Faster training of the Scene Landmark Detection for Camera Localization

Alen Bišanović
ETH Zurich
Switzerland
abisanovic@ethz.ch

Johanna D'Ciofalo
ETH Zurich
Switzerland
jdciofalo@ethz.ch

Owen Du
ETH Zurich
Switzerland
owen.du@inf.ethz.ch

Mathieu Lutfallah
ETH Zurich
Switzerland
mlutfallah@ethz.ch

Abstract

Image-based camera localization concerns determination of a camera's pose within a scene given an image. Traditional methods often require extensive memory storage for 3D point clouds, posing challenges related to efficiency and privacy. This paper explores enhancements to the Scene Landmark Detector (SLD) technique, which leverages convolutional neural networks (CNNs) to detect scene-specific landmarks and calculate camera poses, thereby eliminating the need for storing extensive 3D maps. The proposed improvements include the introduction of a scene-agnostic backbone and scene-specific heads, significantly reducing the training time required for each new scene. Additionally, we investigate the creation and refinement of 3D point clouds by integrating Monocular Depth Estimation (MDE) with COLMAP, using a trained MLPRegressor to estimate and fill missing depth values. The results demonstrate a reduction in training time and improved performance in depth estimation, offering a more efficient and scalable approach to camera localization.

1. Introduction

Visual localization involves using query image and a prebuilt map of the scene to determine the pose of the camera, used for robotics and VR applications. These methods usually fall into one of the following categories: retrieval, structure-based, learning-based localization or Scene Coordinates Regression. Some of these approaches require storing a 3D point cloud of the scene or long-term storage of images and features, requiring high memory storage and poses privacy concerns. Methods that do not require storing a 3D map of the scene fail to reach the same accuracy as methods that do.

Do et al. [5] propose a new visual localization technique, Scene Landmark Detector (SLD) and SLD* [6] that evade the challenges aforementioned and manage to reach higher performance than other current methods that are storage-free. The method takes inspiration from keypoint and landmark recognition to determine the position of the camera. For each scene, certain 3D points are selected as scene landmarks, which are detected in query images using a CNN. After obtaining the 2D-3D correspondences the pose is calculated using a PnP algorithm. As the scene is encoded in the CNN layers, this approach evades the privacy issues. Since each CNN is scene-specific, the SLD approach requires training a new CNN for each scene which is time consuming as training can take tens of hours.

This project explores ways to improve the speed and performance of the SLD. Firstly, we investigate ways to speed up the training time of the SLD. By introducing a scene-agnostic backbone that is shared across multiple scenes, and only training a scene-specific head, the training time can be reduced. The second part of the project concerns scene dataset creation and 3D point cloud reconstruction. This is relevant for the performance of the SLD as it relies on 3D point clouds of the scene created using COLMAP. The accuracy of the point cloud is thus fundamental for the performance of the SLD. We create our dataset of office scenes and then explore ways to improve the 3D point cloud created by COLMAP. This is achieved using Monocular Depth Estimation (MDE), which is assured to create a depth value at every pixel, and then train a regressor to determine the missing depth values from COLMAP from the relative-depth values obtained by MDE.

We propose a novel training approach for SLD, where we first train a scene-agnostic backbone. For new scenes, we only need to train a scene-specific head, which reduces the training time by a factor of 20. The accuracy of the new approach is shown to be close to state-of-the-art for specific scenes.

2. Related Work

2.1. Camera localization methods

Traditional camera localization methods rely on feature matching or image retrieval. The former stores a 3D point cloud of the scene to which the features are mapped to and obtains the camera position by applying pose estimation[5].

Image retrieval methods are not able to produce as accurate camera poses as map-based methods. These methods interpolate poses of retrieved images to evaluate the camera pose of the query image by using scalable techniques.

The second category of camera localization methods are learning-based approaches[5]. This category can be further divided into two sub-categories, namely Learned absolute pose regression (APR) methods and Relative pose regression (PRP) methods. APR approaches do not require storing a 3D point cloud of the scene and regress the camera extrinsics directly from the input image but require homogeneously distributed training sets to reach high accuracy. PRP methods, use stored database images to determine the relative pose of the camera and have a higher degree of generalization but higher memory requirement.

In Scene Coordinate Regression (SCR), for each pixel in the input image the corresponding 3D point in the scene is determined, and the 2D-3D correspondences are used to estimate the camera position [5].

2.2. SLD and SLD*

Do et al. [5] present a new memory efficient camera visualization algorithm based on scene landmark detection. The idea behind the method is to select a few salient scene landmarks and detect these in the input image to determine the camera position.

Given a scene, structure from motion (SfM) is used to obtain the 3D point cloud denoted P of the scene from RGB images. Instead of utilizing each point in the point cloud to determine the camera extrinsic, the method focuses on a few points in the point cloud for mapping, referred to as landmarks. Landmarks are selected for each scene based on a saliency score in a greedy fashion, in which a point is selected as a landmark depending on the following properties: robustness, repeatability and generalizability meaning that the point is visible from several angles in the scene [5]. For each point $x \in P$ a saliency score is determined. In addition to selecting a subset of points x with high saliency score, the selected points must cover the whole scene and to ensure that for all viewing angles some landmarks are visible. The latter is achieved by using a greedy algorithm that assures a minimum distance r between each scene landmark. Furthermore, for each image there is mapping π between the 3D point x and the corresponding 2D image coordinate. Using this each landmark is projected onto its 2D image point and the 2D coordinates of the landmark is determined [5].

The next step of the method includes training a CNN, referred to as Scene Landmark Detector (SLD), to detect the landmarks from the input image I. The SLD outputs a heat map showing the likelihood of each landmark that is visible in the image. There have been two iterations of the SLD architecture. The first version called SLD utilises a second complementary CNN architecture called the Neural Bearing Estimator (NBE). For each image used to create the point cloud, the camera intrinsics are assumed to be known, which allows the 2D coordinates of the landmarks to be converted to 3D bearing vectors. The NBE regresses bearing vectors for all scene landmarks, independent if they are visible in the image or not.

The newer version, SLD* [6], does not utilise NLB to estimate the camera position and instead uses more landmarks (~ 1500) compared to SLD ($200 - 400$). We will briefly go through the architecture of the SLD*, depicted in fig 1, which is the variant we will refer to in this project. SLD* uses EfficentNet as backbone to reduce the storage requirement and memory required for the CNN. This is followed a downsampling layer and a dilated convolutional layer. The final layer is a 1×1 convolutional layer that outputs the heat maps for each landmark.

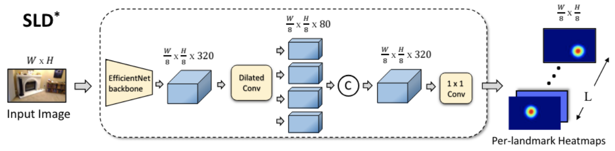

Figure 1. Architecture of SLD*.Image retrieved from [5]

3. Dataset Creation and Depth Map Fusion

3.1. Dataset Generation Using COLMAP

The motivation to generate more scene datasets comes primarily from [3], where the authors claimed to have trained a scene-agnostic backbone across over 100 scenes. As SLD is proven to scale better than SCR methods for larger scenes, we explore the generation of new scene datasets of larger proportion and under different illumination settings.

The framework for creating the dataset consisted of five steps. Initially, frames were extracted from videos at a predefined rate of four frames per second, determined based on some preliminary tests. These frames were then divided into different folders based on the videos; that is, frames corresponding to each video were placed in separate folders. It should be noted that frames could have identical names across different folders; for example, the first frame of video 1 and video 2 had the same name but were placed in different folders. Colmap was then used for sparse reconstruction, resulting in sparse landmarks, camera poses, and intrinsics for the images. We also employed Colmap to generate dense maps of the images for later comparison with those obtained from monocular depth images. The data was stored in files with the ".bin" extension, and we relied on scripts from the official Colmap documentation to parse it.

The data was stored in a structure defined by the provided training scripts, with some C++ code omitted due to copyright restrictions. The undistorted images, along with their corresponding poses and intrinsics, were saved in three separate files within the same folder. The intrinsics were saved in a specific format: W, H, f, px, py, Kappa, PATH. The PATH data was crucial, as it ensured that image names were not duplicated and individual IDs were assigned to each image to avoid confusion. To facilitate tracking and organization, a pickle file was created to store the IDs of all new images and divide them into training, testing, and validation sets.

For the intrinsics file, the camera rotation and translation were saved in a 4×3 matrix according to the format $R \mid t$. The dense maps were also extracted, and geometric images were used rather than photometric ones due to their superior quality. Using the pickle file, extracted dense maps, intrinsics, poses, and undistorted images, we were able to utilize the provided scripts to map the landmarks based on the approach presented in the initial work. In total, three datasets were created for different offices in the ETH building. Figure 2 shows one of the landmarks in four different images.

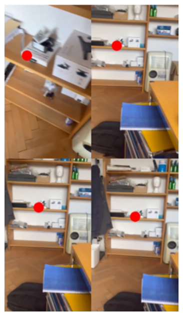
Figure 2. Image showing the same landmark in multiple scene of the created dataset.

3.2. Image Dense Reconstruction using MDE and COLMAP

The second part of the project explores ways to enhance the dense reconstructions. The particular approach that was investigated during the project, was to create a fusion model that utilized monocular depth estimation to improve the depth map generated by COLMAP.

The original paper uses COLMAP to create 3D point clouds and to calculate the camera poses for the scenes [5]. COLMAP is a pipeline that combines Structure-from-Motion (SfM) and Geometry-based Multi-View Stereo (MVS). SfM and MVS based algorithms face several difficulties when it comes to dense reconstruction and the quality of the constructed dense map varies for different scenes. The algorithms face decreasing performance on image regions with occlusions, low texture and repetitive patterns resulting in a reconstruction with holes and noises [8].

3.2.1 Methodology

Monocular Depth Estimation (MDE) methods produce a depth value at every pixel given a single RGB image. However, the produced depth values are scale-ambiguous and do not correspond to real metric predictions of the depth [7]. We train a regressor to map the depth values obtained by MDE to the depth values obtained by COLMAP. We then use the regression to estimate the depth value for missing pixels from COLMAP.

The MDE algorithm used in this project was *Depth-Anything* [13], a new MDE algorithm released in the beginning of 2024. It utilizes unlabeled images apart from labeled images to generate higher performance compared to other state of the art algorithms such as MiDaS [2] and ZoeDepth [1]. Given an RGB image I of a scene, we created the corresponding depth maps of the image using a Depth-Anything and COLMAP, we will refer to these as Z_1 and Z_2 respectively. We iterate over the depth values of Z_2 and mask all NaN entries, generating a new depth map Z_2' not containing NaN entries. We then train a $MLPRegressor$ to map the depth values generated by $DepthAnything$, $z_{1_l} \in Z_1$, to the corresponding depth value $z_{2_l}' \in Z_2'$ generated by COLMAP. The $MLPRegressor$ has hidden layer of size (100×50) and we set the random state to 42. We use a 80% and 20% division of the training and test set respectively and use 1000 iterations for training. We then use the $MLPRegressor$ to estimate the depth values of the NaN entries of Z_2 using the corresponding depth values in Z_1, to create an improved COLMAP depth map.

The predict depth function predicts the depth value for a given pixel using a trained MLP regressor. It takes in the following inputs: the trained regressor, a scaler object, the Depth-Anything depth map, the RGB image, and the pixel coordinates. The function extracts features from these inputs, converts them to a numpy array, scales them, and uses the regressor to make a prediction. The predicted depth value is then returned as a float.

3.2.2 Results and Analysis

For each image, a new $MLPRegressor$ was trained. 3 shows the Training and Validation Loss for one of the Images in the dataset we generated. The improved depth-map in which the missing depth values have been estimated using the $MLPRegressor$ is shown in 4.

The performance of the method is evaluated by removing a subset of depth values from a COLMAP depth map and asking an MLP regressor to estimate them. The estimated values are then compared to the actual COLMAP values to assess their similarity.

We achieved a Mean Squared Error (MSE) of 988.5062 and a Mean Absolute Error (MAE) of 19.2427 for the estimated depth values. Considering the inherent limitations of MDE, these values indicate that the model effectively captures the relative depth structure while filling in the missing information in the COLMAP depth map.

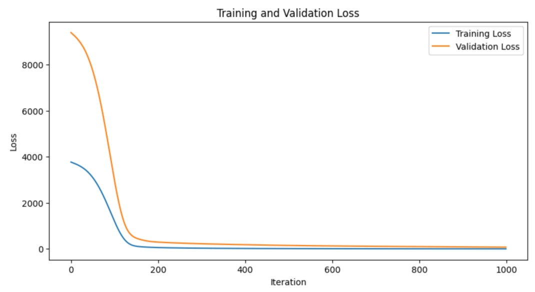

Figure 3. Shows the Training and Validation Loss when training the $MLPRegressor$ in an image from the dataset generated by us.

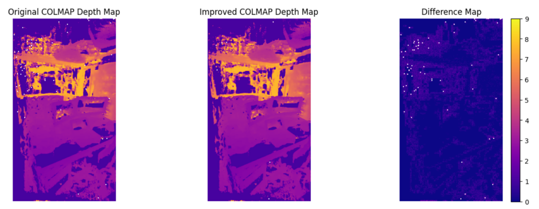

Figure 4. Shows the Training and Validation Loss when training the $MLPRegressor$ in an image from the dataset generated by us.

4. Faster Training of the Scene Landmark Detector

4.1. Methodology

Despite the advantages of the Scene Landmark Detector (SLD) approach being a storage-free visualization method, one drawback is the training time. The improved SLD approach [6] relies on an ensemble of models, each responsible for a non-intersecting subset of landmarks. Each convolutional neural network need to be trained from scratch for each scene, taking upwards of 60 hours on a GTX 1080ti depending on the size of the scene. As the long training time poses limitations for practical usage, we investigate methods for speeding up the training time.

Brachmann et al. [3] achieve a speed-up of over 100 times by employing a scene-agnostic backbone architecture and only training a smaller scene-specific head for each new scene. We also follow this approach and investigate whether a backbone can reliably provide features for general 3D landmarks. We propose a backbone-head architecture as seen in figure 5. After obtaining a suitable backbone, our models trains on each new scene by freezing the backbone and initializing a new scene head, which is smaller and thus faster to train.

To train the scene-agnostic backbone, we propose a multi-head training regime which trains on N scenes in parallel, illustrated in figure 6. We load the pre-trained checkpoint of ResNeXt-101 32x8d [10], which is a ResNeXt [12] architecture trained on billions of images from the internet. To achieve a 8x downsampling rate, we modify the architecture by adjusting some downsampling layers to retain spatial resolution and change the default 32x downsampling rate to 8x. Furthermore, the final feature dimension size is also reduced to 512 from 2048 originally. Then, N heads are initialized, one for each scene. The head model consists of a skip connection block of 1x1 convolutions and a final 1x1 convolution to produce one heatmap per landmark. For the backbone training pipeline to work, we implement a custom batch sampler which produces scene-homogeneous batches, ensuring that each batch contains samples from the same scene. After forming the batches, we then apply a random shuffling process to the batches across all scenes, not just within individual scenes. This approach ensures that while each batch is internally consistent in terms of scene context, the overall training process benefits from a diverse and randomized order of scene batches. To reliably train the backbone on N scenes, we need N different models, each sharing a single backbone. This means, in each update step, the backbone and one specific head, depending on the scene of the batch, is optimized. One epoch of training iterates through all batches of the N scenes exactly once. During the training of the backbone, we use a simpler head architecture than for the final scene training pipeline to lower the training time and prevent overfitting of the backbone.

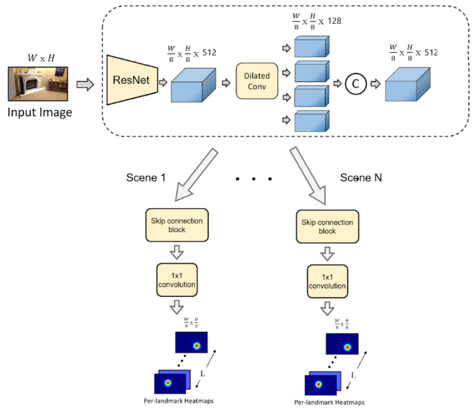
W x H
Input Image
ResNet
$\frac{W}{8}$ x $\frac{H}{8}$ x 512
Dilated Conv
$\frac{W}{8}$ x $\frac{H}{8}$ x 128
C
$\frac{W}{8}$ x $\frac{H}{8}$ x 512
Scene 1
Scene N
Skip connection block
1x1 convolution
$\frac{W}{8}$ x $\frac{H}{8}$
L
Per-landmark Heatmaps

Figure 6. Training setup for the backbone.

After training the backbone network, we leverage it to detect landmarks in new scenes by pre-filling a buffer with features generated by the backbone. This buffer is then used to train a scene head, comprising two convolutional blocks with dilated convolutions and skip connections. The head produces one heatmap per landmark, which is downsampled by a factor of 8 and then averaged for sub-pixel accu-

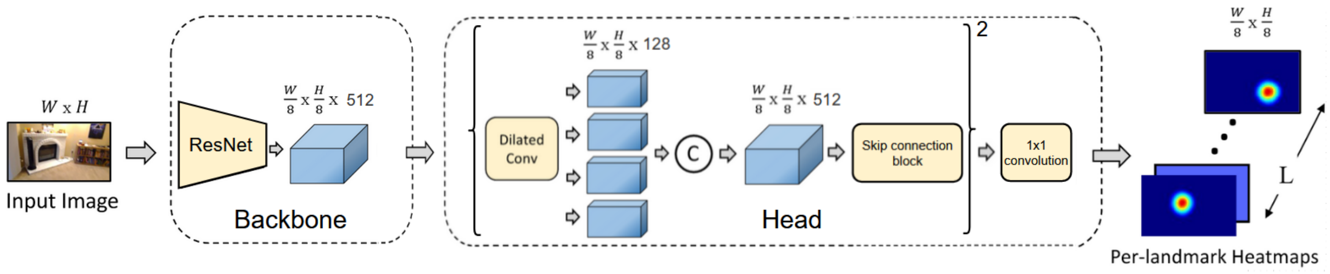

Figure 5. Network architecture of our proposed model. During training of a new scene we initialize a scene-specfic head and keep the backbone frozen. The exponent 2 in the figure denotes 2 consecutive blocks. For the backbone we use a pre-trained ResNeXt-101 32x8d architecture presented in [10] and fine-tune it on multiple scenes to achieve scene-agnostic capabilities. The head consists of two stacked dilated convolution and skip connection blocks, where the skip connection blocks contain 1x1 convolutions. Finally, a 1x1 convolutional layer outputs per-landmark heatmaps.

rate detection. By training only the scene head, we achieve a significant speedup of over 20 times compared to training the entire network from scratch. Additionally, utilizing PyTorch's Distributed Data Parallel (DDP) with 4 GPUs enables us to further accelerate the process by 75 percent, allowing us to quarter the time it takes to fill the buffer. This efficient approach enables us to quickly and accurately detect landmarks in new scenes.

4.2. Experiment

We perform all backbone training experiments on a RTX 3090 GPU and all head training experiments on GTX 1080ti GPUs. Even using the RTX 3090, which offers a 5x speedup over the GTX 1080ti in the backbone training pipeline, a single epoch for a single scene takes around 5-10 minutes depending on the size of the scene. Thus, we are limited in the amount of scenes we can train for in parallel. For the first backbone training experiments, we choose N = 3 scenes to train the backbone on. We use the Indoor6 [5] dataset, and we opt to use scenes 2a, 4a and 6 for the first experiments, as they contain the fewest amount of images. Even with these settings, a run of the backbone training pipeline takes around 24 hours by training for 200 epochs.

SLD* [6] uses an ensemble model by employing 8 scene landmark detector CNNs, each responsible for detecting 125 distinct 3D landmarks, totaling 1000 3D landmarks that are predicted in each input image. As we do exploratory research, all our experiments are only performed with a single model, where we use either 100 or 200 landmarks to detect. We found that using fewer landmarks helps as the backbone needs to learn from the set amount of landmarks of each scene, thus, multiplicatively many as in the standard SLD* approach.

We evaluated different backbones: EfficientNet [11] and ResNeXt (original SLD choice). We explored configurations with dilated convolutions in the backbone or scene-specific heads (backbone/head). A larger head with a smaller backbone improved recall but required longer training. Inspired by feedback, we investigated using a frozen Swin transformer [9] (hierarchical vision transformer) as the feature block. Due to limitations (memory, downsampling rate), only the first 2 layers were used, resulting in poor scene head accuracy. We believe this is due to the smallest Swin architecture and the lack of global self-attention.

Brachmann et al. [3] propose an alternative batch sampling strategy for backbone training. They accumulate gradients with a single forward pass for all scenes, followed by a single backward pass and optimization. We tested this approach on 3 scenes but observed no speedup due to imbalanced scene image counts. Training requires batch duplication for smaller scenes to ensure all scenes are iterated over once per epoch. This approach yielded similar pose recall and no significant speedup. While it uses fewer optimization steps, the benefit is negated by batch duplication, especially for few scenes.

We train our final backbone using all scenes from Indoor6 except scene 3 using a ResNext backbone and use a dilated convolution block plus skip connection block and a 1x1 convolution as scene head. We train the backbone using 100 landmarks per scene, batch size 8 for 300 epochs, taking multiple days on a single RTX 3090. The scenes are evaluated using 5cm/5° recall, the fraction of test images where the pose difference is less than 5 degrees and 5 cm. Head training is then performed with the trained backbone, where we evaluate the performance on unseen scenes, scene 3 from Indoor6 and our own generated scene, which we denote as scene 7. The head architecture can be seen in figure 5, we train for 200 epochs with a batch size of 8 and 100 landmarks. A direct comparison of recall values of SLD* and our model is shown below:

Scene	3	7
SLD*	34.3	53.2
Our	31.0	11.6

Table 1. Recall comparison of SLD* versus our model on unseen scenes.

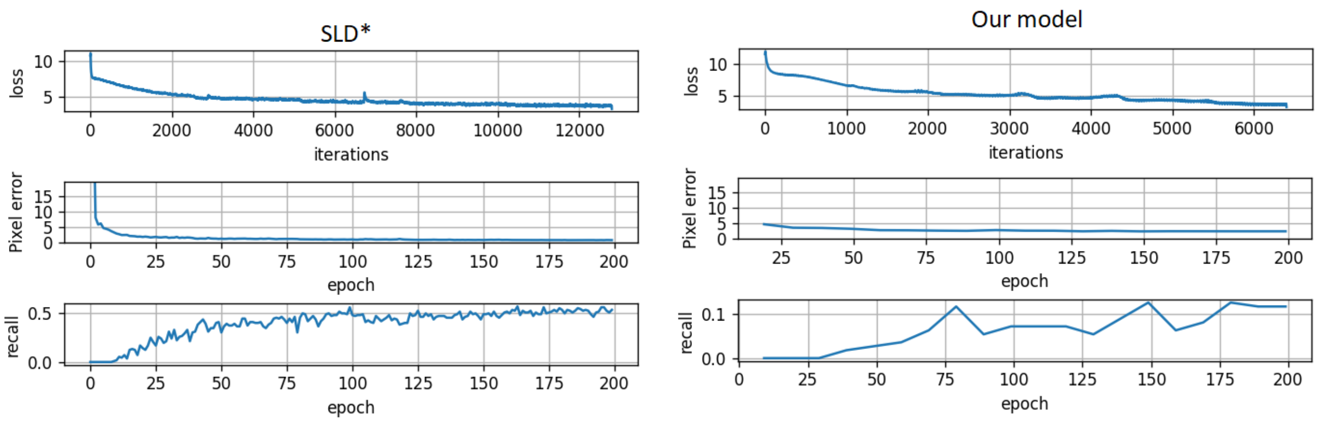

Figure 7. Loss, pixel error, recall curves during training of a scene landmark detector on scene 7. Left shows training using SLD* and the right figures show training using our head architecture.

4.3. Analysis

Looking at the results of the backbone training, we see that even in our final experiment the pose recall for specific scenes is significantly lower than the result set by the original SLD* model. Table 2 shows a direct comparison between our model and SLD*, where our backbone model is trained on 100 landmarks per scene and 5 scenes from Indoor6 and SLD* is trained on 100 landmarks for each specific scene of Indoor6.

Scene	1	2a	4a	5	6
SLD*	34.7	31.5	46.2	28.5	43.3
BB	25.8	25.3	33.3	12.7	33.1

Table 2. Comparison of recall between SLD* on 100 landmarks and multi-headed backbone (BB) model trained on 100 landmarks for each of the 5 scenes of Indoor6.

The disparity in recall shows that in our training regime, a single backbone was not able to provide sufficient features for all 5 scenes at the same time. Improvements for backbone training could be made in multiple aspects. As mentioned before, the training for ACE [3] involves 100 scenes from ScanNet [4] in parallel. This was done over 7 days. As we do not have access to a multi-GPU cluster with more capable GPUs, we are limited to using only few scenes and epochs.

When we compare the recall in the head training pipeline of Indoor6 scene 3 against scene 7, we see a stark difference in accuracy. While for scene3 our head is comparable to SLD* with the benefit of training 20x faster, for our own generated scene 7 our model performs significantly worse. Taking a closer look at the training curve of scene 7 in figure 7 however, we see that both loss and pixel error on the 2D landmark detections behave similarly, only showing a large difference in the recall scores. Our model likely overfits due to erroneous labels based on COLMAP depth maps. Generating better depth maps could improve training robustness. Additionally, landmark selection is crucial due to performance variation across scenes. As the backbone is trained on the first 100 landmarks of 5 scenes, similar features in the first 100 landmarks of all scenes would be beneficial. Modifying the landmark selection algorithm in [5] could address this.

5. Conclusion

We propose a novel training pipeline for the scene landmark detector approach, which can train a scene-agnostic backbone and a smaller head for each new scene. The new head training pipeline can train 20x faster than the original approach and we show that this result can achieve close to state-of-the-art recall on certain unseen scenes. The monocular depth estimation (MDE) is used to improve the dense reconstructions. It combined depth data from COLMAP (existing pipeline) with Depth-Anything (MDE model) to create a more complete depth map. The model successfully filled in missing information in COLMAP's depth maps, achieving good relative depth structure.

6. Contributions

Owen implemented the parallel scene training pipeline for the backbone and the head training pipeline, conducting all related experiments and analysis. Matthieu was involved with dataset creation, and Alen and Johanna improved depth maps using monocular depth estimation.

The SLD codebase provided the training framework (including dataset preparation and validation). We modified the pipelines for parallel backbone training, faster head training, and implemented new features: various architectures, batch sampling strategies, DDP, and pre-trained model adjustments.

References

[1] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023.

[2] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation, 2023.

[3] Eric Brachmann, Tommaso Cavallari, and Victor Adrian Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using rgb and poses, 2023.

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[5] Tien Do, Ondrej Miksik, Joseph DeGol, Hyun Soo Park, and Sudipta N. Sinha. Learning to detect scene landmarks for camera localization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11122–11132, 2022.

[6] Tien Do and Sudipta N. Sinha. Improved scene landmark detection for camera localization, 2024.

[7] Vitor Guizilini, Igor Vasiljevic, Dian Chen, Rares Ambrus, and Adrien Gaidon. Towards zero-shot scale-aware monocular depth estimation, 2023.

[8] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation, 2021.

[9] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[10] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[11] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[12] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017.

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