From BIM to Autonomous Navigation: Using BIM Models to Enable Autonomous Navigation and Localization in Construction Environments

Jörg Husemann[1], **Maximilian Kunz**[1], **Marcel Suiker**[1] and **Karsten Berns**[1]

[1]University of Kaiserslautern-Landau, Germany

joerg.husemann@cs.rptu.de, maximilian.kunz@cs.rptu.de, suiker@rptu.de, karsten.berns@cs.rptu.de

Abstract -

The automation of construction vehicles and digitization using building information modeling (BIM) are currently two of the most popular research topics in the area of construction. This work combines the two areas by utilizing the information from a BIM model to enable a vehicle to localize and navigate through a building autonomously. A robot equipped with an industry arm should receive marking tasks, which specify a position to navigate to and positions to mark with the robotic arm. These tasks should mimic typical tasks on a construction site, like drilling or cutting holes. The localization within the building is based on LiDAR data, which are compared to the structure of the building, which was received from a BIM model. The path planning uses the BIM model to find a path to the target area. The concept is first tested in a simulation using the Unreal Engine 5, which imports the BIM model and uses simulated LiDAR scanners. Furthermore, the concept is evaluated using a real robot and a building consisting of three rooms and an area of roughly 100 square meters.

Keywords -

autonomous navigation, localization, building information modeling

1 Introduction

Digitalization using Building Information Modeling (BIM) and advancements in automation of construction machines are currently two popular research topics. On the one hand, the advancements with BIM aim to have fully digital documentation of construction sites to monitor the worksite's progress continuously. On the other hand, research in autonomous construction machinery aims to reduce risks for construction workers and mitigate the effects of the shortage of skilled labor. While there is research in both fields, only a few works aim to try to use digital models to improve the processes of autonomous robots. This paper presents an approach in which a robotic platform equipped with a robotic arm should drive through a building imitating a drilling task by marking spots on the inside walls. A BIM model of the environment is used as a knowledge base to localize within the building and to find a path to navigate to given target positions. The localization is done using a LiDAR scanner and is performed in two steps. First, the current room is detected by detecting the outline of the current room and comparing it to the outlines of the rooms in the BIM model. For precise localization, a match between the current scan and a scan of the earlier detected room is compared. By calculating the offset between the two scans and taking the robot's odometry into account, a more precise location within the room is determined. The navigation algorithm uses an elastic band fusion algorithm to navigate the building. As the knowledge base for this algorithm, a grid map is created out of the BIM model. Out of the map, some initial waypoints are calculated using a flood fill algorithm. These points are given to the elastic band fusion algorithm, which coordinates the robot's navigation. The concept is implemented in the Finroc framework[1], which is a framework for intelligent robot[1][2].

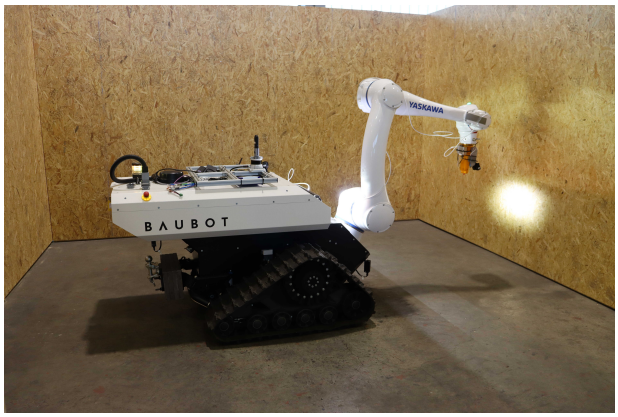

Figure 1. Image of Baubot MRS5-1920 from Baubot equipped with an MOTOMAN HC20DTP produced by YASKAWA.

The experiments are executed using the Baubot MRS5-1920[2], a robotic platform equipped with a

[1]https://www.finroc.org/
[2]https://www.baubot.com/

MOTOMAN HC20DTP robotic arm produced by YASKAWA[3](Figure 1). The experiments aim to drive to a given position within the model and then do a marking on a specified position at the wall. Experiments are performed in both simulations and real scenarios. Unreal Engine 5 is used for the simulation. Here, the BIM model is directly imported as a test environment, and a 3D model of the robot is imported. Both navigation and localization are tested using simulated LiDAR data [3]. For the experiment on the real robot, a setup with three rooms and a size of roughly 100 square meters is built. In this environment, the robot is moving through several rooms to a given target location. After reaching the location, the robotic arm marks a set of points on the wall. The position marked are adapted based on the position of the mobile platform. By doing so, some inaccuracies of the navigation can be compensated during the marking process.

The paper is structured in the following way. Section 2 introduces some previous works utilizing BIM models in the scope of robotic tasks and other related aspects as localization. Then in section 3 our developed system is introduced including the localization, path planning algorithm and the navigation. Section 4 shows the experiments in the simulation environment and with the real robot. Finally the work is summarized in the conclusion 5.

2 Related Work

While there are many theoretical concepts on how the information saved in BIM models could be used to support the use of autonomous robots in construction, only very few works use these concepts in real applications. The most common application utilizing a combination of BIM models and autonomous vehicles is the scanning and monitoring of construction progress. In this area, Pietro et al.[4] proposes a concept to monitor the progress of a construction site. They propose a robotic platform with a sensor setup suitable for the application. A general idea of how a map can be created based on a BIM model and which sensors are needed to localize within the environment is sketched. To validate the concept, simulated tests in a simplified environment are run. The same domain is tackled in the approach of Park et al.[5]. They are using a Spot robot to scan the progress in an indoor environment. They utilize BIM models to plan and optimize the scan positions that the robot has to approach. The target is to minimize the number of static scans needed to cover the whole environment, thus minimizing the needed time to scan the whole building. In future works, the scans should also be registered automatically to achieve a fully autonomous process. A similar approach with a different application is presented by Hu et al.[6]. They present an approach to monitor the air quality in an office environment, documenting it inside a BIM model. An autonomous robot drives through an office environment, measuring the air quality. The measurements are incorporated into the BIM model to help identify problems with air conditioning. Their experiments state a precision of roughly 0.10 m using a Velodyne-16 LiDAR scanner mounted on a four-wheeled mobile platform. The information from BIM models can also be utilized to improve various aspects of robotics, such as localization. Zhao et al.[7] use BIM models to solve the wake-up problem in indoor environments. By detecting features of the environment and matching them with the features saved in a BIM model, they manage to find the robot's position in an indoor environment. If there are not enough features at the robot's current position, the environment is explored until the position can be verified. Depending on the environment, the robot had to travel between 15 m and 50 m until finishing the initial positioning. Doing so, they achieve a translational precision of 0.15 m – 0.21 m and a rotational precision of1 1.2° – 2.4°. Another approach in which an autonomous robot is navigating based on a BIM model is presented by Iqbal et al.[8]. The approach uses a small miniature robot to draw floor plans on a sheet of paper. While this work integrated information from a BIM model to an autonomous robot, the application is far from real scenarios since it uses a small robot with a pen to draw two-dimensional floor plans on a sheet of paper. There are also general concepts on how to use BIM models without a specific application. For instance, Chen et al.[9] present a concept of using BIM models to do global path planning within an indoor environment. They use an Industry Foundation Class (IFC) file of the environment as a knowledge base and generate a map from it. Then, they use a modified A* algorithm to plan a path. The experiments are performed in a simplified building simulation and a model of an example robot. While there are already various concepts on how the models can be used in different scenarios, only very few works show how an autonomous robot can participate in the building progress while utilizing the information saved in BIM models.

3 Localization and Navigation Approach

In this work, two major problems of autonomous locomotion are tackled by utilizing the information saved in a BIM model. First, the robot should localize within the building, which includes identifying the current room and then detecting the precise position of the robot within this room. Second, the model is used to create a map of the building, which is the basis for planning a path to the target position and then navigating it while adapting to the current situation within the environment.

[3]https://www.yaskawa.com/

3.1 Localization

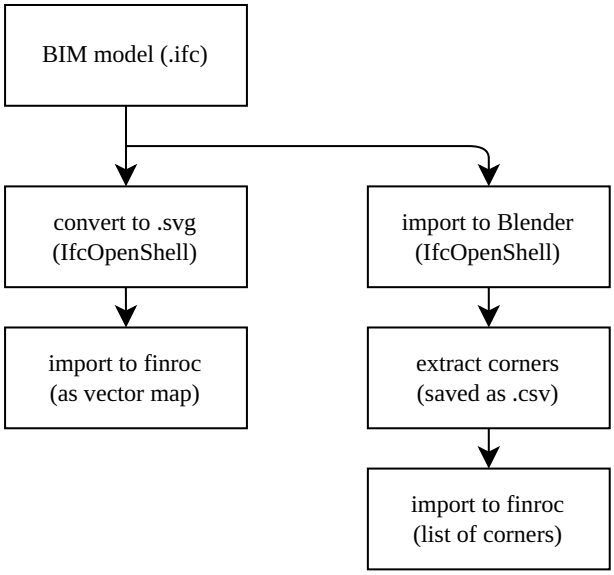

Figure 2. Processing pipeline to use the BIM model for the localization approach.

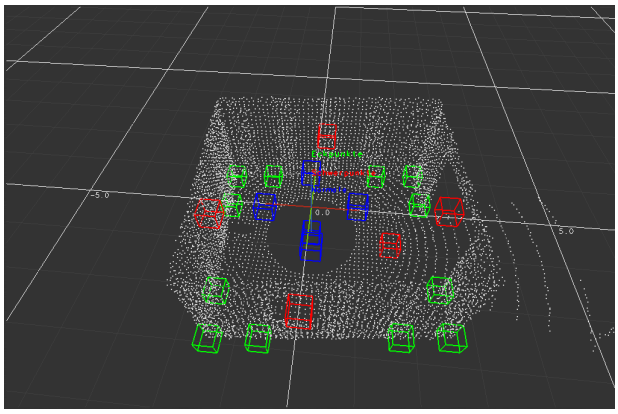

Figure 3. Visualization of feature extraction of a current LiDAR scan in the simulation. The green squares show the corners of the room together with the adjacent walls. The red squares represent the center of gravity of the wall depending on the number of points. The normals of every wall are visualized with the blue squares.

The foundation of our localization approaches is always the BIM model of the building we are currently in. We assume that the 3D model is up to date and contains all necessary components for localization and navigation. As explained in the introduction, we only utilize a LiDAR sensor to detect our environment, thus we are restricted to 3D geometries like walls and corners. We experimented with different pipelines for processing the BIM model stored in a .ifc file to generate a map we can use efficiently, resulting in the pipeline shown in Figure 2. In all of our approaches, we used the 3D modeling software Blender[4] which features an addon for visualizing, creating and working with .ifc files called BlenderBIM[5]. This allows us to use existing Blender features and export functionalities as well as writing our own python scripts inside Blender. Additionally, we made use of the open source IFC toolkit IfcOpenShell[6], e.g. for converting the .ifc file. For our first approach, we used the latter tool to export the BIM model as a .svg file. By doing so, we can interpret the basic geometrical shape of the rooms at a specific height and read the file into our Finroc framework. The downside of losing the Z dimension is neglectable since we extract the data at a specific height, only omitting the wall size. After having access to the building data, we can extract the corners with their corresponding wall normals. After we have created our map of the floor, we do the same for our current scan via the LiDAR sensor. A visualization can be seen in Figure 3. First, we filter the point cloud, then we extract the corners (green) and normal vectors of the wall (blue). As a result, we get a simplified point cloud consisting of corners and normals of the corresponding walls. We match this reduced room representation to our map of all corners using point cloud registration via RANSAC to get the estimated transform. This localization methods works well if we are in a relatively small room where we can see at least two corners. As a fallback, we use basic geometrical features in a similar matter. Similar to the approach by [10] we identify parallel and orthogonal walls since they are invariant to rotational changes and can be found both in the BIM model and the current LiDAR scan quite reliably. For the digital model, our preprocessing steps are similar to the ones we already explained. With the data in .svg format, we extract the walls, convert them into basic lines and extract orthogonal walls (corners) and parallel walls (corridors). In our current scan, we first filter the point cloud, project it to 2D and use the PCL library [11] to extract the walls. In the final step, we compare which features we currently see to what we have stored in the map and try to find the best match. The last influence of our localization is the position estimation based on odometry data we receive from the robot. For short distances, i.e. driving from one room to another, this data was accurate enough for our navigation. After reaching the middle of the room the other localization methods worked as a recalibration for the odometry error.

[4] https://www.blender.org/
[5] https://blender-addons.org/blenderbim-addon/
[6] https://docs.ifcopenshell.org/index.html

3.2 Navigation

The navigation approach consists of two parts. First, a map is generated from the BIM model. Then, the map is used to calculate a preliminary path, which points are used as initial guesses for an elastic band fusion algorithm.

3.2.1 Map Generation from BIM Model

For the navigation, the aim is to use the knowledge saved in the BIM model to create a map of the surroundings. In our scenario, we used the model to create a grid map with the basic structure of the building to be later enhanced with obstacles that can dynamically appear. To do so, some processing steps have to be performed. First, the BIM model available in the Industry Foundation Classes (IFC) format is converted to the Scalable Vector Graphics (SVG) format. The SVG format is usually used for vector graphics and is an efficient way to save geometries. The geometries of the SVG file are rendered to a two-dimensional matrix, which corresponds to the size of the grid map. The geometries' position relative to the BIM model's origin and the map's cell size are considered when mapping the geometries. Finally, the matrix is used to create a grid map in which each element intersecting with one of the geometries is marked as an obstacle in the map, and the other cells are marked as free. Figure 4 shows the BIM model of the test environment and the resulting gird-map. The small cell size for the grid map causes some issues during the calculation of the path planning algorithms. However, it is needed since the passages in the experiment environment are very narrow, considering the large size of the robot. Therefore, the small cell size is used to avoid further reduction of the available space by the map's resolution.

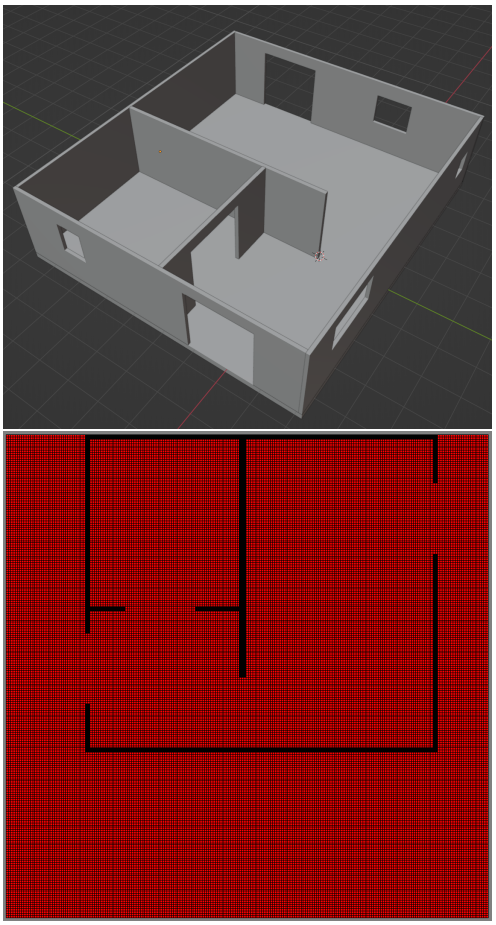

Figure 4. On top the BIM model of the test environment is shown. The image on the bottom shows the resulting grid map with a cell size of 0.06 m. Red indicates free areas and black indicates obstacles.

3.2.2 Elastic Band Fusion

For the actual driving, an elastic band fusion algorithm is implemented [12]. The elastic band method requires a path of target points, which finally leads to the target. Then, the robot always navigates to the next target point. The target points are adapted during navigation based on detected obstacles and the vehicle's position change. As soon as a point is reached, the next point is targeted. This procedure is done until the target is reached. In the case of the marking task, the robot turns to the correct orientation towards the wall after reaching the target point. To calculate the initial points for the algorithm, a flood fill algorithm [13] is used. The algorithm starts at the target pose and assigns to all neighboring cells the distance to the target, which is one in this case. Then, the algorithm iterates to all newly assigned cells and adds the previous cell's distance increased by one to the neighbors that do not have a distance assigned yet. After assigning a distance to the robot's current position, a path can be found by starting at the robot's position and always selecting a neighboring cell with a lower distance than the current one. Theoretically, each of the cells visited when selecting the path can be used as one initial point of the elastic bands algorithm. Since the cell size in our example map is extremely small, the number of points is too high for the algorithm. Therefore, only a subset of the cells is used as the initial target point. Therefore, using a simplified map for the flood fill algorithm, which has a larger cell size, is more meaningful.

The bubbles are created around the calculated target points and are later modified based on the robot's movement. Their size varies based on the critical obstacle, which is the closest obstacle to the bubble 5. The position of each intermediate bubble can be changed based on internal forces between neighboring bubbles, which represent the cohesion strength, and due to an external force, which pushes the point away from the critical obstacle.

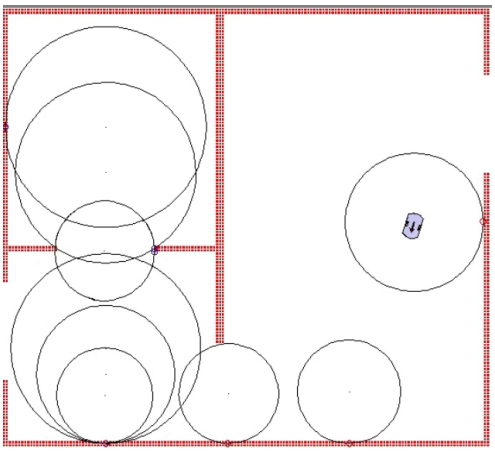

Figure 5. Initial points of the elastic bands fusion. The circles indicate the distance to the next obstacle. If the robots position changes or new obstacles are introduced to the map, the points adjust.

The internal force between two bubbles is defined in the following way

$$\vec{f}_{\text{int},ij} = \alpha_{\text{int}} \cdot \begin{cases} 0 & \text{if } ||\vec{c}_{bi} - \vec{c}_{bj}|| \leq \epsilon \\ \frac{\vec{c}_{bj} - \vec{c}_{bi}}{||\vec{c}_{bi} - \vec{c}_{bj}||} & \text{otherwise} \end{cases} \tag{1}$$

where c_{bi} and c_{bj} are the center points of the two adjacent bubbles. The external force is calculated based on the bubble's distance to the critical obstacle ($\vec{p}_i^*$).

$$\vec{f}_{\text{ext},i} = \alpha_{\text{ext}} \cdot \begin{cases} 0 & \text{if } r_{bi} \leq \epsilon \text{ or } r_{bi} \geq r_{\text{lim}} \\ \frac{r_{\text{lim}} - r_{bi}}{r_{bi}} \cdot (\vec{c}_{bi} - \vec{p}_i) & \text{otherwise} \end{cases} \tag{2}$$

The external force is only larger than zero if the critical obstacle is within the distance threshold r_{lim}. The only exceptions to this behavior are the first bubble, which is fixed to the robot, and the last one, which is fixed at the target position. The shift of a bubble c_{bi} in a discrete time step $t \rightarrow t+1$ is defined in the following way

$$\vec{c}_{bi,t+1} = \vec{c}_{bi,t} + \Delta\vec{c}_{bi} \tag{3}$$

where the delta is calculated based on the two internal forces between the bubble and its neighboring bubbles and the external force on the c_i. The delta is calculated by the sum of the two internal forces and the external force weighted with the value $\alpha_{tot,i}$.

$$\Delta c_{b,i} = \alpha_{\text{tot},i} \cdot \left(f_{\text{int},i,i-1} + f_{\text{int},i,i+1} + f_{\text{ext},i} \right) \tag{4}$$

Whenever the robot approximately reaches the next bubble, the reached bubble is removed, and the next one is chosen as the new target point. This procedure is continued until the bubble representing the target position is reached.

3.3 Marking

The marking task is performed by the robotic arm mounted on the platform. Due to the high precision achievable by the arm, it is responsible for compensating for inaccuracies in the platform's placement. During the marking task, the robot's base is not moving to avoid introducing additional errors. The general process of the marking task is simple. For the movement of the arm, the target positions are first transformed into the platform coordinate system and then into the local coordinate system of the robotics arm. Then, the TCP of the arm is moved to the desired positions. The movement of the arm is done by a function provided by the robotic arm, which is able to move the TCP to a desired local target position. Since only a global target position is given as input, the starting position of the marking progress is calculated to be 0.5 m in front of the marking position. Then, the arm is moved in small steps orthogonally towards the wall. In the target point calculation, the pose of the platform is considered to compensate for inaccuracies in its placement. Since there is no feedback from the arm indicating contact with the wall, a marking tool is used. This tool is equipped with a pen, which is pushed into the tool with a contact. A sensor indicates the contact of the pen. On contact, a new position is set on the arm, which is in front of the wall. As soon as this position is reached, the arm is either moved to the next marking point if another point is reachable from the current position, or the arm is moved back to the driving configuration to enable the platform to drive to the next position.

4 Experiments

To evaluate the concept, tests in simulation and a real test environment are performed. For both experiments, the same BIM model was used. For the simulated tests, the gaming engine Unreal Engine 5 is used. The real experiments are performed in an industry hall where an experimental building with three rooms is built. The setup of the two environments can be seen in Figure 6.

4.1 Hardware Setup

The mobile robot Baubot MRS5-19201 developed by the company Baubot[7] is used for the real experiments. The platform is equipped with the robotic arm MOTOMAN HC20DTP developed by Yaskawa. Additionally, the robot is equipped with an Ouster OS0 64 LiDAR scanner, which is used for perception, and an Intel NUC 13 pro as computing unit. The robot has a width of 1.26 m and a length of 1.88 m excluding the arm. Since the arm has a safety mechanism that is triggered whenever a specific torque

[7] https://www.baubot.com/

Figure 6. Test environment in the Unreal Engine simulation and the real test environment.

threshold is exceeded on the joint, the arm has to be placed in a specific driving position to avoid triggering this mechanism. Due to that, the length of the robot with the arm increases to roughly 3 m.

4.2 Simulated Experiments

A virtual environment based on the model is created for the simulated tests by importing the model into the simulation engine Unreal Engine 5 (Figure 6). Furthermore, a 3D model of the robot is used, which has the same properties as the real machine. Also, the sensors of the real machine are simulated, namely an Ouster OS0 64, the sensor of the marking tool, and the sensors measuring the inertial properties of the robot. In the simulated experiments, the parameters for the elastic band fusion algorithm are verified. Figure 7 shows the setup of the bubbles for a robot, starting in the small room and driving to the large one. The second image of Figure 7 shows the situation after the intermediate room is reached.

The resulting parameters are shown in table 1. The algorithm is also tested in larger environments proving the functionality.

Parameter	Value
r_{lim}	2.00
r_{robot}	0.90
α_{Int}	0.0001
α_{Ext}	0.0001
ϵ	0.050

Table 1. Parameter setup for the elastic band fusion algorithm

4.3 Experiments in Real Test Environment

In the experiments in the real building, the robot is placed in the small room, as also shown in the simulated experiments. Then, a position to drive to in the large room is selected, and a point to mark from this position. Then, the robot is enabled and starts driving to the target position. The initial localization is done using the LiDAR scanner. While driving mainly the odometry is used while resetting it from time to time with a new LiDAR position. The localization is done this way because the odometry is much more stable for short distances. In some places, especially while driving through a door, the LiDAR localization has large errors. Here, the odometry proved to be the best localization method. While driving through the doors, the navigation algorithm also tracks the robot's current room. By that, it is only needed to detect the current room in the beginning and track the current room afterward using the driven trajectory. In the real experiments, the elastic band failed in some cases. Those occurred due to the very narrow passages the robot has to drive. Because of that, fixed bubbles are generated within the middle of the doors and in front of the door in both connected rooms. By that, the robot was guaranteed to be approaching the door at an angle that enabled it to pass through. The most critical passage is shown in Figure 8. It can be seen that the margin for error while driving to the door is minimal, mainly because of the large length of the robot and the arm in the driving position. This issue only appeared in the shown passage. In the other cases, where the door is in the middle of the wall, the navigation is much easier since there is more space to maneuver the robot to pass through.

Figure 9 shows the two most relevant states of the marking process. The left image shows the inital position of the marker. Then the arm slowly moves perpendicularly towards the wall. Since there is very little room for the marker to move into the tool this movement was performed very slowly. As soon as the marker has contact to the wall as seen in the right image, the arm moves back to the driving position and the task is complete.

For obvious reasons, the main property of evaluating such a marking task is to check the accuracy of the marking position. During the process, there are multiple sources of

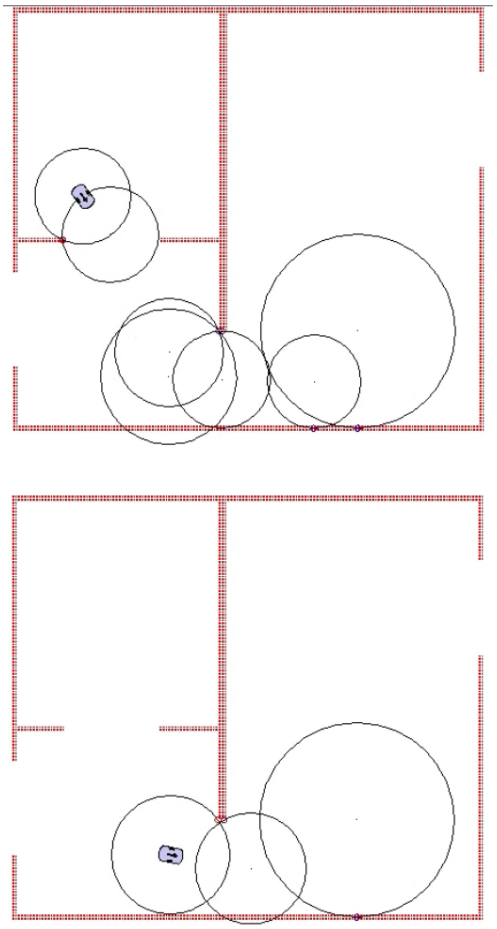

Figure 7. Bubbles of the elastic band fusion algorithm at the start and after driving half of the distance to the target position.

Figure 8. Robot driving through narrow passage in test building.

error, of which some can be optimized by the robotic platform, and some are introduced by the BIM model. During the test run, the precision of the marking varied between 0.2 m and 0.5 m. During the experiments, we found several error sources which have room for improvement. First, adjusting the arm movement to the possible imprecise positioning towards the wall introduced an error. This error grows with the difference between the desired perpendicular position and the actual position toward the wall. This can be either optimized by improving the precision of the driving of the platform or by improving the adaptation of the marking trajectory. The localization error introduces another issue. The quality of the LiDAR localization depends on the visual features as described in section 3. If the marking position is in an area with few features, the error increases. Another aspect that could improve the accuracies mentioned before would be to calibrate the sensors and the arm's position precisely. Currently, the positions are measured manually, which reduces overall precision. Those are the main error sources during our experiments. There are some smaller sources like the repetition accuracy of the arm, which can be up to 0.02 m, and the accuracy of the BIM model compared to the real building. While the accuracy is not yet suitable for real construction applications, improving the previously mentioned points could enable the robot to achieve much higher accuracy.

5 Conclusion

In this paper, an approach to using information from a BIM model to develop an autonomous robot for construction tasks is presented. The proposed concept uses the information to implement a localization approach based on point clouds and a planning and navigation approach. The aim is to navigate through a building while performing marking tasks with the robot's arm. The localization approach detects features of the current room to first identify the room the robot is placed in and then uses the features to precisely position itself within the room. For path planning, a grid map is created using the BIM model. Then, an elastic band fusion algorithm is implemented. A flood fill algorithm is used to find a suitable path to the target point. Then, this path is adapted based on the robot's current position and the LiDAR scanner's measured obstacles. Experiments in both a simulation and a real scenario evaluated the concept. The real tests are executed by a mobile

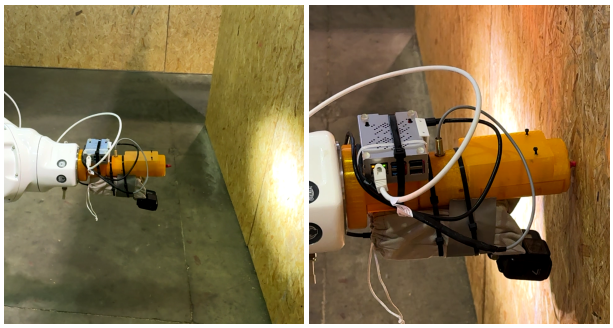

Figure 9. Robotic arm during the marking process. The left image shows the arm after moving into the initial position. The right image shows the arm after a contact before moving back to the initial position.

platform equipped with an arm. For these tests, a building with three rooms is constructed, and the robot navigates through this building to reach a target position. Then, it marks a given position with the arm. The experiments show that using the BIM model is sufficient for the localization and navigation tasks. Nevertheless, there is still room for improvement, especially regarding accuracy.

Acknowledgement

This work is part of the HumanTech project and has received funding from the European Union under Grant Agreement No. 101058236. HumanTech is a three-year project that started on the 1st June 2022 and terminates on the 31st May 2025.

References

[1] Max Reichardt, Tobias Föhst, and Karsten Berns. Introducing finroc: A convenient real-time framework for robotics based on a systematic design approach. *Robotics Research Lab, Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany, Technical Report*, 2012.

[2] Max Reichardt, S Schütz, and Karsten Berns. One fits more-on highly modular quality-driven design of robotic frameworks and middleware. *Journal of Software Engineering for Robotics (JOSER)*, 8:141–153, 2017.

[3] Manuel Philipp Vogel, Maximilian Kunz, Eike Gassen, and Karsten Berns. Lidar-geil: Lidar gpu exploitation in lightsimulations. In *SIMULTECH*, pages 71–81, 2023.

[4] Samuel A Prieto, B Garcia de Soto, and Antonio Adan. A methodology to monitor construction progress using autonomous robots. In *Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 1515–1522. International Association for Automation and Robotics in Construction (IAARC), 2020.

[5] Sangyoon Park, Sanghyun Yoon, Sungha Ju, and Joon Heo. Bim-based scan planning for scanning with a quadruped walking robot. *Automation in Construction*, 152:104911, 2023.

[6] Xi Hu and Rayan H Assaad. A bim-enabled digital twin framework for real-time indoor environment monitoring and visualization by integrating autonomous robotics, lidar-based 3d mobile mapping, iot sensing, and indoor positioning technologies. *Journal of Building Engineering*, 86:108901, 2024.

[7] Xinge Zhao and Chien Chern Cheah. Bim-based indoor mobile robot initialization for construction automation using object detection. *Automation in Construction*, 146:104647, 2023.

[8] Fahad Iqbal, Shiraz Ahmed, Fayiz Amin, Siddra Qayyum, and Fahim Ullah. Integrating bim–iot and autonomous mobile robots for construction site layout printing. *Buildings*, 13(9):2212, 2023.

[9] Zhengyi Chen, Keyu Chen, Changhao Song, Xiao Zhang, Jack CP Cheng, and Dezhi Li. Global path planning based on bim and physics engine for ugvs in indoor environments. *Automation in Construction*, 139:104263, 2022.

[10] Wolfgang D Rencken, Wendlin Feiten, and Raoul Zöllner. Relocalisation by partial map matching. In *Sensor Based Intelligent Robots: International Workshop Dagstuhl Castle, Germany, September 28-October 2, 1998. Selected Papers*, pages 21–35. Springer, 1999.

[11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[12] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.

[13] SV Burtsev and Ye P Kuzmin. An efficient flood-filling algorithm. *Computers & graphics*, 17(5):549–561, 1993.