Towards Real-time Image Localization with BIM models

Rafaella Dantas[1], Simone Peter[1], Xingzhou Wang[1], Miguel Vega Torres[1] and Ann-Kristin Dugstad[1]
[1]Chair of Computational Modeling and Simulation, TU Munich, Arcisstr. 21, 80333 Munich, Germany
E-mail(s): miguel.vega@tum.de

Abstract: In der vorliegenden Arbeit wird die technische Versuchsumgebung *BIMGaze* vorgestellt, welche die experimentelle Beobachtung von Modellierer:innen bei der Erstellung von Building Information Models erlaubt. Für Untersuchungen zum Verständnis der Schnittstelle zwischen Mensch und Maschine haben sich Eyetracking-Systeme bewährt. Insbesondere Usability-Tests und Interaktionsanalysen werden mit dieser Technologie erfolgreich durchgeführt und generieren quantitativ auswertbare Forschungsdaten. Das System *BIMGaze* überträgt diesen Ansatz auf Forschungsfragen der AEC-Branche, die bisher hauptsächlich qualitativ behandelt wurden, sodass mit diesem innovativen System neue Erkenntnisse für die Automatisierung des BIM-Prozesses möglich werden.
BIMGaze simuliert hard- und softwareseitig eine 3D-CAD-Arbeitsumgebung zur Erstellung von Bestandsmodellen. Kombiniert mit einem Trackingsytem, werden dem Nutzer Bestandsdaten in Form von 2D-Plänen und 3D-Punktwolken bereitgestellt. Dessen Interaktion mit diesen Informationsquellen wird bei der Modellierung eines Gebäudes in Revit detailliert automatisch protokolliert. Somit lassen sich vom Modellierer angewandte Suchstrategien und die damit verknüpfte Aufmerksamkeitssteuerung erfassen und untersuchen. Die quantitative Auswertung dieser kognitiven Vorgänge wird durch *BIMGaze* ermöglicht und einer breiteren Forschergruppe zur Analyse zugänglich. Es wird dargelegt, wie die gewonnenen Rohdaten aufbereitet werden und eine intuitive Visualisierung der Ergebnisse erfolgen kann. Dazu wird mittels 2D-Heatmaps und 3D-Heatclouds die Aggregation der Points of Interest des Nutzers durchgeführt. Anschließend erfolgt in einer grafischen Oberfläche die zeitliche Gegenüberstellung der Aggregationen mit den sukzessiv durchgeführten Modellierungsschritten. Zudem wird die Aufmerksamkeitsverteilung des Probanden über die Versuchsdauer illustriert. So können Einblicke in Lösungsstrategien und den Workflow der Bestandsmodellierung gewonnen werden. Ein eigenes Datenschema erlaubt die einfach auswertbare Archivierung der erhobenen Forschungsartefakte und Protokolle sowie deren Wiederverwendung in weiterführenden Untersuchungen.

Keywords: Eyetracking, Datenaufnahme, Laserscan, BIM

1 Introduction

The replacement of human workload by computational support is a recurring request during the last decades. Especially the monitoring of indoor construction progress is nowadays usually conducted manually, leading to error-prone comparisons between the acquired data and actual construction progress.

The enhancement of indoor monitoring progresses with automatic data acquisition and alignment in real-time is thus a promising step towards a dynamic, less error prone and automated construction procedure. Existing approaches require a time consuming Structure from Motion (SfM) procedure to find the correct camera pose in the BIM-coordinate system. This paper aims to overcome this drawback by proposing a method that combines point cloud acquisition with SLAM, perspective detection in BIM view and keyframes and localization improvement to discover the fine camera poses.

The paper is structured as follows: Related work is presented in Section 2. The steps of the proposed concept are described in Section 3. The approach was tested in a case study involving a rectangular-shaped corridor. Results and analysis are presented in Section 4. Section 5 concludes this work.

2 Related Research

Some studies have use the BIM model for *global localization* (without estimate about its initial position) or to solve the *kidnapped robot problem* (in which you know where the robot was but lost communication for a moment).

Acharya, Khoshelham, and Winter [1] introduced BIM-PoseNet, which leverages a deep CNN that uses synthetic images obtained from the 3D indoor model to regress the camera pose without an accurate initial position, achieving an accuracy of 2 m. Haque, Elsaharti, Elderini, *et al.* [2] locate an UAV in the BIM coordinate system by detecting doors and windows in a sequence of images. They use YOLO to detect bounding boxes of these objects in the RGB images captured with a depth camera. This camera simultaneously generates a 3D map with ORB-SLAM2, allowing them to project the detected objects in 3D. Once a sufficient number of doors and windows are detected and localized, they can match with the BIM model. However, once the robot finds its location on the reference map (in this case, the BIM model), the problem is a *pose tracking* task, in which the location and orientation of the robot want to be known all the time while the robot moves. This problem is of more practical importance since while the approximate or corrected starting position could be provided manually, tracking the robot's pose is a problem that the robot itself should be able to perform automatically in order to navigate effectively in an environment autonomously.

To address this problem with the help of a BIM model, Kropp, Koch, and König [3] carried out a study on image to 4D BIM registration using camera pose discovery for each image frame concerning the BIM coordinate system. This method uses line segments as the features. Although the initial registration needed manual intervention, the consecutive images were registered automatically. However, since an

SfM algorithm was used for the rough camera motion estimation, the whole registration pipeline is not applicable in real-time. Asadi, Ramshankar, Noghabaei, *et al.* [4] propose an augmented monocular SLAM algorithm achieving real-time localization performance. They do so iteratively reducing the distance between vanishing points and lines in the image frames and their corresponding BIM views.

Boniardi, Valada, Mohan, *et al.* [5] propose a method that can deal with clutter environments leveraging a convolutional neural network for layout prediction, a particle filter algorithm and a floor plan. Winter, Acharya, M.Ramezani, *et al.* [6] introduced BIM-Tracker, a model-based visual tracking framework, achieves an accuracy of more than 10 cm in environments where even dynamic agents are present, however, there is not much clutter present on their tests.

3 Methodology

The proposed methodology can be divided into three main steps: 1. Point cloud acquisition with SLAM; 2. Perspective detection in the BIM view and in keyframes, and 3. Localization improvement to find the fine camera poses. An overview of the workflow can be seen in Figure 1.

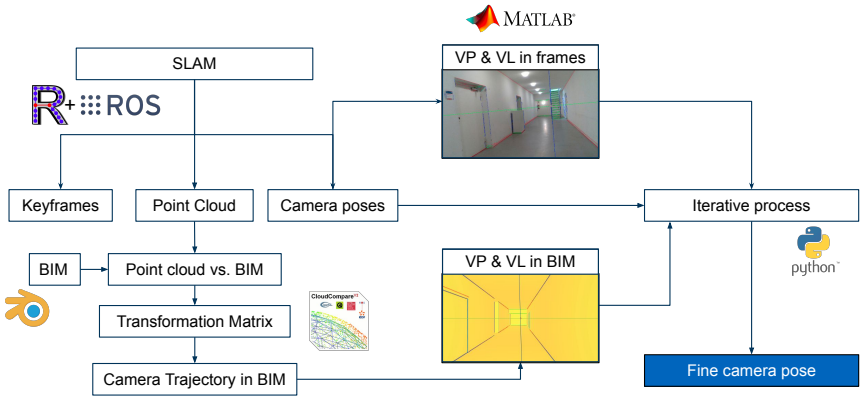

Figure 1: Proposed workflow for fine camera pose correction with a BIM model.

3.1 Point Cloud Acquisition and First BIM Alignment

In order to obtain the rough camera poses, a point cloud acquisition was performed using a RealSense D435i camera with the RTAB-Map SLAM framework [7]. Once a bag file is recorded, a database is created with the help of the “2019-UGRP-DPoom” project [8].

The point cloud database is then opened in RTAB-Map where the poses in the trajectory are optimized, and from where the final rough camera poses and frames can be exported. Besides that, creating a point cloud database is important since it will later be used as one part of the input data in *CloudCompare* (CC) for the alignment between the point cloud and the BIM model.

In order to find the camera poses in the BIM coordinate system the resulting point cloud has to be aligned with the model. This first alignment is done in a semi-automatic manner using *CC*.

The BIM model, originally in IFC format, needs to be saved as an STL file, which can be conducted using Revit. Afterward, the BIM model can be imported into *CC*. Additionally, to render the BIM keyframes, the model is imported into Blender, where the camera object and scripting tools are primarily used. The keyframes are frames extracted every second from the video stream.

The transformation matrix that aligns the point cloud with the BIM ensures that the initial BIM keyframe of the executed trajectory is in the scope of the one from SLAM.

The alignment in *CC* is conducted after manually selecting three reference points in the point cloud and their correspondences in the BIM model. Finally, a transformation matrix from point cloud to BIM is obtained.

Once the correct transformation matrix and scale of the point cloud are found with *CC*, the coordinates of the rough camera trajectory obtained previously with RTAB-Map are transformed to the BIM coordinate system.

3.2 Perspective Detection

One significant task in the computer vision community is to extract 3D information from two-dimensional (2D) images [4]. The estimation of vanishing points (VPs) is important to perform the localization improvement step. In this paper, the algorithm proposed in [9]was applied in order to conduct this step. The vanishing points are points on the image plane, where 2D perspective projections of mutually parallel lines in 3D space intersect. Hedau, Hoiem, and Forsyth [9] propose to compute the parallel lines by the edge detection approach. Afterward, the triplet of orthogonal vanishing points and two vanishing lines are calculated.

In order to obtain the VPs and VLs in the BIM view, a virtual camera is simulated in Blender at the rough camera poses. The parameters of the camera object are set according to the intrinsic parameters of the D435i real depth camera. To be able to generate frames in the BIM view directly in Python and avoid a manual step in Blender, an adaptor was written. Blender is called in the background using the command line rendering.

The script extracts the location, rotation and frame destination from the passed command and renders the respective keyframe in Blender. Then, the module “Data Access” is used to obtain Blender’s internal data and set the new camera information.

3.3 Image Localization improvement

The localization improvement is based on the distance error and the angular error (see Δd and $\Delta\theta$ respectively in eq. 3 between the VPs and VLs of the keyframe and BIM, which are depicted in Figure 2.

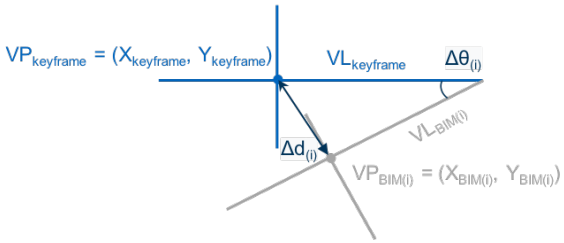

Figure 2: Distance error Δd and angular error $\Delta\theta$ between the VPs and VLs of the current key frame and the corresponding BIM frame. The BIM frame has to be corrected until it matches the current real-world frame, in this way the real camera pose is found in the BIM coordinate system (own illustration based on [4]).

$$\begin{aligned} \Delta d &= \sqrt{(X_{BIM} - X_{keyframe})^2 + (Y_{BIM} - Y_{keyframe})^2} \\ \Delta\theta &= \tan^{-1}\left(\frac{VL_{BIM} - VL_{keyframe}}{1 + VL_{BIM} * VL_{keyframe}}\right) \end{aligned} \tag{1}$$

At first, the location values of the camera poses are corrected. The two corresponding VPs located in the image frame are backward projected from 2D pixel (u, v) to 3D camera coordinates (X_C, Y_C, Z_C) using the intrinsic parameters of the camera.

A correction based on the obtained difference is applied. However, it turns out that the influence on the VPs is relatively small compared to the influence of the VLs. The second part is the improvement of the rotation angles, based on the definition by Euler.

This step is conducted by an iterative, stepwise correction process until a defined threshold is reached. The Euler rotation around the y-axis, also referred to as *pitch*, is directly related to the in-plane computed angle $\Delta\theta$, so that it can be directly applied as a correction for this value.

The *yaw* rotation around the z-axis mainly influences the x-coordinate of the VP.

This coordinate is changed stepwise until a threshold of 3 % for the difference between the x-coordinates of both VPs is reached.

After each correction step, a new BIM frame is generated with Blender. The VPs and thereby the updated distance and angular error are computed. A similar approach is used for the y-coordinate and the *roll* rotation around the x-axis.

4 Results and Analysis

To test the image localization system, a section of an uncluttered corridor was selected. The resulting point cloud and the corresponding BIM model are illustrated in Figures 3 and 4, respectively.

Figures 5 and 6 show the angular and translational errors of the initial rough camera poses (estimated with RTABMAP), and the improved errors after the proposed image localization pipeline is executed. Figure 7 presents the average computational time of each step of the pipeline.

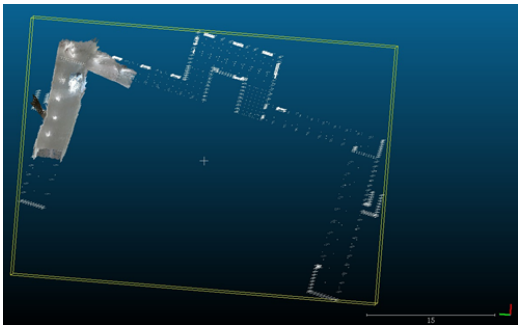

Figure 3: Sparse point cloud reconstructed with RTAB-MAP for a corner with the real-world camera frames used to evaluate the proposed image localization system.

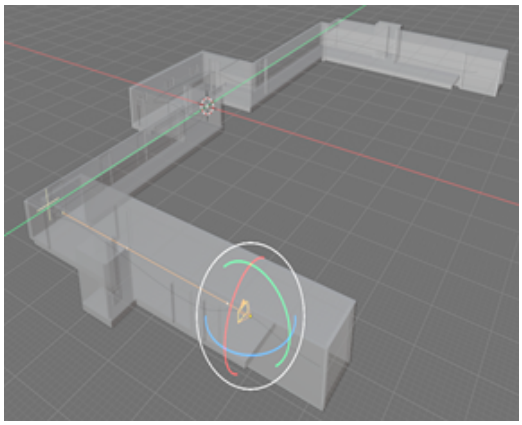

Figure 4: Corresponding BIM model of the whole corridor.

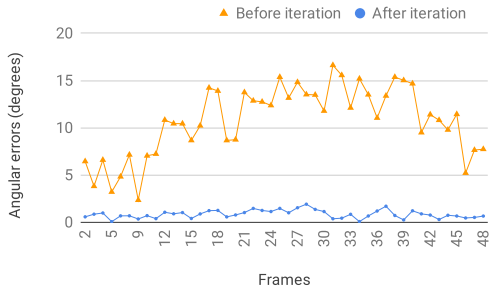

Figure 5: Performance of the image localization system angular errors in degrees.

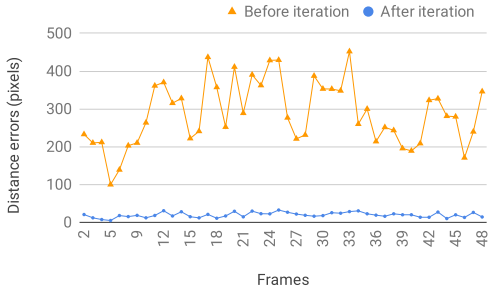

Figure 6: Corresponding distance errors in pixels.

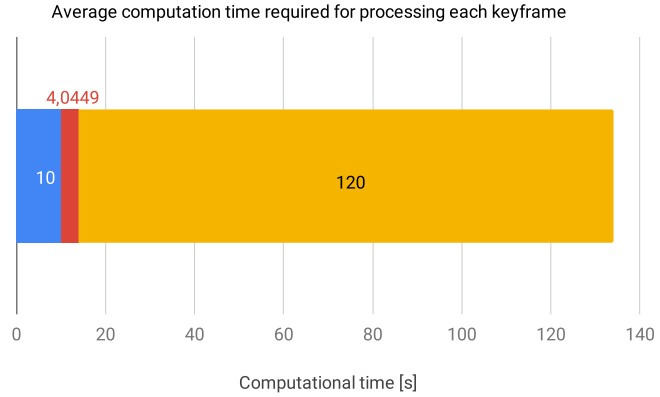

Figure 7: Average computational time performance of the image localization system. In red the time for estimating the rough camera pose with SLAM, in blue for perspective detection (computation of VPs and VLs) and in yellow the time to recover the fine camera pose through localization improvement.

The proposed image localization algorithm achieved a reduction of the average distance error between the VPs of the video and BIM of 93 % and of the average angular error of 92 %, with an average processing duration of 134 s) per frame.

5 Conclusions and Future Work

The experimental results show that the proposed method has a significant potential to improve the camera pose by leveraging an existing BIM model and perspective detection.

The foremost limitation lies in the computational time. By definition, if the speed of processing each keyframe is higher than the speed of the frame sampling, the process can be considered real-time. The proposed method requires, on average, 134 s to process one keyframe. As it can be seen in Figure 7, it does not work in real-time (considering each keyframe is extracted every 1 s). Nonetheless, it is important to consider that the required time per frame varies depending on the quality of the initial pose estimation.

A larger step size or the implementation of the code on a statically typed programming language like C++ could increase the performance of the method substantially.

In addition, this method is based on the computation of VPs. If the algorithm can't detect enough edges required for this computation, which might happen around a corner, close to the wall, in the non-usual case of rounded walls, or in a cluttered room, the VPs cannot be obtained accurately. Consequently, the technique will fail in such cases. One possible solution is to take advantage of other approaches based on plane detection or layout predition instead of only the VP calculation, as it is done in [6] or [5].

Furthermore, a more accurate transformation of the point cloud to the BIM coordinate system could be useful to generate a more precise initial keyframe in the BIM view. For this, visual-based *global localization* or algorithms based on Machine Learning, for example *PoseNet* like in [1], [10], can be implemented, which could serve to partially omit the manual step in CloudCompare. Another possibility is to use a more precise point cloud, which can be generated with the help of other SLAM algorithms such as ORB-SLAM3 [11].

Acknowledgements

The presented research was conducted in the frame of the project "Intelligent Toolkit for Reconnaissance and assessmEnt in Perilous Incidents" (INTREPID) funded by the EU's research and innovation funding programme Horizon 2020 under Grant agreement ID: 883345.

References

[1] D. Acharya, K. Khoshelham, and S. Winter, “Bim-posenet: Indoor camera localisation using a 3d indoor model and deep learning from synthetic images”, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 150, pp. 245–258, 2019, ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2019.02.020.

[2] A. Haque, A. Elsaharti, T. Elderini, M. A. Elsaharty, and J. Neubert, “Uav autonomous localization using macro-features matching with a cad model”, *Sensors (Basel, Switzerland)*, vol. 20, no. 3, 2020. DOI: 10.3390/s20030743.

[3] C. Kropp, C. Koch, and M. König, “Interior construction state recognition with 4d bim registered image sequences”, *Automation in construction*, vol. 86, pp. 11–32, 2018.

[4] K. Asadi, H. Ramshankar, M. Noghabaei, and K. Han, “Real-time image localization and registration with bim using perspective alignment for indoor monitoring of construction”, *Journal of Computing in Civil Engineering*, vol. 33(5), 2019. DOI: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000847.

[5] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard, *Robot localization in floor plans using a room layout edge extraction network*, Mar. 5, 2019. [Online]. Available: http://arxiv.org/pdf/1903.01804v2.

[6] S. Winter, D. Acharya, M.Ramezani, and K. Khoshelham, “Bim-tracker: A model-based visual tracking approach for indoor localisation using a 3d building model.”, *ISPRS Journal of Photogrammetry and Remote Sensing.*, 2019. DOI: 10.1016/j.isprsjprs.2019.02.014.

[7] M. Labbé and F. Michaud, “Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation”, *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[8] Shinkansan, “Slam - 2019-ugrp-dpoom”, *GitHub - https://github.com/shinkansan/2019-UGRP-DPoom/blob/master/SLAM*, 2021.

[9] V. Hedau, D. Hoiem, and D. Forsyth, “Thinking inside the box: Using appearance models and context based on room geometry”, in *European Conference on Computer Vision*, Springer, 2010, pp. 224–237.

[10] D. Acharya, R. Tennakoon, S. Muthu, K. Khoshelham, R. Hoseinnezhad, and A. Bab-Hadiashar, “Single-image localisation using 3d models: Combining hierarchical edge maps and semantic segmentation for domain adaptation”, *Automation in Construction*, vol. 136, p. 104 152, 2022, ISSN: 09265805. DOI: 10.1016/j.autcon.2022.104152.

[11] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM”, *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.