ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

Floorplan Localization
Visual Localization in 2D Floorplans

Master's Thesis

Martin Inauen
Department of Mechanical and Process Engineering

Advisors: Lukas Bernreiter, Magic Leap
Marco Karrer, Magic Leap
Patrik Schmuck, Magic Leap
Julia Jiaqi, CVG
Supervisor: Prof. Dr. Marc Pollefeys
Computer Vision and Geometry Group
Department of Computer Science D-INFK

September 29, 2024

Abstract

The challenge of visual localization is traditionally addressed by matching images to a pre-collected database of images or a 3D map of the environment. In this thesis, an alternative approach that utilizes a 2D floorplan of an indoor environment for localization is proposed. This removes the need of mapping the environment first before using it for localization. We propose a method that learns to predict a bird's-eye view (BEV) of the room layout from a given image. Unlike conventional methods, this BEV prediction is treated as a probability distribution over various depth hypotheses. By directly leveraging this distribution, the model is able to account for uncertainty more effectively. In the inference step, this BEV is matched over rendered views from the floorplan, resulting in a probability distribution of a discretized state space. Additionally, the method addresses the challenge of non-upright images through a data augmentation step, ensuring robustness across various image orientations present on mobile devices. Finally, single-image predictions are efficiently combined in a histogram filter, enabling the model to track the belief over time to do sequential localization. This approach removes the ambiguities inherent in single-image localization, resulting in more accurate and reliable outcomes. Due to the lack of indoor datasets with a semantically labeled floorplans for sequential localization, a dataset of real environments was collected in this work.

Acknowledgements

First and foremost, I would like to thank my supervisors Lukas Bernreiter, Marco Karrer and Patrik Schmuck for giving me the opportunity to complete this thesis with the Localization Team at Magic Leap Zurich. Their regular feedback and guidance throughout the process were invaluable to the development of this thesis. I am especially grateful for their assistance in creating the dataset, as well as for their insightful advice on both the presentation and writing of this thesis. I would also like to thank the Computer Vision and Geometry Group (CVG) at ETH Zurich, with special appreciation to Jiaqi Chen and Paul-Edouard Sarlin for their administrative support. Finally, I would like to thank my family and friends for their encouragement during the time of writing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Localization is an important area of research in computer vision, playing a crucial role in numerous AR/VR (Augmented Reality/Virtual Reality) applications for both head-mounted and handheld mobile devices. It is also of significant practical importance to the field of robotics. Depending on the environment, different sensor measurements can be used to do localization. While for outdoor scenes GPS can give a good prior for localization, in indoor environments this signal is often distorted and not helpful. WiFi fingerprinting is one of the widely used techniques for indoor localization. However, not all indoor environments have these access points and thus a visual localization using only the sensors of the device is desired. Visual localization or image-based localization involves determining the camera's position and orientation based on a given image or sequence of images within a known environment. Most of the existing work use a pre-collected database or 3D model of the environment to do localization [20, 21]. One drawback of these approaches is the storage and upkeep of the maps, which may become outdated quickly as environments change over time. In contrast, 2D floorplans are readily available for most buildings, are lightweight and usually don't change over time. These floorplans offer a generic representation of the space, free from changing elements like lighting or furniture. Still they contain sufficient information for humans to orient themselves within the floorplan. Therefore, this work focuses on the challenge of self-localization using such floorplans.

Visual localization in 2D floorplans has already been tackled by previous work. While some of these work only use panorama images [14, 8], other work use LiDAR or depth images [13, 2] which are usually not present in mobile devices. Furthermore most existing work only uses upright camera images to do localization, or make certain assumptions of the ceiling height which restricts the usage. A goal of this work is to only use the 2D floorplan without making any assumptions about camera or ceiling height. 2D floorplans are not standardized and contain different information. While some floorplans also label the different room types such as bathroom, kitchen or office space, other plans don't cover this information. Since the doors and windows are recognizable in most floorplans, in this work we want to incorporate the usage of this information. The sparse information present in floorplans make the problem of single-image localization inherently ambiguous. Thus, combining the predictions from a single-image over time is crucial to increase performance. It is very important to keep track of multiple pose hypotheses in this ambiguous setting.

The goal of this work is to showcase a visual localization pipeline that works for single image localization as well as sequential localization in real indoor environments. Since there is a lack of realistic indoor datasets for sequential localization, the collection of a dataset is an important step. The ultimate goal of this project is to develop a demo that operates on an AR device and showcases indoor localization in a 2D floorplan in a realistic setting.

Chapter 2

Related Work

Most well-established methods in indoor visual localization try to estimate the 6-DoF pose by matching image features to a set of reference images or a 3D model. Image retrieval methods [1] find the most similar image in a database of image and estimate the pose based on the retrieved image. Other methods using the SfM (Structure from Motion) map mostly rely on creating 2D-3D correspondences between the image and the pre-collected map by matching local descriptors [18, 2]. A disadvantage of these approaches is, that the environment has to be explored a-priori in order to build a map.

The idea of using pre-existing floorplans for localization has been tackled with several methods. While some of the methods try to localize in a floorplan using a LiDAR scan [2, 13], other methods also use an image-based approach [4, 11, 14]. [5] uses the image to build a online pointcloud and use this pointcloud to compare it against a 3D-floorplan by aligning the structural lines. [3] extracts the room edges from the image and compare them against the floorplan layout. These methods usually assume a ceiling height to infer the 3D geometry and often assume a camera height as well. In [8] a network is trained to hallucinate the 3D structure of the floorplan and used in a second step to compare a rendered view to the image. The rendering step of this method is quite expensive and the method lacks the usage of semantics. [14] extracts circular features from sampled points on the floorplan and compares them against image features. Similarly [4] uses the predicted depth from the image and compares it against 1D-depth rays sampled from the floorplan. Although this method predicts the depth uncertainty in a first step, this uncertainty is not used in the downstream tasks since the mean depth is used. Furthermore no semantics are used, which are usually present in floorplans and provide a strong cue as shown in [14].

Using multiple image-frames can improve the robustness of localization, eliminate the scene ambiguities and boost the performance. The usual approach for sequential localization utilizes a bayesian filter to maintain a probability distribution over all states in an online fashion. When the belief is assumed to be gaussian, the bayes filter is equal to the kalman filter [10]. While a histogram filter as used in [4] represents the belief distribution as a grid of discrete probabilities, the particle filter [11] approximates the distribution using a set of particles. [11] uses a particle filter as a recurrent neural network RNN specifically for floorplan localization. This approach suffers from particle degeneracy and the resampling of the particles takes a lot of time. [4] uses a histogram filter with grouped convolutions for the measurement update and maintains a posterior distribution in an efficient way. This approach is especially desirable if parts of the processing are computationally expensive and as the state space is static this computation can be done offline.

Monocular depth estimation, which refers to estimating depth from a single image, has attracted significant attention recently [23, 16]. Monocular depth estimation suffers from scale ambiguity and thus adaptation to different sensor configurations is hard. As shown in [6], aligning the field-of-view (FOV) of two datasets prior to training can improve the performance for domain adaptation. Monocular depth estimation can rely on semantic cues and it has been shown that both tasks have beneficial synergy [19, 9]. Therefore

it can be beneficial for depth estimation to include the semantics of the floorplan as well.

BEV prediction has gained a lot of attention, especially in the autonomous driving domain. [15] combines several camera views and flattens the features to predict a BEV. Similarly [19, 17] maps image columns to a polar ray to estimate a segmented BEV in an end-to-end fashion. Other works try to estimate the position of objects in the image and project those object in a second step into a BEV [24].

Chapter 3

Method

3.1 Problem Formulation

The goal of this work is to localize a RGB-image $\mathbf{I}$ or a sequence $\mathcal{I}$ of k images starting at time t such that $\mathcal{I} = \mathbf{I}_t, \mathbf{I}_{t+1}, \ldots, \mathbf{I}_{t+k}$. with respect to a given 2D-floorplan. This statement leads to the two different tasks of single-image localization and sequential localization. Under realistic assumptions the localization of the 6-DoF system can be reduced to the problem of finding the 3-DoF pose $\mathbf{s} = (x, y, \theta)$ consisting of a location $(x, y) \in \mathbb{R}^2$ and a heading angle $\theta \in (-\pi, \pi]$. This simplification can be made because the gravity direction is assumed to be known and the z-coordinate will not change much in carry-on devices. The gravity direction can be estimated well by an inertial measurement unit (IMU) present in most devices. Furthermore the relative pose $\mathbf{\Delta s}$ between two frames is also assumed to be known as it can be extracted from the IMU. Moreover the camera used in the problem is assumed to be calibrated, which means the camera intrinsic matrix is known. The floorplan is assumed to be a 2D-representation of the environment, in which the camera is located somewhere inside the floorplan. The floorplans from different buildings often contain different information about the building, such as where the kitchen or the bathroom is located. However, not all floorplans contain such information. Hence it is assumed that a floorplan only depicts the walls, the windows and the door of an environment. An example of such a floorplan is shown in 3.1. Here it is further assumed that the scale of the floorplan is known, such that a transformation from floorplan distances to real-world depth is possible.

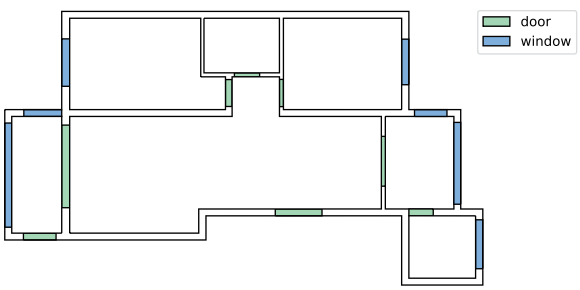

Figure 3.1: Floorplan containing the walls, windows and the doors.

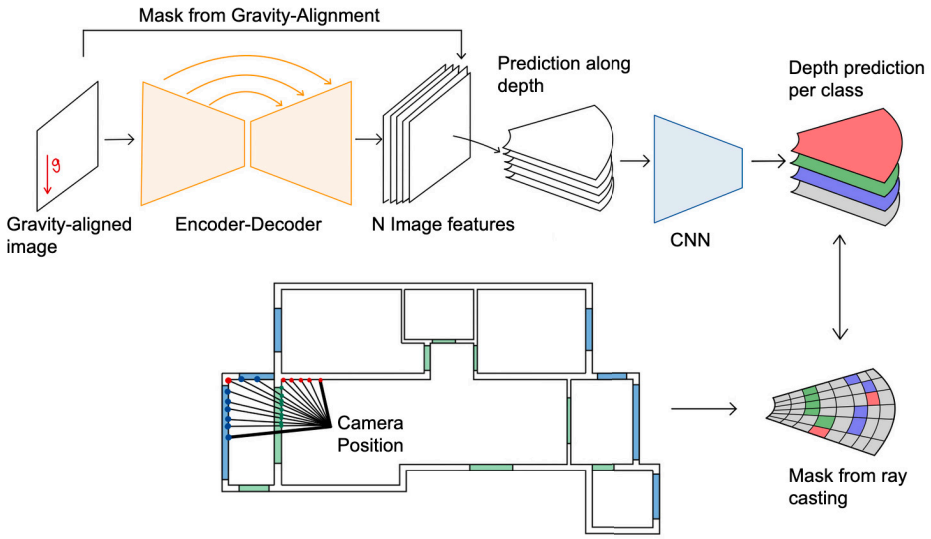

Figure 3.2: Overview of the network training routine.

3.2 Overview

The network is trained to predict a bird-eye view (BEV) from the given image. This BEV should contain the different labels of the floorplan, such as walls, windows and doors. The pipeline consists of a gravity-alignment step, where the image is warped such that its roll and pitch are zero, its principal axis is then horizontal. From this image, the network extracts the features $\mathbf{F} \in \mathbb{R}^{W \times H \times N}$ where W and H are the width and height of the image respectively. This N-dimensional features are then flattened down to the features $\mathbf{T} \in \mathbb{R}^{W \times D \times N}$ where D are log-distributed depth bins in the horizontal direction. This N-features are then aggregated with another CNN to output a class-wise prediction over the depth axis of the floorplan labels. The different steps are explained in detail in the following sections.

3.3 Bird-Eye View Prediction

3.3.1 Gravity Alignment

To deal with non-upright camera images the first step of the pipeline augments the images such that the principle axis of the image is horizontal. As described in 3.1, the gravity-direction is assumed to be known. From the gravity direction in the world coordinate frame, the roll angle ϕ and the pitch angle ψ can be calculated. The transformation between the original image and the gravity-aligned image can be described by using a simple homography as follows:

$$\hat{\mathbf{p}} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{p} \tag{3.1}$$

where $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{R}$ is the rotation matrix with the angles ϕ and ψ. $\mathbf{p}, \hat{\mathbf{p}}$ are the homogeneous image coordinates in the original and the gravity-aligned image respectively. As seen in 3.3 this leads to some pixels being unobservable and some pixels being left empty. The empty parts of the image are filled with black pixels, but a mask $\mathbf{M}$ from this step is calculated to mask out invalid pixels.

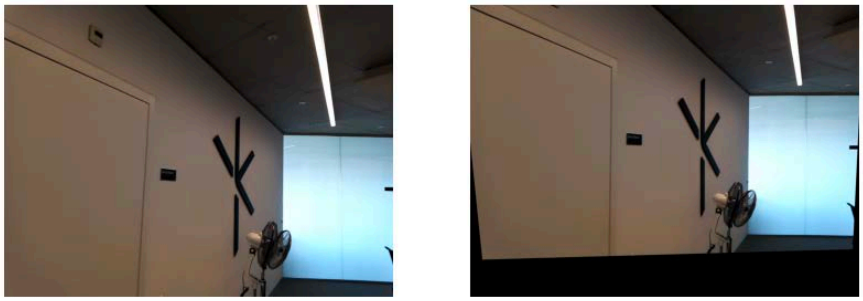

Figure 3.3: Original image on the left and gravity-aligned image on the right.

3.3.2 Encoder-Decoder

To extract the features from the image a U-net architecture is used. This encoder-decoder architecture is used to extract image-level features, similar to a semantic segmentation task. As shown in [9, 12] the tasks of semantic segmentation and monocular depth estimation have beneficial synergy. Monocular depth estimation can rely on semantic cues, which motivates the usage of an encoder-decoder architecture. Most semantic segmentation networks also use skip-connections [7], which improve the level of detail in the upsampling process. Thus the encoder-decoder converts the gravity aligned RGB image into image-level features $\mathbf{F} \in \mathbb{R}^{W \times H \times N}$, where each feature column is also gravity aligned. The mask $\mathbf{M}$ is applied at the end of the upsampling step to mask out invalid pixels.

3.3.3 Depth Prediction

Because the input to the encoder-decoder step is gravity aligned, each of the W feature columns in $\mathbf{F}$ should correspond to a vertical plane in world coordinates. Thus each of the columns in $\mathbf{F}$ is flattened in the horizontal direction into a polar representation such that $\mathbf{T} \in \mathbb{R}^{W \times D \times N}$ as seen in 3.4. This polar representation consists of D log distributed depth bins. For each depth bin a probability distribution $\boldsymbol{\alpha}_{w,d} \in [0, 1]^H$ over the image column is predicted. This transformation can be expressed as follows:

$$\mathbf{T}_{w,d} = \sum_h \boldsymbol{\alpha}_{w,d,h} * \mathbf{F}_{w,h} \tag{3.2}$$

where $d \in [0, D]$, $h \in [0, H]$ and $w \in [0, W]$. This formulation is similar to an attention mechanism from polar rays to image columns as seen in [15]. When the depth is difficult to infer, the probability distribution should be uniformly distributed and the features are spread along the ray in the polar representation.
These BEV features $\mathbf{T}$ are then fed through another CNN architecture with 2D-convolution and normalization layers to aggregate the features. The final BEV $\mathbf{B} \in \mathbb{R}^{W \times D \times C}$ contains the predictions for each class $c \in C$.

3.3.4 Ray Casting

An integral part of this work is the ray casting approach used in the training as a supervision signal and later in the inference for the matching step. A mentioned in 3.1, the floorplan contains the information of walls, windows and doors. In the ray casting step, a ray is shot across the floorplan, where the doors and windows are assumed to be permeable, which means they let the ray pass through. In contrast, where a ray hits a wall, it is assumed to be stopped. In 3.5 this step is visualized. After this ray casting step the continuous values are then sampled to a BEV occupancy grid where each cell takes a class label. This BEV occupancy grid is of the same shape as the BEV features $\mathbf{B}$.

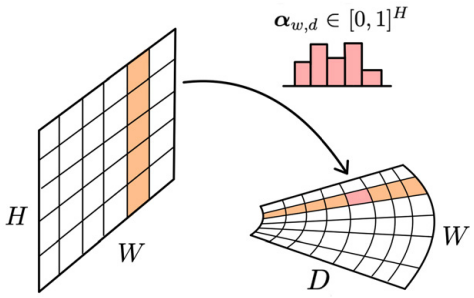

Figure 3.4: Transformation of a image feature $\mathbf{F}$ to a polar feature $\mathbf{T}$.

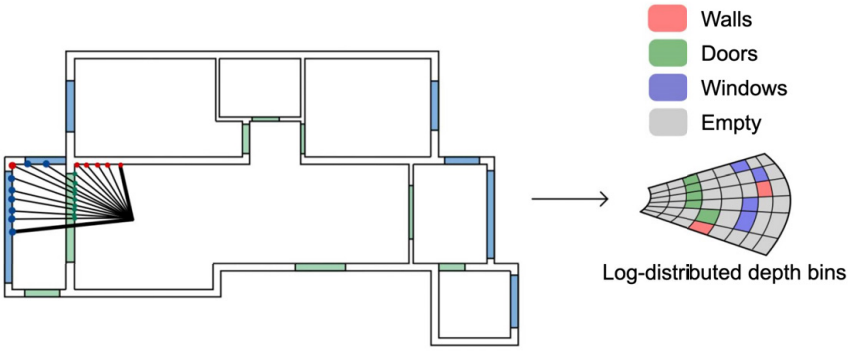

Figure 3.5: Ray Casting approach used to extract a supervision signal.

3.3.5 Training

In the training step the image is fed trough the network as explained previously. The ground-truth position $\mathbf{s_{gt}}$ is used to extract the BEV occupancy grid $\mathbf{B_{gt}}$ from the floorplan. With this occupancy grid and the class-wise BEV predictions $\mathbf{B}$ a balanced cross-entropy loss is used as a supervision signal to learn to predict a BEV from a given image. In unbalanced semantic segmentation settings, it is often beneficial to use a IoU (Intersection over Union) or a dice-loss as demonstrated in [22]. However, using such a loss leads to sharper edges in the distribution. In contrast, a balanced cross-entropy loss leads to a soft distribution over the depth when the depth is difficult to infer.

3.4 Matching for Inference

In the inference step the map is discretized into a grid of $X \times Y$ positions. At each location $\mathbf{s} \in X \times Y$ a 360 degree view of the BEV occupancy grid is collected with the approach shown in 3.3.4. This computation is quite expensive and but can be done upfront in a preprocessing step for each given map used for inference. With the BEV prediction $\mathbf{B}_{pred}$ of the network, a probability volume $\mathbf{P} \in X \times Y \times K$ can be constructed, where K is the number of rotations. At each pose $\mathbf{s} = (x, y, \theta)$ in this probability volume, a score can be calculated as follows:

$$L_{\mathbf{s_{x,y,\theta}}} = \frac{1}{DW} \sum_{d,w,c} \mathbf{B}_{pred} \odot \mathbf{B}_{\mathbf{s_{x,y,\theta}}} \tag{3.3}$$

with $c \in C$ being the class. Calculating this score for each pose $\mathbf{s}$ and normalizing over the entire volume leads to a valid probability distribution $\mathbf{P}$.

3.5 Sequential Localization

As the problem statement of estimating the pose in a floorplan using monocular depth is inherently ambiguous, the single-image localization predictions can be accumulated over time to achieve a better prediction. Tracking the belief over time can eliminate those ambiguities and increase the robustness of the framework. Usually a bayes filter is used to update the prior belief with a new observation. There are different variants of the bayes filter, such as the particle filter, which tries to represent the belief distribution by sampling particles in the continuous domain. As shown in [4] this approach is slower in this setting and furthermore suffers from particle degeneracy. This happens, when one particle has a weight close to one while all the other weights are close to zero. Because the ray-casting step is computationally expensive it is beneficial to use a histogram filter, which represents the belief with a discrete distribution. Since the state space doesn't change in a histogram filter, the ray-casting step can be computed offline.

3.5.1 Histogram Filter

A histogram filter is used to keep track of the belief over the state space. The probability distribution from the matching step given an image at time t is given as $P(\mathbf{s}_t \mid \mathbf{I}_t)$. The relative pose $\Delta\mathbf{s}$ between the images is assumed to be known and can be used to formulate a motion model as follows:

$$\mathbf{s}_{t+1} = \mathbf{s}_t \oplus \Delta\mathbf{s}_t + \boldsymbol{\omega}_t \tag{3.4}$$

where $\boldsymbol{\omega}_t = (\omega_{x,t}, \omega_{y,t}, \omega_{\theta,t})$ being the transition noise and $\oplus$ the pose composition operator. Assuming the transition noise w_t follows a gaussian distribution the transition probability can be written as:

$$P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \Delta\mathbf{s}_t) = \exp\left(-\frac{1}{2}(\mathbf{s}_{t+1} - \mathbf{s}_t \oplus \Delta\mathbf{s}_t)^\top \Sigma^{-1}(\mathbf{s}_{t+1} - \mathbf{s}_t \oplus \Delta\mathbf{s}_t)\right) \tag{3.5}$$

where $\Sigma = \text{diag}(\sigma_x, \sigma_y, \sigma_\theta)$ the covariance matrix of a gaussian distribution. The update for the motion model then can be calculated using bayes rule:

$$P(\mathbf{s}_{t+1}) = P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \Delta\mathbf{s}_t)P(\mathbf{s}_t) \tag{3.6}$$

The entire framework is based on $SE2$ and hence the translations in the world frame are dependent on the different orientations. As seen in [4] the translation and rotation can be decoupled and applied in sequence to the probability volume. The translation can be implemented as grouped 2D-convolution, where each orientation is a group. By applying this translational filter to the volume, each orientation layer is shifted with the respective translation. The rotation filter is implemented as a normal 1D-convolution over the orientation axis using circular padding. Using this formulation the belief from a single-image is translated and rotated using the relative motion. After shifting the predictions, the prediction can be updated with the prediction of the next frame using bayes rule as visualized in 3.6.

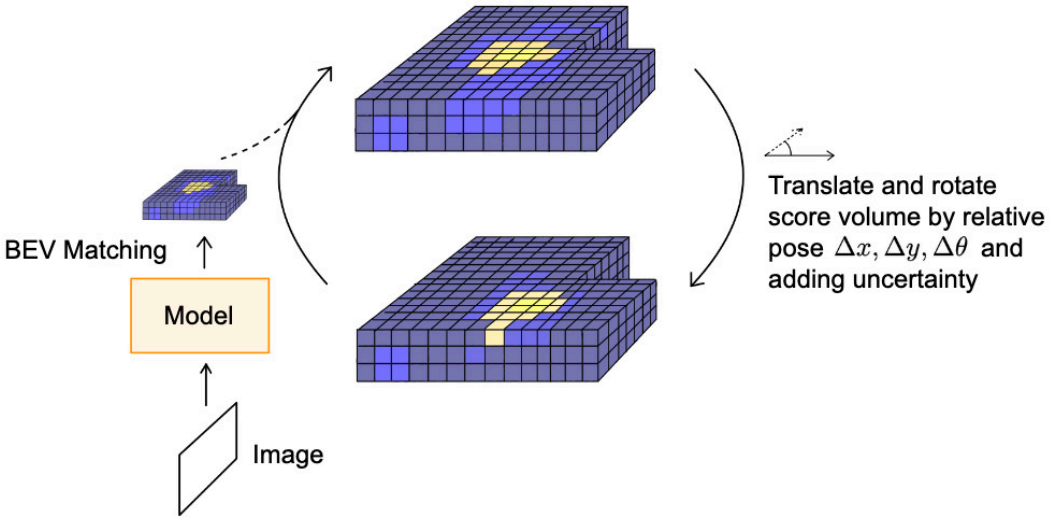

Figure 3.6: The belief is shifted using the relative pose and then updated with the new observation at each timestep.

Chapter 4

Experiments

4.1 Datasets

For the training and evaluation of the framework, a dataset with given floorplans and corresponding images with ground-truth pose is needed. The floorplan should contain the labels for doors, windows and wall and should be a 2D representation of the building with known scale. For the sequential localization the dataset should also consist of poses and images that are nearby each other, such that the poses can be viewed as a trajectory. Ideally this sequence should be taken at the same time, such that it reflects the usage in the real world, where e.g. an AR device is trying to localize itself in the environment. As mentioned in 1, there is a lack of indoor datasets consisting of trajectories. For this reason a custom dataset was collected in this work.

4.1.1 Structured3D

Structured3D [25] is an extensive dataset consisting of 3500 different scenes and corresponding floorplans of mostly residential housings with 196k photorealistic images. The dataset is split up into panoramic and perspective images. Since the goal of this work is the localization using mobile devices, the perspective images are used here. Furthermore, the Structured3D dataset provides the images in empty and fully furnished configuration. For the experiments the fully furnished dataset was used, since it better replicates a realistic environment than completely empty rooms. The xy-position of the images is randomly sampled across the floorplan, while the z-position (camera height), is normally distributed around 1.5m in this dataset. The images also contain a small pitch and roll component, which helps to evaluate our data augmentation approach. In all experiments we used the official train-test split.

4.1.2 Custom Dataset

Due to the aforementioned lack of realistic dataset with trajectory, a custom dataset was created. The floorplans from different office spaces at Magic Leap in Zurich, Sunnyvale and Plantation had to be polygonized and labeled with the semantics of doors, windows and walls. Using a mobile mapping system images in these three environments were collected. A pre-existing pointcloud of the environment together with an internal pipeline was used to get the ground-truth poses of the images in the pointcloud. In a next step, the floorplan is aligned with the pointcloud in a 2D-space in order to get the floorplan to camera transformation. This was achieved by sampling points on the polygonized floorplan, cropping the floor and ceiling from the pointcloud and projecting it to 2D. After masking out the tables and other furnishing an iterative closest point (ICP) algorithm was used to finely register these two instances. An example floorplan and the corresponding pointcloud can be seen in 4.1. Using this approach, a dataset with real images in indoor office

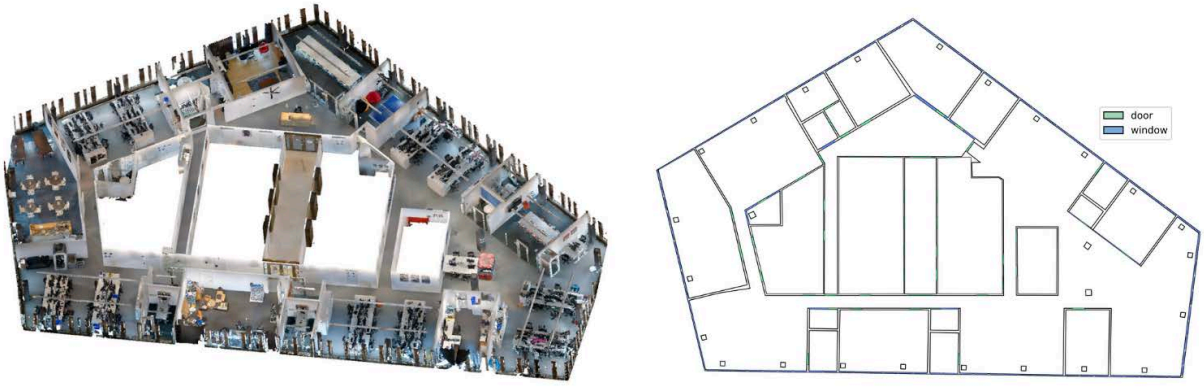

Figure 4.1: Pointcloud and floorplan of the Magic Leap office in Zurich. By registering the images in the pointcloud and aligning the floorplan with this pointcloud the ground-truth pose was obtained.

environment was created. Details of the dataset and a comparison to the Structured3D dataset can be found in 4.1.

Dataset	**Structured3D**	**Custom Dataset**			
Locations	3500 Scenes	Zurich	Sunnyvale	Plantation	Total
Number of Images	196'000	5'500	2'100	20'600	28'200
Area [m^2]	376'500	1'170	1'815	11'125	14'110

Table 4.1: Details of the generated dataset compared to the Structured3D dataset. For the Structured3D dataset the area is summed up over all floorplans, a single floorplan covers an area of approximately $105m^2$.

4.2 Evaluation Metrics and Benchmark

As evaluation metrics the location error e_{pos} and angle error e_θ of the pose with the highest probability with respect to the ground truth pose is calculated. From this errors the position recall at different thresholds is calculated to evaluate the performance of the localization. In addition, we calculate a recall for position and angle (logical and) to evaluate if the model correctly predicts the rotation as well. As the problem setting for single-image localization is ambiguous, the recall for the top-K predictions is calculated as well. For the sequential localization, the recalls are calculated per frame to see if the model manages to converge to a true solution.
As a benchmark to our model the SOTA methods of *F3Loc* [4] and *LASER* [14] are used. As Structured3D does not contain any trajectories, the sequential localization evaluation can only be done on the custom dataset. The single-image localization performance is shown for both datasets.

	Structured3D	Custom Dataset
Single Image Localization	✓	✓
Sequential Localization	✗	✓

Table 4.2: Evaluation of the used Datasets.

4.3 Results

4.3.1 Single-image Localization

Structured3D

Using the official train-test split and evaluating on the entire test dataset (6266 images) the recalls in 4.3 were achieved. To evaluate the need of semantics, a model was trained using only occupancy information from the floorplan. This means that the doors were removed from the floorplan and the window and wall labels were labelled as occupied. The method stayed the same but had only 2 classes (occupied/unoccupied) and the model was trained with a balanced cross-entropy loss with adapted weights. The LASER pipeline was adapted to work on the perspective Structured3D dataset, because originally only the panoramic dataset was used in this work. The adapted version was then trained for 50k iterations, since the model overfitted when using the 300k iterations that were used in the original paper. The recalls for F3Loc were taken from the paper[4], since the full pipeline for the evaluation was not published at the time of completing of this thesis.

Recall@ [%]	1m	1m30deg	0.5m	0.2m
F3Loc	22.4	21.3	14.6	-
LASER	6.1	3.7	2.7	0.4
Ours (no semantics)	12.0	9.5	4.9	1.3
Ours	**36.1**	**31.7**	**19.2**	**4.5**

Table 4.3: Recall at different thresholds for single-image localization on Structured3D.

Custom Dataset

For the evaluation on the custom dataset, two different versions of the model were trained. The first version follows a valid train-test split. In this version, the locations of the environment for training and testing are completely independent from each other and a different part of the dataset for the validation was used during the training than for the testing. Here the validation loss was not decreasing after the 3rd epoch, where we assumed that the domain gap between the different environments is too big. In a second version, parts of the training dataset were assigned to the validation set and parts of the initial test dataset were used in the training as seen in 4.4. The train-test split in this version is not ideal, as the training environment overlaps with the test dataset. However, given the limited data available, this approach was taken to create a proof of concept. These versions of the model are used for the evaluation. The single-image localization for this two versions is reported in 4.5.

	Version 1	**Version 2**
Train	Plantation	$\frac{4}{5}$ Plantation
	Sunnyvale	Sunnyvale
		$\frac{1}{2}$ Zurich
Validation	$\frac{1}{2}$ Zurich	$\frac{1}{5}$ Plantation
		$\frac{1}{2}$ Zurich
Test	$\frac{1}{2}$ Zurich	$\frac{1}{2}$ Zurich

Table 4.4: Train-test split for the custom dataset for two different versions of the model.

Recall@ [%]	2m	1m	1m30deg	0.5m	0.2m
Version 1	8.6	3.1	2.2	0.6	0.1
Version 2	16.8	11.6	11.4	4.0	0.2

Table 4.5: Recall at different thresholds for single-image localization on the custom dataset.

4.3.2 Sequential Localization

The sequential localization results are reported for Version 2 of the model on the custom dataset over different lengths of the trajectory. The entire test dataset is split up into trajectories of length N and the recall is recorded for each frame. In 4.6 the recall for different trajectory lengths is reported as an average over the entire trajectory.

Trajectory	Mean Recall@ [%]			
Length	2m	1m	1m30deg	0.5m
2	14.4	8.4	8.2	3.2
5	13.5	8.2	7.6	3.5
10	15.9	10.9	10.4	5.2
20	15.1	9.9	9.5	4.9
50	17.3	10.7	10.1	4.5
100	14.2	9.3	8.7	3.8

Table 4.6: Mean recall for trajectories of different length. The number of available trajectories decreases with increasing trajectory length, as the total test dataset contains only 1546 images.

Chapter 5

Discussion

5.1 BEV Prediction

The model is trained to predict a soft distribution over the depth for the different classes as seen in 5.1. It has to be noted that the model also predicts a distribution for the empty space in the BEV. For the output of the model, the softmax is applied over the class dimension of the BEV. In 5.1 it can be seen, that the model is able to predict the room layout well. Only the three relevant classes of doors, windows and walls are shown here. It has to be noted that the empty area of the BEV also accounts to the total score in the matching for inference. The empty class is the reminder of the three shown classes. When the scale is difficult to infer e.g. where the window exactly is, the model spreads the prediction over the depth bins. It can also be seen that the model is able to mask objects that are not relevant to the scene such as the sofa or other furnishing.

In other examples the BEV predictions is not performing well as visualized in 5.2. We can already see from the image, that the view in both of these examples is occluded by furnishing. In the left image the position is just in front of a door, but that door is not recognized at all. The BEV poorly represents the scene structure in this case, as the walls on the right side are in a wrong scale. We can also see that the painting is recognized as a window. In the right image the view is occluded by a wardrobe and the door is hidden behind the occlusion. Interestingly the window seen on the left is recognized, but not represented in the floorplan. This could stem from a wrong labeling of the floorplan in the Structured3D dataset or it is possibly just a painting/photo on the wall.

The BEV predictions on the custom dataset show varying results, with some images yielding accurate predictions, while others exhibit significant discrepancies. Figure 5.3 displays two examples of successful predictions. In the left image, the model accurately assigns a likelihood to the window at the back, as it is visible through the front window, also the wall corner is predicted with notable precision. Similarly, the right image shows a BEV prediction that captures the room layout effectively. However, when analyzing the BEV predictions across the custom dataset, numerous distortions and unintended predictions become apparent.

In 5.4, the left image highlights a common issue where the model assigns a likelihood behind closed doors across all classes. This occurs because the rays pass through doors, regardless of whether they are open or closed. While removing labels behind closed doors might help reduce these distortions, it could also negatively affect prediction quality in other scenarios. Similarly, in the right image, the likelihood is spread over the entire depth, with only a few distinguishable peaks in the prediction. Overall, the BEV predictions on the custom dataset tend to be more diffuse and less sharp.

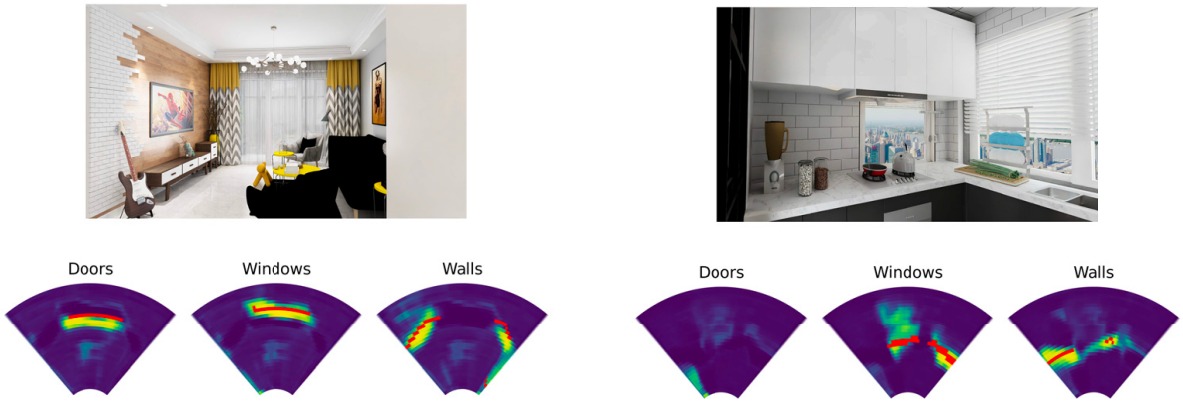

Figure 5.1: BEV prediction for two example images from Structured3D with good predictions. The ground truth is depicted in red and the likelihood from purple (low) to yellow (high).

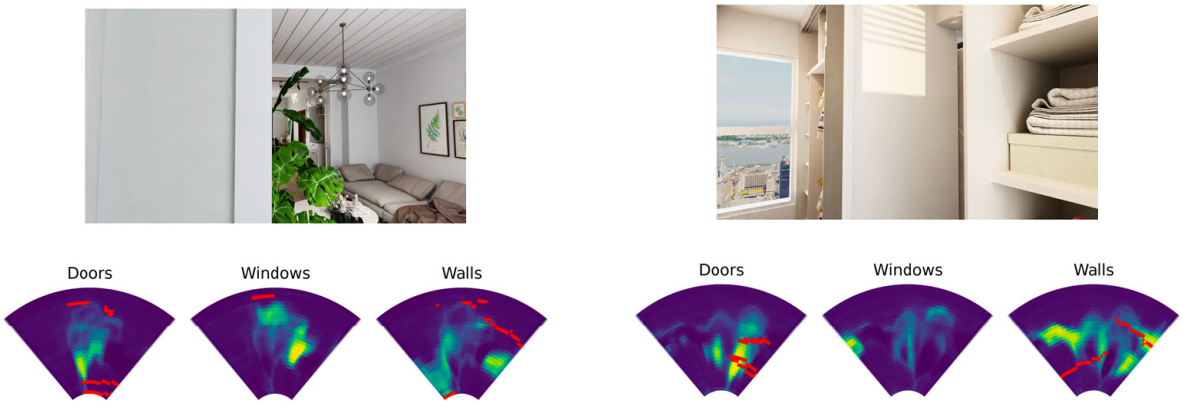

Figure 5.2: BEV prediction for two example images from Structured3D with bad predictions. In both cases the view is occluded by the furnishing of the environment. The ground truth is depicted in red and the likelihood from purple (low) to yellow (high).

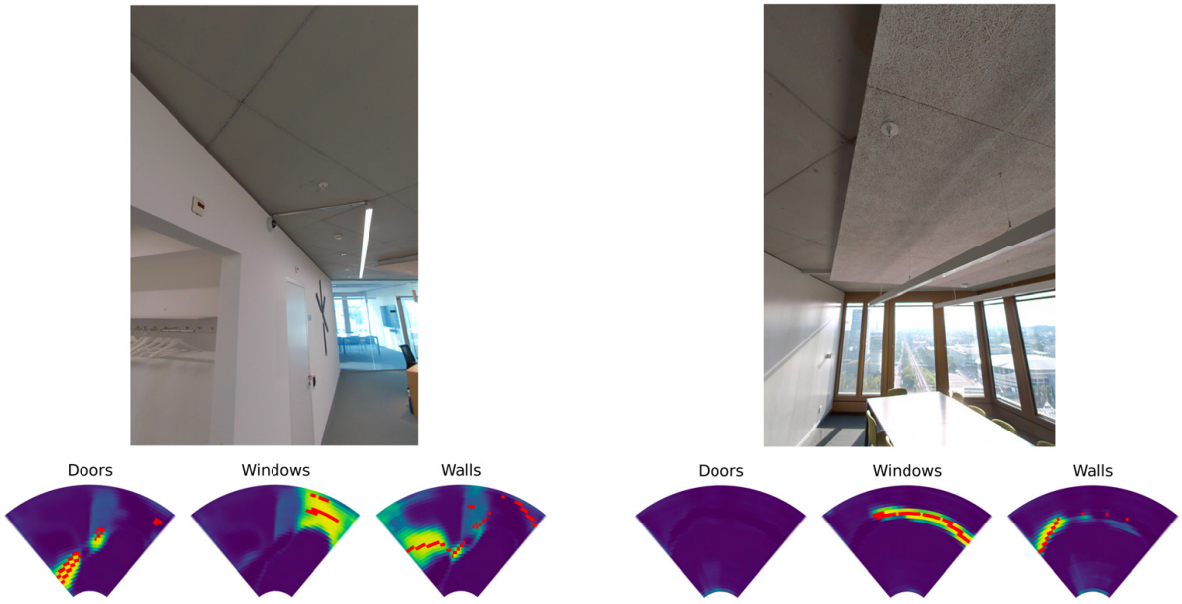

Figure 5.3: BEV prediction for two example images from the custom dataset with good predictions. The ground truth is depicted in red and the likelihood from purple (low) to yellow (high).

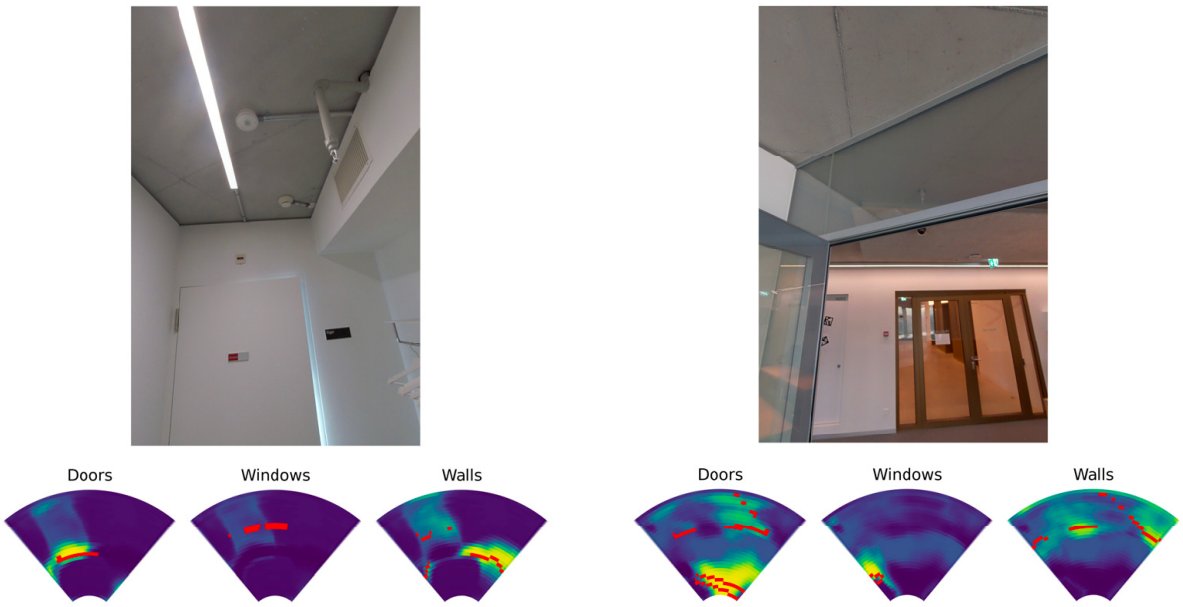

Figure 5.4: BEV prediction for two example images from the custom dataset with bad predictions. The ground truth is depicted in red and the likelihood from purple (low) to yellow (high).

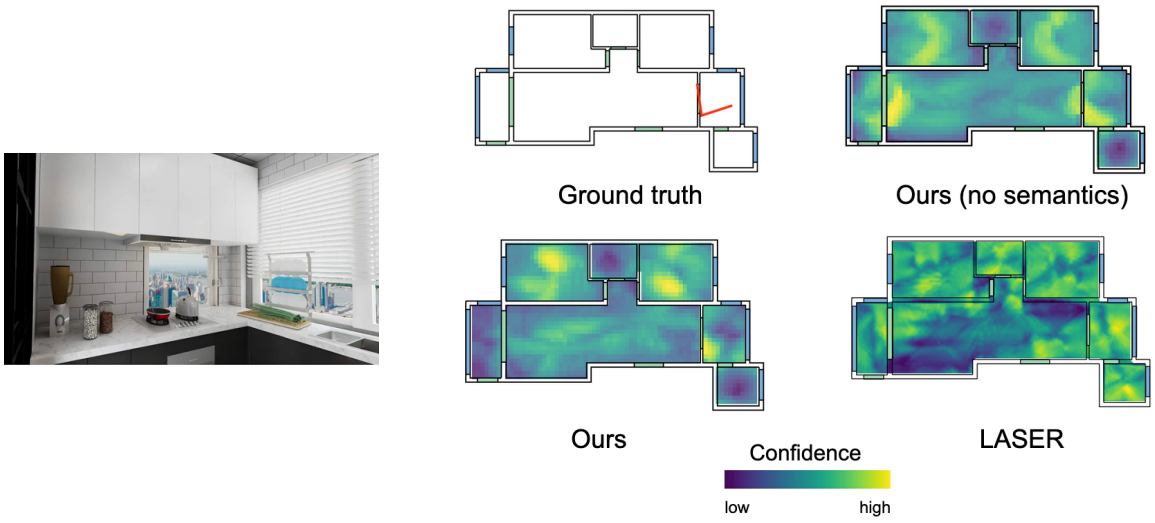

Figure 5.5: Maximum prediction over the orientation axis of the score volume shown for different models. for an example image.

5.2 Inference

In the inference step the BEV is matched across the entire floorplan to get a score volume. The normalized score volume is reflecting the probability distribution over the state space. This probability distribution can not be visualized easily, since it covers a dense three dimensional space over x, y and θ. Hence we visualize only the xy plane here and take the maximum over the orientation dimension. In 5.5 this probability distribution is visualized for the model with and without semantics as mentioned in 4.3. In this example it can be seen, that by using the semantics from the floorplan, the model can remove some of the ambiguities present in the floorplan. However, there are still three main peaks with different viewing directions in this example, which could all be possible locations from the room layout. We can also see that the prediction of our model leads to a smoother distribution than the prediction from LASER, which could stem from the fact that a soft distribution over the depth is predicted. The prediction from LASER has a poor performance when looking at this example, but also has been seen in 4.3. During the evaluation of LASER we noticed a significant gap to the performance mentioned in [14], which was also noticed in [4]. A big reason for this is the different dataset that was used in the original paper. Although the panoramic images were cropped to the same FOV as the images in the perspective dataset, there are a few differences in the two datasets. During the evaluation we found that the camera position for the panoramic images is not fully random as in the perspective dataset. The camera position in the panoramic dataset has a heavy bias towards the center of the room, while the perspective dataset does not have this bias. Furthermore the images in the perspective dataset contain some roll and pitch distortion and are also sampled from different camera heights. The panorama images are always taken from the same height without roll.

As already mentioned previously, conducting inference on the custom dataset presented several challenges. One reason for this is that the floorplans in this dataset were a factor of 10-100 bigger than the floorplans in Structured3D. This caused issues during the matching step, which was originally designed for smaller floorplans. While it was still feasible to run the matching on the smaller floorplans of the custom dataset with a batch size of one image, the largest floorplan, Plantation (11,000m^2), could not be processed due to its high memory requirements. An example of a single image localization can be seen in 5.6. There

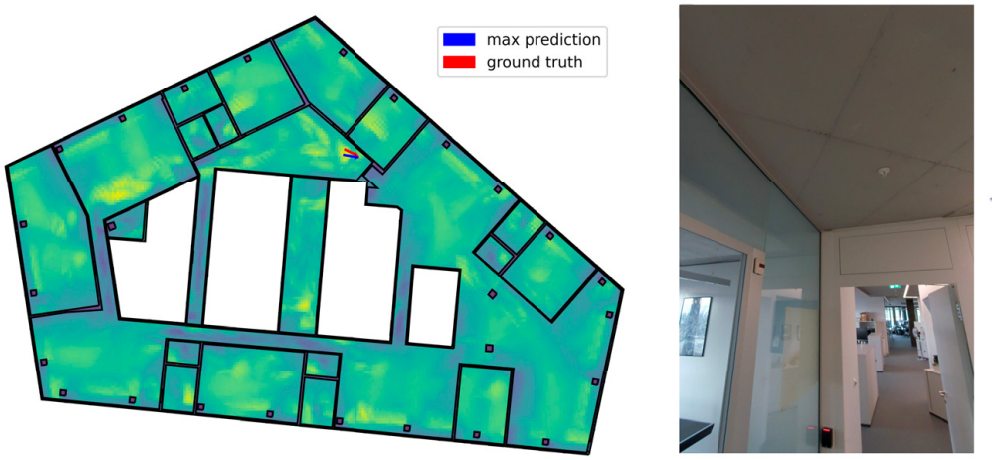

Figure 5.6: Single image localization on a real image for the custom dataset.

are a lot of ambiguous peaks across the entire floorplan, still the maximum prediction of the model was correct in this example. In this image, the room layout was predicted accurately, closely aligning with the ground truth, resulting in a strong prediction. In this example, two doors are open, and the BEV prediction successfully captures the wall behind the doors.

5.3 Sequential Localization

The results from the sequential localization in Section 4.3 indicate that the mean recall over trajectory length does not increase linearly. Typically, mean recall should improve as trajectory length increases, assuming each individual image prediction is reasonably accurate. However, the results show this is not the case. One possible explanation is that while some single-image localizations are quite accurate, others are significantly off, leading to inconsistency. In some instances, the prediction is spread across the entire map, causing the prior belief to become more uniform rather than refined. An area for potential improvement could be to update only the top-K peaks from the predictions, which could help sharpen the belief and accelerate convergence. Additionally, it is important to note that the sample size in Table 4.6 decreases as the trajectory length increases. The total test set contains 1,546 images, meaning 154 trajectories were evaluated for a trajectory length of 10, whereas only 15 trajectories were available for lengths of 100 images. This small sample size further increases the uncertainty of the results.

Since the results of the sequential localization experiments did not meet the expectations, further analysis was performed. Given the high ambiguity in combining single predictions to track belief over time, especially in large floorplans, we also evaluated the top-K peaks of the belief distribution. To identify the top peaks, a maximum filter of size $5 \times 5 \times 3$ was applied, representing an area of $1m \times 1m \times 30°$. This filter was applied across the entire score volume (with circular padding) to isolate the top peaks within the belief. These peaks were then ranked based on their maximum likelihood. In Figure 5.7, the top-K peaks over time are shown. A sample is considered a true positive if the ground truth is within the top-K peaks. The plot illustrates that by using the top-5 peaks, recall can reach up to 32

5.4 Timing

Timing is a critical factor for deploying the pipeline on a real device. Since the matching time and histogram filter scale with the size of the floorplan, the total processing time decreases when running the model on

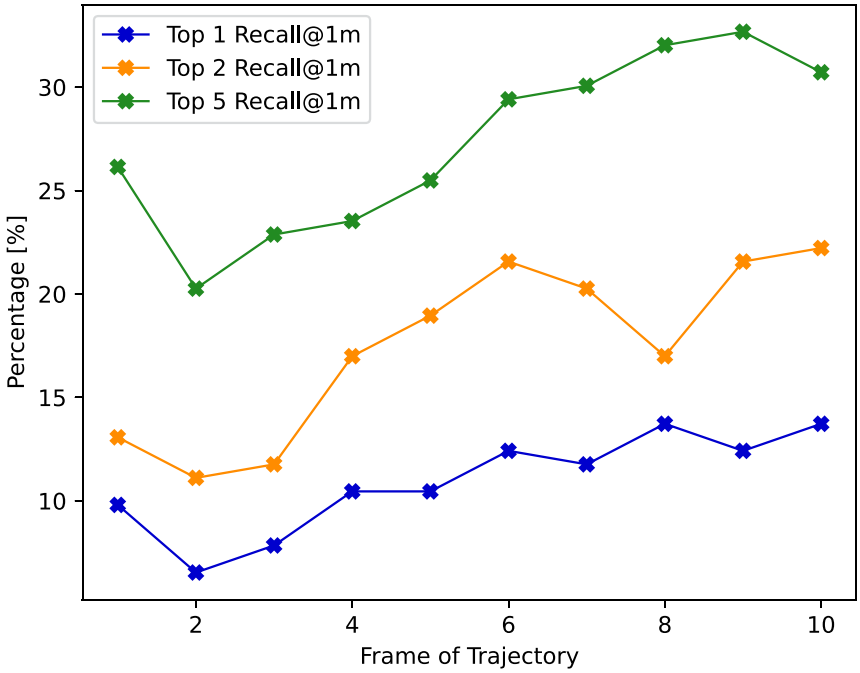

Figure 5.7: The recall over the frames of the trajectory calculated for the top-K peaks.

smaller floorplans, such as those in Structured3D. Table 5.1 presents the timing results for the various steps on the Zurich floorplan. It is evident that the total time is primarily influenced by the matching time. This duration could be reduced by either using a smaller floorplan or decreasing the spatial resolution. Consequently, the pipeline is capable of running at a maximum frame rate of 0.52Hz on the Zurich floorplan with the specified spatial resolution of $0.2m \times 0.2m \times 10\,\mathrm{deg}$.

Step	**Time [s]**	**Share of Total Time [%]**
BEV prediction	0.14	7
Matching	1.67	87
Histogram Filter Update	0.1	6
Total	**1.91**	**100**

Table 5.1: Timing for the different steps of the pipeline for inference on the floorplan in the Zurich office. The model was running on a MacBook Pro, M1 Max 2021.

Chapter 6

Conclusion & Future Work

During the process, we quickly recognized the shortage of indoor datasets that include both trajectories and 2D floorplans. To address this, we created a custom dataset at the Magic Leap offices, which mitigated the issue to some extent. However, having access to a larger, more diverse dataset covering various environments with corresponding floorplans would still be highly beneficial. A few datasets are available that include trajectories along with registered positions in indoor environments and the corresponding point cloud. These datasets could potentially be used by extracting the floorplan directly from the point cloud (e.g. dataset with for semantic segmentation), although this presents its own set of challenges. While creating the custom dataset, we also realized that manually labeling the floorplans is highly labor-intensive. Scaling this process to a large number of scenes would require an automated approach, as manual labeling is too time-consuming and inefficient for broader applications.

Despite limited data, we successfully developed a localization pipeline using only 2D floorplans. The method presented in this work can predict a BEV and match it across the entire floorplan. This approach resulted in an interpretable pipeline that outperformed state-of-the-art methods for single-image localization. While integrating single-image localization predictions into a histogram filter proved challenging, it still managed to reduce some of the ambiguity inherent in the problem. Bridging the gap between simulated and real data was difficult but still effective for BEV prediction. However, the inference step posed significant challenges due to the much larger floorplan sizes compared to the rendered image dataset. A promising direction for future work would be to improve scalability for larger floorplans and make the entire framework more lightweight. This would not only enhance efficiency but also increase the pipeline's frequency by reducing matching times. Another area worth exploring is incorporating more semantic information into the framework, such as identifying areas like kitchens or bathrooms from the floorplan. Additional data from building information models (BIM), such as electrical wiring visible in warehouses, could further enrich the framework's capabilities.

Another challenge is the adaptation of the framework to different datasets. Currently, the model learns to predict a depth distribution for a given image, which is closely tied to the specific camera used in the dataset. As a result, training the model embeds the camera's intrinsic parameters into the model's weights. Making the model independent of these camera intrinsics would improve its adaptability, allowing it to quickly generalize to other datasets or previously unseen data.

Bibliography

[1] Vassileios Balntas, Shuda Li, and Victor Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[2] Hermann Blum, Julian Stiefel, Cesar Cadena, Roland Siegwart, and Abel Gawel. Precise robot localization in architectural 3d plans, 2020.

[3] Federico Boniardi, Abhinav Valada, Rohit Mohan, Tim Caselitz, and Wolfram Burgard. Robot localization in floor plans using a room layout edge extraction network, 2019.

[4] Changan Chen, Rui Wang, Christoph Vogel, and Marc Pollefeys. F^3loc: Fusion and filtering for floorplan localization, 2024.

[5] Hang Chu, Dong Ki Kim, and Tsuhan Chen. You are here: Mimicking the human thinking process in reading floor-plans. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2210–2218, 2015.

[6] Alexandra Dana, Nadav Carmel, Amit Shomer, Ofer Manela, and Tomer Peleg. Do more with what you have: Transferring depth-scale from labeled to unlabeled domains, 2024.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Henry Howard-Jenkins and Victor Adrian Prisacariu. Lalaloc++: Global floor plan comprehension for layout localisation in unvisited environments, 2022.

[9] Lukas Hoyer, Dengxin Dai, Yuhua Chen, Adrian Köring, Suman Saha, and Luc Van Gool. Three ways to improve semantic segmentation with self-supervised depth estimation, 2021.

[10] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[11] Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization, 2018.

[12] Dong Lao, Fengyu Yang, Daniel Wang, Hyoungseob Park, Samuel Lu, Alex Wong, and Stefano Soatto. On the viability of monocular depth pre-training for semantic segmentation, 2024.

[13] Oscar Alejandro Mendez Maldonado, Simon Hadfield, Nicolas Pugeault, and Richard Bowden. Sedar: Reading floorplans like a human—using deep learning to enable human-inspired localisation. *International Journal of Computer Vision*, 128:1286 – 1310, 2019.

[14] Zhixiang Min, Naji Khosravan, Zachary Bessinger, Manjunath Narayana, Sing Bing Kang, Enrique Dunn, and Ivaylo Boyadzhiev. Laser: Latent space rendering for 2d visual localization, 2023.

[15] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d, 2020.

[16] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020.

[17] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks, 2020.

[18] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale, 2019.

[19] Paul-Edouard Sarlin, Daniel DeTone, Tsun-Yi Yang, Armen Avetisyan, Julian Straub, Tomasz Malisiewicz, Samuel Rota Bulo, Richard Newcombe, Peter Kontschieder, and Vasileios Balntas. Orienternet: Visual localization in 2d public maps with neural matching, 2023.

[20] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning robust camera localization from pixels to pose. In *CVPR*, 2021.

[21] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 09 2016.

[22] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. *Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations*, page 240–248. Springer International Publishing, 2017.

[23] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024.

[24] Yunpeng Zhang, Zheng Zhu, Wenzhao Zheng, Junjie Huang, Guan Huang, Jie Zhou, and Jiwen Lu. Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving, 2022.

[25] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling, 2020.