2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Detecting spatio-temporal Relations by Combining a Semantic Map with a Stream Processing Engine

Lennart Niecksch[1], Henning Deeken[2], Thomas Wiemann[1,2]

***Abstract*— Changes in topological spatial relations of objects are often strong indicators for state transitions in the underlying processes they are involved in. While various aspects of semantic mapping have been extensively researched, the reasoning about the temporal development of spatial relations of instances is often neglected. This paper presents a concept to combine a semantic map with a stream processing framework for live analysis of the spatio-temporal relation of objects, based on the map and information inferred from sensors streams. To demonstrate the functionality of our concept, we implemented a proof-of-concept system to track everyday events in an office environment. The presented application scenario clearly demonstrates the benefits of the proposed architecture for detecting and handling complex spatio-temporal events.**

I. INTRODUCTION

Knowledge about spatial relations between objects and actors is required for many applications in artificial intelligence and robotics. Approaches that combine geometric, topological and symbolic information for reasoning about the current state of the environment based on known concepts are critical for semantic mapping [1].

Semantic mapping algorithms extract semantic information from sensor data and store it together with metric and topological information in a representation that provides means to retrieve the gathered knowledge. For many tasks in robotics, information needs to be available timely, but extensive environment analysis can often not be achieved in a live system. Some tasks cannot be solved without all high level information present, but often enough a less accurate, but timely available analysis is desirable to detect relevant changes. In this paper, we propose an architecture to interweave a semantic mapping framework with a stream processing engine to analyze the temporal development of topological spatial relations. The system generates and analyzes spatio-temporal event logs based on information from a semantic map and observations from live sensor data.

The architecture is inspired by the *Lambda Architecture* [2] common in the *Big Data* domain. The aim is to reduce the gap between extensive and slow batch analysis systems and the need for real-time analytics, by combining batch processing analysis with a stream processing component, which provides less accurate but faster analysis, based on new input and previously available information. By utilizing a *Complex Event Processing (CEP)* [3] library, we detect spatio-temporal changes and infer additional information about their relation in near real time based on predefined patterns, the relation of detections in sensor streams over time and information from the map through spatial queries. This work is a first step towards a lambda architecture in a semantic mapping framework. It provides a compromise between the high computational costs of full spatio-semantic analysis and the need of updates with high frequency.

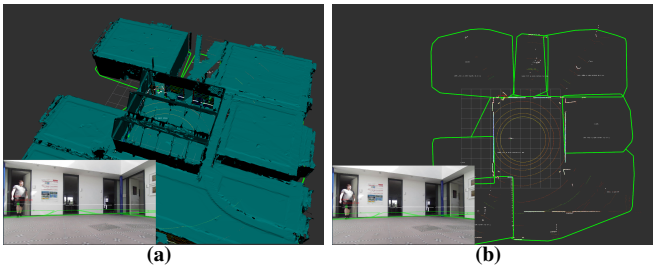

Fig. 1: Different views and representations of the same scene in which a person that left a room. In addition to the sensor data, the left image shows a mesh representing the room's geometries. The right image shows the footprints generated by the semantic mapping framework. The markers show the position of the tracked person and the recently detected event.

In this paper, we present a concept that links a stream processing framework with a semantic map to generate semantic events, which signal state transitions in a live system. For that, we bridge ROS to the Apache Flink stream processing system through a Kafka message broker. In this stream processing system we a pipeline to infer relevant spatial relations from the sensor data stream via typical classifiers and spatial stream processing. Finally, we use CEP on the resulting event streams to monitor and analyze the spatio-temporal development of the observed objects over time. We demonstrate the feasibility of our approach by a proof-of-concept pipeline for a typical indoor scenario, where a robot observes doors opening and closing and people leaving and entering rooms.

II. RELATED WORK

Semantic maps have a long history in autonomous mobile robotics. Kostavelis et al. [4] provide an overview on semantic mapping in different use-cases including task planning [5] and plan execution and monitoring [6]. In recent years, extensive research has been done in the field of semantic SLAM. Nüchter et al. [1] build semantic maps by registering

[1]Lennart Niecksch and Thomas Wiemann are with the German Research Center for Artificial Intelligence (DFKI), DFKI Niedersachsen, Plan-based Robot Control Group, Osnabrück, Germany firstname.lastname@dfki.de

[2]Henning Deeken and Thomas Wiemann are with the Knowledge Based Systems and Autonomous Robotics groups at the Institute of Computer Science, Osnabrück University, Osnabrück, Germany firstname.lastname@uni-osnabrueck.de

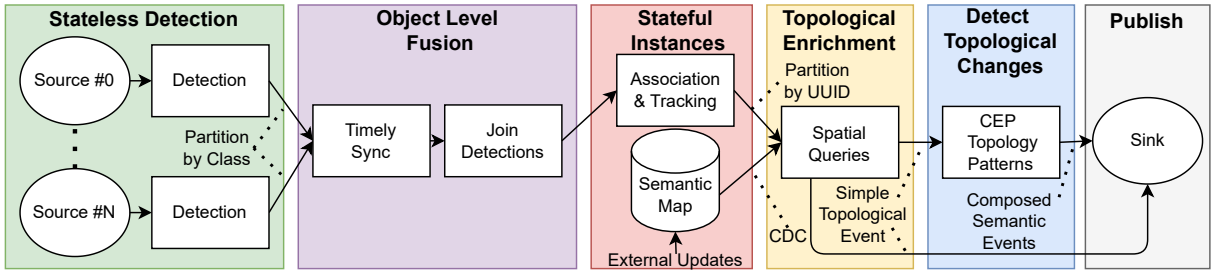

Fig. 2: The steps in our concept to analyze the temporal development of topological spatial relations of entities present in our map and sensor based perceptions.

3D laser scans, extracting surfaces, labeling them by their orientation and detecting known objects in 2D renderings. Günther et al. [7] extract semantic information from a RGB-D sensor by surface reconstruction and extraction of planar features. Object detection is done by applying SWRL rules on the extracted features. Semantic Fusion [8] combines Convolutional Neural Networks for semantic labeling with a SLAM system to fuse multiple viewpoints from an RGB-D camera into a map in real time. Kimera [9] provides a library for real-time metric-semantic mapping with stereo cameras. Instance level semantic mapping across multiple frames in static environments is covered in [10], [11], [12]. Runz et al. [13], [14] are explicitly focusing on mapping dynamic objects.

Another aspect of semantic mapping is to derive and represent spatio-temporal and topological information about the mapped environment. SEMAP by Deeken et al. [15] grounds concepts and instances in a spatial database and implements operators to reason about relations and topology between objects. SEMAP itself is agnostic to the origin of the perceptions. Rosinol et al. [16] build multilayer 3D dynamic scene graphs for actionable spatial perception based on [9]. In contrast to SEMAP it maintains a history of object poses in a pose graph, but to our knowledge this is not utilized to reason about changes in topological spatial relations. Such spatial relations are exploited in [17], were SEMAP was used to infer high level information about agricultural processes by constantly updating and querying the system externally. The resulting event log was then analyzed offline in an external tool. Machine telemetry typically already has a strong semantic. This is different in the context of autonomous robotics, where such information typically has to be inferred from sensor data.

Tenorth and Beetz [18] implemented reasoning about spatio-temporal change by storing multiple perceptions over time as a special case. In DyKnow [19], Heintz et al. describe an approach for temporal stream reasoning using metric temporal logic by focusing on the stream-like nature of sensor data. It was extended to allow C-SPARQL [20] queries [19] and spatial reasoning based on different region connection calculi [21]. Unfortunately, to our knowledge DyKnow is not freely available.

In recent years, several open source distributed stream processing frameworks have emerged. They focus on real time streaming analysis of high volume unbounded data streams and offer support for pipelined stream transformations. Such transformations are often represented as a Directed Acyclic Graph (DAG). They provide certain guarantees like at-least/exactly-once processing, fault tolerance and managed states [22]. Apache Spark [23], [24] and Apache Flink [25] offer an high level declarative API, while Apache Samza [26] and Apache Heron [27] provide low level APIs for DAG definition. These frameworks also provide additional tools, e.g., for machine-learning [28], graph analysis or CEP [25].

Based on the evaluation of [29] *Apache Flink* is a good fit for the application in robotics, mostly due to low latency, processing and state guarantees and its support for CEP [25] for high-level analysis of temporal relation of events. In this paper, we propose an architecture to interweave SEMAP with Flink to analyze the temporal development of topological and spatial relations based on the information being present in the map and live sensor data from a robot. We show using a *CEP* library, we are able to detect spatio-temporal changes, and infer additional information using defined patterns about the relation of these changes in near real time.

III. Concept

Our idea is based on the concepts of SEMAP, but instead of querying the map externally, we apply a pipeline implemented in Flink on the input streams to continuously analyze the relation between detected objects, while still using the abstractions derived from the geometric aspect of the map. The concept similar to the Waterfall Model [30] closely related to the JDL Data Fusion Model [31]. For object detection, stateless stream operators are used, which transform input data streams to derive information based on pre-defined models. The output streams are synchronized timely and detections from different sensors are fused. These fused detections are then aggregated and processed in stateful streaming operators. In these operators, observation-to-track association is done and the state of tracks is predicted and estimated. We propose to treat the map updates of the semantic map as input streams as well. This allows to do spatio-temporal joins of the updated tracks with the relevant objects from the map, while maintaining the same grounding of the underlying concepts. This way, the current topological spatial relations of objects are inferred. Partitioning and querying of the streams is possible with object identifiers,

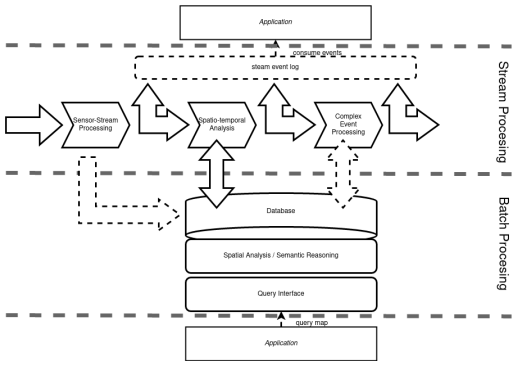

Fig. 3: Interaction of the stream processing component and the SEMAP framework.

which map to concrete instances of defined concepts. To monitor the temporal development of said relations, we use CEP to detect state transitions. The proposed concepts are visualized in Fig. 2.

IV. Implementation & System Architecture

A. Apache Flink

Flink programs are structured as a directed acyclic graph (DAG). Stateful operators transform and produce data streams. Operators are parallelized into subtasks based on stream partitions and state is scoped to the partition [25]. The execution is possibly distributed, and the data exchange can be parametrized for latency or throughput. Flink differentiates between event and processing time semantics. Temporal progress in streams is measured through watermarks, which are generated on event timestamps. Features like windowing, aggregation, and the internal operator state are based on these watermarks. Flink's *DataStream API* offers a set of predefined operators like *map, flatmap and window*. Object detection could be implemented as a flatmap operator producing an output stream containing *0-n* detections from *1* image in the input stream. Flink also provides stateful versions of said operators. The API also allows to access low level functionality like timers, event timestamps, state and the key of the partition through its *ProcessFunction* interface. Additionally it provides libraries like FlinkCEP and a high level structured and declarative *Table & SQL API* based on the concept of dynamic tables, which are build on top of the low level *Streaming API*.

B. Connection to ROS

For robotic applications communication with the Robot Operating System (ROS) is desirable. In Flink, data sources and sinks are typically made available through connectors. We opted for bridging ROS and Flink through a message broker (Apache Kafka) as shown in Fig. 4. We chose Kafka because of its excellent integration into Flink, its high throughput capabilities and low latency. Our bridge is based on *librdkafka*, which maps ROS topics to Kafka topics, allowing to use default ROS serialization and custom conversions. This broker, while adding an overhead, allows to easily exchange message between both systems, simplifies the implementation and benefits from the good integration of back pressure handling in Flink.

op	id	name	alias	frame_id	rel_desc_id	abs_desc_id
+I	1	object1	hallway	2	1	29

(a) An excerpt from the instance table showing the hallway instance how it is equally represented in the Flink application and SEMAP.

op	type	geometry	abstraction_desc
+I	FootprintHull	POLYGON ((...	1
+I	FootprintHull	POLYGON ((...	29

(b) The 2D convex hull abstractions (relative and absolute) selected from the geometry model table which are referenced by the hallway instance.

TABLE I: An example of *CDC* tables in Flink mirroring the equivalents in SEMAP.

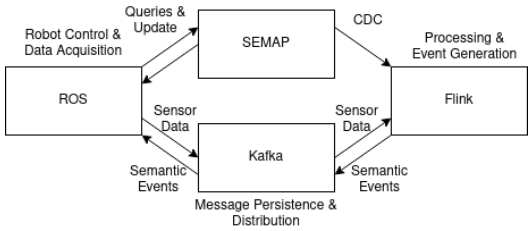

Fig. 4: Current data flow between the different components in our architecture.

C. SEMAP Integration

SEMAP is an instance-based framework using a PostGIS database. It provides different levels of geometric abstractions for objects, operators to reason about spatial relations and an interface to ROS for data management and querying. An environment map in SEMAP consists of triangle meshes representing the geometry of objects.

In our work, we monitor SEMAPs internal instance table for changes using a *Change Data Capture (CDC)* framework. The *Flink-cdc-connector* is used to stream changes of the database into the stream processing system, where parts of SEMAPs internal schema are mirrored. This way, the object instances and their absolute geometric abstractions (see Fig. 1) are available in the stream processing system using the same representation. Therefore, the streaming queries done with Flink's Table and SQL API as shown in Fig. 3 are still referring to the same concepts and entities, which are present in the map as shown in Tab. I.

```
// SQL API
// Select only 2D convex hulls
Table footprints = tableEnv.sqlQuery(
"SELECT" +
    "type," +
    "ST_geomFromWKB(geometry) as geometry," +
    "abstraction_desc" +
"FROM geometries" +
"WHERE type='FootprintHull'");

// Table API
// Get only absolute footprints for instances
Table abstractions = footprints.
            join(instances).
                where($("absolute_description_id").
                    isEqual($("abstraction_desc")))
                    .
            select($("*"));
```

Listing 1: Selecting the 2D convex hull abstractions and join the result with the instance table on the condition that the geometry is absolute.

D. Spatial Stream Processing

For spatio-semantic analysis of the streaming data, we mirror SEMAPs functionality in the streaming pipeline to a certain degree. For that, our system performs selected spatial queries over streams. The continuous spatial queries implemented in our pipeline are based on Apache Incubator Sedona (formerly known as GeoSpark [32], [33]), which extends Flink's SQL API with constructors, functions and predicates. We extended Sedona with constructors to efficiently convert the binary CDC input to spatial data types and spatial aggregators to collect partial results. An example can be seen in the Listing 1.

V. Proof of Concept

To demonstrate the feasibility of the proposed concept and the functionality of our implementation, we tested our approach in a robotic indoor scenario. The robot is equipped with a Velodyne VLP-16, a wide-angle Logitech webcam and a SICK Tim LiDAR. The goal is to detect persons within the field of view of the sensors that leave or enter rooms. In addition, it tracks when doors open or close to generate semantic events. In the following sections we describe our concrete implementation and the algorithms used to detect and track persons, and how the streams are successively semantically enriched. Best to our knowledge, there is no baseline for comparison, so we also provide the data sets used for performance evaluation.

```
Table tracksLocated = tableEnv.sqlQuery(
"SELECT" +
    "r.name as room," +
    "t.name as track_id," +
    "t.pointGeom," +
    "t.rowtime AS rowtime " +
"FROM trackTable as t, " +
    "LATERAL (SELECT * FROM roomTable) AS r" +
"WHERE "
    ST_Contains(r.polyGeom, t.pointGeom)");

// Resulting schema
(
  `room` STRING NOT NULL,
  `track_id` STRING,
  `pointGeom` RAW('org.locationtech.jts.geom.
      Geometry', '...'),
  `rowtime` TIMESTAMP(3) *ROWTIME*
)
```

Listing 2: Joining the track table with the intermediate table of rooms with the previously intersected 2d convex hull based on the containment predicate.

A. Detection & Tracking Pipeline

Image based detection is done in stateless operators with the YOLOv4-Tiny [34] model, which was pre-trained on the COCO [35] data set. It was chosen for its well known low inference time. Object detections are tracked across multiple sensor frames by an operator using a combination of Kalman filters and Hungarian method [36] with the Jaccard index as cost. For laser scan based detection, we use the 2D SICK Tim LiDAR and apply the clustering and random forest classification presented in [37]. Different operators are implemented for the synchronization of the detection streams and the synchronization with a stream containing the relevant *tf* information. Up to two clusters are matched to an image object detection based on the containment in the bounding box after projecting them on the image plane. The matched clusters are tracked in an operator by assignment through perfect matching and a constant velocity Kalman filter. The produced stream contains the tuples of the unique identifier of a track and its 2D geometric position. The position is transformed in the world frame and the stream is interpreted as a streaming table with each event being an upsert.

```
Pattern<Tuple2<String, String>, ?>
    openedAndEntered = Pattern.
    <Tuple2<String, String>>begin("Opened").
    where(new SimpleCondition<Tuple2<String, String
        >>(){
            @Override
            public boolean filter(Tuple2<String,
                String> value) {
                return value.f1.equals("Opened");
            }
    }).next("Entered").
    within(Time.seconds(6)).where(new
        SimpleCondition<Tuple2<String, String>>() {
            @Override
            public boolean filter(Tuple2<String,
                String> value) {
                return value.f1.equals("Person
                    entered");
            }
    });
```

Listing 3: Simple pattern withe the sequence of the door of a room being opened and someone entering the same room within 6 seconds.

B. Semantic enrichment, change detection and temporal relation

We show two different ways to capture spatial changes in Flink data streams. One operator relies directly on the exact geometric transition. The other one captures the transition on a topological and semantic level. To detect door events, lines are extracted from the laser scans, which are converted to a streaming table. These are joined with *ST_Contains* predicate via room id and a bounding box of the door's possible range of motion. Every line that fulfills the predicate, is enriched with the semantic information that it is the door of a certain room instance. Subsequently, another operator compares the angle of the line with the reference of a closed door. Its internal state is per door and depends on a threshold for the angle. Events are produced whenever the internal state changes.

We first select all 2D convex hull abstractions from the tables streamed from SEMAP (see Listing 1). The hallway's footprint abstraction is intersected with each room, resulting in a better approximation of its concave geometry. In the next step the tracking stream is joined with this table based on the *ST_Contains* spatial predicate (see Listing 2) using the declarative SQL API and the Sedona extensions. The resulting table is converted to a data stream containing tuples of the track and room identifier it is localized in, thus containing the topological relation between room and track. To detect when of people leave or enter rooms, the stream is partitioned by the track identifiers and simple CEP patterns

for the transition from room to hallway and vice versa, producing an event tuple when matched, which contains the same entity identifier of the rooms in the map.

For the inference of spatio-temporal relations of events, a union data stream of the topological spatial event streams is created. The identifier of these events associates them unambiguously with spatio-semantic entities. This allows to define CEP patterns capturing predefined relations based on the order of occurrence within temporal constraints. Listing 3 shows such a simple pattern. One limitation in the current implementation is the hard temporal threshold and the fixed sequence of events - the pattern obviously generates wrong results if the door would be opened by someone else. Another limitation is that, e.g., both the patterns for entering an leaving have to be manually defined.

C. Watermarks and Flink & Kafka Configuration

For all data sources, we tightly generate watermarks monotonously increasing timestamps for every incoming record with a larger timestamp then the current watermark. This is feasible, since we process data from one system, so records appear in order for one data source. Currently, everything is computed locally, so the lowest latency is achieved with Flink's internal buffering disabled when partitioning streams. For the same reason, buffering is currently also disabled in both the ROS producers and Flink consumers and the broker immediately flushes messages. Processing timers and idleness configurations in the watermark generators may be used for an upper bound for the delay introduced from a source or an operator, but these result in non-deterministic execution and will practically disable the processing and state guarantees.

VI. Evaluation

A. Datasets

Because of the lack of a fitting dataset for evaluation, we provide a simple one to demonstrate and evaluate our approach. The robot is placed in the hallway of an office building, observing the entries of the rooms. The recorded bagfile is 93 s long. Image data is recorded with 30 Hz, 2D laser scans with 15 Hz and 3D laser scans with 10 Hz. The robot is localized in the map providing the *tf* information. We labeled the events of interest which happened in the field of view of the robot's camera. The reference sensor for timely annotation of events is the 2D laser scanner and an event was labeled, when the transition was complete, e.g., when all points belonging to the person in the 2D laser scan are inside the room's bounding footprint. For doors, we labeled both the start and the end of the opening and closing process.

B. Event delays

All experiments were done by playing the data set 25 times, on a Thinkpad X1 extreme with 64 GB RAM, an Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz octa core, a NVIDIA Quadro T2000 with 4 GB VRAM (NVIDIA 510.47.03, CUDA 11.6, CUDNN 8.3.2). The Kafka broker, the Flink application and the ROS nodes were all deployed locally on the same machine. For the evaluation of the

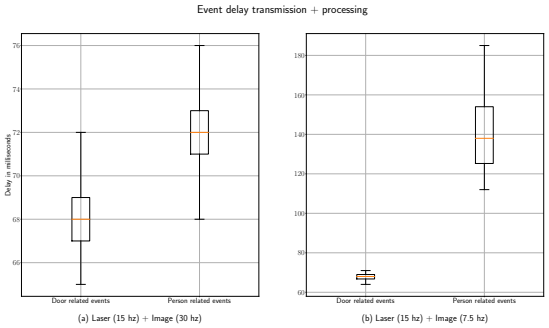

Fig. 5: Delays between event emission and start of transmission of the last record affecting the detection.

delays in the event generation, we differentiate between two types of events: The door events, which are purely based on laser data, the transformation into the map frame, spatial queries and state update in an operator and the person events, which dependi on multi-modal sensor input, which requires transformation of both sensor streams, accumulation in two tracking operators, the timely synchronization and fusion of detections, the transformation into the map frame, spatial queries and finally the matching to a simple CEP pattern. We measured the difference from the point of time the messages were sent to the broker, and the time the event was eventually generated. The first plot in Fig. 5 shows that the latency for the door events is approximately the sensor's frequency, which is plausible given that the watermark is increased with every incoming record.

For the person based events, the delay mainly consists of two parts. The largest part of the delay is a result of the watermark generation. When connecting two streams, e.g. for synchronization, the watermark of the resulting stream is the minimum of the two input streams. Thus, the time delta between two records in the slower stream (in our case the laser scan stream) directly impacts the delay. On top of that is the runtime cost of the assignment, tracking and pattern matching, where events are accumulated and only processed when the watermark progresses.

To demonstrate this effect, we limited the image stream frequency to 7.5 Hz, which is half of the frequency of the scan stream. The second plot in Fig. 5 shows, that the median delay approximately doubles for the person events, while for the door events it remains unchanged. The main source of the delay is the timely synchronization necessary between image and laser scan detection, where records are only send downstream, if they were successfully matched. The watermarks only progress when records with a higher timestamp arrive, which now happens with a median frequency of 7.5 Hz (the throttled image stream's frequency), affecting the progress in the following operators, e.g., tracking and the evaluation of the CEP patterns. One explanation for the higher deviation is that in the synchronization operator, the best timely match found does not result in an emitted record, if it does not lie fulfill a time constraint, which happens more often because of the lower frequency of the image stream. Overall, the

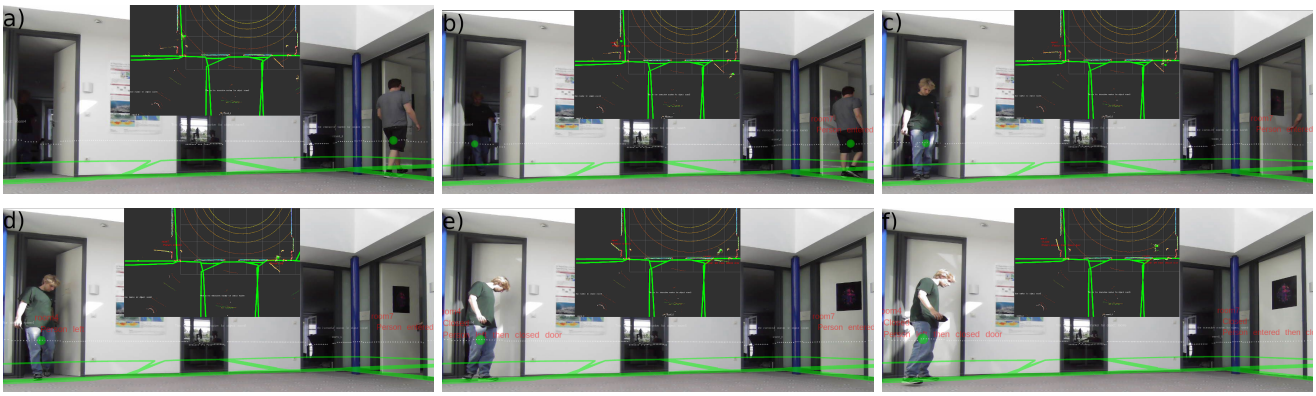

Fig. 6: Example of temporal event detection: (a) No event (b) Person entered Room07 (c) No event (d) Person left Room04 (e) Room04 closed $\rightarrow$ Person left then closed Room04 (f) Room07 closed $\rightarrow$ Person entered then closed Room07

numbers show that the detectors of choice are fast enough to keep up with the input data and introduce no back pressure or additional delay. We also see that the overhead of the message broker is negligible, although it is possibly the source for the small deviations evident in all measurements.

C. Detection

An example for the resulting event log of our pipeline are shown in Fig. 6. It displays a sequence of topological spatial changes and the generated events and relations inferred by our pipeline. It was able to detect all 16 transitions in the topological spatial relations of persons and rooms within $500\,\mathrm{ms}$ around their occurrence in event time (not processing time) in the labeled logs. Every 6 times the doors were opened, it was detected before persons left or entered the corresponding room. As a result, 12 spatio-temporal relations of the opening/closing of doors, when persons entered or left, were all detected correctly.

Reasons for the temporal imprecision stem directly from the quality of detectors and representation. A person is currently just represented by a center point in the Kalman filter and spatial queries. The Kalman filter also takes the mean of up to two associated clusters centers, which further results in a loss of geometric precision of the position and the constant velocity model used is also imprecise. Better detection and tracking, together with, e.g., a mesh-based representation of persons would most likely lead to better results, since the result of the predicates in the spatial streaming queries are directly constrained by the accuracy of the tracking and detection algorithms.

D. Discussion

While the results in Sec. VI-C definitely show room for improvement. In Sec. VI-B we showed the order of magnitude of delays introduced by the stream processing framework, through the concept of watermarks, which is the base for its processing guarantees. As discussed in IV-A, latencies could be reduced but would weaken the guarantees given by the framework. Currently, the latencies negligible, as we see the use case of this approach as an input or trigger for symbolic task planning. In this case, the delay and temporal precision is sufficient, as it would require additional more extensive and computationally expensive analysis. While the detectors, sensor fusion and trackers performed well enough to demonstrate the concept on the presented data set, it is unclear how the perform in more complex situations. In this paper, we processed all data in the Flink pipeline. While this might be useful in cases where it is important to provide guarantees, e.g, on the order in which sensor data is processed, this approach has some disadvantages. Especially in distributed systems the high amount of data that has to be transmitted might become a communication bottleneck. Processing all sensor data in the Flink application also means that many already existing algorithms have to be ported. An alternate approach would be to do the raw sensor data processing in ROS and send the derived information to the Flink application, still benefiting from the processing guarantees, query language and specialized tools like CEP.

VII. Conclusion & Future Work

We showed that our concept is feasible to detect the change of topological relations of objects based on multi-modal sensor input and a semantic map. Analyzing the resulting log with CEP patterns to derive additional information about spatio-temporal relations delivered promising first results. Next, we plan to integrate such information into high level task planning, execution and monitoring. For that, a common multi-modal query and maintenance interface for the semantic map and the stream processing component is required. Also, it will be necessary to synchronize all groundings made by the streaming pipeline back into the database. Currently all sensor data processing is done in the stream processing framework. In the future, we plan to do it partially in ROS. This is an advantage, since there is currently no need to have the guarantees provide from our framework on this level of processing. Since the detection and tracking directly impacts the results of event detection, we want to deploy state-of-the-art object detectors, trackers and semantic mapping frameworks as input for our system.

Acknowledgment

Part of this research has been funded by the Federal Ministry of Education and Research of Germany (BMBF) in the project DAKIS (grant number 031B0729B).

REFERENCES

[1] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, pp. 915–926, 2008.

[2] J. Lin, "The lambda and the kappa," *IEEE Internet Computing*, vol. 21, no. 05, pp. 60–66, 2017.

[3] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, pp. 1–62, 2012.

[4] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[5] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008, semantic Knowledge in Robotics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889008001188

[6] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Monitoring the execution of robot plans using semantic knowledge," *Robotics and autonomous systems*, vol. 56, no. 11, pp. 942–954, 2008.

[7] M. Günther, T. Wiemann, S. Albrecht, and J. Hertzberg, "Model-based furniture recognition for building semantic object maps," *Artificial Intelligence*, vol. 247, pp. 336–351, 2017, special Issue on AI and Robotics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000437021400157X

[8] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[9] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1689–1696.

[10] Y. Nakajima and H. Saito, "Efficient object-oriented semantic mapping with object detector," *IEEE Access*, vol. 7, pp. 3206–3213, 2018.

[11] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3d object discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3037–3044, 2019.

[12] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr, "Eao-slam: Monocular semi-dense object slam based on ensemble data association," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4966–4973.

[13] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4471–4478.

[14] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2018, pp. 10–20.

[15] H. Deeken, T. Wiemann, and J. Hertzberg, "Grounding semantic maps in spatial databases," *Robotics and Autonomous Systems*, vol. 105, pp. 146 – 165, 2018.

[16] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.

[17] H. Deeken, T. Wiemann, and J. Hertzberg, "A spatio-semantic approach to reasoning about agricultural processes," *Applied Intelligence*, pp. 1–13, 2019.

[18] M. Tenorth and M. Beetz, "Knowledge processing for autonomous robot control," in *2012 AAAI Spring Symposium Series*, 2012.

[19] F. Heintz, "Semantically grounded stream reasoning integrated with ros," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5935–5942.

[20] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus, "C-sparql: a continuous query language for rdf data streams," *International Journal of Semantic Computing*, vol. 4, no. 01, pp. 3–25, 2010.

[21] F. Heintz and D. De Leng, "Spatio-temporal stream reasoning with incomplete spatial information." in *ECAI*, 2014, pp. 429–434.

[22] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan, "A survey of distributed data stream processing frameworks," *IEEE Access*, vol. 7, pp. 154 300–154 316, 2019.

[23] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: Fault-tolerant streaming computation at scale," in *Proceedings of the twenty-fourth ACM symposium on operating systems principles*, 2013, pp. 423–438.

[24] M. Armbrust, T. Das, J. Torres, B. Yavuz, S. Zhu, R. Xin, A. Ghodsi, I. Stoica, and M. Zaharia, "Structured streaming: A declarative api for real-time applications in apache spark," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 601–613.

[25] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.

[26] S. A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, and R. H. Campbell, "Samza: stateful scalable stream processing at linkedin," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1634–1645, 2017.

[27] S. Kulkarni, N. Bhagat, M. Fu, V. Kedigehalli, C. Kellogg, S. Mittal, J. M. Patel, K. Ramasamy, and S. Taneja, "Twitter heron: Stream processing at scale," in *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*, 2015, pp. 239–250.

[28] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[29] J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl, "Benchmarking distributed stream data processing systems," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1507–1518.

[30] D. De Leng, *Robust Stream Reasoning Under Uncertainty*. Linköping University Electronic Press, 2019, vol. 2006.

[31] A. N. Steinberg and C. L. Bowman, "Revisions to the jdl data fusion model," in *Handbook of multisensor data fusion*. CRC press, 2017, pp. 65–88.

[32] J. Yu, J. Wu, and M. Sarwat, "Geospark: A cluster computing framework for processing large-scale spatial data," in *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, 2015, pp. 1–4.

[33] J. Yu, Z. Zhang, and M. Sarwat, "Spatial data management in apache spark: the geospark perspective and beyond," *GeoInformatica*, vol. 23, no. 1, pp. 37–78, 2019.

[34] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, 2021, pp. 13 029–13 038.

[35] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[36] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[37] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, pp. 726–733, 06 2015.