Details on the instantiation of the Video Process Mining RA for EVAL3

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 [1] 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 [2]. FFmpeg facilitates the conversion of video data, and we primarily use it to adjust the framerate based on the user's settings. We extract frames with OpenCV's VideoCapture class [3]. These may be further processed depending on the requirements of the current Information Extractor components (e.g., rescaling [4] or normalizing color dimensions).

Information Extractor Subsystem: Since we target manual activities concurrently conducted by multiple actors, we incorporate spatio-temporal AR. 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 framerate 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 AR 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 OD and OT 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 AR 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, AR models trained on custom datasets might reveal more valuable information for specific process types (e.g., assembly processes). In our evaluation, we use pre-trained models for the Object Detector and Object Tracker and train only the Activity Recognizer.

Object Detector: We integrate the Detectron2 library, which comprises state-of-the-art OD algorithms [5]. Detectron2 is publicly available and includes various pre-trained OD models for person detection [5]. 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 [6,7] 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 [7]. 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 AR [8]. SlowFast networks consist of a Slow pathway with a low framerate but higher computational complexity and a Fast pathway with a high framerate and lower computational complexity [4]. They achieve very high accuracy for notable benchmark datasets including video classification (e.g., Kinetics [9]), temporal AR (e.g., Charades [10]), and spatio-temporal AR (e.g., Atomic Visual Actions (AVA) [11]) [4]. The

SlowFast architecture promises high accuracy for AR, which is of the utmost importance for creating trustworthy event logs [12]. 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) [11]. 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 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 process instances. Consequently, we leave the event-case correlation open to the process mining application layer.

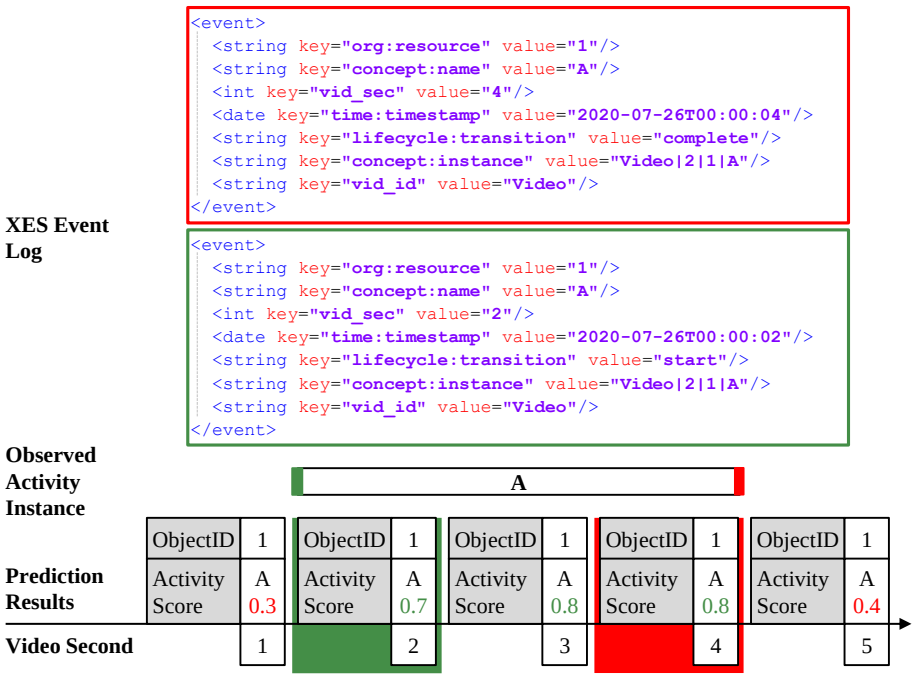

Figure 1: Event abstraction logic and event log generation based on Figure 4 (export threshold = 0.5)

Event Aggregator: Figure 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 percent, 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 could result 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. Thus, the event aggregator creates a unique key in the format "vid_id|vid_sec|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 individually applies 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 [13]. Since no case correlation is implemented, this results in one trace for all events.

The software prototype demonstrates the operationality of our RA. Its 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 our RA.

Overview of Computer Vision Implementations

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

Reference	*Artifact*	*Implements/considers these computer vision capabilities*								
		BS	IC	OD	SeSe	HPE	OT	FR	RID	AR
Lim et al. [14]	Framework	x					x			
Nazare Jr. and Schwartz [15]	Framework	x		x		x	x	x	x	x
Cooharojananone et al. [16]	Framework	x		x			x			
Lotfi et al. [17]	Framework			x			x			
Weissenfeld et al. [18]	Architecture	x		x			x			
Wang et al. [19]	Architecture	x		x			x			
Räty [20]	Review	x		x			x			x
Shidik et al. [21]	Review	x		x			x	x		x
Sikandar et al. [22]	Review	x		x						

Process Discovery Results

Table 2: Comparison of process models resulting from different thresholds against four quality criteria

ID	*Threshold*	*Simplicity*	*Generalization*	*Precision (%)*		*Fitness (%)*	
				GT_LOG_{12}	GT_LOG_{53}	GT_LOG_{12}	GT_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

References

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

[2] FFmpeg, 2019. https://ffmpeg.org/ (accessed 31 July 2020).

[3] OpenCV: cv:VideoCapture Class Reference, 2020. https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html#a57c0e81e83e60f36c83027dc2a188e80 (accessed 31 July 2020).

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

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

[6] ZQPei, Deep Sort with PyTorch, 2020. https://github.com/ZQPei/deep_sort_pytorch (accessed 2 August 2020).

[7] 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.

[8] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, Christoph Feichtenhofer, PySlowFast, 2020. https://github.com/facebookresearch/SlowFast (accessed 11 September 2020).

[9] 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.

[10] G.A. Sigurdsson, et al., Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding, in: Computer Vision – ECCV 2016, Cham, Springer International Publishing, Cham, 2016, pp. 510–526.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] G.F. Shidik, et al., 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.

[22] 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.