Sensors & Transducers, Vol. 273, Issue 2, June 2026, pp. 39-49

www.sensorsportal.com

Sensors & Transducers

Published by IFSA Publishing, S. L., 2026
http://www.sensorsportal.com

Automated Mission Planning for Mobile Robots Using Building Information Modeling (BIM) and Digital Twins

[1] Annamria ARNOUK, [1] Martin BEHM, [1] Christian BORCK, [2] Tim HOFFMANN, [2] Florian DIETRICH, [2] Matthias MELZER, and [2, *] Jan DÜNNWEBER

[1] Brandenburgische Technische Universität Cottbus – Senftenberg, Germany
[2] Ostbayerische Technische Hochschule Regensburg, Germany
[2] Tel.: +49 941 943-7186
[*] E-mail: jan.duennweber@oth-regensburg.de

Received: 8 April 2026 /Revised: 20 May 2026 /Accepted: 5 June 2026 /Published: 29 June 2026

Abstract: This article presents a conceptual framework for automated mission planning of mobile robots in smart factory environments, integrating Building Information Modeling (BIM) with industrial digital twins. Our first contribution is BIM2Robot, an open-source framework that systematically transforms building models, given in a standards-compliant IFC description, into a ROS-compatible topological navigation graph including a geometric environment model and an executable mission plan, enabling Automated Guided Vehicles (AGVs) to be deployed in the described building without manual waypoint definitions. The second contribution is the Assembly Twin, a production digital twin built on NVIDIA Omniverse that integrates design and process data from PLM and ALM platforms, real-time sensor streams via the Adaptive Framework for Optical Sensors (AFOS), and AI-based assembly monitoring via the Confirmation and Tacit Data System (CTDS). The Assembly Twin provides real-time worker guidance through extended reality interfaces and continuously tracks workbench states and production progress. The integration of both systems establishes a closed loop between production intelligence and robot navigation: When the Assembly Twin detects a production disruption, such as an unexpected workbench failure, it runs a script provided by the BIM2Robot framework that generates an updated mission plan and reroutes the affected AGV(s) autonomously. Relying on open standards including IFC, USD and OPC UA, the combined framework is adaptable and accommodates continuous changes in the workplace layout. Generating robot code directly from a building model eliminates the need for manual reconfiguration. Combined with a live digital twin of the assembly hall, AGVs benefit from a continuously synchronized understanding of both the static geometry and the dynamic production state, enabling context-aware navigation decisions that transcend purely geometric path planning.

Keywords: Robot motion planning, Digital twins, Assembly automation, Building Information Modeling (BIM), Robot Operating System (ROS 2), Robot programming methodology, Code generation.

1. Introduction

In modern smart factories, manual and automated manufacturing processes are deeply intertwined to ensure high efficiency, quality, and flexibility. Central to this environment are interconnected workbenches that serve as pre- and end-assembly stations, including integrated quality control. At these stations, operators are supported by worker guidance systems utilizing smart sensors, machine vision, and digital twins to execute complex assembly tasks. To sustain this production process, a seamless flow of inbound and outbound components is required, which is typically facilitated by Automated Guided Vehicles (AGVs) handling the intralogistics between workbenches.

However, maintaining continuous operation in such dynamic environments poses significant challenges, particularly when the manufacturing

process is impacted by unforeseen events. Two frequently occurring scenarios highlight the limitations of current static production systems:

First, unexpected failures can disrupt the workflow. If a workbench goes offline unexpectedly – for example, due to a broken tool – the production process must be rerouted dynamically. Components currently in transit or waiting must be swiftly transferred to alternative workbenches via AGVs to minimize idle times and prevent bottlenecks.

Second, the concept of a living shopfloor introduces continuous spatial and logistical changes. Modifications in the shopfloor layout directly influence intralogistics. Consequently, AGVs and mobile robots must autonomously find new routes to meet their objectives. This requires the system to constantly update and understand environmental models and contextual data to identify the exact start and goal locations of components, as well as the newly designated stations for assembly and warehousing.

To address these scenarios, this article proposes a novel framework that bridges the gap between dynamic production management and robot spatial planning by combining two approaches to automation: the Assembly Twin, implemented via NVIDIA Omniverse, and the open-source framework Robot BIM (BIM2Robot). While the Omniverse-based Assembly Twin acts as the cognitive core for the production process – managing real-time workbench states, worker guidance, and the dynamic rerouting of tasks – BIM2Robot provides the semantic and geometric understanding of the facility. By automatically translating standardized Building Information Models (IFC) and spatial annotations into ROS-compatible mission plans, the system ensures that AGVs can instantly adapt to living shopfloor changes.

In combination, the two technologies establish a resilient, fully synchronized manufacturing environment where both the production logic and the ROS-driven mobile robots act on a shared, real-time understanding of the factory floor. This article integrates and extends two prior conference papers. The first introduced the Assembly Twin as a production digital twin for real-time worker guidance and workbench monitoring [1]. The second presented BIM2Robot as a code generation framework for IFC-based robot navigation [2].

The remainder of this article is organized as follows. In Section 2, we discuss related work on factory automation with digital twins and on indoor path planning for robots. Section 3 uses the case study of producing aircraft drives to highlight features of our Omniverse-based Assembly Twin. In Section 4, we explain how BIM2Robot facilitates automatic path planning in a factory hall. Section 5 shows how an automated factory benefits when a digital twin is combined with a robot mission plan derived from the building description, using failure recovery as a possible use case. Finally, in Section 6 we conclude the article with an outlook on future work.

2. Related Work

Manufacturing digital twins have evolved into connected components of immersive environments referred to as the industrial metaverse. Siemens connected its Xcelerator platform with NVIDIA Omniverse in 2022 [3] and later added OpenUSD [4] support to Teamcenter X, letting engineering teams collaborate in a shared 3D environment [5]. Dassault Systèmes follows a different path with its 3DEXPERIENCE platform, where the DELMIA division provides virtual manufacturing planning and process optimization across the product lifecycle [6]. Where Siemens uses OpenUSD [4] as its interoperability layer, Dassault relies on IFC and STEP. NVIDIA Omniverse serves as a platform for physically accurate digital twins. Its USD (Universal Scene Description [4]) scene graph lets assets from CAD, BIM, and simulation tools be combined [7].

2.1. Digital Twins in Manufacturing

A virtual commissioning methodology was proposed to integrate digital twins into a flow-shop manufacturing system, validating digital-twin-driven reactive scheduling in response to machine breakdowns before physical deployment [8]. Furthermore, fourteen open-source digital twin frameworks were compared, showing that no single framework covers all needs, so practitioners often combine multiple frameworks [9]. Among these, Eclipse BaSyx implements the Asset Administration Shell aligned with RAMI 4.0, which is relevant here, as our earlier RAMI4.0-based asset combination [10] took a similar layered approach to connect sensing and context generation to the visualization layer.

XR-based worker guidance in manual assembly is closely related to the architecture presented in this article. XR applications across assembly stages were reviewed from a human-centric Industry 5.0 perspective, mapping benefits at the operator, enterprise, and industry level [11]. For tracking assembly progress without manual input, YOLO-based object detection was combined with pose estimation to recognize repetitive worker actions, achieving 92.8 % recognition accuracy [12]. Zero-shot or few-shot models extend this capability by generalizing unseen objects without retraining, relying only on reference images and CAD models. This property is relevant for high-variability production where components change frequently.

Our architecture builds on two research threads from the COCKPIT4.0 and DIREKT projects. A modular pipeline for mixed reality environments was built to structure visualization assets from different sources into reusable workflows [11]. Furthermore, a RAMI4.0-based asset combination was proposed to connect AFOS, CTDS, EAS, and VIS as coordinated assets for operating a digital twin in human-centered assembly [8]. Building on this RAMI4.0-based asset combination, our earlier work advanced this

foundation toward an Omniverse-based Assembly Twin for manual assembly. The study did not redefine the RAMI4.0 architecture itself; rather, it transferred this earlier conceptual basis into a concrete digital-twin environment centered on NVIDIA Omniverse, with a particular focus on the visualization layer. Its added contribution was the real-time integration of design and production data streams, the organization of assembly states through USD scene hierarchies and sublayers, and the coupling of AI-based tracking with XR-based worker guidance. A twin for manual production, as both works showed, requires a pipeline with sensing, context interpretation, and operator feedback. We extend this toward the industrial metaverse with USD-based scene composition, synchronized design and production twins, and XR-based guidance. To keep the spatial model current, the Environment Acquisition System (EAS) [8], a mobile robot equipped with LiDAR, acceleration, and RGBD sensors, is used. The joint use case of component commissioning across multiple workstations with load-dependent rerouting requires both this up-to-date spatial model and robot navigation between stations. For this purpose, we consider the derivation of navigation plans from BIM data.

2.2. BIM and Robot Path Planning

Mobile robots increasingly reduce the reliance on manual labor in logistics [13]. Transferring existing robotic solutions to new facilities, however, remains a persistent challenge, as environmental models, navigation graphs, and waypoint structures are typically tied to specific floor layouts and require substantial redesign effort when the environment changes [14].

BIM data provides structured sources of environmental information for robotic systems. In the construction domain, sensor data has been aligned with geometric information from IFC files to improve robot localization accuracy [15], and IFC-based semantic building models have been used to support robot task planning and execution in built environments [16].

More recent work demonstrates that the direct generation of robot-specific navigation maps from BIM data reduces computational overhead, compared to traditional SLAM-based navigation. BIM data also include high-level semantic descriptions of building structures, thereby enabling robots to navigate with greater autonomy [17]. Object detection techniques have also been integrated with BIM data to initialize pathfinding strategies in unknown environments [18], and ontology-based building models have been proposed to categorize building elements and their functionality for inspection tasks [19].

Beyond the BIM domain, digital building twins, created, e.g., via 3D laser scanning, have been applied to support indoor localization across multiple user platforms, demonstrating the broader relevance of semantically enriched environment models for robot navigation [20]. Model-driven approaches to robot programming have demonstrated the potential of generating application code directly from formal models [21].

In contrast to these concepts, where BIM data serves as a supplementary geometric information source or requires manual enrichment of building elements with robot-specific instructions, BIM2Robot uses the standardized building model itself as the primary source for new robot applications. Building on our previously proposed concept [22], the framework systematically transforms IFC descriptions into topological navigation graphs, geometric environment models, and executable mission plans through a structured parsing pipeline. BIM2Robot bridges the gap between descriptive building models and executable robot programs without requiring proprietary map files or waypoint definition.

3. Assembly Twin: Our Architecture for Digital Twins in Manual Assembly

Our proposed assembly digital twin architecture is shown in Fig. 1.

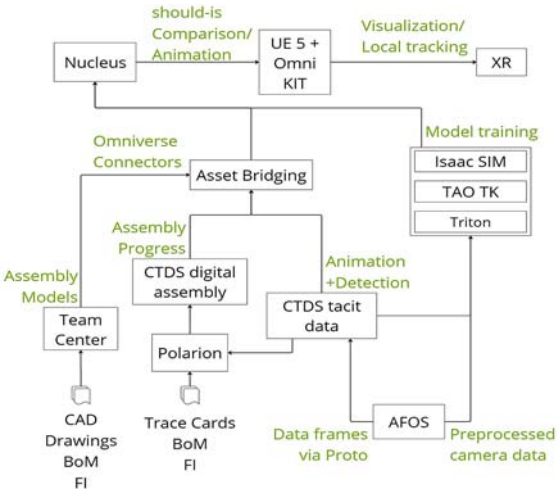

Fig. 1. Reference architecture.

3.1. Methodology for Real-Time Integration of Design and Production Digital Twins

All technological components are built around NVIDIA Omniverse as the central integration environment [23].

3.1.1. Technological Framework

The main components of our Assembly Twin are Omniverse Nucleus [24], Omniverse Kit, used for project-specific extensions [25], and software-specific connectors that enable synchronization between heterogeneous tools and data sources [26].

Sensor data is acquired via the Adaptive Framework for Optical Sensors (AFOS) [10]. The Confirmation and Tacit Data System (CTDS) processes assembly progress and implicit signals like time expenditure.

At the design level, BIM authoring environments, such as Autodesk Revit [27], can be integrated through Omniverse connectors. Connector-based synchronization is particularly relevant, since changes in the BIM model can be propagated to Omniverse applications.

Augmented reality (AR) and extended reality (XR) interfaces are available to factory workers and can close the loop between digital planning and physical execution. This is helpful for validating manual steps or for guiding workers based on operator behavior.

3.1.2. Data Acquisition and Transfer Pipeline

Our data integration pipeline is shown in Fig. 2. We explain its components using motor assembly as an example. Production-related information such as composition rules, Computer-Aided Design (CAD) models, and bills of materials (BOM) are retrieved from cloud-based Product Lifecycle Management (PLM) platforms, such as Teamcenter, and from Application Lifecycle Management (ALM) platforms, such as Polarion, through dedicated Omniverse connectors. Additional inputs include post-processed simulation models and animation sequences generated in Visual Components, 3ds Max, or Blender. Scanned physical prototypes can also be integrated into the pipeline.

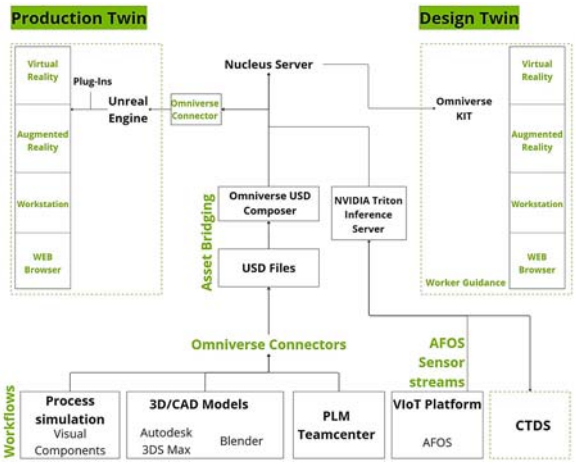

Fig. 2. Modular frameworks for mixed reality environments.

At the physical workspace level, the Adaptive Framework for Optical Sensors (AFOS) as a visual IoT platform provides a decentralized infrastructure for the integration of a strategically positioned camera system on the workbench consisting of two Orbbec Femto Bolt RGBD cameras. It also enables the creation of distributed data pipelines to allow real-time big data preprocessing of the camera streams including data fusion, e.g. for colored point cloud creation, multi-camera synchronization and the publication of the result to other applications via gRPC (Remote Procedure Calls). These data streams, combined with CAD assets from Teamcenter, feed into the Confirmation and Tacit Data Acquisition System (CTDS), which employs deep learning models to track worker actions and identify assembly deviations. Afterward, structured data flows into a centralized database optimized for cross-disciplinary collaboration. The high-fidelity digital twin, rendered in real-time, serves as the backbone of this knowledge base, delivering it to the operator through XR-enabled devices. The assembly information from Polarion is compared in order to check the sequence of the activities performed.

At the same time, the system automatically records the best fitting activities or delays from workers, improving the ideal time needed for each step and identifying when it takes too long, which helps to continuously improve the workflow using data. Beyond operational guidance, the digital environment serves as an immersive training platform, enabling operators to rehearse tasks in a risk-free virtual setting, thereby accelerating skill acquisition and increasing task confidence.

The same pipeline is used for the design digital twin and the production digital twin, with significant differences in the data feed and features, as shown in Table 1.

Table 1. Functional comparison between design and production digital twins.

Feature	Design Digital Twin	Production Digital Twin
Phase	conceptual/ pre-production	in-operation/ execution
Source of Data	PLM, ALM (Fitting Instructions), EAS (environment scan)	AFOS + CTDS (real-time sensor & ML input), ALM (Trace Cards)
Purpose	Simulation, Layout validation	Process validation, live monitoring
Dynamics	Static or versioned (e.g. planned state)	Dynamic and continuously updated
Usage	Virtual commissioning, planning	Monitoring, feedback, worker assistance

3.2. Automated Compositing

We present an overview of the detection logic for assembly monitoring, USD-Sublayer configuration and stage control via Extensions, bridging the gap between digital representations and manual execution. As an example, we discuss the assembly of electric aircraft propulsion systems for a hybrid engine stator.

3.2.1. Use Case Description: Digital Twin for Hybrid Aircraft Drive Assembly

The Nucleus Workstation serves as our local Omniverse server, which represents all application objects using the standard USD format and gives us access to the control infrastructure. It enables collaborative interactions with USD assets through tools like USD Composer, Isaac Sim and various third-party applications through dedicated connectors and custom extensions. The Omniverse USD API wrappers allow us to aggregate diverse data sources into a multi-layered USD hierarchy, as illustrated in Fig. 3. Modular Scene composition in Omniverse Composer. Layers stack assets, animations, or modifications, with higher layers overriding lower ones where necessary to assemble a cohesive scene.

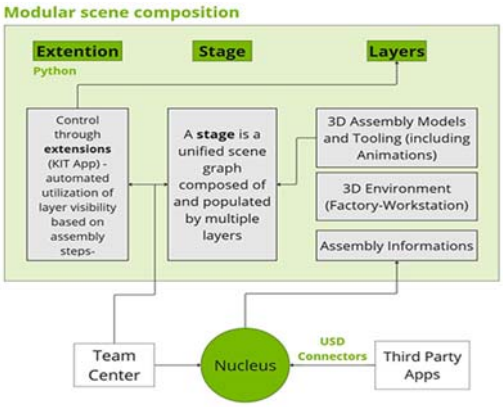

Fig. 3. Modular Scene composition in Omniverse Composer.

The USD structure mirrors the physical assembly hierarchy: Each assembly level is represented as an Xform, which is a scene node controlling the position, rotation and scale of its children. Mechanical relationships are reflected via nesting. In the aircraft use case, e.g., screw tightening operations are nested within the motor subassembly. Sublayers define assembly progression ("aircraft base frame installed") and bundle the 3D models required for each step. This modular approach supports animation integration (e.g. training systems toggling sublayer visibility to guide assembly workflows) and environmental elements (e.g. workbenches, manufacturing floors to support XR training environments and increase immersion), which are added as compressed payloads to balance detail and performance. Additionally, fitting instructions extracted from trace cards outlining the assembly steps can be integrated as custom USD metadata, including positioning and textual information in USD format.

Our architecture clearly separates design concerns encompassing CAD data, assembly logic, and animation workflows, from operating aspects driven by sensor inputs, contextual inference, and operator guidance.

3.2.2. Increased Factory Performance with AGVs

Each workbench display presents the relevant 3D views of the aircraft drive to the respective workers without requiring their intervention. The assignment of assembly stages to the workers is configured in our worker assistance system built upon the Omniverse USD Composer App Template (from the "kit-app-template"toolkit[1]) with KIT extensions [28-30].

These extensions facilitate the end-to-end monitoring of the aircraft drive assembly, and also the cooperation between automated guided vehicles (AGVs) and our Assembly Twin. AGVs are provided with stage position data from the factory OPC UA server and send back their locations, allowing their visualization within the 3D scene in real time.

While the Assembly Twin monitors AGVs, it does not maintain a spatial model of the factory from which navigation plans for the AGVs are derived from. Such plans that govern how AGVs physically move, which doors are passed, and which rooms are traversed must be provided by an external system.

Despite the added complexity, the use of AGVs is worth considering since they significantly contribute to the performance of processes such as component commissioning, which is time-critical in the aircraft drive assembly use case. The AGVs distribute parts across pre-assembly and end-assembly workstations taking the current machine loads into account. The real-time visualization of this process allows operators and facility managers to observe this load-balancing behavior within the digital twin. Adapting to layout changes requires deriving new, executable navigation programs reflecting the physical reconfiguration. Generating such routes from a revised floor plan is outside the scope of the Assembly Twin.

The visualization remains continuously synchronized with the factory workflows, as the KIT event message bus toggles the relevant sublayers to reflect the corresponding transition, whenever the completion of a step is detected. The fitting instructions for the following step are then rendered using the Omni.UI API, and the corresponding assembly-state animation is initiated.

3.2.3. AI-Powered Guidance

The AI-based guidance subsystem provides operators and facility managers with a live view of the assembly state at each workstation, including the steps required for completion and real-time feedback during the assembly process. Zero-shot and few-shot models allow process-agnostic analysis, reducing the need for

[1] Version 106.5-109.3

retraining when components or assembly sequences change. The AI-powered action detection and worker guidance components are shown in the right part of the architecture diagram in Fig. 4.

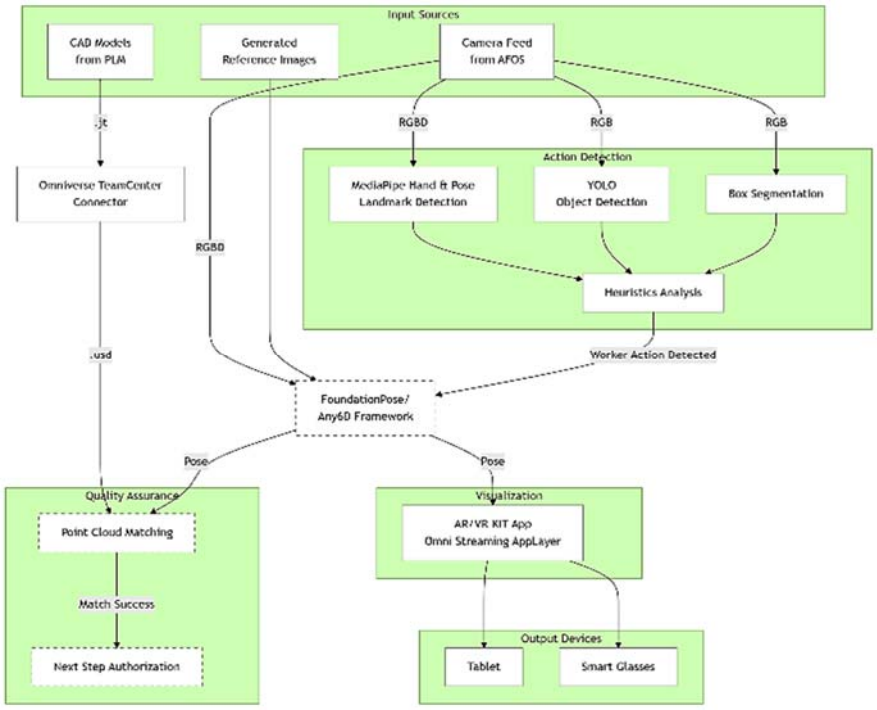

Fig. 4. AI-Powered Assembly Twin Components.

When a worker begins an action, MediaPipe Hands [31] detects the corresponding state transition. For reliable action detection, our movement tracking includes lift-off distances, workbench zones, hip-to-arm distances, and wrist-elbow-shoulder angle ratios. From this data, observed movements are classified. Hand proximity to commission boxes indicates, e.g., grasping events.

Object detection is performed with YOLO11 to verify that the required components have been retrieved from the commission boxes and are present at the workbench. Pose estimation based on FoundationPose [32] complements the object detection by providing continuous 6D tracking of components, including partly occluded objects, under variable lighting conditions. FoundationPose requires an initial bounding box and a CAD model in OBJ format for each target object. To eliminate the need for a manual initialization, we use Any6D [33] in combination with FoundationPose, SAM2 segmentation [34] and InstantMesh mesh generation [35], allowing us to start from an anchor RGBD image instead of a CAD model.

Accurate pose estimates are a prerequisite for AR-based assembly guidance, where virtual overlays are registered to physical components in real time. Our AR interface works with tablets, smart glasses, and VR headsets.

In the targeted component commissioning scenario with multiple workstations, the same tracking pipeline can operate at each station independently, as the zero, few-shot and model-free approaches do not require station-specific training data.

As Fig. 4 shows, not only the worker movements (tracked by the MediaPipe Hands) but also the aforementioned components, AFOS and CTDS (Section 3.1), provide input from cameras and other sensors to our guidance system. To orchestrate the multiple AI models involved, we use the NVIDIA Triton Inference Server [36]. AFOS packages camera streams together with the static reference data including CAD files, initial bounding boxes and anchor images. CTDS receives this bundled input, recognizes objects and assembly states and eventually submits inference requests to the Triton server via gRPC. The server executes the models and returns results in the protobuf format [37, 38]. Triton enables concurrent model execution and dynamic batching, which improves throughput and efficiency.

4. BIM2Robot: Keeping AGVs Up-to-Date with the Factory Environment

BIM2Robot complements the Assembly Twin presented in Section 3. We developed this open-source framework independently of NVIDIA Omniverse. However, it proves particularly valuable in automated factory environments such as the shopfloor scenario which we described above to introduce our Assembly Twin.

4.1. Sideloading the Robot Workplace

While the Assembly Twin seamlessly integrates animated visualization with operating procedures, BIM2Robot separates the static environment in the form of a building model from these dynamic processes. BIM2Robot can generate navigation plans from building models, but has no awareness of why a reroute is needed. The Assembly Twin can detect production disruptions and manage workflows, but cannot autonomously translate a production event into updated robot navigation. This directly addresses the gap identified in Section 3.2.2: the Assembly Twin coordinates AGV destinations but cannot supply the underlying navigation plan. BIM2Robot fills this role by deriving executable routes from the building's IFC description, providing the spatial navigation layer that the Assembly Twin relies on but does not contain itself.

BIM2Robot takes IFC-formatted building description files, as they are commonly used in architecture, as input for a conversion pipeline that tokenizes the contained data and re-interprets the resulting token stream as a robot-compatible navigation plan that ROS-compliant AGVs can execute. When these AGVs are deployed to an automated factory, detailed workplace layout information is transferred or "sideloaded" into assembly scenarios like the one described in Section 3.

Unlike additional sensors or cameras, which merely produce more data or alternative views, BIM2Robot generates executable source code for robot tasks directly from the workplace description. This generated source code enables the rapid development of new AGV applications which can go beyond load balancing in an assembly process, such as the one described above. BIM2Robot includes its own GUI for IFC manipulation (shown in Fig. 5) and task authoring: users can interactively select relevant spaces in a factory and integrate robot tasks with the IFC file as mission entities, e.g., the origin and destination of a transportation task, in the form of annotations. With its GUI BIM2Robot supports a human-in-the-loop workflow for task specifications outside the standard manufacturing procedures. These robot tasks may include:

- **Facility inspection:** robots regularly scan technical rooms, record sensor values, and check conditions. Our GUI supports assigning inspection tasks to specific rooms and linking them to the mission;
- **Intralogistics and supply-chain operations:** robots move safely through warehouses and production halls, recognize doors, walls, and paths from the BIM model, and support order picking, material transport, empty-container return, automatic inventory scans and cleanup.

The second gap from Section 3.2.2 is addressed here as well: when the shopfloor is reconfigured, re-running the BIM2Robot pipeline on the updated IFC file produces a revised topological graph and a new mission plan that immediately reflects the new layout.

4.2. The BIM2Robot Translation Pipeline

The BIM2Robot framework is organized as a pipeline-based system that processes a mission-annotated IFC building model and derives multiple robot-compatible outputs. The architecture separates (i) task authoring within the IFC model, (ii) structured parsing and conversion, and (iii) the generation of deployable representations for robotic execution. Fig. 6 summarizes these components and data flow.

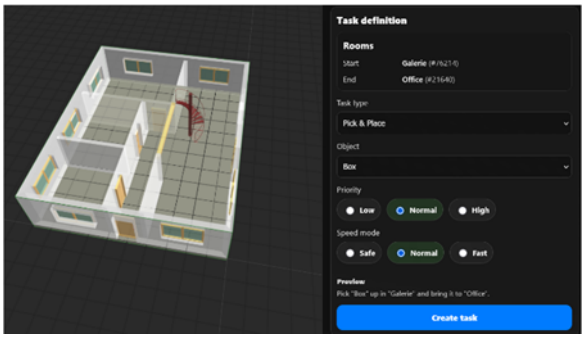

Fig. 5. The BIM2Robot GUI / IFC Editor.

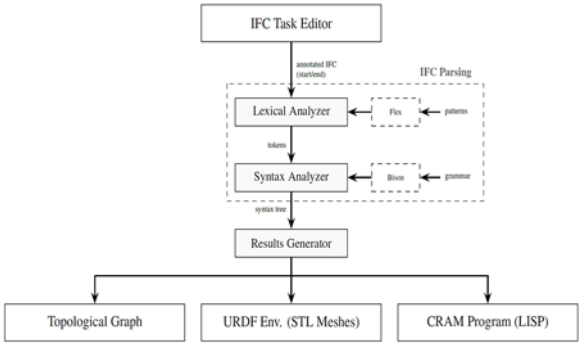

Fig. 6. The BIM2Robot IFC conversion pipeline.

The system workflow starts with the integration of robot tasks added by the users as annotations in the IFC building model. Using our GUI, users can open the IFC Task Editor as a context menu for any entity shown in a building, and assign task-related annotations such as start and target markers, task types, and execution parameters to process elements marked previously. These graphical editing steps result in an annotated IFC file that serves as the input for the subsequent processing steps.

The IFC data is a single source of truth, containing both the architectural building description and the robot-specific task definitions. No separate task description files or external mission specifications are required, as all task-relevant information is embedded in the BIM model.

The pipeline is organized as a classical parsing process consisting of lexical analysis, syntax analysis, and result generation. During this process, relevant IFC entities such as spaces, boundaries, and task annotations are identified, structured, and mapped into an internal representation that serves as a common basis for all subsequent outputs. The lexical and syntactic analysis stages are implemented using the GNU tools Flex and Bison[2] (as we have shown in a concept paper before [22]). Our design is inspired by compiler construction and allows the conversion pipeline to remain independent of specific robotic frameworks (i.e., it is forward compatible with future ROS versions) while ensuring that both geometric information and task-related annotations are consistently interpreted.

In the last processing step, three robot-compatible outputs are produced:

1. A topological graph of the robot workspace capturing rooms as nodes and doors or passages as edges. This graph represents the spatial connectivity of the environment and serves as a high-level navigation structure independent of metric planning;
2. A geometric environment model formatted as a URDF description with associated STL meshes. This representation enables the use of the building geometry in simulation and collision checking in physics-based environments or robot simulators;
3. A LISP program (with ROS library extensions), which encodes the mission logic derived from the task annotations in the form of an abstract syntax tree. This functional code comprises high-level instructions that can be executed using the Cognitive Robot Abstract Machine library (CRAM [39]) and the roslisp[3] interpreter for controlling the robot application.

By separating these outputs, the architecture allows each representation to evolve independently while remaining consistent through a shared source model.

4.3. Methodology

We now examine the BIM2Robot components comprising the data transformation pipeline [22].

4.3.1. Task Encoding

In the first step, robot task-related information, such as start and target locations, e.g. for filming, are added as semantic annotations, allowing the building model to serve as a unified source of structural and task data. Robot task specifications are thus defined at a high level of abstraction and independently of any specific robot platform or execution framework.

The interpretation of the task annotations and their transformation into robot-compatible representations are addressed in the subsequent processing steps.

4.3.2. Token Streaming and Parsing

During token streaming, the IFC file is processed sequentially by a lexical analyzer, which we implemented via GNU Flex[4].

The IFC analyzer (and the other BIM2Robot components) are available via github[5].

Relevant constructs are identified in the textual IFC representation and translated into a stream of symbolic tokens representing spatial entities, relationships, and task annotations. This step abstracts from the original IFC syntax and isolates information required for robotic navigation and task execution.

The token stream is then passed to a syntax analyzer implemented with Bison. Based on a predefined grammar, the tokens are combined into a structured intermediate representation that encodes containment relations, connectivity between spaces, and references to task-relevant locations. This representation preserves the logical structure of the environment while remaining independent of the IFC file format. By specifying a different grammar, users may also switch to a USD[6] building description, as used internally by Omniverse's visualization components. Thus, BIM2Robot can be used with Omniverse (as we show in Section 5) or independently.

4.3.3. Output Representations

The first output is a topological graph, such as the one shown in Fig. 7. Here, rooms are nodes and passable connections are edges. This graph captures the navigational structure of the building and serves as a high-level abstraction for route selection. Internally, the graph is stored as an adjacency matrix, but it can be stored in a comma-separated file and plotted using, e.g., GNUPlot.

A geometric environment model is derived in the form of a URDF description as the second output, including associated mesh files (see Fig. 8). This representation provides collision geometry and spatial context for simulation and execution, enabling the use of standard robotic tools for animated experiments with virtual robots in ROS-compliant simulation software.

[2] https://www.gnu.org/software/bison
[3] https://wiki.ros.org/roslisp
[4] https://westes.github.io/flex/manual
[5] github.com/TimMHoffmann/digital-building-models-to-cram
[6] https://openusd.org

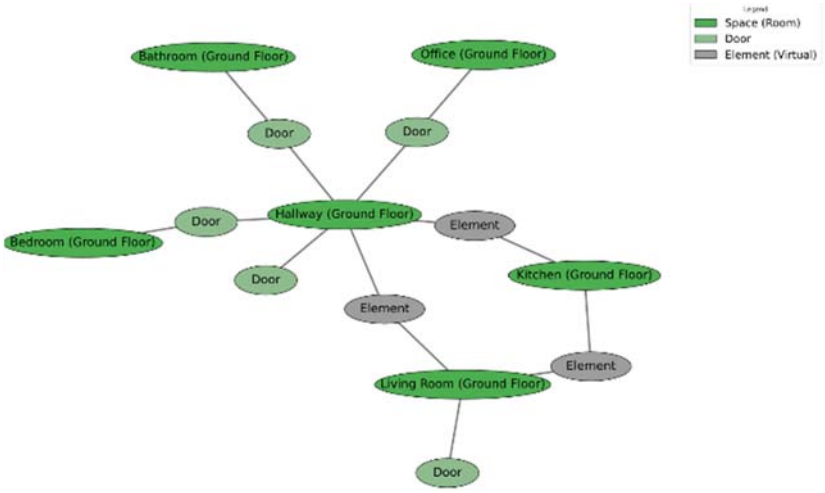

Fig. 7. Topology of the floor of the building from Fig. 5.

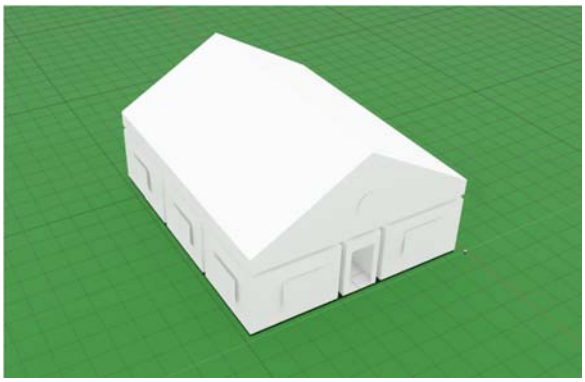

Fig. 8. URDF-based geometric environment model used.

The third output is functional code that encodes task logic derived from the annotated building model, in the form of an executable program for a ROS-compliant robot. Using the structural information from the topological graph and the geometric reference frame of the environment model, the CRAM layer synthesizes executable robot actions such as navigation between rooms or task-specific operations at defined locations.

5. Toward a Unified Framework: Integrating BIM-Based Navigation and Digital Twin-Driven Assembly

BIM2Robot excels at transforming standardized building descriptions into executable robot navigation plans, providing a semantically rich, geometry-aware model of the factory floor that can be easily updated whenever the physical environment changes by re-running the appropriate script. The Assembly Twin, on the other hand, delivers real-time awareness of the production process, monitoring workbench states, guiding workers through complex assembly tasks, and managing the dynamic rerouting of components in response to live production events.

5.1. Complementary Capabilities

These strengths are complementary. Where BIM2Robot understands space, the Assembly Twin understands the process. Both technologies keep humans in the loop: in the case of BIM2Robot, typically software developers, and in the Assembly Twin, typically factory workers. By combining both technologies, we gain what neither system offers alone: a closed loop between production intelligence and spatial navigation, a shared environment model that is simultaneously process-aware and geometrically precise, and the ability to autonomously adapt to both production disruptions and shopfloor layout changes without manual intervention.

5.2. Conceptual Framework Architecture

The proposed framework is structured around a shared data model in which workbenches, storage locations, AGVs, and assembly tasks are consistently referenced across both systems. Workbench identifiers used in the Assembly Twin must correspond directly to spatial entities annotated in the IFC model, as this alignment is the technical prerequisite that makes event-driven communication between the two layers possible. Both layers retain their own internal logic and interact exclusively through defined interfaces, requiring no centralized controller.

5.3. Scenario Walkthrough

To illustrate how the framework operates at runtime, consider the following scenario: a workbench goes offline unexpectedly due to a tool failure. The Assembly Twin detects the failure via AFOS and CTDS monitoring, identifies all components currently queued for the affected workbench, and issues an event message containing the workbench identifier and rerouting priority to BIM2Robot. BIM2Robot

queries the topological graph, determines a viable alternative workbench, and returns an updated ROS-compatible mission plan. The affected AGVs are rerouted accordingly while the Assembly Twin updates the Omniverse visualization to reflect the new production state in real time.

6. Limits and Perspectives

The framework presented here lays the conceptual foundation for an event-driven integration of production intelligence and robot navigation, with practical implementation and experimental validation remaining as the next steps. Event communication requires that workbench identifiers are uniquely mapped to IFC entities. In practice, IFC models are not always kept up to date with the physical shopfloor state, and discrepancies between the building model and the actual layout would result in incorrect navigation plans without the system detecting the inconsistency. Ensuring continuous synchronization between the IFC model and the physical environment is therefore a central focus of the planned implementation work. The integration of dynamic obstacle detection into BIM-derived navigation plans is a natural extension for future work, as is the formalization of the criteria by which BIM2Robot selects an alternative workbench, such as shortest route, available capacity, or task priority. Once the framework is fully implemented, BIM2Robot scripts will no longer be called manually. A production event such as a workbench failure will then automatically trigger a complete navigation replanning and AGV rerouting.

7. Conclusion

This article proposes a conceptual framework that combines BIM2Robot and the Assembly Twin to establish a closed loop between production intelligence and robot navigation. Using an example scenario, we have illustrated how the combined framework enables autonomous adaptation to both unexpected production disruptions and continuous shopfloor layout changes. The reliance on open standards such as IFC, ROS, and OPC UA ensures interoperability and reduces dependence on proprietary infrastructure.

References

[1]. A. Arnouk, N. Behm, C. Borck, Architecture for digital twins in manual assembly within the industrial metaverse, in *Proceedings of the 6th IFSA Winter Conference on Automation, Robotics & Communications for Industry 4.0/5.0/6.0 (ARCI)*, 2026, pp. 5-12.

[2]. T. Hoffmann, F. Dietrich, M. Melzer, J. Dünnweber, A code generation framework for indoor robot applications based on building information modeling (BIM), in *Proceedings of the 6th IFSA Winter Conference on Automation, Robotics & Communications for Industry 4.0/5.0/6.0 (ARCI)*, 2026, pp. 40-44.

[3]. NVIDIA, Siemens and NVIDIA to enable industrial metaverse, 2022, https://nvidianews.nvidia.com/news/siemens-and-nvidia-to-enable-industrial-metaverse

[4]. OpenUSD, Introduction to USD – Universal Scene Description 26.03 documentation, https://openusd.org/release/intro.html

[5]. NVIDIA, NVIDIA announces Omniverse Cloud APIs to power wave of industrial digital twin software tools, 2024, https://nvidianews.nvidia.com/news/omniverse-cloud-apis-industrial-digital-twin

[6]. Dassault Systèmes, Virtual twin experiences for manufacturing industries, https://www.3ds.com/virtual-twin/manufacturing-industries

[7]. NVIDIA, NVIDIA Omniverse for physical AI, https://www.nvidia.com/en-us/omniverse/

[8]. G. Barbieri, A. Bertuzzi, A. Capriotti, L. Ragazzini, et al., A virtual commissioning based methodology to integrate digital twins into manufacturing systems, *Production Engineering Research and Development*, Vol. 15, Issue 3-4, 2021, pp. 397-412.

[9]. S. Gil, P. H. Mikkelsen, C. Gomes, P. G. Larsen, Survey on open-source digital twin frameworks – A case study approach, *Software: Practice and Experience*, Vol. 54, Issue 6, 2024, pp. 929-960.

[10]. R. Schmitt, C. Borck, M. Behm, J. Böhnke, Approach of an asset combination based on RAMI4.0 for the digital transformation and operation of a digital twin, *Procedia CIRP*, Vol. 130, 2024, pp. 724-729.

[11]. B. Wang, L. Zheng, Y. Wang, W. Fang, et al., Towards the Industry 5.0 frontier: Review and prospect of XR in product assembly, *Journal of Manufacturing Systems*, Vol. 74, 2024, pp. 777-811.

[12]. C. Chen, T. Wang, D. Li, J. Hong, Repetitive assembly action recognition based on object detection and pose estimation, *Journal of Manufacturing Systems*, Vol. 55, 2020, pp. 325-333.

[13]. G. Fragapane, R. De Koster, F. Sgarbossa, J. O. Strandhagen, Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda, *European Journal of Operational Research*, Vol. 294, Issue 2, 2021, pp. 405-426.

[14]. Í. R. Da Costa Barros, T. P. Nascimento, Robotic mobile fulfillment systems: A survey on recent developments and research opportunities, *Robotics and Autonomous Systems*, Vol. 137, 2021, 103729.

[15]. M. S. Moura, C. Rizzo, D. Serrano, BIM-based localization and mapping for mobile robots in construction, in *Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2021, pp. 12-18.

[16]. K. Kim, M. Peavy, BIM-based semantic building world modeling for robot task planning and execution in built environments, *Automation in Construction*, Vol. 138, 2022, 104247.

[17]. R. G. Braga, M. O. Tahir, S. Karimi, U. Dah-Achinanon, et al., Intuitive BIM-aided robotic navigation and assets localization with semantic user interfaces, *Frontiers in Robotics and AI*, Vol. 12, 2025, 1548684.

[18]. X. Zhao, C. C. Cheah, BIM-based indoor mobile robot initialization for construction automation using object detection, *Automation in Construction*, Vol. 146, 2023, 104647.

[19]. F. Bahreini, M. Nasrollahi, A. Taher, A. Hammad, Ontology for BIM-based robotic navigation and inspection tasks, *Buildings*, Vol. 14, Issue 8, 2024, 2274.
[20]. M. Kümpel, J. Dech, Semantic digital twins for omni-channel localisation, *IFAC-PapersOnLine*, Vol. 59, Issue 10, 2025, pp. 613-618.
[21]. S. Ebert, J. Mey, R. Schöne, S. Götz, et al., Distributed Petri nets for model-driven verifiable robotic applications in ROS, *Innovations in Systems and Software Engineering*, Vol. 20, Issue 4, 2024, pp. 531-557.
[22]. T. Hoffmann, M. Melzer, J. Dünnweber, Telling robots to do what we want them to, in *Proceedings of the IEEE International Conference on Control, Mechatronics and Automation (ICCMA)*, 2025, pp. 69-75.
[23]. NVIDIA GTC 2024: AI and robotics innovations overview, 2024, https://www.researchgate.net/publication/386442028_NVIDIA_GTC_2024_AI_AND_ROBOTICS_INNOVATIONS_OVERVIEW
[24]. Nucleus overview – Omniverse Nucleus, https://docs.omniverse.nvidia.com/nucleus/latest/index.html
[25]. Introduction – Omniverse Developer overview, https://docs.omniverse.nvidia.com/dev-overview/latest/index.html
[26]. USD connections overview – Omniverse Connect, https://docs.omniverse.nvidia.com/connect/latest/index.html
[27]. Revit – Omniverse Connect, https://docs.omniverse.nvidia.com/connect/latest/revit.html
[28]. NVIDIA Omniverse, kit-app-template, https://github.com/NVIDIA-Omniverse/kit-app-template
[29]. Overview – Omniverse Kit, https://docs.omniverse.nvidia.com/kit/docs/kit-manual/latest/guide/kit_overview.html
[30]. API (python) – Omniverse Kit, https://docs.omniverse.nvidia.com/kit/docs/kit-manual/109.0.3/API.html
[31]. F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, et al., MediaPipe Hands: On-device real-time hand tracking, *arXiv*, 2020, arXiv:2006.10214.
[32]. B. Wen, W. Yang, J. Kautz, S. Birchfield, FoundationPose: Unified 6D pose estimation and tracking of novel objects, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 17868-17879.
[33]. T. Lee, B. Wen, M. Kang, G. Kang, et al., Any6D: Model-free 6D pose estimation of novel objects, arXiv, 2025, arXiv:2503.18673.
[34]. N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, et al., SAM 2: Segment anything in images and videos, *arXiv*, 2024, arXiv:2408.00714.
[35]. J. Xu, W. Cheng, Y. Gao, X. Wang, et al., InstantMesh: Efficient 3D mesh generation from a single image with sparse-view large reconstruction models, *arXiv*, 2024, arXiv:2404.07191.
[36]. Triton inference server: An optimized cloud and edge inferencing solution, https://github.com/triton-inference-server/server
[37]. Triton Inference Server, Inference protocols, https://github.com/triton-inference-server/server/blob/main/docs/customization_guide/inference_protocols.md
[38]. Triton Inference Server, Client source code, https://github.com/triton-inference-server/client/tree/main/src
[39]. L. Mösenlechner, M. Beetz, M. Tenorth, CRAM – A cognitive robot abstract machine for everyday manipulation in human environments, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1012-1017.

Published by International Frequency Sensor Association (IFSA) Publishing, S. L., 2026 (http://www.sensorsportal.com).