Analytics Pipeline for Process Mining on Video Data

Arvid Lepsien[1](✉), Agnes Koschmider[2], and Wolfgang Kratsch[3]

[1] Department of Computer Science, Kiel University, Kiel, Germany
ale@informatik.uni-kiel.de

[2] Chair of Business Informatics and Process Analytics, University of Bayreuth, Bayreuth, Fraunhofer FIT, Bayreuth, Germany
agnes.koschmider@uni-bayreuth.de

[3] Technical University of Applied Sciences Augsburg, Fraunhofer FIT, Augsburg, Germany
wolfgang.kratsch@fim-rc.de

Abstract. Process mining has shown that it provides valuable insights in terms of uncovering bottlenecks and inefficiencies in processes or identifying tasks for automation. However, process mining techniques expect structured input data that is at a high (business) level of abstraction. Recently, the benefits of process mining for unstructured data which is at a much lower level of abstraction have been demonstrated, e.g., for IoT data or time series data. It can be expected that the demand for methods efficiently processing these kinds of data for process mining will continuously increase. Hence, in this paper, we present an approach that allows the translation of video data into higher-level, discrete event data, thus enabling existing process mining techniques to work on data tracked in videos. Particularly, we used a combination of object tracking, spatio-temporal action detection, and techniques for raising the abstraction level of events. The evaluation results show that meaningful event logs can be extracted from an unlabeled video dataset, validating both the implementation and the feasibility of our approach.

Keywords: Process mining · Event log extraction · Unstructured data · Activity recognition

1 Introduction

Process mining (PM) strives to discover, monitor, and improve processes by extracting knowledge from structured event logs typically sourced from core information systems (e.g., ERP systems) [1]. However, for many processes (e.g.,

We thank Jan Bosselmann for his support with the implementation, and the Institute of Agricultural Engineering at Kiel University for providing a dataset and use case. This project has received funding from the State of Schleswig-Holstein under the Datencampus project grant no. 220 21 016.

C. Di Francescomarino et al. (Eds.): BPM 2023 Forum, LNBIP 490, pp. 196–213, 2023.
https://doi.org/10.1007/978-3-031-41623-1_12

highly manual processes), structured data is not available, resulting in blind spots that are not covered by traditional PM approaches. Consequently, PM often does not deliver a full, end-to-end process analysis, but only insights into digitized parts of processes. However, process-related behavior may also be captured by various types of unstructured data, i.e., data that is not organized in a pre-defined manner, or at least no data scheme that is directly applicable for PM purposes. Video cameras are an especially promising source of process-related unstructured data because (1) video cameras are inexpensive and easy to set up and (2) a high amount of diverse information can be extracted from video recordings using modern computer vision techniques.

First approaches concerned with the analysis of video data for PM have been proposed, which were used to examine structured processes from highly specific contexts in logistics and production, recorded in laboratory settings [11,14]. In reality, however, processes are often chaotic and unstructured, due to irrational actors or unpredictable internal and external disorders. To the best of our knowledge, no existing approach offers a solution to analyze these rather chaotic and unstructured processes using video data. Therefore, we present a novel, end-to-end analytics pipeline for performing PM using video data, that enables the video-based analysis of unstructured processes with the help of systematic event abstraction and event log processing. By evaluating our pipeline on videos capturing the daily activities of fattening pigs, we employ a chaotic process with a limited set of known activities. This allows us to perform a technical evaluation of activities but also allows for the incorporation of feedback from domain experts (i.e., agricultural scientists) to confirm the validity of our automatically mined insights.

The paper is structured as follows. First, we introduce our end-to-end analytics pipeline for PM on video data (Sect. 2). Then, we introduce our implementation of this pipeline (Sect. 3) and our use case (Sect. 4), and report the evaluation of our pipeline including the implementation and application to real-world data (Sect. 5). Finally, we outline related work (Sect. 6) and conclude our research and discuss its limitations as well as potential avenues for future research (Sec. 7).

2 Process Analytics Pipeline

This section presents our analytics pipeline for PM for video data, which consists of six steps, as depicted in Fig. 1. The main objective of this pipeline is the discovery of process models from unstructured data, specifically video data. The pipeline enables the identification of structure in terms of a process model from unstructured data, facilitating the detection and explanation of bottlenecks, anomalies, and causalities. In order to apply the pipeline, an analysis goal or problem statement must be defined, which significantly influences the method selection and parameters for each step of the pipeline. The design of the pipeline is based on the pipeline originally presented in [17]. Compared to this pipeline, the proposed approach was extended by (1) renaming of the preprocessing steps

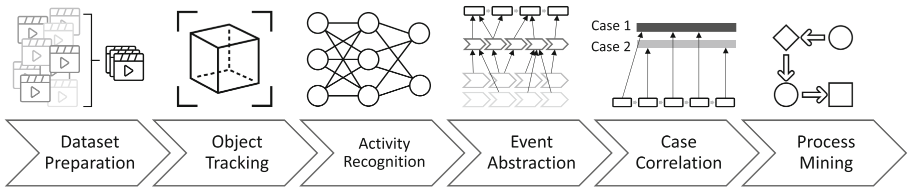


Fig. 1. Process analytics pipeline for process mining on video data (based on [12,17])

to better reflect the tasks related to *Dataset Preparation* and *Object Tracking*, (2) refinement of the *activity recognition* step into the three separate steps of *Activity Recogniton*, *Event Abstraction*, and *Case Correlation*, and (3) removal of the step *mine domain specific knowledge* and the refinement loop. Additionally, this paper contains details on how to select appropriate methods for each pipeline step, a full implementation of the analytics pipeline, and the evaluation of the approach in a real-world use case.

A critical and necessary requirement of the pipeline is the extraction of *events* with *timestamps* from unstructured data. Events are defined as any relevant observations related to a process and do not need to be linked to a specific process activity at the outset. Without the notion of *events* with *timestamps*, the pipeline cannot generate any meaningful results.

2.1 Dataset Preparation

In the first step of the pipeline, a raw video dataset is processed in order to transform it into a unified format. The challenges of unifying video data are to bridge the gap between different resolutions as well as frame rates that arise from using different cameras or recording devices and to combine contiguous recordings that are split across multiple video files.

First, relevant video segments are selected based on the analysis goal. The relevancy of video segments depends on factors such as the presence of specific objects or activities, or on their context. For instance, if the goal is to analyze the performance of a process in a specific part of the day, only video recordings from that part of the day are relevant. Once the relevant segments have been identified, they are resampled to a common frame rate and their resolution is scaled down to reduce the computational load. Furthermore, the degree of resolution reduction depends on the detail required by subsequent pipeline steps, which, in turn, depends on the analysis goal. It is important to reduce the resolution in a controlled manner, as otherwise this could result in the loss of utility of the video segments.

2.2 Object Tracking

In the second step of the pipeline, relevant objects are detected and tracked within the selected video segments. Objects are relevant to the analysis if they

Table 1. Exemplary inputs and outputs of spatio-temporal action detection

Frame	Bounding Box	Track ID	Activities
...	...	...	...
13759	[261, 314, 453, 432]	1	{A: 0.601, B: 0.102}
13791	[274, 315, 566, 410]	1	{A: 0.629}
13823	[314, 295, 638, 399]	1	{A: 0.489, C: 0.222}
13855	[459, 248, 696, 374]	1	{A: 0.699, C: 0.217}
...	...	...	...

are related to the observed process, which are typically the actors performing the process activities. The objective of this step is to extract context information for subsequent steps. Specifically, the bounding boxes of relevant objects are required as input for *Activity Recognition*, and the movement trajectories for *Event Abstraction* as well as *Case Correlation*. The quality of object tracking significantly influences the quality of the subsequent steps.

A tracking-by-detection approach is used for *Object Tracking*, which consists of separate object detection and tracking models [19]. Tracking-by-detection is a suitable approach for this pipeline step, because it has state-of-the-art quality, and only the object detection model needs to be customized for different use cases. Deep learning models, which have been pre-trained on large and heterogeneous image datasets, can be adapted for object detection in order to detect customized objects with few labeled examples, even if these objects were not contained in the original training dataset [26]. *Object Tracking* outputs the bounding boxes of relevant objects in all frames of the selected videos, as well as movement trajectories that identify these objects throughout video segments with tracking IDs.

2.3 Activity Recognition

In the third pipeline step, low-level events are extracted from the selected videos using deep learning. Specifically, a spatio-temporal action detection technique [8] is used, which can detect multiple actions performed concurrently by multiple objects in the same video. Since this is a supervised learning task, a dataset of videos labeled with actions related to the analysis goal must be manually created to train the deep learning model. To simplify the subsequent step of *Event Abstraction*, the training dataset is labeled with actions that directly correspond to the activities of the analyzed process.

An excerpt of inputs and outputs of this pipeline step is listed in Table 1. Inputs for the detector include the previously detected bounding boxes and tracking IDs of relevant objects for each frame. The trained model then analyzes segments of the videos at a fixed interval of frames to detect the activities performed by each object. As seen in the *Activities* column, multiple activities are detected with varying confidence scores for each object and frame, rather than

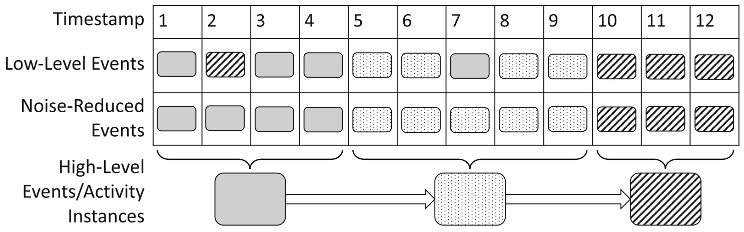


Fig. 2. Abstraction from low-level events to high-level activity instances

single specific activities, which needs to be addressed in the following pipeline step. In summary, *Activity Recognition* outputs a list of low-level events, each referring to the activities that were observed for an object at a specific time.

2.4 Event Abstraction

In the next step of the pipeline, the low-level events from *Activity Recognition* are abstracted to high-level events and finally correlated with activity instances (see Fig. 2). *Activity Recognition* generates multiple, fine-granular events for each observed activity. However, PM algorithms expect abstract, high-level events that denote only the status of the activity, such as the start and end times of execution. Because the activity classes used for *Activity Recognition* directly correspond to the activities of the analyzed process, conceptual abstraction of low-level events is not required. Rather, the low-level events need to be aggregated *temporally* to high-level events corresponding to entire activity instances. Consequently, *Event Abstraction* is done in two steps: first, in order to reduce local noise caused by imperfectly detected activities, a moving average is applied across the confidence scores assigned to the activities of each object, which removes implausibly short or sudden changes of the detected activities. An appropriate size for the moving average window needs to be determined based on the typical duration of relevant process activities, in an effort to avoid removing short activities unintentionally, while simultaneously removing as much noise as possible. After noise reduction, if multiple activities are detected for the same object within the same low-level event, the activity with the highest score is selected. Second, the noise-reduced events are abstracted to high-level events by aggregating repeated observations of the same activity in the sequence of events for each object. Occurrences of activities, i.e., activity instances, can then be identified through the contextualization of high-level events into the realm of a specific process [12,32]. For this purpose, we assume that a one-to-one mapping can be constructed from high-level events to activity instances, and directly treat the high-level events as activity instances. The output of *Event Abstraction* is a log of high-level activity instances, which we consider as an event log without a case allocation.

2.5 Case Correlation

The objective of the fifth pipeline step is to organize the activity instances identified from *Event Abstraction* into cases. Case correlation is a pivotal step of the pipeline since the cases are interpreted as process instances, which implicitly define the process that will be discovered. Case correlation can be approached in three distinct ways: (1) in certain scenarios, case identifiers may be provided externally, either explicitly (e.g., through manual annotations) or implicitly (e.g., one case per video file, as in [11]). (2) Depending on the analysis goal, case correlation can also be performed using information extracted in the previous steps together with domain-specific assumptions. For instance, if the analyzed process is characterized by well-defined start as well as end activities and each process instance is executed by precisely one object, the sequence of activities for each object can be split into cases that run from each start activity to the first following end activity [14]. (3) When domain-specific assumptions are inadequate to construct cases that align with the analysis goal, advanced algorithmic techniques for case correlation, such as those described in [7], can be applied. Nevertheless, knowledge about the analyzed process is still required to choose an algorithm with appropriate properties, such as support for loops or parallelisms inside cases. The output of this step is an event log suitable for PM.

2.6 Process Mining

The final step of the pipeline addresses the application of PM algorithms for process discovery or conformance checking. Usually, PM techniques are designed for structured processes with a limited set of process variants [6], and may not provide satisfying results when directly applied to event logs of unstructured processes. For instance, the application of process discovery algorithms on event logs of unstructured processes will commonly result in either highly overgeneralized or complex process models, which allow little insights into the process.

To address this issue, we divide the event log of an unstructured process into multiple, more structured sub-logs of similar process variants using trace clustering [31]. For trace clustering approaches, it is essential to define features related to the event log, which are then used to cluster the event log of interest. The way the features are defined and selected significantly impacts how the event log is split into clusters. Trace clustering divides an event log into multiple independent sub-logs, and PM techniques are then applied to each sub-log separately, enabling more accurate capturing of the underlying structure of an unstructured process, and providing more valuable insights than an unclustered log.

3 Implementation

We implemented a framework that consists of modular components, each corresponding to one step of the analytics pipeline as described in Sect. 2. For each module, users can select an appropriate method out of a set of methods. These methods are implemented in a generalized way and are not limited to a specific use case. Our implementation is available on GitHub[1]. In the following, the implemented methods are described for each module.

Dataset Preparation. In order to combine video sequences split into multiple files, rescale video resolution and resample video frame rate, we use FFmpeg[2]. To filter out areas of content not relevant to the analysis, we provide an option for using a static mask.

Object Tracking. We integrated interfaces for YOLOv7 [28] and Detectron2 [30] as object detectors. ByteTrack [33] and OC-SORT [4] were implemented as detection-based trackers. Labelme [27] is used to create training datasets for object detection. In addition, we implemented filters to address common issues in object detection such as low confidence, strongly overlapping, and implausibly small bounding boxes.

Activity Recognition. We employ MMAction2 [21] and SlowFast [8] for spatio-temporal action detection. To allow integration of custom training datasets, we added a feature in a labeling tool [2] that allows transferring training datasets into the format of the Atomic Visual Actions (AVA) dataset [9].

Event Abstraction. For event abstraction, we implemented the temporal aggregation technique as described in Sect. 2.4.

Case Correlation. We offer three options for case correlation: (1) tracking ID-based correlation to construct one case for each object for a complete video sequence (e.g., a day), (2) temporal segmentation-based correlation to construct multiple cases for each object based on the time of day, and (3) correlation based on pre-defined start/end activities (see Sect. 2.5). In the third option, further filters can be applied, for example, to require multiple repeats of the end activity before a case terminates or define the length of a start activity instance. If required, the interfaces of the Case Correlation module also support the application of advanced case correlation techniques.

Process Mining. We implemented multiple case-level features, which can be extracted from an event log, and used as input for trace clustering using the algorithms provided by scikit-learn [23]. In particular, we use a combination of standard scaling, principal component analysis (PCA), and k-means clustering. We use the heuristic miner [29] and inductive miner infrequent [15] algorithms for process discovery and token-based replay for conformance checking, both provided by pm4py [3]. Moreover, we also support the export of event logs for use by external PM tools such as Disco[3].

[1] https://github.com/arvidle/video-process-mining-public.
[2] https://ffmpeg.org/.
[3] https://fluxicon.com/disco/.

4 Use Case

We applied the analytics pipeline to a real-world video dataset of surveillance recordings from a conventional pig farming environment. The benefits of the use case are as follows: (1) the behavior of pigs is limited to a few activities, which significantly simplifies activity recognition compared to recognition of human activities in smart homes or smart factories. (2) Pig behavior is well-researched, and the knowledge on pig behavior can be used to verify the analysis results. Finally, (3) no privacy concerns need to be considered. In the future, however, we plan to transfer the pipeline to more complex use cases.

The behavior of fattening pigs is generally limited to resting (lying, sitting, and standing), locomotion (moving, and investigating their surroundings), feeding/drinking, defecating, and playing with toys [35]. Fattening pigs in particular spend between 60–85% of the day lying [35]. Pigs autonomously divide their pens into functional areas associated with specific activities [22]. For instance, defecation is typically done in a small (partially) sheltered area, e.g., near walls or in a corner, which is located opposite to the feeding area.

Previously, video-based research of pig behavior was done manually, i.e., a person observed the behavior of pigs. The goal of our data analysis was to *automatically* monitor common behavioral patterns and evaluate the division of the pigpens into functional areas.

We recorded video material of a pigpen with eleven pigs at a resolution of 1920×1080 pixels and 18.75 FPS from 6:00 a.m. to 6:00 p.m. over a period of four weeks. To reduce processing times, the video files of five consecutive days were sampled from the complete dataset. For each of those days, the full recordings were selected as relevant, to capture behaviors throughout the whole day, and the video files were consolidated into one preprocessed video for each day. For this particular use case and analysis goal, video resolution was reduced to 854×480 pixels, significantly reducing processing times while keeping sufficient visual detail.

Although numerous approaches exist to extract behavioral information from surveillance video of pigpens [5], these approaches are limited to the detection of isolated activities and not a process composed of several activities. Therefore, our approach has the novel potential to explain the influences and causalities of activities in this context.

5 Evaluation

This section summarizes the evaluation of all steps from *Event Abstraction* to *Process Mining* for our use case. After detecting activities in the selected videos, we used the procedure shown in Fig. 3 to evaluate our approach. To validate the result stability, we applied conformance checking, while the meaningfulness of the extracted event logs and process models was confirmed by discussing the results of our analysis with domain experts. By synthesizing the findings from these separate evaluations, we can show that our approach is able to extract meaningful, PM-compliant event logs.

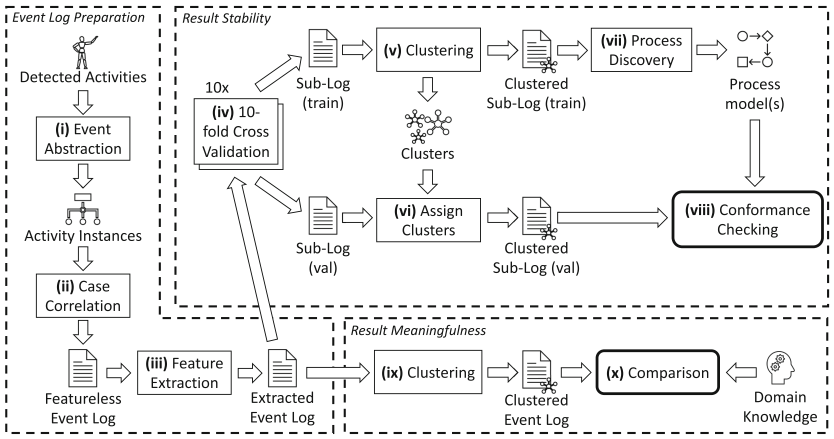


Fig. 3. Procedure to evaluate our analytics pipeline.

5.1 Pre-Processing

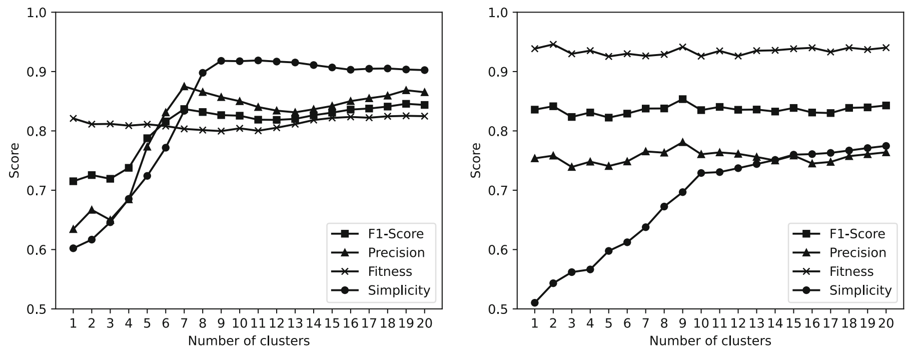
First, activities needed to be detected in the selected videos by conducting the first three steps of the analytics pipeline. A training dataset for object detection was created by sampling a total of 614 frames from the complete dataset, and pigs were labeled with bounding boxes in each frame. This training dataset was used to train a pre-trained YOLOv7 model, which was used in combination with the ByteTrack algorithm to track the pigs in the selected videos. Then, the relevant activities requiring labeling for *Activity Recognition* were determined by interviewing a domain expert. This included the following eight activities: *lying*, *sitting*, *standing*, *moving*, *investigating*, *feeding*, *defecating* and *playing*. Behavior that could not be described by one of these activity classes was categorized to the class *other*. A total of 9240 of these activities were sampled from the complete dataset and manually annotated, and then used to train a pre-trained SlowFast 4×16 model, which was used to detect the pigs' activities in the selected videos.

Preparing the labeled training datasets for the object detection and spatio-temporal action detection models proved to be the most labor-intensive manual tasks required to instantiate the analytics pipeline. Additionally, applying the trained deep learning models to the selected videos was the most computationally-intensive task in the analysis. On average, the object detector recognized 10.6 of 11 pigs per frame, and the pigs were tracked for 18 min before being lost by the tracker. The SlowFast model reported a validation mean average precision (mAP) of 0.7365, which was considered significant to continue the analysis.

5.2 Event Log Preparation

An event log was prepared from the detected activities and used to evaluate both result stability and event log meaningfulness. The same event log was used for both steps of the evaluation.

We used a moving average window size of 20 low-level events for the noise reduction in *Event Abstraction*, which is shorter than the duration of common pig activities, but sufficiently long to reduce local noise. Then, we applied the *Case Correlation* based on specific start/end activities separately for each pig (identified by its respective tracking ID), with *lying* being both the start and end activity. Thus, the process starts with a pig standing up and ends with the pig lying back down. Feature vectors were extracted from each case consisting of activity counts, total duration per activity, case duration, and directly-follows relations. Then, we scaled the feature vectors and reduced them using PCA.



(a) Heuristic Miner (b) Inductive Miner Infrequent

Fig. 4. Aggregated conformance measures by number of clusters for two different process discovery algorithms

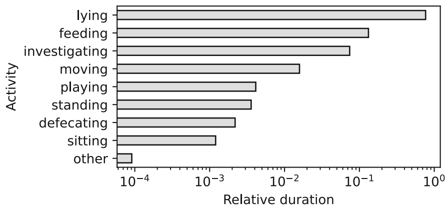
5.3 Result Stability

We assessed the stability of our approach with conformance checking. In particular, we evaluated if our approach was able to repeatably produce PM-compliant event logs. Generally, PM methods assume that all events and cases of an event log refer to the same notion of abstract tasks of the same process [1]. By partitioning the extracted event log into two distinct sub-logs (i.e., a training sub-log used for process discovery, and a validation sub-log to evaluate the discovered process models with conformance checking) conformance checking can be used to evaluate whether events with the same activity refer to the same notion of abstract steps in a process across the complete event log. If this assumption is not fulfilled, process discovery and conformance checking cannot produce reliable results. Instead, analysis results and quality measures would vary unpredictably

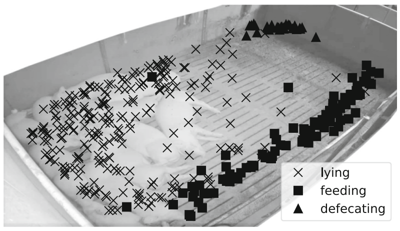
between multiple executions of the analytics pipeline on the same dataset and different partitionings of the same event log. Conversely, if quality measures are stable, it is demonstrated that the pipeline reliably extracts PM-compliant event logs.
To ensure replicability, 10-fold cross-validation was applied for splitting, which itself was repeated 10 times, resulting in a total of 100 runs (see (i) in Fig. 3). For each run, first, the cases of the training sub-log were clustered with k-means (ii). The cases of the validation sub-log were then assigned to the same clusters (iii), and a separate process model was discovered for each cluster of the training sub-log (iv). Fitness, precision, F1-score, and simplicity were then calculated for each cluster, using the cases of the validation sub-log assigned to the clusters accordingly (v). Quality measures for the complete event log were constructed by combining the measures of each cluster using a harmonic mean weighted by the number of cases in each cluster, which were then averaged over all runs. This evaluation scheme was repeated with different cluster configurations of k-means and using both the heuristic miner and inductive miner infrequent algorithms, for which the results are summarized in Fig. 4. Fitness, precision, and the F1-score were largely stable across runs with the same parameters, and their average values are acceptable considering that the analyzed process was inherently unstructured. As expected, model simplicity correlates with the number of clusters. For five or more clusters, the heuristic miner outperformed the inductive miner in F1-score, precision, and simplicity. In summary, the evaluation of result stability shows that the pipeline repeatably extracts activities and cases that are similar over the complete event log, and organizes the cases into behaviorally homogeneous clusters.

5.4 Result Meaningfulness

The approach was further evaluated by analyzing whether the extracted activities, cases, and clusters were *meaningful* with respect to the existing knowledge in the domain (of agriculture). This was evaluated by comparing the event log and process models extracted for one exemplary run of the analytics pipeline with domain knowledge (i.e., confirming the results through domain experts). We organized the resulting event log of this run into 15 clusters.
First, we analyzed the temporal and spatial distribution of the activities. The relative duration of the detected activities (see Fig. 5a) matches the expected ranges. For instance, 75% of all activities are *lying*, which is within the expected range of 60–85%, and the occurrence of *feeding* (13%) is largely similar to the analysis conducted in [2]. The spatial distribution of activities associated with the three functional areas (*lying*, *feeding* and *defecating*) is shown in Fig. 5b. The positions of these three activities reflect three largely separate clusters. *Feeding* was correctly detected among the feeding area, and *defecating* was localized in one corner of the pigpen and separated from the *lying* area. A manual inspection of the selected videos confirmed that these clusters match the actual functional areas commonly used by pigs.



(a) Relative duration of detected activities (total duration an activity is detected over total duration of all activities, logarithmic x-axis)



(b) Positions of a sample of detected *lying*, *feeding* and *defecating* activities

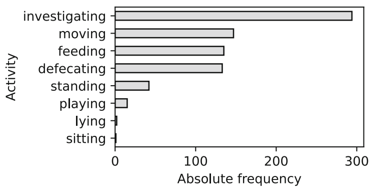
Fig. 5. Spatial and temporal distribution of detected activities

We then evaluated the cases and clusters of the event log, specifically if each cluster contained a set of similar cases of a specific behavioral pattern. The occurrences of activities in two clusters are shown in Fig. 6. Usually, two to four specific activities could define over 90% of all the events in a cluster. For instance, the cluster that was mostly composed of *moving*, *feeding* and *investigating* contained a behavioral pattern that connects these three activities. This implies that the clusters contain specific behavioral patterns. To analyze if the behavioral patterns are meaningful, a process model was discovered for each cluster using Disco. For instance, Fig. 7a shows an excerpt from the process model of a cluster mostly containing *moving*, *standing*, and *investigating*. In this pattern, pigs start with *moving* to a location, and then *investigate* their surroundings with intermittent *standing* pauses, which is indicated by a loop between these two activities. In the model of the cluster that contains 82% of all observations of pigs *defecating* (Fig. 7b), a loop exists between *defecating* and *investigating* (i.e., they are often executed in sequence). This order of activities corresponds to current domain knowledge on pig behavior, as pigs are known to typically investigate their surroundings before and after defecation.

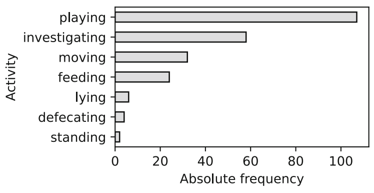
The findings of our analytics pipeline were discussed with two domain experts from agricultural science, who confirmed that the extracted event log and behavioral patterns were meaningful with respect to the knowledge in the domain.

5.5 Reproducibility and Data Availability

Due to copyright restrictions, we are unable to publish the full recorded video dataset. However, we provide the detected bounding boxes of objects, tracking information, and the recognized activities for the videos (see [18]) allowing reproducibility. This also includes the training datasets for object detection and spatio-temporal action detection as well as the trained deep learning models. Also, the implementation includes the extracted event log, scripts to prepare the event log from the recognized activities and reproduce the results, as well as all process models discovered for the evaluation of event log meaningfulness.

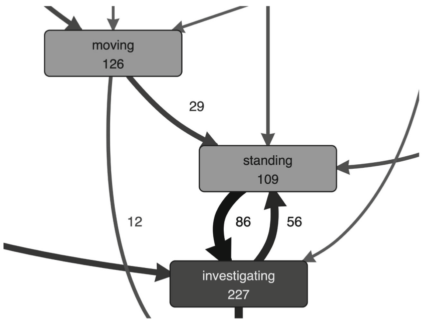


(a) Cluster 10

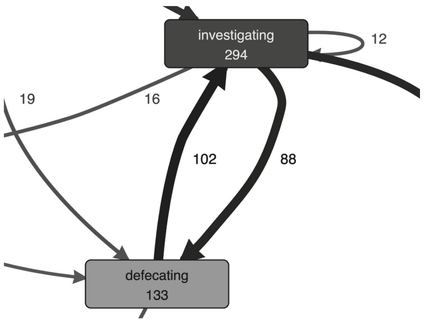


(b) Cluster 11

Fig. 6. Occurrence of activities in two clusters.



(a) Cluster 9



(b) Cluster 10

Fig. 7. Excerpts from process models for two clusters generated using Disco

6 Related Work

In recent years, there has been a growing interest in exploring unstructured data for PM. Thus, many approaches have been proposed for various types of data sources. For instance, Janssen et al. [10] proposed an approach to extract event logs from smart home/IoT motion sensor data. The log of sensor activations is divided into sequential sections, which are clustered into patterns of similar sensor activation sequences. Then, the clusters are mapped to activities, and the activities are grouped into cases by assuming that cases are started and ended by specific sensors. Process discovery is applied to the extracted event log to discover models of human daily routines. Rebmann et al. [25] presented an approach to PM using time series sourced from motion sensors worn by workers. The time series are segmented into fixed-width sub-series, and supervised activity recognition is performed for each sub-series. If an activity cannot be classified from just sensor data, image data is used for disambiguation. An event log is constructed from the recognized activities and used for process discovery. Koschmider et al. [13] proposed an abstract method to extract event logs from general time series data. Similarly to Janssen et al. [10], time series are split into sub-series, similar sub-series are identified using clustering, and the resulting clusters are then mapped to process activities.

In contrast, research on PM specifically using video data is still at an early stage, with the few identified approaches published very recently (i.e., since 2020). Lepsien et al. [17] designed an abstract pipeline for PM on video data. The pipeline outlines the steps from dataset preparation to event log construction and the application of PM to this event log. The steps of the pipeline include the extraction of relevant video data, object and activity recognition, and process discovery. The pipeline also includes the explicit extraction of domain-specific knowledge and a refinement step. The contribution is limited to the abstract approach, and neither an implementation nor evaluation of the steps after pre-processing is provided. Knoch et al. [11] presented an unsupervised approach to process discovery from video recordings of manual assembly tasks. Process activities are recognized in the video recordings from overhead cameras mounted at specially designed assembly workstations by (1) tracking workers' hands throughout videos, (2) clustering hand trajectories and (3) assigning the trajectory clusters to work steps using the location of assembly parts on the workstation. As the clusters are directly assigned to pre-defined work steps, this approach is unable to discover activities not known a priori. Kratsch et al. [14] presented a reference architecture for PM on video data, outlining the steps from raw video data to event logs. They described a selection of different computer vision techniques that can be applied to extract information from video data, and provided guidance on choosing appropriate techniques for specific video PM use cases. The correlation of activity instances to cases is not included in the architecture but needs to be done externally. Further, the authors point out the limitation that their prototypical implementation and evaluation took place in a rather structured process context, and evaluation of more complex (i.e., less structured and more chaotic) processes would be beneficial.

7 Conclusion

In this paper, we presented a pipeline for extracting PM-compliant event logs and process models from unstructured video data. Structure is imposed into this unstructured data step by step, by extracting low-level events from the preprocessed video dataset using spatio-temporal action detection, and raising these events to a higher abstraction level using event abstraction, case correlation, trace clustering, and finally process discovery. We demonstrated the efficiency of our approach by implementing a modular framework that can easily be configured for application on video datasets from different domains and applying the implementation to surveillance footage of fattening pigs. The evaluation indicates that our approach extracts meaningful event logs in a reproducible manner. A review of related literature shows that, to the best of our knowledge, our analytics pipeline is the first fully validated approach that addresses the challenging task of analyzing an unstructured process through unstructured video data.

While evaluating our approach, we identified several limitations that need to be addressed in the future. Firstly, the requirement of supervised activity recognition to pre-define the activities to be detected may hinder the analysis of processes where limited information is available before the analysis. This could be addressed by integrating unsupervised activity recognition into the pipeline, which would also require advanced event abstraction algorithms capable of conceptually abstracting low-level to high-level activities [32]. Secondly, the evaluation confirmed that the pipeline can successfully be instantiated for a real-world application and produces stable and meaningful results, but the benefits added for end users were not quantified. We plan to evaluate the benefits added with user studies. Thirdly, expertise in deep learning is required to prepare the models required for *Object Tracking* and *Activity Recognition*. In the future, we plan to provide an interface to simplify the labeling of datasets and training of the deep learning models. Fourthly, while the evaluation confirmed the general efficiency and validity of our approach, the evaluation on a single use case does not enable conclusions about the generalization of our approach. To address this, a reference dataset and evaluation scheme would be beneficial. However, compiling a dataset that is large and heterogeneous enough for this purpose is highly time-consuming. This could be solved by synthetic evaluation data, which has already been proposed for other types of unstructured data in PM [34]. Fifthly, the current implementation is limited to settings where all process activities are visible from a single camera perspective (i.e., are executed in a single, constrained area) and actors can be tracked without interruption. A solution addressing this limitation would require the implementation of multi-camera tracking or object re-identification [14] to handle actors moving between areas observed by different camera perspectives, and improved event abstraction and case correlation techniques. Finally, the current methods implemented for *Case Correlation* limit the approach to actor-centric processes (objects performing activities). Implementing advanced case correlation methods would enable the analysis of processes that can be characterized with the more general notion of subjects performing activities on objects. Further possibilities to extend the applicability of the pipeline include privacy-preserving analysis techniques to address regulatory limitations [14,20], and techniques to propagate uncertainty (e.g., from *Activity Recognition*) through the pipeline (e.g., [16,24]) to quantify the result confidence.

In conclusion, our work provides a promising approach to integrating unstructured data sources into PM pipelines.

References

1. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_19
2. Bergamini, L., et al.: Extracting accurate long-term behavior changes from a large pig dataset. In: VISAPP 2021, vol. 4, pp. 524–533. SCITEPRESS, Online Streaming (2021). https://doi.org/10.5220/0010288405240533

3. Berti, A., van Zelst, S.J., van der Aalst, W.: Process Mining for Python (PM4Py): bridging the gap between process- and data science (2019). https://doi.org/10.48550/arXiv.1905.06169
4. Cao, J., Weng, X., Khirodkar, R., Pang, J., Kitani, K.: Observation-centric sort: rethinking sort for robust multi-object tracking. arXiv preprint (2022). https://doi.org/10.48550/arXiv.2203.14360
5. Chen, C., Zhu, W., Norton, T.: Behaviour recognition of pigs and cattle: journey from computer vision to deep learning. Comput. Electron. Agric. **187**, 106255 (2021). https://doi.org/10.1016/j.compag.2021.106255
6. Diamantini, C., Genga, L., Potena, D.: Behavioral process mining for unstructured processes. J. Intell. Inf. Syst. **47**(1), 5–32 (2016). https://doi.org/10.1007/s10844-016-0394-7
7. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. WIREs Data Min. Knowl. Discov. **10**(3), e1346 (2020). https://doi.org/10.1002/widm.1346
8. Feichtenhofer, C., Fan, H., Malik, J., He, K.: SlowFast networks for video recognition. In: ICCV 2019, pp. 6201–6210. Seoul, Korea (South) (2019). https://doi.org/10.1109/ICCV.2019.00630
9. Gu, C., et al.: AVA: a video dataset of spatio-temporally localized atomic visual actions. In: CVPR 2018, pp. 6047–6056 (2018). https://doi.org/10.1109/CVPR.2018.00633
10. Janssen, D., Mannhardt, F., Koschmider, A., van Zelst, S.J.: Process model discovery from sensor event data. In: Leemans, S., Leopold, H. (eds.) ICPM 2020. LNBIP, vol. 406, pp. 69–81. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72693-5_6
11. Knoch, S., Ponpathirkoottam, S., Schwartz, T.: Video-to-Model: unsupervised trace extraction from videos for process discovery and conformance checking in manual assembly. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 291–308. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_17
12. Koschmider, A., Mannhardt, F., Heuser, T.: On the contextualization of event-activity mappings. In: Daniel, F., Sheng, Q.Z., Motahari, H. (eds.) BPM 2018. LNBIP, vol. 342, pp. 445–457. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11641-5_35
13. Koschmider, A., Oppelt, N., Hundsdörfer, M.: Confidence-driven communication of process mining on time series. Informatik Spektrum **45**(4), 223–228 (2022). https://doi.org/10.1007/s00287-022-01470-3
14. Kratsch, W., König, F., Röglinger, M.: Shedding light on blind spots - developing a reference architecture to leverage video data for process mining. Decis. Support Syst. **158**, 113794 (2022). https://doi.org/10.1016/j.dss.2022.113794
15. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 66–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
16. Lepsien, A.: Quantifying uncertainty for explainable process mining. In: Proceedings of the 13th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2023). CEUR Workshop Proceedings, vol. 3397. CEUR-WS.org, Stockholm, Sweden (May 2023). https://ceur-ws.org/Vol-3397/
17. Lepsien, A., Bosselmann, J., Melfsen, A., Koschmider, A.: Process mining on video data. In: Manner, J., Lübke, D., Haarmann, S., Kolb, S., Herzberg, N., Kopp, O.

(eds.) ZEUS 2022. CEUR Workshop Proceedings, vol. 3113, pp. 56–62. CEUR-WS.org, Bamberg, Germany (2022). https://ceur-ws.org/Vol-3113/paper9.pdf
18. Lepsien, A., Koschmider, A., Kratsch, W.: Video process mining evaluation data (2023). https://doi.org/10.5281/zenodo.7763839
19. Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Kim, T.K.: Multiple object tracking: a literature review. Artif. Intell. **293**, 103448 (2021). https://doi.org/10.1016/j.artint.2020.103448
20. Mannhardt, F., Koschmider, A., Baracaldo, N., Weidlich, M., Michael, J.: Privacy-preserving process mining. Bus. Inf. Syst. Eng. **61**(5), 595–614 (2019). https://doi.org/10.1007/s12599-019-00613-3
21. MMAction2 Contributors: openmmlab's next generation video understanding toolbox oand benchmark. https://github.com/open-mmlab/mmaction2 (2020)
22. Nannoni, E., Aarnink, A.J.A., Vermeer, H.M., Reimert, I., Fels, M., Bracke, M.B.M.: Soiling of Pig Pens: a review of eliminative behaviour. Animals **10**(11), 2025 (2020). https://doi.org/10.3390/ani10112025
23. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
24. Pegoraro, M., Uysal, M.S., van der Aalst, W.M.P.: Efficient construction of behavior graphs for uncertain event data. In: Abramowicz, W., Klein, G. (eds.) BIS 2020. LNBIP, vol. 389, pp. 76–88. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53337-3_6
25. Rebmann, A., Emrich, A., Fettke, P.: Enabling the discovery of manual processes using a multi-modal activity recognition approach. In: Di Francescomarino, C., Dijkman, R., Zdun, U. (eds.) BPM 2019. LNBIP, vol. 362, pp. 130–141. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-37453-2_12
26. Shermin, T., Teng, S.W., Murshed, M., Lu, G., Sohel, F., Paul, M.: Enhanced transfer learning with imagenet trained classification layer. In: Lee, C., Su, Z., Sugimoto, A. (eds.) PSIVT 2019. LNCS, vol. 11854, pp. 142–155. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34879-3_12
27. Wada, K.: LabelMe: image polygonal annotation with Python (2023). https://doi.org/10.5281/zenodo.5711226, https://github.com/wkentaro/labelme
28. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors (2022). https://doi.org/10.48550/arXiv.2207.02696
29. Weijters, A., van Der Aalst, W.M., De Medeiros, A.A.: Process mining with the heuristics miner-algorithm. TU Eindhoven, Tech. Rep. WP 166(July 2017), 1–34 (2006)
30. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
31. Zandkarimi, F., Rehse, J.R., Soudmand, P., Hoehle, H.: A generic framework for trace clustering in process mining. In: ICPM 2020, pp. 177–184. IEEE, Padua, Italy (2020). https://doi.org/10.1109/ICPM49681.2020.00034
32. van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A.: Event abstraction in process mining: literature review and taxonomy. Granular Comput. **6**(3), 719–736 (2020). https://doi.org/10.1007/s41066-020-00226-2
33. Zhang, Y., et al.: ByteTrack: multi-object tracking by associating every detection box. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022, pp. 1–21. LNCS, Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20047-2_1

34. Zisgen, Y., Janssen, D., Koschmider, A.: Generating synthetic sensor event logs for process mining. In: De Weerdt, J., Polyvyanyy, A. (eds.) CAiSE Forum 2022. LNBIP, vol. 452, pp. 130–137. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07481-3_15
35. Zoric, M., Johansson, S.E., Wallgren, P.: Behaviour of fattening pigs fed with liquid feed and dry feed. Porc. Health Manag. **1**(1), 14 (2015). https://doi.org/10.1186/s40813-015-0009-7