Exporter: To export an XES event log and, thereby, transmit the information past the boundaries of the prototype, we use the PM4Py Python process mining library [32]. Since no case correlation is implemented, this results in one trace for all events.

The software prototype's generic implementation guarantees its suitability for various naturalistic application scenarios. Furthermore, the successful export of meaningful event logs enables several process mining use cases and confirms the internal consistency of the ViProMiRA.

D. Process Discovery Results

Table D.1: Comparison of process models resulting from different thresholds against four quality criteria

ID	Threshold	Simplicity	Generalization	Precision (%)		Fitness (%)	
				True_Log$_{12}$	True_Log$_{53}$	True_Log$_{12}$	True_Log$_{53}$
1	0.00	-	+	Not defined	Not defined	98.61	98.83
2	0.05	-	+	Not defined	Not defined	98.61	98.83
3	0.10	0	+	46.52	51.18	95.73	93.69
4	0.15	0	+	56.65	59.72	94.70	90.61
5	0.20/0.25	0	+	56.68	57.68	91.10	87.60
6	0.30	+	0	78.40	78.56	87.42	83.91
7	0.35	+	-	78.57	85.71	66.03	61.00

Note: Simplicity and generalization are rated positive (+), neutral (0), or negative (-).

The high fitness values of models 1 and 2 in Table D.1 express that they allow for the behavior contained in the event logs. On the other side, their low precision in line with the high generalization, as well as their lack of simplicity, indicate the allowance of considerable deviation from the traces in the logs. Model 7 shows opposite results and poorly represents the traces in the event logs, as it is oversimplified and generalizes insufficiently. The increasing filtering of infrequent behavior (i.e., models 3-5) improves the models' simplicity and precision at the cost of their fitness. For example, model 4 is the first to include the correct, sequential order of the activities "stir" and "pour." Overall, model 6 yields the best trade-off between all quality criteria. While it exhibits high fitness and precision values, it is still simple and generalizes satisfactorily.

References

[1] M.K. Lim, S. Tang, C.S. Chan, iSurveillance: Intelligent framework for multiple events detection in surveillance videos, Expert Systems with Applications 41 (2014) 4704–4715. https://doi.org/10.1016/j.eswa.2014.02.003.

[2] A.C. Nazare Jr., W.R. Schwartz, A scalable and flexible framework for smart video surveillance, Computer Vision and Image Understanding 144 (2016) 258–275. https://doi.org/10.1016/j.cviu.2015.10.014.

[3] N. Cooharojananone, S. Kasamwattanarote, R. Lipikorn, S. Satoh, Automated real-time video surveillance summarization framework, J Real-Time Image Proc 10 (2015) 513–532. https://doi.org/10.1007/s11554-012-0280-7.

[4] M. Lotfi, S.A. Motamedi, S. Sharifian, Time-based feedback-control framework for real-time video surveillance systems with utilization control, J Real-Time Image Proc 16 (2019) 1301–1316. https://doi.org/10.1007/s11554-016-0637-4.

[5] A. Weissenfeld, A. Opitz, R. Pflugfelder, G. Fernández, Architecture for Dynamic Allocation of Computer Vision Tasks, in: Proceedings of the 10th International Conference on Distributed Smart Camera, Paris, France, ACM, New York, NY, 2016, pp. 50–55.

[6] G. Wang, L. Tao, Di H, X. Ye, Y. Shi, A Scalable Distributed Architecture for Intelligent Vision System, IEEE Transactions on Industrial Informatics 8 (2012) 91–99. https://doi.org/10.1109/TII.2011.2173945.

[7] T.D. Räty, Survey on Contemporary Remote Surveillance Systems for Public Safety, IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 40 (2010) 493–515. https://doi.org/10.1109/TSMCC.2010.2042446.

[8] G.F. Shidik, E. Noersasongko, A. Nugraha, P.N. Andono, J. Jumanto, E.J. Kusuma, A Systematic Review of Intelligence Video Surveillance: Trends, Techniques, Frameworks, and Datasets, IEEE Access 7 (2019) 170457–170473. https://doi.org/10.1109/ACCESS.2019.2955387.

[9] T. Sikandar, K.H. Ghazali, M.F. Rabbi, ATM crime detection using image processing integrated video surveillance: a systematic review, Multimedia Systems 25 (2019) 229–251. https://doi.org/10.1007/s00530-018-0599-4.

[10] S. Pospiech, R. Mertens, S. Mielke, M. Städler, P. Söhlke, Creating Event Logs from Heterogeneous, Unstructured Business Data, in: F. Piazolo, M. Felderer (Eds.), Multidimensional Views on Enterprise Information Systems, Springer International Publishing, Cham, 2016, pp. 85–93.

[11] S. Pospiech, S. Mielke, R. Mertens, K. Jagannath, M. Stadler, Exploration and Analysis of Undocumented Processes Using Heterogeneous and Unstructured Business Data, in: 2014 IEEE International Conference on Semantic Computing, Newport Beach, CA, USA, 2014, pp. 191–198.

[12] H. Yang, M. Park, M. Cho, M. Song, S. Kim, A system architecture for manufacturing process analysis based on big data and process mining techniques, in: 2014 IEEE

International Conference on Big Data, Washington, DC, USA, IEEE, Piscataway, NJ, 2014, pp. 1024–1029.

[13] M.L. van Eck, N. Sidorova, W.M.P. van der Aalst, Enabling process mining on sensor data from smart products, in: IEEE RCIS 2016, Grenoble, France, IEEE, Piscataway, NJ, 2016, pp. 1–12.

[14] M. Cameranesi, C. Diamantini, D. Potena, Discovering Process Models of Activities of Daily Living from Sensors, in: Business Process Management Workshops, Springer International Publishing, Cham, 2018, pp. 285–297.

[15] R. Raso, A. Emrich, T. Burghardt, M. Schlenker, T. Gudehus, O. Sträter, P. Fettke, P. Loos, Activity Monitoring Using Wearable Sensors in Manual Production Processes - An Application of CPS for Automated Ergonomic Assessments, in: Tagungsband Multikonferenz Wirtschaftsinformatik (MKWI), Leuphana Universität Lüneburg, 2018, pp. 231–242.

[16] S. Knoch, S. Ponpathirkoottam, P. Fettke, P. Loos, Technology-Enhanced Process Elicitation of Worker Activities in Manufacturing, in: Business Process Management Workshops, Springer International Publishing, Cham, 2018, pp. 273–284.

[17] S. Knoch, N. Herbig, S. Ponpathirkoottam, F. Kosmalla, P. Staudt, P. Fettke, P. Loos, Enhancing Process Data in Manual Assembly Workflows, in: Business Process Management Workshops, Springer International Publishing, Cham, 2019, pp. 269–280.

[18] S. Knoch, S. Ponpathirkoottam, T. Schwartz, Video-to-Model: Unsupervised Trace Extraction from Videos for Process Discovery and Conformance Checking in Manual Assembly, in: D. Fahland, C. Ghidini, J. Becker, M. Dumas (Eds.), Business Process Management, Springer International Publishing, Cham, 2020, pp. 291–308.

[19] A. Rebmann, A. Emrich, P. Fettke, Enabling the Discovery of Manual Processes Using a Multi-modal Activity Recognition Approach, in: Business Process Management Workshops, Springer International Publishing, Cham, 2019, pp. 130–141.

[20] ProM: The process mining toolkit, 2009.

[21] FFmpeg, 2022. https://ffmpeg.org/ (accessed 6 March 2022).

[22] OpenCV: cv:VideoCapture Class Reference, 2020. https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html#a57c0e81e83e60f36c830 27dc2a188e80 (accessed 6 March 2022).

[23] C. Feichtenhofer, H. Fan, J. Malik, K. He, SlowFast Networks for Video Recognition, in: International Conference on Computer Vision (ICCV), 2019, pp. 6201–6210.

[24] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, Ross Girshick, Detectron2, 2019. https://github.com/facebookresearch/detectron2.

[25] ZQPei, Deep Sort with PyTorch, 2021. https://github.com/ZQPei/deep_sort_pytorch (accessed 6 March 2022).

[26] N. Wojke, A. Bewley, D. Paulus, Simple online and realtime tracking with a deep association metric, in: IEEE International Conference on Image Processing (ICIP), 2017, pp. 3645–3649.

[27] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, Christoph Feichtenhofer, PySlowFast, 2020. https://github.com/facebookresearch/SlowFast (accessed 6 March 2022).

[28] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, A. Zisserman, The Kinetics Human Action Video Dataset, 2017.

[29] G.A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, A. Gupta, Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding, in: Computer Vision – ECCV 2016, Cham, Springer International Publishing, Cham, 2016, pp. 510–526.

[30] C. Gu, C. Sun, D.A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, J. Malik, AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions, 2017.

[31] W. van der Aalst, Process Mining: Data Science in Action, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[32] A. Berti, S.J. van Zelst, W. van der Aalst, Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science, in: Proceedings of the ICPM Demo Track 2019, 2019.