Knowledge Graph-Driven Reasoning for Automated Crane Lift Monitoring

Junlin Wang[1], Songbo Hu[2] and Yihai Fang[1], Hongling Guo[3]

[1]Department of Civil and Environmental Engineering, Monash University, Australia
[2]School of Built Environment, University of New South Wales, Australia
[3]Department of Construction Management, Tsinghua University, China
Junlin.Wang@monash.edu, Songbo.Hu@unsw.edu.au, Yihai.Fang@monash.edu, hlguo@tsinghua.edu.cn

Abstract

Crane lifting operations strongly influence construction productivity, yet current monitoring approaches offer limited ability to interpret lifting activities beyond motion tracking. This study presents a knowledge graph approach that uses a Neo4j property graph with in-database reasoning to interpret crane lifts. A conceptual graph schema is proposed to organize fused sensor and vision data, while Cypher rules operate on the graph to infer eight transient behaviors and segment continuous monitoring into discrete lifts. The method is validated on a field deployment with 21 lifts, and 2,974 labeled samples captured every 3 seconds. The system reaches accuracy 0.911 and F1 0.907 for transient behavior recognition and segments all lifts with median timing errors below 10s per lift. The results show that the proposed rule-based reasoning inside the graph database enables efficient automated monitoring of crane lifts and provides a basis for productivity evaluation and progress tracking.

Keywords –

Crane lift; Monitoring; Knowledge graph; Reasoning; Graph database

1 Introduction

Crane lift operations are pivotal to the productivity of modern construction projects, particularly with the increasing adoption of prefabrication and modular construction [1]. However, monitoring these operations remains predominantly manual and fragmented, leading to errors and delays in decision-making [2]. As projects grow in complexity, automated monitoring technologies have been increasingly adopted. Internet of Things (IoT) sensors and computer vision (CV) are widely used to capture motions of cranes [3] and identify hazards [4]. While these methods effectively improve safety, they lack the semantic reasoning capacity to interpret crane lifting progress for productivity enhancement. More recent studies have proven that interpreting crane lifts is feasible in specific scenarios, such as crane-assisted concrete pouring [5] and façade installation [6]. In practice, however, cranes are highly versatile, handling various materials in complex environments. Approaches that rely on a single data source or target specific activities (e.g., concrete pouring) tend to have limited generalizability and are inevitably sensitive to occlusions or sensor failure. This motivates more generalizable and robust approaches. To address these challenges, researchers have utilized Semantic Web and ontology-based methods to model crane lift knowledge [7]. These technologies resulted in enhanced generalizability in lab tests. Nevertheless, their implementation typically relies on an algorithm architecture where data storage and semantic reasoning are separated: data resides in databases while logic is executed as an external reasoner. This introduces redundant data movement and implementation complexity, particularly when streamlining continuous on-site lifting data processing. As the industry moves towards data-intensive monitoring, there is a need to explore architectures that bring reasoning logic closer to where the data resides.

To bridge these gaps, this paper proposes an automated crane lift monitoring approach based on a Knowledge Graph-driven reasoning framework that adopts an in-database reasoning architecture. Instead of relying on external reasoner, the approach utilizes a Property Graph model (via Neo4j) to integrate multi-source heterogeneous data such as CV results and sensor signals into a unified graph structure, allowing hierarchical reasoning to be executed directly via Cypher queries. Building on the developed ontology schema in [7], the framework infers transient crane behaviors and segments complete lift cycles. This approach enables site managers to track lifting progress, identify inefficiencies, and make prompt adjustments to construction schedules and crane resource allocation.

2 Related Work

The foundation of crane monitoring lies in the acquisition of physical data. Early research focused on sensor networks that use Inertial Measurement Units (IMUs), and encoders to track crane poses [3] and detect safety hazards such as excessive load sway [8]. With the advancement of deep learning, CV has become a dominant modality. Tang et al. [4] utilized CV to identify struck-by hazards in lifted loads, and later works adopted YOLO-based detectors to track hook and load trajectories for productivity analysis [9]. To address visibility limitations in complex scenarios, researchers have also developed multi-sensor systems for 3D workspace reconstruction and improved situational awareness [10]. Another study has applied vision-based methods to identify specific steps and calculate cycle time for concrete pouring [11]. To improve robustness, recent studies combined multiple sensors and digital models, such as integrating 4D point clouds with BIM to track hoisting activities [12] and fusing cameras with LiDAR to delineate risk zones around cranes [13]. Building on these technologies, digital twin-based monitoring has also emerged. Hussain et al. [14] fused sensor streams with finite element analysis within a digital twin model to assess lifting capacity. Yuan et al. [15] linked physical crane states to a virtual twin for closed-loop monitoring. These studies suggest that advanced sensing, computer vision and digital twins jointly enhance crane monitoring. However, most systems still focus mainly on trajectories and equipment motion status rather than explicit semantics of varied lift activities or task-level progress reasoning.

To enrich the semantic context, many studies have introduced ontologies and the broader Semantic Web stack to formalize construction knowledge and connect heterogeneous data sources. Ontologies provide machine-readable representations of domain concepts and relationships, enabling systems to interpret and relate objects, sensor readings and textual documents in a unified way [16]. Du et al. [17] constructed a Tunnel Defect Analysis Ontology (TDAO) that links BIM, inspection reports and sensor measurements to diagnose defects and their causes across multi-source datasets. In construction safety, Speiser et al. [18] proposed the Unified Ontology for Construction Safety (UNOCS) to integrate regulations, work processes and risk mitigation measures into a shared vocabulary, supporting consistent safety checking. For construction planning, Linked-Data-based Constraint-Checking (LDCC) by Soman et al. [19] encodes look-ahead scheduling constraints in RDF and SHACL and validates them against OpenBIM datasets, and was later extended with reinforcement learning for automated decision-making. Existing works demonstrate that ontology and Semantic Web technologies can support constraint checking, planning and hazard assessment over distributed construction data. These approaches typically use RDF and OWL to model static relationships. While highly effective for knowledge sharing and rule checking, the traditional semantic stack treats "reasoning" as a distinct batch process separated from data management. Optimizing this computational architecture to minimize the gap between storage and inference remains necessary for high-frequency monitoring data and for executing fine-grained temporal reasoning close to the underlying data management infrastructure.

Building on semantic representations, a small but growing stream of work applies knowledge graphs (KG) directly to construction and crane operations. A KG represents domain entities and relations using property graphs. Property graph databases such as Neo4j store labelled nodes and relationships with arbitrary properties and support pattern-based querying in Cypher, making it easier than RDF triple stores to attach rich attributes to edges and to run graph traversals and analytics directly inside the database. For example, Jiang et al. [20] integrated a tower crane digital twin with a safety knowledge graph to recognize unsafe hoisting patterns and trigger early warnings. Pfitzner et al. [5] combined computer vision with a knowledge graph capturing concrete pouring progress. Chen et al. [21] constructed a Crane Safety Knowledge Graph from accident reports and trained a graph attention network on this graph to learn accident patterns for deriving crane safety rules. These studies show that KGs are effective for capturing crane-related safety and process knowledge and for linking low-level observations to higher-level reasoning and assessment. However, these KGs mainly served as relatively static knowledge bases for management or analysis, and the graph database was used primarily for storage and querying while core reasoning was implemented in external engines or learning models. There are still limited works modelling crane operations, task states and multisensor-derived events directly as a property graph and encodes domain rules as queries executed inside the database.

3 Methodology

The primary goal of this research is to develop a knowledge graph for automated crane lift monitoring by integrating multi-source heterogeneous data into a Neo4j-based property graph and realizing in-database reasoning for transient crane behaviors and task-level lift progress inference. As illustrated in Figure 1, the proposed method consists of three phases. Phase 1 defines the knowledge graph schema that formalizes the crane-related entities, relations and reasoning logic. Phase 2 acquires and processes multi-source heterogeneous data and instantiates them as time-ordered

frame nodes in the graph. Phase 3 performs Cypher-based reasoning rules in Neo4j to label frame nodes with transient behaviors and segment complete lift cycles from the labeled frame sequence.

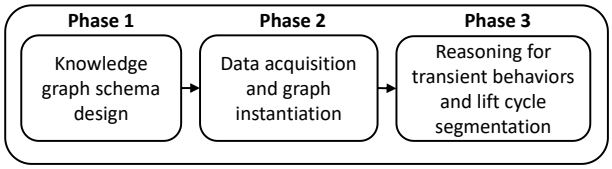

Figure 1. Workflow of the proposed method

3.1 Knowledge Graph Schema Design

Phase 1 designs a knowledge graph that organizes crane monitoring data and reasoning outputs into a two-layer property graph (Figure 2). This schema provides the organizing principles for structuring instance data in the graph. The schema layer defines core node types including Crane Transient Status and Crane Lift Cycle. Crane Transient Status groups frame-level attributes such as weight on hook, boom status, slewing status, lifting height status and transient behaviors. Crane Lift Cycle groups task-level attributes including start and end time, load weight, load color, shape and material, and the landing height and area. A "Contains" relation connects Crane Transient Status and Crane Lift Cycle, expressing that frame-level behaviors are organized into complete lift cycles executed by the same crane.

The data layer instantiates this schema as instance data with two main node types: Time Frame nodes and Lift nodes. Each Time Frame node represents one fused observation window and stores transient attributes. Consecutive Time Frame nodes are connected by a "NextIs" relation to preserve temporal order. Each Time Frame node is linked to a Lift node through a "Contains" relation, indicating which lift cycle it contributes to, while Lift nodes store aggregated instances attributes such as lift start and end times, load weight and landing location.

Rule sets for transient behaviors inference and lift cycles segmentation are applied on this schema, using the defined node types and relations as the basis for Cypher-based in-database reasoning, and the details are elaborated in section 3.3.

3.2 Data Acquisition and Graph Instantiation

The monitoring system integrates multi-source heterogeneous data from on-site sensors and a camera mounted on the mobile crane (Figure 3). Encoders and an IMU package measure boom pitch angle, slewing angle, boom length and weight on hook, providing numerical

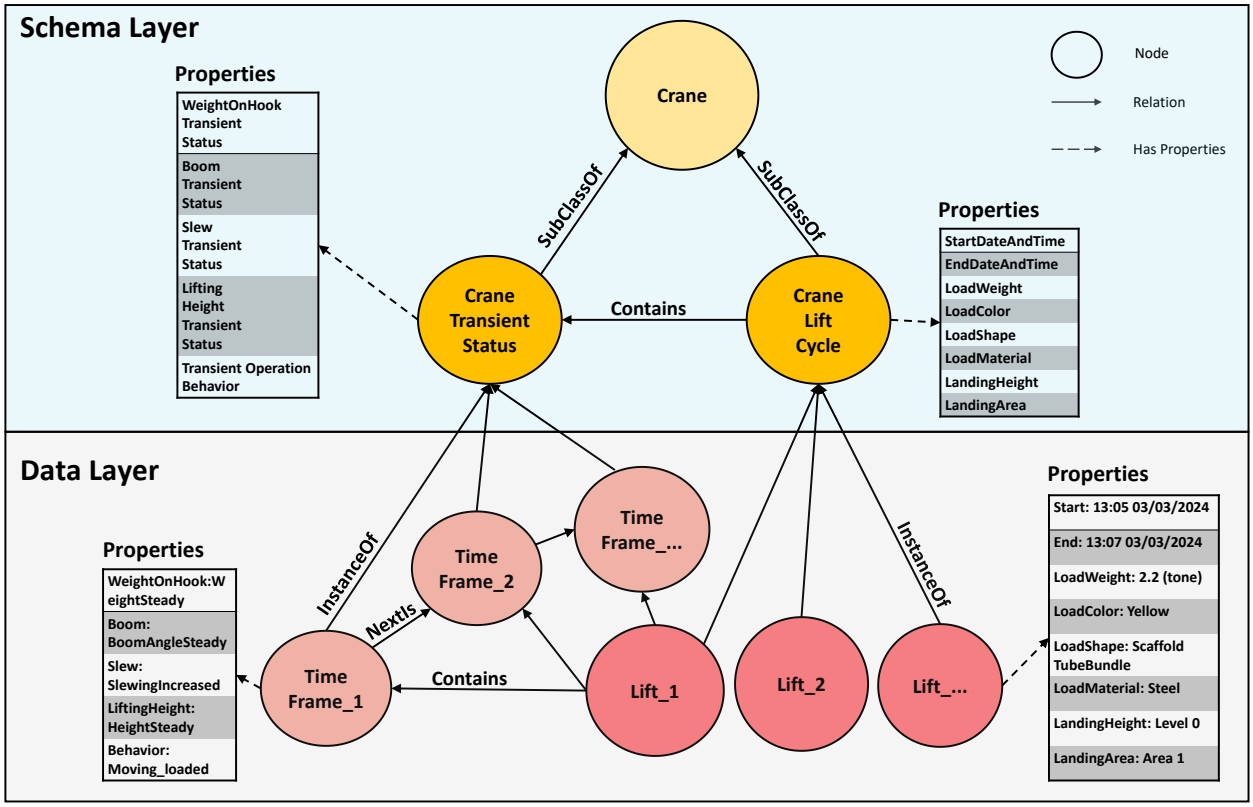

Figure 2. Schema and data layer of KG

signals that describe the crane's mechanical status. In parallel, an RGB camera observes the workspace, capturing the lifted load and hook region so that lifting height, basic load appearance and geometric data can be extracted and processed by a deep learning neural network based onYOLOv8-OBB.

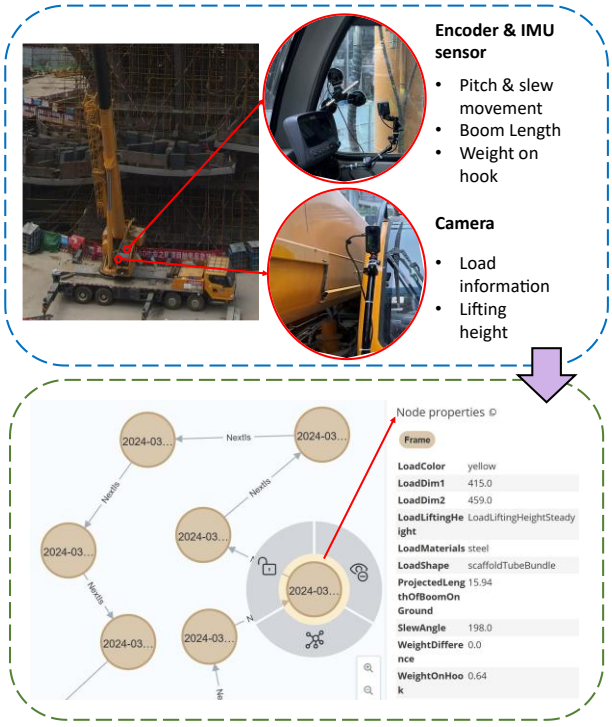

Figure 3. Data acquisition and graph instantiation

All numerical signals are first converted to consistent physical units (degrees for angles, meters for lengths and heights, and tonnes for weight) and then aggregated to a frame with a three-second interval. Sensor traces that are sampled at higher frequencies are averaged with a boxcar filter applied to each three-second window without overlapping, which filters short-term noise while preserving the temporal resolution required to distinguish crane behaviors. Measurements and semantic attributes derived from the camera are given precise time stamps and aligned to the same timeline using synchronized clocks, so that each interval contains a coherent description of crane and load across different sources.

These aligned records are then instantiated as Frame nodes in the Neo4j graph. For each three-second interval, a Frame node is created with properties including date and time, load weight, boom angle and length, slewing angle, lifting height, and load attributes. Frame nodes are ordered by index ID and connected through "NextIs" relations, forming a time-ordered chain that preserves the sequence of crane operations. This graph instantiation step converts the aligned sensor and vision streams into instance data that follows the organizing principles defined by the schema in Section 3.1, providing the basis for subsequent in-database reasoning. Figure 3 illustrates this process: the upper box shows the deployment of encoders, IMU and camera on the crane, while the lower box shows an example of Frame nodes connected by "NextIs" relations in Neo4j, together with the corresponding node properties populated from the aligned data.

3.3 Reasoning for Transient Behaviors and Lift Cycle Segmentation

The third phase performs in-database reasoning in Neo4j to derive transient crane behaviors at the frame level and to segment complete lift cycles from the frame sequence. The rule sets operate directly on the instance data introduced in Sections 3.1–3.2 and are implemented as Cypher queries (Figure 4).

```
// Part-level motion & transient opertaion behaviors
MATCH (prev:Frame)-[:NEXT]->(cur:Frame)
WITH prev, cur,
    cur.WeightonHook - prev.WeightonHook  AS d_w,
    cur.HookLiftingHeight - prev. HookLiftingHeight AS d_h,
    ...
SET  cur.WeightStatus = CASE WHEN d_w >  $th_w THEN 'Increased'
                     WHEN d_w < -$th_w THEN 'Decreased'
                     ELSE 'Steady' END,
    cur.LiftingHeightStatus = CASE WHEN d_h >  $th_h THEN 'Increased'
                     WHEN d_h < -$th_h THEN 'Decreased'
                     ELSE 'Steady' END,
    ...
// Derive transient operation behaviors from part-level motion
SET  cur.CraneTransientBehaviors  = CASE
       WHEN w_status = 'WeightSteady'
          AND h_status = 'LiftingHeightSteady'
          AND b_status = 'BoomPitchAngleSteady'
          AND s_status = 'SlewingSteady'
          AND cur.WeightOnHook = 'Equal'  // WeightOnHook Equal empty hook
          THEN  'Idle'
       ...
            END;
```

```
// Calibrated CraneTransientBehavior & segment sequence → lift cycles
MATCH (f:Crane)-[:HAS_FRAME]->(fr:Frame)
WITH f, fr ORDER BY fr.row_idx
WITH f, collect(fr) AS frames

// Debounce & refine status
WITH f, [x IN frames | CASE WHEN x. CraneTransientBehavior IN
            ['Takeup_load','Lower_load'] AND x.segment_len <= $short_max
        THEN CASE WHEN x.hasNonSteadyPart THEN 'Moving_unloaded' ELSE 'Idle' END
      ELSE x. CraneTransientBehavior      END] AS CraneBehavior_calibrated
    ...
// Segment lift cycles
WITH f, frames, CraneTransientBehavior_calibrated,
    apoc.coll.indexOf(CraneTransientBehavior_calibrated,'Loading')   AS s,
    apoc.coll.lastIndexOf(CraneTransientBehavior_calibrated,'Unloading') AS e
CREATE (lift:Lift {crane_key:f.crane_key, start_idx:s, end_idx:e, ...})
FOREACH (i IN range(s,e) | MERGE (lift)-[:Contains]->(frames[i]));
```

Figure 4. Cypher-based in-database reasoning

The first group of rules infers the motion status of crane parts and eight transient behaviors (Idle, Loading, Takeup_load, Moving_loaded, Loaded_hold, Lower_load, Unloading, and Moving_unloaded) from each Frame node. For every pair of neighboring frames connected by the relation, Cypher computes differences in weight on hook, hook height, boom angle and slewing angle. These differences are compared with calibrated thresholds to classify each component as "Increased",

"Decreased" or "Steady". These values were obtained from representative calibration sequences collected on the instrumented crane and refined with field observation and operator feedback to separate genuine operational events from sensor noise and empty-hook oscillations. These status of components are combined to determine a transient behavior using compact decision logics. For example, all components steady and weight on hook below or equal to the empty-hook reference correspond to an "Idle" behavior; any non-steady motions with load on the hook correspond to "Moving_loaded". These logics are encoded as the CASE expressions in Cypher and stored back as properties of the Frame nodes, creating a time-ordered sequence of transient behavior labels along the frame chain.

The second group of rules calibrates this labeled frame sequence and segments it into complete lift cycles. In this process, segments of lift cycles with time below the minimum duration threshold are debounced and calibrated to suppress noise from sensors. Once this calibration finishes, the frame labeled "Loading" and the frame labeled "Unloading" are identified as the temporal boundaries of a lift cycle. A new Lift node is then created with start and end indices, timestamps, basic attributes, and all frames between these indices are linked to the Lift node through the "Contains" relation.

Through the three steps, the in-database reasoning procedure transforms the aligned multi-source data into a structured sequence of transient behaviors and complete lift cycles that can be queried and analyzed directly in the knowledge graph.

4 Field Experiments

The proposed approach was evaluated through a field deployment on an active construction site. Multi-source heterogeneous data from two consecutive days of crane lifting were captured and populated into the knowledge graph, and the reasoning rules were applied to infer transient behaviors and segment lift cycles within the graph. The experiment validates the approach on real crane lifts and quantifies its performance in terms of accuracy and robustness.

4.1 Experimental Setup and KG Instantiation

The field experiment was conducted on a construction site that builds a 50 m steel observation deck. The structure created a confined working space for a 50-tonne mobile crane. As shown in Figure 5, the site layout included a truck loading and unloading area, a staging area storing scaffold tube bundles, I-beams, square steel tubes, glass bundles and other prefabricated materials, designated load landing areas on the deck, and a crane deployment and stabilization area. Over two consecutive workdays, the crane completed 21 lifts that covered typical material handling tasks such as moving scaffold bundles and large prefabricated elements between the staging areas and the landing positions.

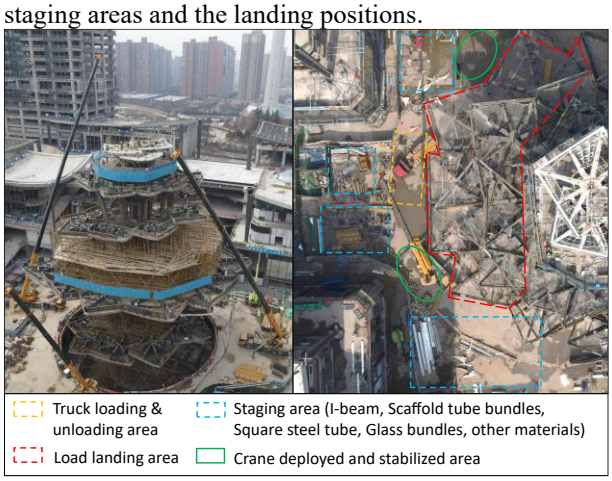

Figure 5. Site layout

These lifting operations produced 2,974 three second frames used in the subsequent evaluation. The acquisition and preprocessing pipeline in Section 3 was deployed on the crane to integrate multi-source heterogeneous data. Encoder and IMU signals were converted to physical units, aligned temporally and combined with camera-based measurements of load category and lifting height. Figure 6 illustrates representative lifting examples, where the detected hook and load with geometric data derived by CV are shown together with the weight on hook, boom pitch angle and slew angle traces obtained by sensors.

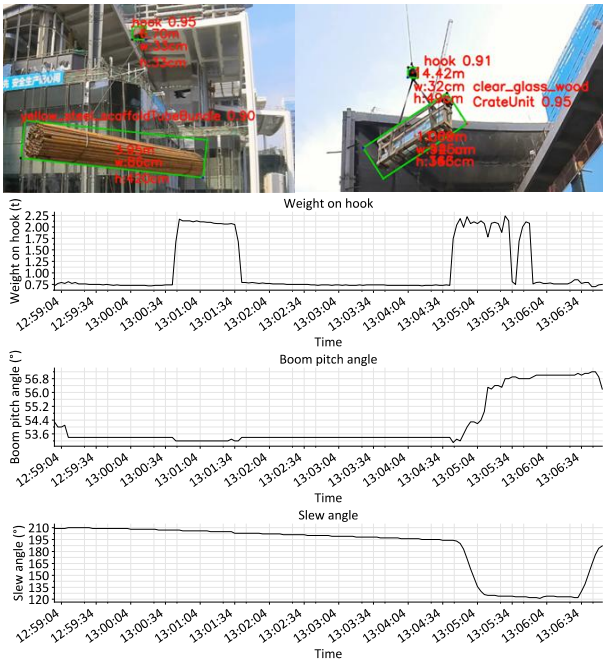

Figure 6. Data collection and preprocessing

The temporally aligned data were populated into a Neo4j graph database-driven KG following the schema in Section 3.1. Each three-second interval became a Frame node linked by "NextIs" relations, and each complete lift was represented as a Lift node connected to its member frames. Figure 7 visualizes the resulting knowledge graph instance for the entire experiment, where clusters of frames (light brown) are organized around the inferred lift segments (pink).

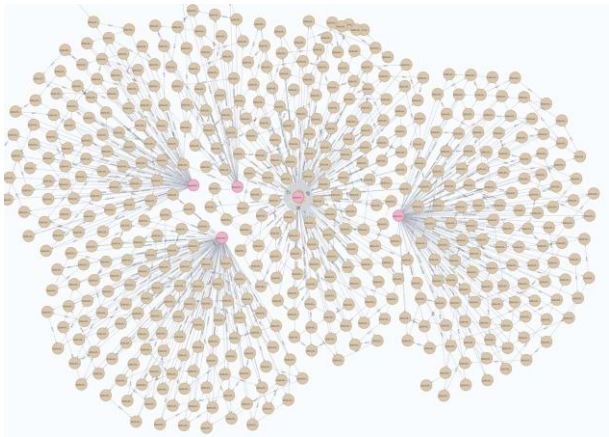

Figure 7. Details of the instantiated KG containing nodes: Frame (light brown), Lift (pink)

Figure 8 zooms into the sample Frame and Lift nodes which store reasoned data to illustrate how inferred transient behaviors and segmented lift cycles are co-located in the same property graph and are ready for subsequent querying and analysis.

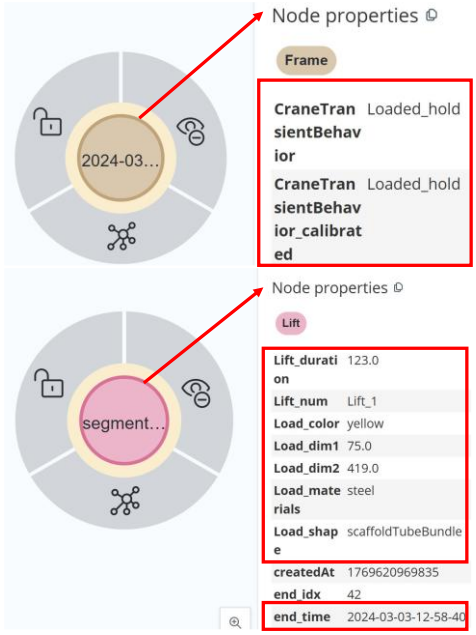

Figure 8. Inferred properties of nodes

4.2 Reasoning Results and Evaluation

The results of transient behaviors reasoning and segmentation of complete lift cycles were evaluated separately. Ground truth was created on the same three-second timelines by manually assigning one of eight behaviors to each record and by marking the start and end time of all 21 complete lifts. System outputs were compared with these labels using accuracy, precision, recall and F1 score, and by analyzing timing differences between inferred and reference lift durations.

For transient behaviors, as shown in Table 1, the 2,974 labeled records yielded an overall accuracy of 0.911 and an average F1 score of 0.907. Behaviors such as Idle (F1 0.987) and Unloading (F1 0.972) were recognized with very high reliability, while Moving_loaded (F1 0.932) and Moving_unloaded (F1 0.912) also showed strong performance.

Table 1. Evaluation matrices of transient behaviors

	precision	recall	F1-score
Moving_loaded	0.975	0.892	0.932
Loading	0.739	0.996	0.849
Unloading	0.981	0.963	0.972
Idle	0.988	0.985	0.987
Moving_unloaded	0.935	0.891	0.912
Loaded_hold	0.968	0.838	0.899
Lower_load	0.867	0.812	0.839
Takeup_load	0.929	0.812	0.867

As shown in Figure 9, the main confusion occurred between Loading and Loaded_hold: Loading achieved very high recall (0.996) but moderate precision (0.739), whereas Loaded_hold showed precision 0.968 and recall 0.838. This phenomenon is consistent with the design of the rules, which rely strongly on temporal reasoning over slewing patterns to distinguish these behaviors. On-site, loading phases are long and variable; riggers repeatedly adjust the hook and slings, producing many short intervals where the crane temporarily holds a partly stabilized load. Using lifting height changes to differentiate these two behaviors could improve accuracy, but detecting the changes depends on a vision-based method that can be affected by occlusion and illumination shifts. Short duration classes such as Takeup_load (F1 0.867) and Lower_load (F1 0.839) are defined to occur only when a coherent transport phase connects them, which makes them rarer, briefer and more sensitive to noise. Although they were less frequent, results still reached acceptable scores. Even with these challenges, the confusion matrix in Figure 9 shows that the reasoning succeeds in turning low-level signals into a

coherent and interpretable sequence of transient crane behaviors.

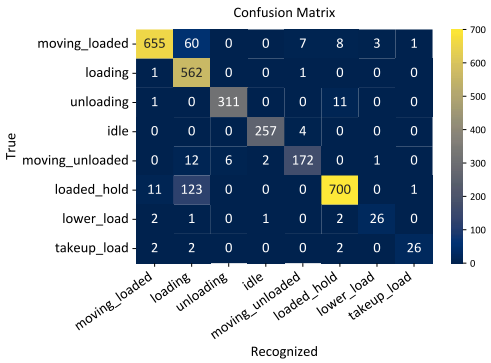

Figure 9. Confusion matrix of transient behaviors

For lift cycle segmentation, the rule-based reasoning procedure correctly identified all 21 complete lifts. The algorithm applies a minimum duration threshold and a slewing pattern threshold, derived from field observations and operator feedback, so that short fluctuations due to empty hook sway or brief takeup or lowering adjustments are filtered out. The timeline comparison in Figure 10 shows close agreement between inferred segments and ground truth on both days, with no missed lifts and no incorrect mergers.

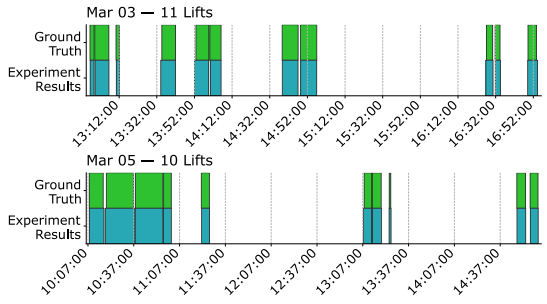

Figure 10. Comparison of segmentation results

The distribution of absolute timing errors in Figure 11 indicates median differences of about 7s and 6s for start and end times, and less than 10s for total lift duration, which is small compared with lift operations that usually last several minutes. A few outliers correspond to lifts with extended adjustments or pauses. The segmentation is therefore robust to occasional misclassifications at the transient-behavior level and, by combining temporal constraints with information on intermediate behaviors, produces lift cycles that are suitable for automated monitoring and performance analysis.

These results demonstrate that the in-database reasoning over the knowledge graph can reconstruct both transient behaviors and complete lift cycles for real crane operations with high fidelity.

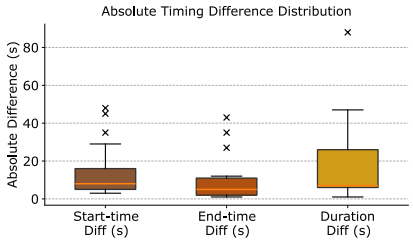

Figure 11. Absolute time difference in the lift cycles segmentation

5 Conclusions

The results show that a property graph and Cypher rules can reconstruct crane transient behaviors and complete lift cycles with high fidelity, demonstrating that in-database reasoning is a practical way to couple multi-source heterogeneous data with semantic interpretation for automated lift monitoring.

The study is based on a single mobile crane in a confined work zone and a moderate number of lifts, so generalizability to other crane types, multi-crane sites and longer projects have not been verified. As the proposed graph schema and reasoning hierarchy are defined at the level of generic crane states and lift-cycle logic, it makes the overall architecture transferable in principle. However, extended rule set and graph representation (e.g., crane identity and workspace-interaction relations) would be required.

Certain reasoning thresholds were tuned from this dataset and may require recalibration under different sensing noise levels or operational practices. In new settings, recalibration is expected to be limited to a small set of site-specific numerical parameters, such as the empty-hook reference, signal-change thresholds, and segmentation tolerances, while the overall graph schema, Cypher workflow, and behaviour definitions remain unchanged. The framework also relies on vision-derived lifting height and load appearance, which can degrade under occlusion and illumination changes.

While executing the reasoning directly in Neo4j reduces data movement between storage and inference, formal benchmarking of latency, indexing strategy, and large-graph scalability during long-term, high-frequency monitoring projects remain future work.

Future research will test the framework across multiple sites and cranes, integrate richer project data such as schedules and BIM into the graph, and explore learning assisted rule discovery while preserving transparent, queryable reasoning inside the database. At current stage, the system supports monitoring of executed lifting progress and operational bottlenecks. Although it does not yet provide full schedule-linked or BIM-linked progress verification, which remains future work, it

already establishes the necessary foundation for more comprehensive tracking of on-site schedule execution and overall construction progress.

References

[1] A. H. Ali, T. Zayed, and M. Hussein, "Crane safety operations in modular integrated construction," *Autom. Constr.*, vol. 164, p. 105456, 2024.

[2] Z. Zhang and W. Pan, "Lift planning and optimization in construction: A thirty-year review," *Autom. Constr.*, vol. 118, p. 103271, 2020.

[3] C. Zhang, A. Hammad, and S. Rodriguez, "Crane pose estimation using UWB real-time location system," *Journal of Computing in Civil Engineering*, vol. 26, no. 5, pp. 625–637, 2012.

[4] Y. Tang, C. Liu, K. Feng, X. Yang, X. Li, and X. Leng, "Identifying Struck-By Hazards in Unknown Lifting Loads: A Hybrid Computer Vision Approach," *J. Constr. Eng. Manag.*, vol. 151, no. 10, p. 04025146, 2025.

[5] F. Pfitzner, S. Hu, A. Braun, A. Borrmann, and Y. Fang, "Monitoring concrete pouring progress using knowledge graph-enhanced computer vision," *Autom. Constr.*, vol. 174, p. 106117, 2025.

[6] I. Jeong, J. Hwang, J. Kim, S. Chi, B.-G. Hwang, and J. Kim, "Vision-based productivity monitoring of tower crane operations during curtain wall installation using a database-free approach," *Journal of Computing in Civil Engineering*, vol. 37, no. 4, p. 04023015, 2023.

[7] S. Hu, J. Wang, and Y. Fang, "Semantic Web–assisted progress monitoring of crane operations in construction projects," *Proceedings of the Institution of Civil Engineers-Smart Infrastructure and Construction*, vol. 178, no. 3, pp. 148–157, 2025.

[8] Y. Fang and Y. K. Cho, "Crane load positioning and sway monitoring using an inertial measurement unit," in *Computing in Civil Engineering 2015*, 2015, pp. 700–707.

[9] H. Wang *et al.*, "Automated productivity analysis of cable crane transportation using deep learning-based multi-object tracking," *Autom. Constr.*, vol. 166, p. 105644, 2024.

[10] L. C. Price, J. Chen, J. Park, and Y. K. Cho, "Multisensor-driven real-time crane monitoring system for blind lift operations: Lessons learned from a case study," *Autom. Constr.*, vol. 124, p. 103552, 2021.

[11] D. Wang *et al.*, "Vision-based productivity analysis of cable crane transportation using augmented reality–based synthetic image," *Journal of Computing in Civil Engineering*, vol. 36, no. 1, p. 04021030, 2022.

[12] D. Liang, S.-H. Chen, Z. Chen, Y. Wu, L. Y. L. Chu, and F. Xue, "4D point cloud-based spatial-temporal semantic registration for monitoring mobile crane construction activities," *Autom. Constr.*, vol. 165, p. 105576, 2024.

[13] F. Wu, M. Hu, F. Xie, W. Bu, and Z. Zhang, "A Multimodal Sensor Fusion and Dynamic Prediction-Based Personnel Intrusion Detection System for Crane Operations," *Processes*, vol. 13, no. 12, p. 4017, 2025.

[14] M. Hussain, Z. Ye, H.-L. Chi, and S.-C. Hsu, "Predicting degraded lifting capacity of aging tower cranes: A digital twin-driven approach," *Advanced Engineering Informatics*, vol. 59, p. 102310, 2024.

[15] E. Yuan, J. Yang, M. Saafi, F. Wang, and J. Ye, "Realization of the physical to virtual connection for digital twin of construction crane," *J. Ind. Inf. Integr.*, vol. 44, p. 100779, 2025.

[16] Z. Fang, S. Hu, J. Wang, Y. Fang, and Q. Lu, "Ontological Models for Construction Scheduling and Resource Planning," in *Routledge Handbook of Smart Built Environment*, Routledge, 2025, pp. 133–161.

[17] J. Du, A. Yu, V. Sugumaran, Y. Hu, G. Yu, and J. Zhang, "Ontological reasoning for automated tunnel defect diagnosis and root cause identification," *Autom. Constr.*, vol. 178, p. 106362, 2025.

[18] K. Speiser, S. Seiß, F. Boukamp, J. Melzner, and J. Teizer, "From fragmented data to unified construction safety knowledge: A process-based ontology framework for safer work," *Autom. Constr.*, vol. 176, p. 106293, 2025.

[19] R. K. Soman and M. Molina-Solana, "Automating look-ahead schedule generation for construction using linked-data based constraint checking and reinforcement learning," *Autom. Constr.*, vol. 134, p. 104069, 2022.

[20] W. Jiang, Y. Liu, K. Chen, Y. Liu, and L. Ding, "Early-warning of unsafe hoisting operations: An integration of digital twin and knowledge graph," *Developments in the Built Environment*, vol. 19, p. 100490, 2024.

[21] J. Chen, H.-L. Chi, B. Xiao, and R. Li, "Interpreting accident reports by integrating a heterogeneous graph neural network and factor analysis," in *30th International Conference on Intelligent Computing in Engineering 2023, EG-ICE 2023*, 2023.