Article

Indoor Localization for Augmented Reality Devices Using BIM, Point Clouds, and Template Matching

Patrick Herbers * and Markus König

Chair of Computing in Engineering, Ruhr-University Bochum, 44780 Bochum, Germany; koenig@inf.bi.ruhr-uni-bochum.de

* Correspondence: patrick.herbers@ruhr-uni-bochum.de

Received: 28 June 2019; Accepted: 6 October 2019; Published: 11 October 2019

Abstract: Mobile devices are a common target for augmented reality applications, especially for showing contextual information in buildings or construction sites. A prerequisite of contextual information display is the localization of objects and the device in the real world. In this paper, we present our approach to the problem of mobile indoor localization with a given building model. The approach does not use external sensors or input. Accurate external sensors such as stationary cameras may be expensive and difficult to set up and maintain. Relying on already existing external sources may also prove to be difficult, as especially inside buildings, Internet connections can be unreliable and GPS signals can be inaccurate. Therefore, we try to find a localization solution for augmented reality devices that can accurately localize itself only with data from internal sensors and preexisting information about the building. If a building has an accurate model of its geometry, we can use modern spatial mapping techniques and point-cloud matching to find a mapping between local device coordinates and global model coordinates. We use normal analysis and 2D template matching on an inverse distance map to determine this mapping. The proposed algorithm is designed to have a high speed and efficiency, as mobile devices are constrained by hardware limitations. We show an implementation of the algorithm on the Microsoft HoloLens, test the localization accuracy, and offer use cases for the technology.

Keywords: Augmented reality; BIM; localization; indoor localization; pose estimation; point cloud; indoor navigation

1. Introduction

Augmented Reality (AR) applications allow a user to interact with digital entities on top of a real environment [1], often linking real objects to digital information. This linking process requires recognizing the semantics of an environment, a task which proves to be difficult. One solution to this problem is localization in digital twins. If the exact location of the device in the environment is known, then a digital twin can provide semantics to the environment. Determining this location is called localization or pose estimation. While systems such as GPS are able to provide a georeferenced location in outdoor environments, they are prone to errors in indoor environments, making them unusable as localization for many applications. Thus, indoor localization can be seen as a separate problem from outdoor localization. In Section 2, we will discuss different existing approaches to global localization and their advantages and disadvantages.

Especially the Architecture, Engineering, and Construction (AEC) industry could profit from using augmented reality in areas yet to be digitalized. One such area is maintenance, where augmented reality can already be used for contextual data visualization and interactive collaboration [2]. Worker efficiency could be increased by allowing direct access of an object's digital twin through augmented reality. Data could be visualized directly on top of the real world, and changes or notes could be sent back to a central service immediately. Direct data availability could also improve the construction process itself. Comparing the as-is state to the as-should-be state of a construction site would be easier and faster, which in turn would uncover mistakes earlier. Applications in these areas would have to rely on localization to attribute data to its real world counterpart.

Global indoor localization requires digital information about the building a device is operated in, such as a digital twin. Digital building information is accumulated through a concept called Building Information Modeling (BIM) [3]. BIM encapsulates geometry, structure, and semantics of a building over the whole building lifecycle. Over the years, BIM has seen a rise in popularity in the construction industries, and thus, the availability and sophistication of digital twins has improved. A more widespread use of digital building models would allow for more applications to procure detailed data directly from available sources, allowing for a direct connection between buildings and users. Therefore, BIM can benefit from a reliable localization as much as localization can be aided by using BIM data.

In this paper, we introduce a novel approach to localization in indoor spaces through the help of BIM. We aim for a computational, efficient 6 degrees of freedom (6-DOF) pose estimation algorithm that can accurately determine the position of an AR device in a known building. To achieve this, we use point clouds, BIM, and template matching.

2. State of Technology

Localization is a subsection of the visual place recognition problem that has been heavily researched in robotics [4]. While visual place recognition concerns itself with the question of whether a place has been seen before, localization also tries to estimate a pose relative to either a previous pose or an arbitrary origin point. This problem is also referred to as global localization, kidnapped robot problem, or wake-up robot problem, where a robot starts from an arbitrary location and has to subsequently determine its location. Generally, the problem of localization can be divided into two categories: localization with and without external sensor input.

2.1. Localization with External Sensors

External sensors can be defined as all active or passive sensors that are considered static in the to-be-localized environment. This can be radio-frequency beacons such as Wi-Fi signals or RFID tags or global navigation satellite systems such as GPS. External sensors are used for estimating a position, which is usually obtained by triangulation of time signals or signal strength. The type of positioning precision can vary widely, from an exact 6D pose (location and rotation) of the device to general descriptors such as floor or room numbers.

The use of GPS is among the most common global localization techniques. The advantage of using the established global network of satellites for positioning is availability, as the infrastructure already exists. While availability is widespread, GPS lacks the precision and reliability needed for localization. Schall et al. [5] have introduced an AR application for utility maintenance that visualizes hidden infrastructure elements. Underground infrastructure could be located directly by workers without having to refer to 2D plans. GPS was used in combination with an inertial measurement unit for pose estimation. The accuracy and reliability of the GPS tracking system was identified as the major shortcoming of the system. Especially in cities with urban canyons, the GPS signals are subject to multicasting as signals are reflected by surrounding buildings, causing the pose to deviate widely [6].

In an indoor or underground environment, large-scale external sensors such as GPS prove to be unreliable. An alternative is resorting to localized sensors called beacons. Beacons emit some form of signal that can be used by the device to estimate the distance to the beacon. One approach is using RFID emitters with a known global location and using the signal's strength as a distance estimator [7]. RFID-based localization can be very accurate but requires extensive deployment effort. As deployment effort is a major concern for widespread use, relying on preexisting beacons can be a major improvement. Especially Wi-Fi signals are commonplace in modern buildings and are usually set up to cover the whole building. Chintalapudi et al. [8] are using Wi-Fi access points as beacons to localize devices, but accuracy is limited to a few meters due to multicasting. Xiao et al. [9] used data fusion to combine multiple radio-frequency sources into one model. Still, the localization depends entirely on the coverage of the beacons, which is not always a given. On a similar note, Blankenbach and Norrdine [10] have achieved indoor localization through the use of magnetic field generators outside of buildings. While magnetic fields can, for the most part. ignore the problem of multicasting, their range is limited to 10–20 m. While beacons can give accurate results, their range is often limited and pre-deployment effort can be a deal breaker for most use cases.

2.2. Localization without External Sensors

The problem of pre-deployment effort can be negated by simply not using external sensors for localization. Localization procedures that do not rely on external sensors have to rely on only the local sensory information and their prior knowledge of the environment. Such a system usually has low running costs, as only the localization device itself has to be kept running. Without external sensors to be installed, such a system can be transferred easily to different locations and environments. A problem arises in self-similar environments. Similar to how a human can be lost in a building by mistaking one place for another, a localization algorithm can be susceptible to self similarities in buildings. In the following, we will discuss different localization methods without the use of external sensors.

The AR localization problem shares similarity with the point-cloud registration problem. The goal of point-cloud registration is to find a transformation to a point cloud such that it aligns optimally with a different set of points. A simple and ubiquitous algorithm is the Iterative Closest Point (ICP) algorithm [11]. While efficient and accurate, the algorithm is often susceptible to local minima. Since localization usually requires registering a subset of a whole model, getting caught in a local minimum is likely. Attempts to make ICP globally optimal have been made [12] but suffer from increased run-time complexity. Magnusson et al. [13] showed that using point clouds for real-time localization is feasible by using normal distribution transforms for loop closure. Using the geometry of the surroundings has the advantage of being independent of lighting conditions, as shown by Desrochers et al. [14]. The authors achieve a localization accuracy of 20 cm and 20° for localizing Unmanned Aerial Vehicles (UAVs) positions in known environments, but execution time for large environments can be up to 60 s. Using point-cloud matching in AR localization offers a significant simplification to the domain: AR localization may apply domain-specific heuristics to achieve a registration of its surrounding environment, while general point-cloud algorithms are built to work on arbitrary geometry. For example, Bueno et al. [15] are able to perform localization by matching point clouds with building models using plane features, as rooms in buildings usually consist of multiple faces perpendicular to each other. Jagbrant et al. [16] are also able to use domain specifics for localization by recognizing and matching tree structures in orchards using a Hidden Markov Model.

A different approach to localization is observing the environment through vision. This is a very similar way to which humans or animals learn to navigate their surroundings: Natural landmarks, such as object edges or lettering, are recognized and remembered. The observer can then use these landmarks to build a map of its surroundings and to simultaneously determine its own position in the map. This procedure is usually called Simultaneous Mapping and Localization (SLAM) and has prominent use in the robotics field. Lowry et al. [4] summarize many of the state-of-the-art visual place-recognition approaches.

Another vision approach is the use of Convolutional Neural Networks (CNNs) to learn mapping between images of the environment and their global positions. Since CNNs are efficient to execute, they can provide a very fast method of localization. CNNs have been successfully applied to localization in outdoor environments using PoseNet [17]. The network that PoseNet provides also performs well in indoor environments [18], with an accuracy of 1–3 m. Acharya et al. [19] have even trained PoseNet on simulated data obtained from BIM models, training on computer-generated images of an indoor space to learn a pose regression. Although SLAM and learning approaches are fast, using them for localization has a significant drawback: The environment has to be traversed to collect features or training data before localization is possible. This training phase takes a significant amount of work as each image has to be mapped to an exact location [20]. Additionally, once the environment changes, the training has to be repeated or improved. Even day and night changes can be enough to confuse a trained network, requiring even more training data [21]. Simulated data can help to mitigate the training problem but ultimately performs worse than using real data [19].

Floor-plan matching is an approach generally limited to indoor localization. The observer generates a 2D top-down representation of the surrounding environment, including the layout of walls and possible landmarks. This representation is then matched to an existing floor plan of the building, resulting in a location relative to the building. Breaking down the problem of localization from three dimensions to only two (or 6-DOF to 3-DOF when including rotation) significantly reduces computation complexity. Of course, this approach requires an existing plan of the building in digital form. Fortunately, with the advent of BIM over the last years, the availability of comprehensive digital models makes this a nonissue in modern buildings. Biswas and Veloso [22] have used floor-plan matching with line segments representing walls to improve robot navigation. Boniardi et al. [23] also make use of floor plans by applying SLAM to LiDAR scans but require an initial state for incremental position correction, making the localization local instead of global. Fast global indoor localization without external sensors is still an open problem.

3. Methodology

For this paper, we approach the localization problem with a floor-plan matching algorithm through template matching. Since point clouds provide a large amount of data, reducing the dimensionality of the data is key. The data is assumed to be in the form of point clouds, as much current hard- and software, e.g., SLAM algorithms and time-of-flight cameras, are able to generate point clouds from depth data. We also assume that the point-cloud data is accumulated over time. Of the incoming frames, every frame is processed into a point cloud. Through Inertial Measurement Units (IMUs) or SLAM, a transform between those frames can be established, linking each point cloud frame relative to each other. The 3D points from multiple frames can then be accumulated to a complete point cloud that includes observed geometry from all frames.

In our methodology, we take advantage of the nature of indoor environments, namely that indoor geometry is not arbitrary. Hallways and rooms are generally made up of straight walls, ceilings, and floors, which are usually perpendicular to each other. Walls are usually perpendicular to the floor and to each other, allowing us to make assumptions about unseen geometry. The most significant information about the layout of the observed environment is, therefore, found in the floor plan of the building. Floor-plan matching reduces the dimensionality of the problem. The position is determined along the x- and y-dimensions, disregarding the height of the pose. Of the rotational pitch, yaw, and roll, only the yaw of the pose can be calculated by floor-plan matching. The missing dimensions (height above floor level, pitch, and roll) can be easily determined by other sensors. The height above the floor level can be independently determined using a ground-plane estimation, which is often a standard feature in common AR frameworks. In our case, we will use a simple ground-estimation algorithm using normal filtering. Rotational pitch and roll can be found by using a device's internal accelerometer, which even in common smartphone hardware is accurate to around 1° for inclination measures. Thus, we assume point clouds to be rotationally aligned with the ground plane. The algorithm is, therefore, able to break down the 6-DOF pose estimation into 3-DOF localization including orientation.

The aim is to find a proper rigid transformation T between the local coordinate system of the AR device and the world coordinate system of a given building model. As the input data, only collected point-cloud data and local position changes of the device are to be used. Finding the transformation should be done in a time-efficient manner so that the localization of the AR device can be done in real time. Matching the update speed of the device is not required, as the device can use existing inertial sensors to track its location between updates of the transformation. The local transformation from the inertial sensors are able to keep the device responsive between updates, allowing the system to keep the necessary responsiveness and interaction required for AR. Run times in the area of milliseconds to a few seconds are, therefore, acceptable. An augmented reality system could request a new transform every few seconds to correct its local transform. The system can use the provided transform between the local coordinate frame and the BIM model coordinate frame to match its surrounding environment to the BIM model. The algorithm's targeted accuracy depends on the applied task. Indoor navigation requires an accuracy of about 1 m, while contextual information displays require to be accurate to a few centimeters.

The proposed algorithm is based on two-dimensional template matching of recorded floor plans onto existing building models. The given BIM model of a building is transformed into a 2D distance map for template matching against a processed point cloud. For a recorded point cloud, we determine several possible rotation candidates around the z-axis. Since it is assumed that the point cloud is rotationally aligned with the ground through accelerometers, only one rotational degree of freedom has to be solved. For the set of rotations, we calculate a transformation over the XY plane using template matching of floor plans. Figure 1 gives an overview of the localization steps. In Section 3.5, we will introduce methods for improving run-time efficiency by caching and updating parts of the matched templates over time.

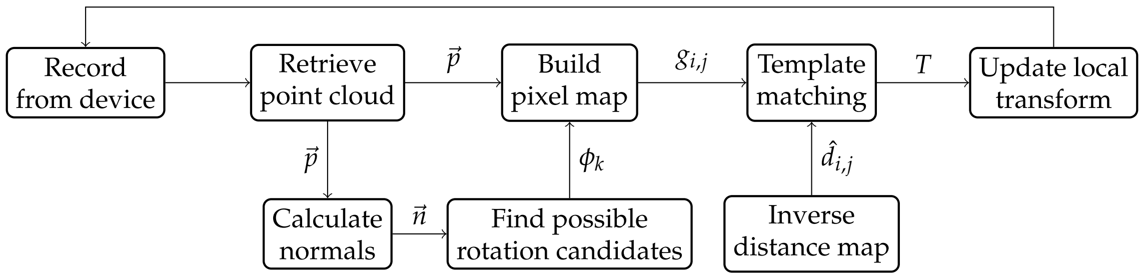


Figure 1. Algorithm overview: $\vec{p}$ is the processed point cloud; $\vec{n}$ is the calculated normals; ϕ_k is the rotation candidates; $g_{i,j}$ is the grid map over the pixel indices i, j; $d_{i,j}$ is the inverse distance map of the building model; and T is the final transformation matrix. The algorithm can run continuously on the device and update when new point-cloud data is available.

3.1. BIM Distance Map

To prepare a BIM model for localization, the 3D model has to be transformed into a 2D map for template matching. For every floor of the BIM model, we set a cross section over the XY plane, which is the basis of the template-matching map. The cross section is placed at roughly the same height as a device would be held above floor level. Since point-cloud data gathered from depth sensors is inherently noisy and the environment may be obstructed by furniture, the recorded point cloud will differ from the original BIM model. These spatial inaccuracies are not accounted for template matching against a binary map representation of the BIM model. An error of 10 cm would have the same negative impact on the correlation as an error of 1 m. Therefore, we match against the inverse distance map of the BIM model. The inverse distance map allows for correction of small errors in the point cloud when template matching while still penalizing inaccuracy. The distance map for the model can also be calculated and cached beforehand and has no impact on efficiency.

To calculate the inverse distance map, every cell position of the distance map is assigned a 3D coordinate on the cross section of the floor. For every cell position $u_{i,j}$, we find the closest point on a wall $v_{i,j}$. The distance to the closest wall equals $d_{i,j} = \|u_{i,j} - v_{i,j}\|$ for every cell. We then calculate the inverse distance map over all pixels i, j:

$$\hat{d}_{i,j} = \frac{1}{d_{i,j} + 1} \tag{1}$$

which results in higher values the closer the position i, j is to a wall. Figure 2 shows an example of the inverse distance map. The closer a pixel on the template is to a wall, the higher the correlation value. Large environments may require a cutoff distance to avoid penalizing inconsistently as the center of a room would be penalizing outliers much more harshly than other places.

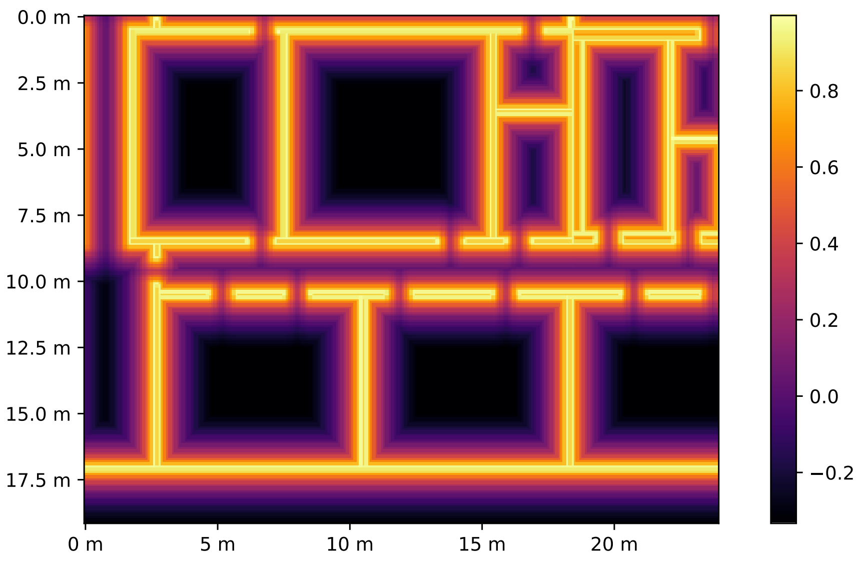


Figure 2. Inverse distance map $\hat{d}_{i,j}$ of a Building Information Modeling (BIM)-model cross section at 1 m above the floor: Higher values are closer to walls. The cutoff distance is set to 2 m in this case. The same scene can be seen as a point cloud in Figure 3. Note that the value range of the map can be neglected, as the map is later used for convolution.

3.2. Finding Rotation Candidates

When a point-cloud frame is recorded, it is processed for template matching. To keep the template-matching run time to a minimum, only a few possible rotations for the point cloud are considered. We therefore determine a set of rotation candidates that are most likely to match the recorded point cloud with the BIM model. Using the knowledge of our domain, we can assume that the walls recorded in the point cloud must face the same direction as the walls in the reference model. We use a convolution of normals in Fourier space to find the best rotational candidates. The rotation candidates are determined as follows:

1. Calculate normals for the point cloud using a normal estimation algorithm. All normal vectors are assumed to be unit vectors.
2. Project every normal vector onto the XY plane.
3. Transform the projected vector from a two-dimensional coordinate system to a polar coordinate system, which yields a rotation and magnitude for every normal vector.
4. Filter out vectors by magnitude. Vectors with a magnitude below a certain threshold can be ignored, as they are most likely situated on a floor or ceiling of the point cloud.
5. Generate a histogram based on the rotation of the vectors. This histogram requires a form of binning.
6. Compute a convolution between the histogram of normal rotations and a target wave, and determine the best fit. The target wave has to match the general directions of walls in the reference model. A reference model with only perpendicular walls would yield four rotational candidates after a convolution with a sine wave $f(\phi) = \sin(4\phi)$. Since the histogram is binned, the convolution can be calculated efficiently by transforming the histogram into Fourier space using a discrete fast Fourier transform algorithm.

The maximum value of the convolution is the optimal rotation ϕ_0 of the point cloud to the reference. Additional rotation candidates ϕ_k can be calculated by adding (in the case of perpendicular walls) 90°, 180°, and 270° to ϕ_0. Depending on the geometry of the building, more rotational candidates may be defined.

3.3. 2D Rasterization

To enable template matching of the point cloud, the point-cloud representation needs to be transformed into a two-dimensional map for every rotational candidate. After applying the rotation to the point cloud, a grid g over the XY plane is created with resolution r. Each pixel on the resulting image represents the number of points contained within a grid cell. Figure 3a shows a floor plan of the BIM model, and Figure 3b shows a corresponding recorded point cloud. The resulting image is then split into three categories: *Wall*, *Floor*, and *No Observation*. This is done through thresholding, where each category is assigned a value of 0, greater than 0, and lower than 0:

$$g_{i,j} = \begin{cases} \beta & \text{if } c_{i,j} \geq t_w, \\ \beta - 1 & \text{if } t_w > c_{i,j} \geq t_f, \\ 0 & \text{else.} \end{cases} \tag{2}$$

The pixel map $c_{i,j}$ contains the number of points of which the xy coordinates correspond to the grid cell $g_{i,j}$. The values t_w and t_f are the thresholds of vertices needed in a cell to be recognized as a wall or floor, respectively. In the resulting image, a wall pixel has a positive value; a floor pixel a negative value; and when no observation has been made, the pixel is set to 0, as seen in Figure 3c. The parameter $\beta \in [0, 1]$ balances floor and wall pixels, giving higher importance to either observed floors or walls. During template matching, walls in the point cloud correlate positively with walls in the model while floors in the point cloud correlate negatively with walls in the model. Cells in which the point cloud shows no observation are set to zero and have no impact on the template matching. This rewards correct matching of walls while punishes missing walls where an observation has been made.

To improve wall matching, we also perform binary dilation on all wall pixels. This dilation creates a fast approximation of the distance calculation done in Section 3.1, improving robustness. The distance δ in pixel of the dilation should roughly match the distance cutoff of the inverse distance map. Without the dilation, large empty spaces would be treated favorably as the amount of wall pixels to floor pixels differs. The dilation is then added to the grid map, resulting in the final grid map as the weighted sum of both maps:

$$g_{i,j} = c_{i,j} + \alpha\, s_{i,j}. \tag{3}$$

The factor α weighs the two maps and should be between 0 and 1. Figure 3d shows an example of a grid map g.

3.4. Template Matching

With the inverse distance map $d_{i,j}$ and the rasterized point cloud $g_{i,j}$ of the floor plan obtained, we try to find the best positional match for the point cloud in the BIM model. For the template matching, we perform a convolution to find the best match for the localization. The convolution is defined as follows:

$$(g * \hat{d})_{x,y} = \sum_i \sum_j g_{x-i,y-j} \cdot \hat{d}_{i,j}. \tag{4}$$

The convolution is performed for every rotational candidate. The maximum value of $(g * \hat{d})$ over all rotational candidates yields the localized position $x_{\max}, y_{\max}$ and the rotation $\phi_{\max}$, which is the most likely transformation T of the point cloud.

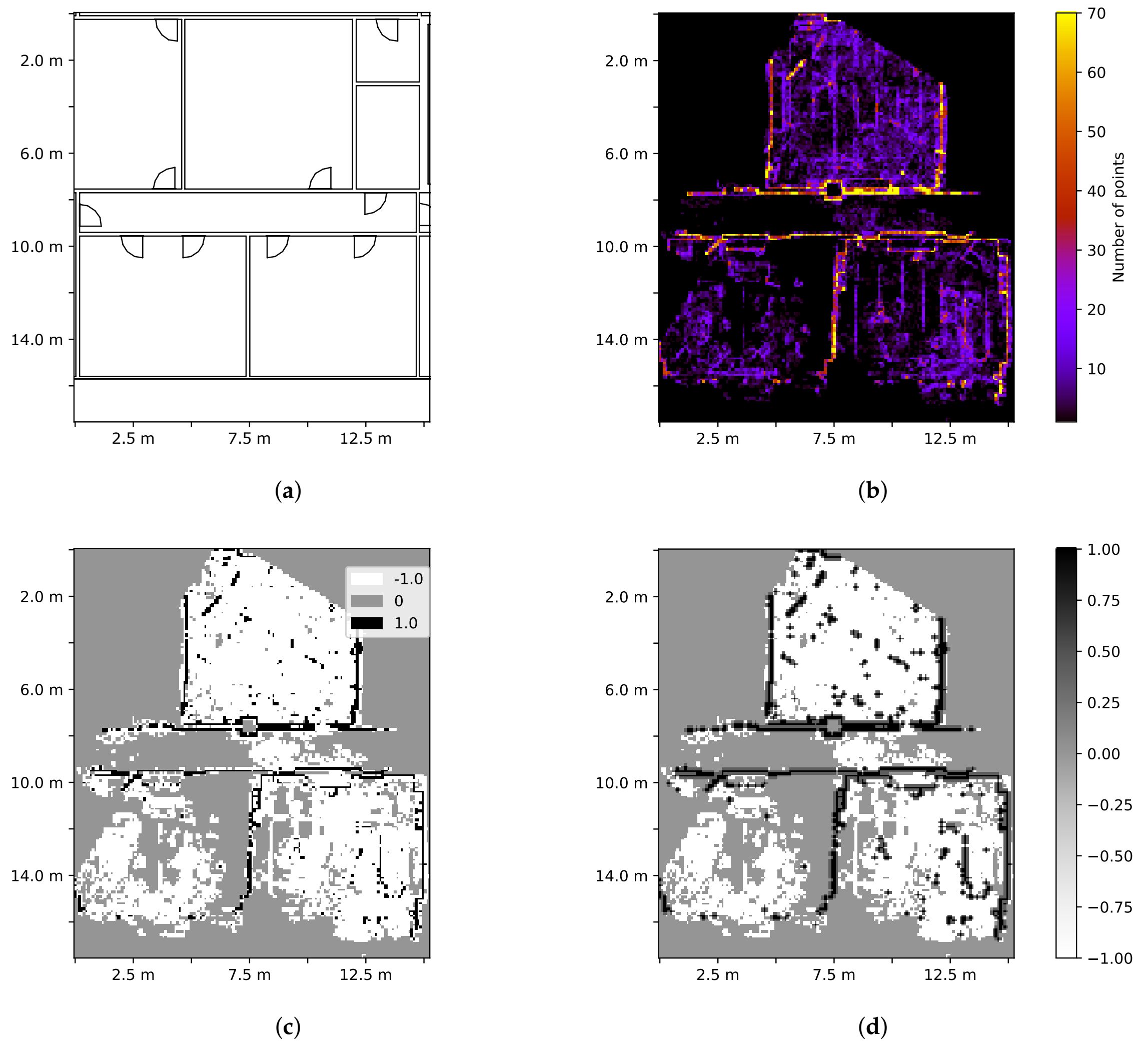


Figure 3. The rasterization process of an axis-aligned point cloud: (**a**) The original layout in the BIM model; (**b**) the axis-aligned floor plan $c_{i,j}$ of a recorded point cloud, where the walls and floors are clearly visible while unobserved locations are in black; (**c**) the floor plan $c_{i,j}$ with applied thresholding, where observed walls have a value of 1, observed floors have a value of -1, and unobserved parts have a value of 0; and (**d**) the combined floor plan $g_{i,j}$.

3.5. Template Caching

The introduced algorithm uses the whole point cloud for each localization pass, but between localization passes, only parts of the point cloud are changed as the environment is observed. Sections of the point cloud that are not observed in an update step are also left unchanged. Since convolutions are expensive, we can take advantage of caching to improve performance. We split up the rasterized grid g from Section 3.3 into multiple two-dimensional blocks G_{kl}, such that we have the following:

$$g = \sum_k \sum_l G_{kl} \tag{5}$$

Since the convolution is distributive, we can compute the convolution of the template matching step for every block individually and sum up the results:

$$g * \hat{d} = \left(\sum_k \sum_l G_{kl} \right) * \hat{d}, \tag{6}$$

$$g * \hat{d} = \sum_k \sum_l (G_{kl} * \hat{d}). \quad (7)$$

This allows us to only recalculate convolutions in areas which have been updated in the last step. When the recorded point cloud increases in size, the computation time of the convolution, thus, stays unaffected.

3.6. Ground Estimation

In the case of matching point clouds of a single floor, a simple ground estimation will suffice for this application. Along the z-axis, points cluster at the floor and ceiling levels. When filtering out points with downward-facing normals, only clusters on the floor are left. The z values of the points are then divided into bins along the axis. The bin with the most points can be established as the ground plane with height $z_{\max}$. The final transformation T can then be calculated:

$$T = \begin{pmatrix} \cos(\phi_{\max}) & \sin(\phi_{\max}) & 0 & x_{\max} \\ -\sin(\phi_{\max}) & \cos(\phi_{\max}) & 0 & y_{\max} \\ 0 & 0 & 1 & z_{\max} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (8)$$

4. Evaluation

To test the accuracy and reliability of the algorithm, we have recorded point-cloud data and pose data using Microsoft HoloLens. Our reference model is a BIM model of a university building, where we match on a 3000 m^2 floor space. The model has multiple self-similar rooms and hallways to test reliability. Figure 4 shows the floor plan of the BIM model used for localization.

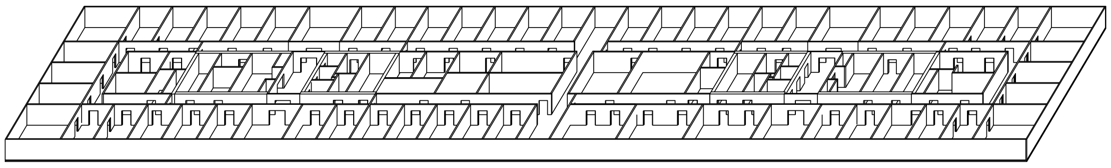
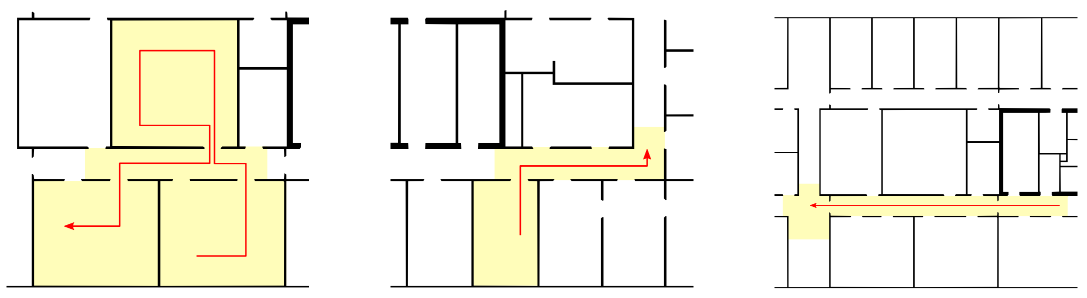


Figure 4. The 3D BIM model used for testing.

4.1. Testing Scenarios

We used Microsoft HoloLens to record point clouds in three different locations in the building. The recorded point clouds cover a number of possible scenarios. Figure 5 shows the three scenarios including the total recorded area in square meters and the walked path. For every scenario, we recorded 20 frames of point-cloud data. The frames were recorded evenly spaced (about 1 m apart) along the walked path. To determine the ground truth, the recorded point-cloud frames have been aligned manually with a ground truth pose $\hat{T}$. The following scenarios have been recorded:

Scenario S1: The device starts in one room, moves out into the hallway, enters another room, turns back into the hallway, and enters a third room (see Figure 5a). The rooms in this scenario are distinct from other rooms in the building model. This scenario tests the accuracy when moving through multiple different rooms. The walked path has a total length of 21.02 m, and the point cloud frames cover a total area of 169.41 m^2.
Scenario S2: The device starts in a single room, exits into the hallway, and moves around a corner (see Figure 5b). Many rooms in the given building model are self similar and can be confused for other rooms. The walked path is 18.65 m in length with 62.45 m^2 of area covered.
Scenario S3: The device moves down a single hallway up until an intersection (see Figure 5c). Straight hallways are often difficult for localization algorithms, as they can be visually indistinguishable at different positions. The path has a length of 24.29 m, and the hallway has an area of 57.94 m^2.

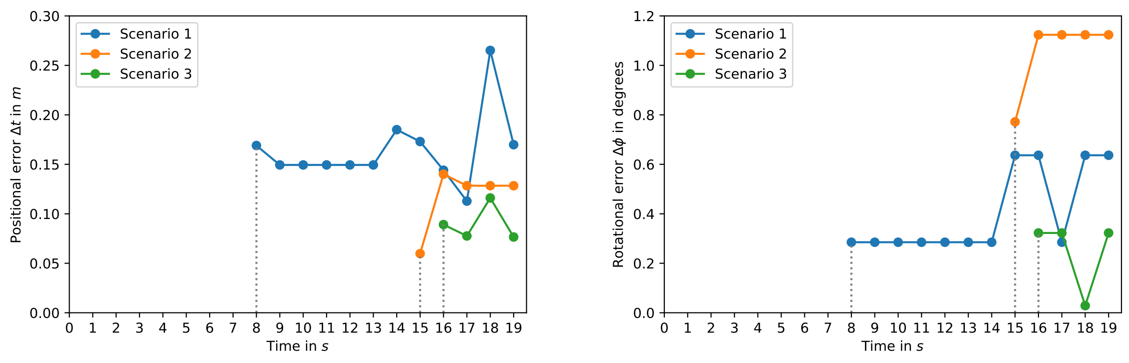


(**a**) (**b**) (**c**)

Figure 5. The three scenarios used for testing localization: Observed geometry is marked in yellow, and the path followed the red line. Red dots show selected points of recording used for testing.

The grid size of the rasterization has been set to 10 cm. The parameters for thresholding have been set to $t_f = 1$ and $t_w = 20$, which depend on the chosen spatial resolution of the HoloLens. To optimize the results, we used a grid search over the recorded data to determine the factors $\alpha = 0.6$ and $\beta = 0.65$.

Each point cloud frame is fed into the algorithm, resulting in an estimated pose T. We then calculate the positional and rotational difference Δt and $\Delta\phi$ as a way of measuring localization accuracy. We categorize a localization as correct if Δt is smaller than 0.5 m and $\Delta\phi$ is smaller than 5°. Figure 6 shows the localization error of correctly localized frames. The average error over all correctly localized point clouds is about 10 cm–20 cm over the xy plane and 1 cm and 5 cm for the ground estimation. Localization errors for incorrect localizations are between 10 m and 100 m. No localization errors between 1 m and 10 m were observed. This gap can be explained as the algorithm mistaking the observed area as a room in a different part of the building. Since rooms are about 10 m in size, localization errors in between are unlikely.



(**a**) Positional error over time. (**b**) Rotational error over time.

Figure 6. Localization error for position and rotation: The gray lines indicate the point in time when the first correct localization was measured. Localization results above 0.5 m and 5° have been omitted from the graph.

The reliability is measured through the point of first correct localization. At this point, the localization error is smaller than 0.5 m and 5°. Table 1 shows the time, distance, and covered area of the first successful localization of each scenario. Notable is that, once the localization is correct, the localization stays correct for the remainder of the scenario.

Table 1. Results of the localization accuracy and reliability for each scenario: $\overline{\Delta t}$ and $\overline{\Delta \phi}$ are the average localization accuracy for correctly localized poses. $\overline{\Delta t_{xy}}$ is the distance error over the template-matched xy plane. $\overline{\Delta t_z}$ is the distance error of the ground-plane estimation along the z-axis. Frame is the point in time when the scenario was first localized correctly, with the according distance walked and area covered at that point.

Scenario	$\overline{\Delta t_{xy}}$ (m)	$\overline{\Delta t_z}$ (m)	$\overline{\Delta \phi}$ (°)	Frame	Distance Walked (m)	Area Covered (m^2)
Scenario 1	0.163	0.041	0.402	8	15.46	111.36
Scenario 2	0.117	0.009	1.053	15	14.25	58.43
Scenario 3	0.089	0.013	0.249	16	19.57	50.03

4.1.1. Scenario S1

In this scenario, we examine localization accuracy in a non-self-similar environment. The second room to be visited is distinct in its dimensions from other rooms. Thus, the first correct localization can be observed in frame 8, when the first room has been completely observed and the second room has been observed halfway. Figure 6 shows that the accuracy of localization is below 30 cm and 1°. The average translational error for correctly localized poses is 16.3 cm, and the average rotational error is 0.402°. The ground estimation has an average error of 4 cm, the highest of the three scenarios, which correlates with the area covered. The best accuracy in this scenario can be found when the device first observes the second room. After this, the localization accuracy decreases again, which points to inaccuracies of the device accumulating over time. Older point-cloud data may become shifted due to accumulative errors in the device's internal tracking systems.

4.1.2. Scenario S2

Scenario S2 starts in a room that is self similar to multiple other rooms in the building. After recording the room, the device is moved outside. When only the room and the adjacent hallway are observed, localization is inconclusive. The guessed position jumps all over as similar-sized rooms can be found through the building. After walking about 14 m, the device observes the corner of the hallway. At this frame, the point cloud is sufficiently distinct and the localization can determine the position with an accuracy of 12 cm and 1°. This scenario shows that self similarities can be mitigated by an increased observed area. The surrounding geometry must be distinguishable from similar environments to give a reliable localization result.

4.1.3. Scenario S3

Scenario S3 shows the localization of a hallway, a structure that can be found at multiple different locations in the building. Similar to Scenario S2, the localization is unreliable until the geometry is sufficiently distinct. During movement, the point cloud is matched to multiple different hallways, as there is not enough information about the hallway for a correct localization. Only when the intersection is reached can the localization accurately match the position with a 9 cm error. Rotational error is the lowest in this scenario at 0.25°.

4.2. Results

As seen in all three scenarios, reliability suffers in self-similar environments. A point must be reached where the geometry of an environment is sufficiently different from other poses. Only then can the localization be reliable, as the localization remains correct after the first correctly guessed pose. In our scenarios, the user had to move the device 15–20 m until a reliable pose could be estimated. The area covered is also related to reliability as first correct localizations were observed from 50–110 m^2. The reliability, thus, depends strongly on the geometry of the building one tries to localize in as well as on the amount and type of environment observed. Especially buildings with multiple self-similar floors would suffer from reliability problems. The more distinct geometry is observed the higher the reliability gets.

If localization is successful, the accuracy is in the range of 5–30 cm to a manually aligned point cloud. This difference can be attributed to inaccuracies during measurement as well as to drifts during recording caused by the device IMU. The naive ground estimation has an accuracy below 5 cm. Due to the rasterization process, the accuracy is bound by the rasterization grid size. A finer grid may improve localization but will decrease performance. This offers a trade-off between accuracy and performance.

The performance of the algorithm is acceptable. On a computer with 4 cores at 3.7 GHz, a localization procedure of the tested scenarios takes less than 100 ms. On the HoloLens itself, the procedure takes about 500–1000 ms. This allows the localization to be considered in real time, as positioning in between localization frames can be handled by existing IMU systems. Since the procedure is highly parallelizable, it is scalable for future hardware. Thus, development in hardware will improve algorithm performance.

5. Conclusions

The proposed algorithm provides a novel way of performing pose estimation of an augmented reality device in a digital twin. The digital twin is obtained through existing BIM models, allowing localization in all buildings which provide such a model. We break the problem of pose estimation down from a 6-DOF problem into multiple independent dimensions. Two rotational degrees of freedom are obtained by hardware sensors, the height above ground is calculated using a ground-plane estimation, and the position in the building is determined by template matching. The input for our algorithm is point-cloud data and sensor information from AR hardware, making the approach independent of external sensors.

Our algorithm achieves higher accuracy than current vision-based approaches such as BIM–PoseNet [19], which is around 2 m, and requires no extensive training phase. While Boniardi et al. [23] achieve a higher accuracy and reliability, the approach requires knowledge about the initial position of the device. The UAV localization presented by Desrochers et al. [14] has a similar accuracy to our algorithm but requires a significantly longer computation time on similar-sized environments, making it not applicable for AR localization. In our algorithm, aligning the recorded point cloud through normal analysis allows for a significant reduction in computation time and applying template matching on floor plans ensures a high accuracy with good performance. Performance of the algorithm proves to be in line with modern AR hardware capabilities, which allows for localization even when no network connection is available. Scalability of the algorithm can be a problem, since complexity scales with the size of the digital twin and the observed environment. Still, the performance impact of the observed environment can be alleviated by using template caching.

The accuracy performs well over all environment sizes and improves with more observed environments. Large localization errors occur in most part when not enough of the environment is observed and may offer widely inaccurate results. This could be mitigated by introducing a certainty value which could inform a user that more information about the environment is necessary for a localization. Since the localization may drift over time, a value for point age could be introduced. Older observations could be weighted less favorably than new observations or be removed from the point cloud entirely. This would allow for dismission of mistakes made by the device's local positioning systems. Self similarities are still a problem for the proposed algorithm, but since all the previously observed environments are considered for localization, we achieve better results in self-similar environments than stateless image-based methods. Accuracy in self-similar environments could be improved by introducing sensor fusion. If external beacons such as Wi-Fi signals are used to narrow in on the correct location, self similarities over multiple floors and reflections could be reduced. Utilizing the semantics of the surroundings may also improve localization, e.g., recognizing certain structures such as doors or windows or recognizing signage.

Our approach shows that floor-plan matching is an appropriate approach for AR localization. With the proposed algorithm, it is possible to compute a mapping between the local device coordinates and the BIM model. From the 2D mapping, the position of the device can be determined to about 10 cm and 1° after exploring the surrounding geometry sufficiently. This exploration phase may require the user to walk 15–20 m, which may depend on the amount of self similarity inherent to the building.

Maintenance workers may benefit from an augmented reality room-layout display with data provided by BIM models or databases. Since reliability can only be assured when enough distinct geometry has been observed, a user would have to explore the space around them before they are correctly localized. This is easily achievable in the case of maintenance work, where multiple locations are likely to be visited. If the device is switched on as soon as the building or floor has been entered, the localization will be reliable by the time a maintenance location has been reached. Especially when using a head-mounted device, the localization can keep observing the surrounding space without interfering with work or requiring special attention. A maintenance worker could then be offered information about the room they are in, could see contextual displays for devices that are at a static location in the building, or could visualize structures hidden behind walls and floors [2,5].

Another possible application for this research is indoor navigation for AR devices. With the mapping between the BIM model and the device coordinate frame, the location of the device in the building can easily be determined and a path to a goal can be calculated by using standard navigation algorithms. To mitigate the requirement of observing sufficiently different geometry, improvements to reliability can be made by including more of the available data of an AR device. External sensors as a secondary source of information may further help against self similarities and computer vision as they can be optionally installed in a building. Learning techniques could improve the knowledge of a building over time and could correct repeated mistakes. Thus, if combined further with other approaches, localization could be made fast, accurate, and reliable.

Author Contributions: Methodology, P.H.; software, P.H.; supervision, M.K.; validation, P.H.; writing—original draft, P.H.; writing—review and editing, M.K.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Azuma, R.T. A Survey of Augmented Reality. *Presence Teleop. Virtual Environ.* **1997**, *6*, 355–385, doi:10.1162/pres.1997.6.4.355. [CrossRef]
2. Abramovici, M.; Wolf, M.; Adwernat, S.; Neges, M. Context-Aware Maintenance Support for Augmented Reality Assistance and Synchronous Multi-User Collaboration. In *Procedia CIRP*; Elsevier: Amsterdam, The Netherlands, 2017; Volume 59, pp. 18–22, doi:10.1016/j.procir.2016.09.042. [CrossRef]
3. Borrmann, A.; König, M.; Koch, C.; Beetz, J. (Eds.) *Building Information Modeling: Technology Foundations and Industry Practice*; Springer International Publishing: Berlin, Germany, 2018.
4. Lowry, S.; Sünderhauf, N.; Newman, P.; Leonard, J.J.; Cox, D.; Corke, P.; Milford, M.J. Visual Place Recognition: A Survey. *IEEE Trans. Robot.* **2016**, *32*, 1–19, doi:10.1109/TRO.2015.2496823. [CrossRef]
5. Schall, G.; Mendez, E.; Kruijff, E.; Veas, E.; Junghanns, S.; Reitinger, B.; Schmalstieg, D. Handheld Augmented Reality for Underground Infrastructure Visualization. *Personal Ubiquitous Comput.* **2009**, *13*, 281–291, doi:10.1007/s00779-008-0204-5. [CrossRef]
6. Xie, P.; Petovello, M.G. Measuring GNSS Multipath Distributions in Urban Canyon Environments. *IEEE Trans. Instrument. Meas.* **2015**, *64*, 366–377, doi:10.1109/TIM.2014.2342452. [CrossRef]
7. Sanpechuda, T.; Kovavisaruch, L. A Review of RFID Localization: Applications and Techniques. In Proceedings of the 2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Krabi, Thailand, 14–17 May 2008; Volume 2, pp. 769–772, doi:10.1109/ECTICON.2008.4600544.

[CrossRef]

8. Chintalapudi, K.; Padmanabha Iyer, A.; Padmanabhan, V.N. Indoor Localization without the Pain. In Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, Chicago, IL, USA, 20–24 September 2010; ACM Press: Chicago, IL, USA, 2010; pp. 173–184, doi:10.1145/1859995.1860016. [CrossRef]
9. Xiao, Z.; Wen, H.; Markham, A.; Trigoni, N. Lightweight Map Matching for Indoor Localisation Using Conditional Random Fields. In Proceedings of the IPSN-14 13th International Symposium on Information Processing in Sensor Networks, Berlin, Germany, 15–17 April 2014; pp. 131–142, doi:10.1109/IPSN.2014.6846747. [CrossRef]
10. Blankenbach, J.; Norrdine, A. Position Estimation Using Artificial Generated Magnetic Fields. In Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation, Zurich, Switzerland, 15–17 September 2010; pp. 1–5, doi:10.1109/IPIN.2010.5646739. [CrossRef]
11. Besl, P.J.; McKay, N.D. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256, doi:10.1109/34.121791. [CrossRef]
12. Yang, J.; Li, H.; Jia, Y. Go-ICP: Solving 3D Registration Efficiently and Globally Optimally. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 1457–1464, doi:10.1109/ICCV.2013.184. [CrossRef]
13. Magnusson, M.; Andreasson, H.; Nuchter, A.; Lilienthal, A.J. Appearance-Based Loop Detection from 3D Laser Data Using the Normal Distributions Transform. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 23–28, doi:10.1109/ROBOT.2009.5152712. [CrossRef]
14. Desrochers, B.; Lacroix, S.; Jaulin, L. Set-Membership Approach to the Kidnapped Robot Problem. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 3715–3720, doi:10.1109/IROS.2015.7353897. [CrossRef]
15. Bueno, M.; Bosché, F.; González-Jorge, H.; Martínez-Sánchez, J.; Arias, P. 4-Plane Congruent Sets for Automatic Registration of as-Is 3D Point Clouds with 3D BIM Models. *Autom. Constr.* **2018**, *89*, 120–134, doi:10.1016/j.autcon.2018.01.014. [CrossRef]
16. Jagbrant, G.; Underwood, J.P.; Nieto, J.; Sukkarieh, S. LiDAR Based Tree and Platform Localisation in Almond Orchards. In *Field and Service Robotics*; Mejias, L., Corke, P., Roberts, J., Eds.; Springer International Publishing: Cham, Switzerland, 2015; Volume 105, pp. 469–483, doi:10.1007/978-3-319-07488-7_32. [CrossRef]
17. Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2938–2946, doi:10.1109/ICCV.2015.336. [CrossRef]
18. Walch, F.; Hazirbas, C.; Leal-Taixe, L.; Sattler, T.; Hilsenbeck, S.; Cremers, D. Image-Based Localization Using LSTMs for Structured Feature Correlation. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 627–637, doi:10.1109/ICCV.2017.75. [CrossRef]
19. Acharya, D.; Khoshelham, K.; Winter, S. BIM-PoseNet: Indoor Camera Localisation Using a 3D Indoor Model and Deep Learning from Synthetic Images. *ISPRS J. Photogramm. Remote Sens.* **2019**, *150*, 245–258, doi:10.1016/j.isprsjprs.2019.02.020. [CrossRef]
20. Huitl, R.; Schroth, G.; Hilsenbeck, S.; Schweiger, F.; Steinbach, E. TUMindoor: An Extensive Image and Point Cloud Dataset for Visual Indoor Localization and Mapping. In Proceedings of the 2012 19th IEEE International Conference on Image Processing, Orlando, FL, USA, 30 September–3 October 2012; pp. 1773–1776, doi:10.1109/ICIP.2012.6467224. [CrossRef]
21. Milford, M.J.; Wyeth, G.F. SeqSLAM: Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 1643–1649, doi:10.1109/ICRA.2012.6224623.

[CrossRef]

22. Biswas, J.; Veloso, M. Depth Camera Based Indoor Mobile Robot Localization and Navigation. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 1697–1702, doi:10.1109/ICRA.2012.6224766. [CrossRef]
23. Boniardi, F.; Caselitz, T.; Kümmerle, R.; Burgard, W. Robust LiDAR-Based Localization in Architectural Floor Plans. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3318–3324, doi:10.1109/IROS.2017.8206168. [CrossRef]