ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

Localization against Universal 3D Models

Master's Thesis

Eric Tüschenbönner
Department of Mechanical and Process Engineering D-MAVT

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

November 4, 2024

Abstract

Visual localization is essential for AR applications, enabling accurate placement of virtual content and collaborative experiences. State-of-the-art solutions rely on environment maps for stable and precise localization. While mapping small-scale spaces like rooms is relatively simple, the task becomes time-consuming in larger environments like warehouses or construction sites. However, existing 3D models of these environments, such as CAD models or LiDAR scans, offer a solution. Making use of these models has the potential to improve the AR experience and open up new applications.

Current visual localization methods primarily focus on real-world data or train and test on the same data type. However, localizing real-world images using synthetic data for training presents a significant challenge due to the domain gap between synthetic and real data, which has not been adequately addressed in existing solutions.

This project investigates adapting Scene Coordinate Regression for visual localization using CAD models, an area with limited prior research. We assess the compatibility of current visual localization methods with different types of 3D models, analyze their limitations, and explore strategies to enhance their performance. Our approach includes developing a synthetic data generation pipeline, implementing supervised training using scene coordinates as privileged information, and exploring transfer learning for domain adaptation to bridge the synthetic-real domain gap.

We conducted experiments on supervised training using scene coordinates and transfer learning for domain adaptation. Supervised training demonstrated faster convergence but did not improve localization performance compared to traditional reprojection loss. Transfer learning experiments, despite testing various approaches, did not yield consistent improvements in bridging the domain gap between synthetic and real data. A limited exploration of end-to-end training provided insights into potential future directions but did not result in significant improvements.

While our current approaches did not surpass existing methods, this work has identified several promising directions for future research in visual localization using synthetic data. The findings highlight the challenges of bridging the synthetic-real domain gap and indicate the need for more comprehensive methods, such as end-to-end training across multiple datasets and refined loss functions. This research contributes to the ongoing efforts in the field and underscores the complexity of leveraging synthetic data for real-world visual localization tasks.

Acknowledgements

I would like to express my sincere gratitude to my advisors at Magic Leap, Patrik Schmuck and Lukas Bernreiter, for their guidance and support throughout this thesis. Their expertise, direct involvement, and feedback have been invaluable in shaping this work. I'm grateful for the opportunity to work with them and for the resources provided by Magic Leap.

I would also like to thank Julia Chen and Paul-Edouard Sarlin from the Computer Vision and Geometry Group (CVG) for their organizational support and assistance during the project.

This experience has been a great learning opportunity. I've gained a lot of knowledge not only about visual localization, but also about long-term research projects and practical implementations. These experiences have improved my technical skills and contributed significantly to my professional and personal growth over the last 6 months.

Finally, I extend my thanks to all those who have supported me throughout this journey. Your encouragement has been crucial in the successful completion of this thesis.

Contents

1 Introduction **1**

2 Related Work **3**

3 Preliminaries **7**

4 Method **11**

4.1 Synthetic Data Generation 12

4.1.1 Rendering Pipeline 13

4.1.2 Scene Coordinates 15

4.2 Supervised Training using Scene Coordinates 17

4.3 Transfer Learning for Domain Adaptation 19

4.3.1 Feature Similarity 20

4.3.2 Feature Contrasting 21

4.3.3 Against Scene Coordinates 22

5 Experiments **23**

5.1 Supervised Training using Scene Coordinates 23

5.2 Transfer Learning for Domain Adaptation 26

5.2.1 Feature Similarity 28

5.2.2 Feature Contrasting 29

5.2.3 Against Scene Coordinates 34

6 Discussion & Conclusion **37**

6.1 Key Findings and Interpretations 37

6.1.1 Supervised Training using Scene Coordinates 37

6.1.2 Transfer Learning for Domain Adaptation 38

6.1.3 End-to-End Training 39

6.2 Limitations 39

6.3 Future Directions 39

6.4 Concluding Remarks 40

List of Figures

2.1 Steps of structure-based visual localization methods 3

3.1 Reference MVS and CAD models for each dataset 9

3.2 MVS-CAD model alignment after 3D registration 10

4.1 Synthetic data generation pipeline 12

4.2 Example of CAD model used for synthetic data generation 12

4.3 Outputs from synthetic data generation visualized 12

4.4 Overlays of real vs synthetic images 13

4.5 Automatically generated orbit poses around 3D model 14

4.6 MVS scene coordinates and synthetic data compared 16

4.7 GLACE [34] training pipeline 17

4.8 Reprojection and 3D loss functions visualized 18

4.9 Regressor consisting of scene-agnostic feature encoder and scene-specific prediction head. . 19

4.10 Encoder fine-tuning via feature similarity. 20

4.11 Encoder fine-tuning via feature contrasting. 21

4.12 Encoder fine-tuning against scene coordinates from scene-specific prediction heads. 22

5.1 Training loss terms for separate feature contrast training. 29

5.2 Validation loss terms for separate feature contrast training. 29

5.3 Training loss terms for combined feature contrast training 32

5.4 Validation loss terms for combined feature contrast training 32

5.5 Scene coordinate evaluation of real image using pre-trained encoder (dataset: Pantheon) . . 34

5.6 Scene coordinate evaluation of fake image using pre-trained encoder (dataset: Pantheon) . . 34

List of Tables

3.1 Datasets and chosen CAD models 8

5.1 Localization results for different loss functions (dataset: Pantheon) 24

5.2 Convergence compared between loss functions at 15K iterations (dataset: Pantheon) 25

5.3 Original localization results with same data domain used for training and testing. 26

5.4 Original localization results with domain gap between data used for training and testing. . . 27

5.5 Weights for loss terms in separate feature contrast training. 29

5.6 Localization results on fake data with fine-tuned encoder and original head for different weights 30

5.7 Localization results on fake data with fine-tuned encoder and updated head for 0.8/0.2 . . . 31

5.8 Localization results on real data with pre-trained encoder and updated head for 0.8/0.2 . . . 31

5.9 Weights for loss terms in combined feature contrast training 32

5.10 Localization results on fake data with fine-tuned encoder and updated head for 0.4/0.3/0.3 . 33

5.11 Localization results on real data with pre-trained encoder and updated head for 0.4/0.3/0.3 . 33

5.12 Scene coordinate evaluation results (real head training, dataset: Pantheon) 35

Chapter 1

Introduction

Accurate visual localization, the task of estimating a camera's pose from images, is crucial for various robotics applications, particularly in augmented reality (AR). AR requires precise alignment of virtual and real scenes, necessitating a detailed environmental representation. While mapping can be time-consuming for larger scenes, using pre-existing 3D models (e.g., CAD models or LiDAR scans) allows direct localization without prior mapping.

Existing localization solutions using 3D representations include structure-based methods and Scene Coordinate Regression (SCR). Both approaches follow similar steps: feature extraction, 2D-3D correspondence determination, and camera pose estimation, differing mainly in the second step. Structure-based methods use explicit 3D models and can be further categorized into reconstruction-based approaches [26, 19, 30, 23, 2, 36] (computationally expensive) and depth-based approaches [15, 16] (requiring depth information). In contrast, SCR [34, 5, 7] uses an implicit 3D representation via a neural network trained on images with known poses to predict 3D coordinates from pixels for a specific scene. Current methods primarily focus on real-world data localization, or when considering synthetic data, train and test on the same data type. However, our use case requires localization in real-world images using synthetic data for training. We therefore need to address the domain gap between synthetic and real data to ensure compatibility with both data types.

This project focuses on adapting Scene Coordinate Regression for visual localization with CAD models, which present unique challenges due to their lack of photorealistic details and varying quality. To our knowledge, this combination has not been previously explored. We investigate the feasibility of this approach by assessing the compatibility with existing methods, analyzing their limitations, and proposing improvements. Our contributions include: (1) a synthetic data generation pipeline enabling CAD model compatibility with various localization methods; (2) supervised training against ground truth scene coordinates from 3D models to improve training efficiency; and (3) transfer learning for domain adaptation by fine-tuning feature extraction to bridge the synthetic-real domain gap.

This report is structured as follows. Chapter 2 provides a detailed overview of related work in the field of structure-based visual localization. Chapter 3 outlines the setup and prerequisites related to the datasets used in this work. Chapter 4 describes the proposed methodology for each contribution in detail, followed by the experiments and results in Chapter 5. Chapter 6 discusses the overall results and implications of the work, and concludes the report.

Chapter 2

Related Work

Visual Localization, the task of determining a camera's 3D pose from images, encompasses various methods including structure-based approaches, Scene Coordinate Regression, Absolute Pose Regression, Relative Pose Regression, and Pose Interpolation. Structure-based methods [19, 36, 15, 23, 2, 30, 26] make use of a 3D scene representation. Similarly, Scene Coordinate Regression [7, 5, 34] uses an implicit scene representation in the form of a neural network to predict 3D coordinates from images. Absolute Pose Regression [12, 35, 22] aims for an end-to-end approach to directly predict the camera pose from the query image, without any intermediate steps. Relative Pose Regression and Pose Interpolation use similar images for pose estimation. Image Retrieval using matching of global features [3, 17, 8, 38] can be implemented as an initial step to improve performance on larger scenes by reducing the search space.

Most state-of-the-art visual localization methods are structure-based, including structure-based Scene Coordinate Regression. Structure-based methods have the following general steps: (1) extract features from the query image, (2) establish 2D-3D correspondences between features and the scene, and (3) estimate the camera pose from the 2D-3D correspondences via a Perspective-n-Point (PnP) and Random Sample Consensus (RANSAC) loop, see Figure 2.1.

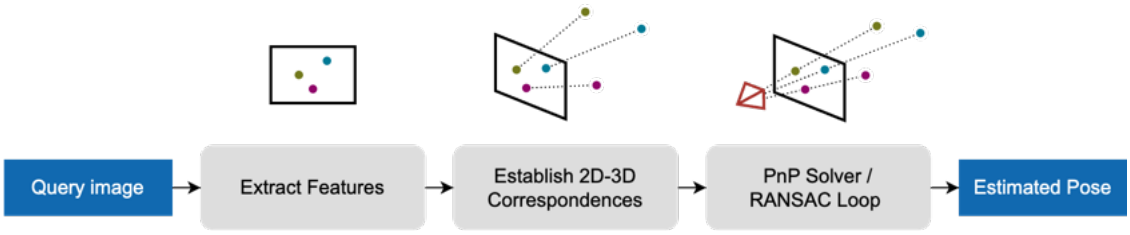


Figure 2.1: Steps of structure-based visual localization methods

The main difference between different structure-based approaches is the second step, i.e. how 2D-3D correspondences are established, whereas the first and third steps tend to be very similar. This can be attributed to feature extraction effectively capturing the most important information from an image, especially with modern learned features [9, 18, 10, 32] or those optimized specifically for localization pipelines [12, 22, 5]. The third step, pose estimation, is also well-established and can be solved robustly with PnP and RANSAC. The main challenge thus lies in the second step, establishing 2D-3D correspondences.

Feature Matching

Feature-matching-based methods establish 2D-3D correspondences using either triangulation or depth maps. Triangulation via Structure-from-Motion (SfM) [26, 27] uses feature matches from multiple images to explicitly calculate 3D coordinates and create a sparse 3D point cloud, which is used to localize a query image by matching its features with the 3D points. Examples include HF-Net [19], InLoc [30], ActiveSearch [23], CANN [2], and, GoMatch [36]. SfM is generally accurate and robust but computationally expensive and require large descriptor storage.

HF-Net [19] is a hierarchical localization method that uses a coarse-to-fine approach, simultaneously predicting global descriptors and local features. It first matches global descriptors via a NetVLAD [3] layer to obtain a location hypothesis, followed by local feature matching to establish 2D-3D correspondences with a SfM reconstruction. HLoc[1] implements HF-Net and is configurable with multiple local features [9, 32, 10, 18, 14] and matchers [20, 13, 29], but works best with SuperPoint [9] and SuperGlue [20].

Depth maps offer an alternative to triangulation, directly using feature matches to convert pixels into 3D coordinates using database depth values. This approach is faster and more memory-efficient than SfM but requires depth data for training.

MeshLoc [15] is a mesh-based localization method that makes use of depth maps rendered from a dense 3D mesh model (instead of a sparse point cloud) of the scene. It matches local features from the query image with database images and uses the rendered depth maps to establish 2D-3D correspondences. Since the representation is geometry-based, it is independent of the feature type and more memory-efficient than SfM. MeshLoc uses Patch2Pix [37], SuperPoint [9] with SuperGlue [20] via the ImMatch[2] toolbox.

CadLoc [16] evaluates the MeshLoc [15] pipeline on CAD models with varying quality, comparing them to reference Multi-View Stereo (MVS) models. Different scenes from the IMC 2021 Phototourism [11, 31] image dataset with available reference MVS models are used, while CAD models are obtained from 3D model sharing websites. The CAD models vary in terms of photorealism of colors and textures and geometric complexity, ranging from simple approximations to fine details. This is reflected in the results, where realistic models with colors and textures perform better than simple geometry approximations.

CadLoc serves as inspiration for our work, and we use the same image dataset and a subset of the CAD models. To our knowledge, CadLoc is the only existing work that evaluates visual localization specifically on CAD models, however, it does not address the domain gap between synthetic and real data since it trains and tests on the same data type. In this work we go beyond feature-matching based approaches and seek to not only evaluate but also adapt to CAD models.

[1] https://github.com/cvg/Hierarchical-Localization/
[2] https://github.com/GrumpyZhou/image-matching-toolbox

Scene Coordinate Regression

Scene Coordinate Regression (SCR) represents an alternative approach to feature matching for establishing 2D-3D correspondences. SCR trains a neural network as an implicit scene representation to regress pixel-wise 3D coordinates. This allows for directly establishing 2D-3D correspondences without explicit feature matching. The output of the network is a scene coordinate (i.e. 3D point) for each patch of pixels in the input image.

DSAC* [7], published in 2020, uses end-to-end training through a deep neural network, including a pose solver via Differentiable RANSAC (DSAC). Information input is flexible, ranging from RGB only, RGB-D (depth), and dense/sparse 3D point clouds for training, with DSAC* adapting its loss function between reprojection error (implicit triangulation) and 3D Euclidean distance between predicted and ground truth 3D coordinates. It achieved state-of-the-art localization accuracy on public datasets, specifically an error of less than 5cm/5° on more than 95% of the test images of 7Scenes [28] and 12Scenes [33] with a model size of 28 MB. In terms of training time, while it was reduced from 6 days down to 15 hours compared to its predecessor DSAC++ [6], this is still an order of magnitude slower than 3D reconstruction required by feature-matching-based approaches.

ACE [5] (Accelerated Coordinate Encoding), published in 2023, pioneered fast training with a more than 100x speed-up to only 5 minutes on a modern GPU. It reduced the model size to 4 MBs while maintaining similar accuracy to DSAC*. This is achieved by splitting the network into (1) a pre-trained scene-agnostic feature encoder that extracts relevant features and (2) a scene-specific MLP prediction head that predicts scene coordinates from features. Training efficiency is improved via simultaneous optimization across multiple viewpoints in large batches and gradient decorrelation that randomly samples patches from different images across the batch. Instead of end-to-end training like DSAC*, ACE uses a robust pose solver. Reprojection loss requires no depth data, but is also the only loss option, not being able to natively take advantage of privileged depth or 3D information.

GLACE [34] (Global-Local Accelerated Coordinate Encoding), published in 2024, builds upon ACE with improved scalability and generalization. It captures co-visibility constraints by introducing global features (off-the-shelf R2Former [38]) and concatenating them with local features (ACE's pre-trained feature encoder) to pass onto the prediction head. Moreover, a position decoder is used to better parametrize the output position on large-scale scenes by using multiple cluster centers. The model size is slightly larger than ACE on small scenes (9-13 MB vs. 4 MB), but can be adapted to larger scenes by increasing the network layers (number of residual blocks) in the prediction head. This allows for a single network to effectively capture large-scale scenes, whereas ACE requires an ensemble of models with larger model size to achieve similar performance (13MB vs 16MB for 4xACE on Cambridge Landmarks [12], 27MB vs. 205MB for 50xACE on Aachen Day [24, 25]), yet training time also scales exponentially.

We use the GLACE architecture as our baseline to evaluate and adapt Scene Coordinate Regression to CAD models, as it is the most recent and efficient method for training and inference. Similarly, we also use ACE's pre-trained feature encoder as a starting point for transfer learning.

Chapter 3

Preliminaries

This chapter provides the setup and prerequisites necessary for the method and experiments presented in this work. Firstly, it covers how and why different types of 3D models are used, outlining their benefits and limitations. Secondly, it introduces the datasets used, including the reference MVS models and CAD models. Lastly, it explains the 3D registration process, which aligns the CAD models to the reference MVS models for evaluation.

Types of 3D Models

3D models primarily come in two forms: point clouds and mesh models. Point clouds are collections of 3D points without defined connections, typically generated from LiDAR scans or 3D reconstructions. Mesh models, often created in CAD software, consist of triangles forming surfaces.

This work focuses on CAD models, which are less explored in visual localization research. CAD models offer the advantage of avoiding time-consuming and computationally expensive reconstruction processes while providing accurate global geometry. However, they often lack photorealistic details such as colors, textures, and fine-grained structures, and their quality can vary significantly.

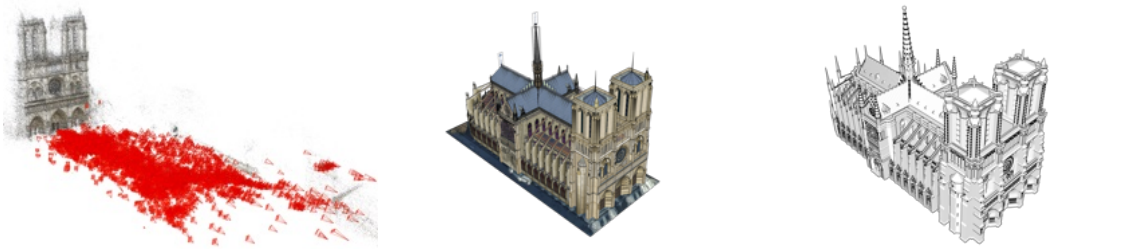
For evaluation purposes, we utilize MVS point cloud reconstructions from real images generated via COLMAP [27, 26] as ground truth models. These MVS models tend to be more spatially accurate and photorealistic than CAD models, but sparser than dense mesh models. In our work, MVS models serve as reference models with ground truth poses, effectively connecting real-world images to CAD models.

Datasets

For this study, we require high-quality datasets comprising diverse real-world images and accurate CAD models of the same scenes. We utilize the Image Matching Challenge (IMC) 2021 Phototourism dataset [11, 31], which contains images and reconstructions of popular European landmarks. Selected MVS models from this dataset were chosen based on the availability of corresponding CAD models, specifically for Notre Dame, Pantheon, Reichstag, and Brandenburg Gate. The CAD models were sourced from 3D model sharing websites, following the approach of CadLoc [16]. Table 3.1 provides an overview of the datasets, chosen CAD models, and their sources. Figure 3.1 offers a visualization of all models

Dataset	**Images**	**3D Points**	**CAD Model 1**	**CAD Model 2**
Notre Dame	3765	488895	Notre Dame B	Notre Dame E
Pantheon	1401	488895	Pantheon B	Pantheon C
Reichstag	75	17823	Reichstag A	Reichstag B
Brandenburg Gate	1363	100040	Brandenburg Gate B	Brandenburg Gate C

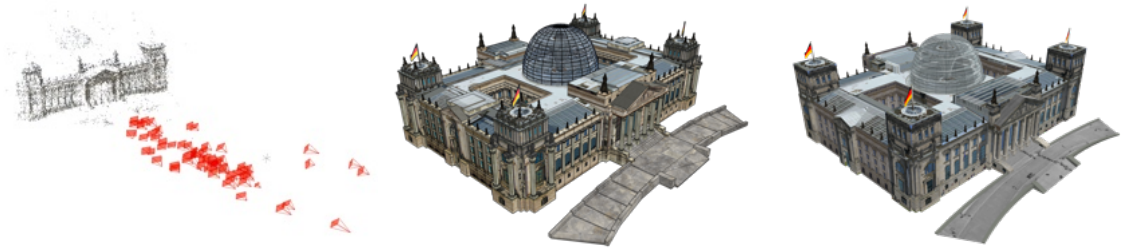
Table 3.1: Datasets and chosen CAD models



(a) Notre Dame Reference (b) Notre Dame B (c) Notre Dame E

(d) Pantheon Reference (e) Pantheon B (f) Pantheon C

(g) Brandenburg Gate Reference (h) Brandenburg Gate B (i) Brandenburg Gate C

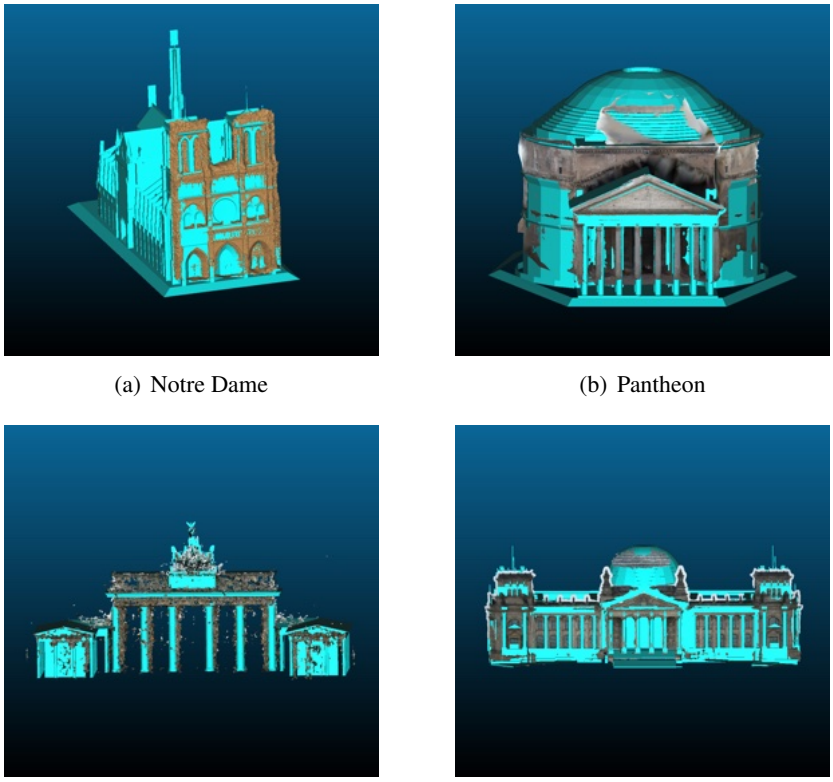


(j) Reichstag Reference (k) Reichstag A (l) Reichstag B

Figure 3.1: Reference MVS and CAD models for each dataset

3D Registration

Accurate registration of CAD models to their reference MVS models is crucial for evaluation. This process establishes a ground truth reference transformation $T_{\text{MVS CAD}}$ (CAD in MVS frame). Registration is performed using CloudCompare [1], involving the selection of equivalent points and/or fine ICP registration. Figure 3.2 illustrates the alignment results.



(a) Notre Dame (b) Pantheon

(c) Brandenburg Gate (d) Reichstag

Figure 3.2: MVS-CAD model alignment after 3D registration

Chapter 4

Method

The aim of this thesis is to investigate how different types of 3D models can be used for the task of visual localization. This includes evaluating how existing structure-based visual localization methods perform on such models, analyzing their limitations, and exploring strategies to enhance performance by leveraging the available 3D information. We focus on addressing the domain gap between real images and synthetic data generated from CAD models, by training Scene Coordinate Regression on synthetic data and querying with real-world images.

Existing 3D models - often available for buildings, construction sites, and warehouses - can be used for visual localization in larger scenes, where mapping the environment becomes time-consuming. Instead of capturing large amounts of real-world data for 3D reconstruction, synthetic data from pre-existing 3D models can be used to accelerate the mapping and localization process. While this method has been explored with realistic point clouds [21] and mesh models [15] from 3D scans, CAD models present a greater challenge and are less explored in research.

The challenge is to overcome the domain gap between synthetic and real-world data, i.e. making the different data types compatible with each other. CAD models usually provide accurate global geometry, but often lack photo-realism and local details. There is also a significant variability in model quality between different CAD models. It is crucial to address these issues to achieve accurate and reliable visual localization.

In particular, we make the following contributions, that will be detailed in the following sections:

1. Section 4.1: An automatic process for synthetic data generation from CAD models to create comprehensive training datasets compatible with existing visual localization methods.
2. Section 4.2: Adapting modern Scene Coordinate Regression to train using scene coordinates, making effective use of the privileged information available from 3D models to improve training efficiency.
3. Section 4.3: Transfer learning for domain adaptation, to bridge the gap between synthetic and real-world data to improve generalization of Scene Coordinate Regression.

4.1 Synthetic Data Generation

This section covers the process for generating synthetic data from CAD models that can be used for training and evaluation of visual localization methods. At first, the rendering pipeline is described, followed by generation of synthetic scene coordinates to be used for Scene Coordinate Regression.

As shown in Figure 4.1, the 3D model is the input to the rendering pipeline, which generates synthetic images and depth maps from different viewpoints. Next, the scene coordinates are reprojected from the depth maps using the camera intrinsics and poses. An example of a CAD model used for synthetic data generation is shown in Figure 4.2 and the outputs are visualized in Figure 4.3, including a rendered image, depth map, and scene coordinates.

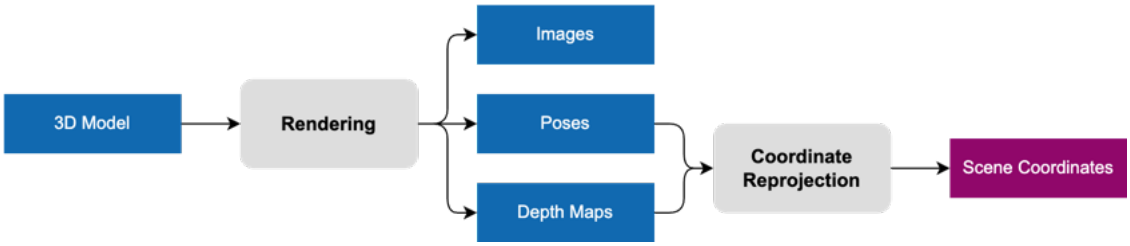


Figure 4.1: Synthetic data generation pipeline

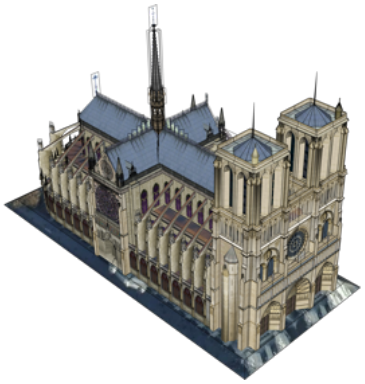
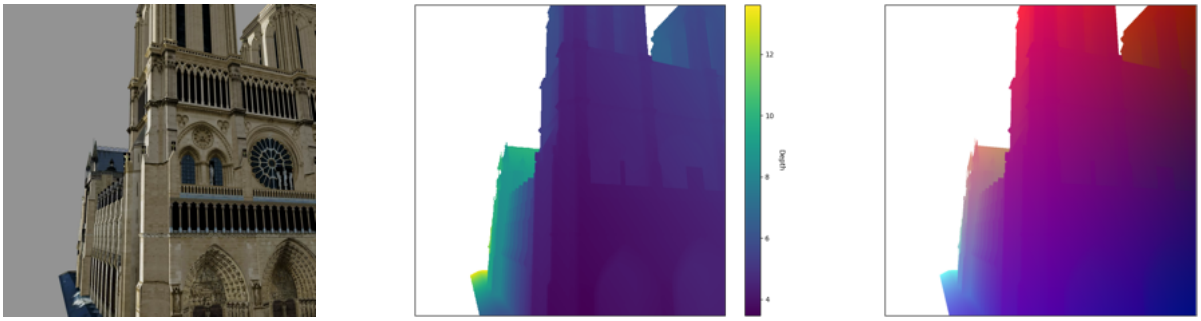


Figure 4.2: Example of CAD model used for synthetic data generation



(a) Rendered Image (b) Depth Map (c) RGB Scene Coordinates

Figure 4.3: Outputs from synthetic data generation visualized

The generated datasets are compatible with different existing localization pipelines. Specifically, format conversions to MeshLoc [15] and GLACE [34] are implemented in the code. The dataset conventions used for these localization pipelines are likely compatible with a wide range of other methods as well. Moreover, synthetic datasets have been tested on MeshLoc for data verification before training Scene Coordinate Regression.

4.1.1 Rendering Pipeline

The task of the rendering pipeline is to generate synthetic images of a CAD model from different camera poses and intrinsics to create a diverse training dataset for visual localization. It is implemented in Blender [4] due to its flexibility, powerful rendering engine, and Python integration. There are two main scenarios: (1) creating renders corresponding to the specific poses of the MVS reference model, and (2) rendering of automatically generated orbit/ground poses around the object. The former is especially useful for evaluation against real-world data, while the latter more closely resembles the actual use case, where real data is not available and there are no ground truth poses.

MVS Reference Poses

To render images from the same viewpoints as the MVS reference model, the poses and intrinsics are extracted from the reconstruction files. Next, the MVS poses need to be converted into the coordinate system of the CAD model to be rendered correctly. This is done using the reference transformation $T_{\text{CAD MVS}}$ obtained from 3D registration in Equation (4.1).

$$T_{\text{CAD cam}} = T_{\text{CAD MVS}} \cdot T_{\text{MVS cam}} \tag{4.1}$$

The output is a set of rendered images from the MVS reference poses, which can be used for evaluation against the reference model. Overlays of real images and corresponding renders are shown in Figure 4.4. The alignment is imperfect due to the nature of the CAD model and registration accuracy.

(a) Notre Dame (b) Pantheon

Figure 4.4: Overlays of real vs synthetic images

Automatic Orbit Poses

To render images without specific ground truth poses, reflecting the actual use case, automatic orbit rendering is used to generate a diverse set of camera poses around the object. The only parameters that need to be specified are the horizontal angles, vertical angles, distances, and intrinsics. The camera pose is automatically calculated such that it tracks the object center. Lighting is dynamically adjusted to follow the camera pose to ensure suitable illumination.

A simplified example of automatic orbit poses is shown in Figure 4.5. Here, there are 72 horizontal angles (every 5 degrees), 2 vertical angles, 2 distances, and fixed intrinsics for a total of 288 renders.

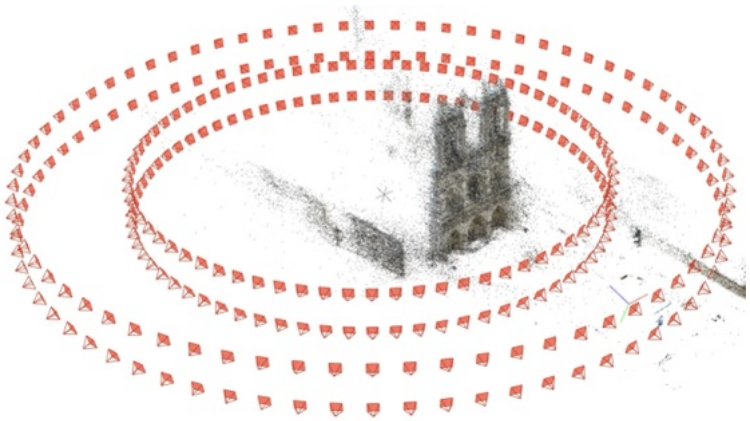


Figure 4.5: Automatically generated orbit poses around 3D model

Additional options include a height setting and offset angles. The height setting specifies the camera heights above the ground instead of the vertical angles to simulate more realistic camera poses, such as a person holding a device rather than aerial views. Offset angles in horizontal and vertical directions can be specified to tilt the camera away from the object center, introducing more variation in the dataset.

By setting additional parameters, such as multiple intrinsics, vertical angles, and offsets, the pipeline is capable of generating thousands of diverse renders. Note that multiple orbit render operations can be combined to achieve specific scenarios without rendering combinations, such as combining wide-angle close-up views with telephoto distant views.

4.1.2 Scene Coordinates

Scene coordinates that can be used for supervised training using 3D loss in Section 4.2 and evaluation are generated from the rendered depth maps or MVS point clouds. Whereas rendered dense depth maps can be projected to scene coordinates, MVS point clouds require a different approach.

Synthetic Scene Coordinates

Using the rendered depth maps, camera poses, and intrinsics, ground truth scene coordinates can be generated for the synthetic images. In Equation (4.2), the conversion from pixel coordinates (u, v) and depth (z) to scene coordinates in the camera frame is shown using the camera intrinsics – principal point (c_x, c_y) and focal length (f_x, f_y) expressed in pixels. Equation (4.3) shows the conversion from local camera frame to global CAD frame using the transformation matrix $T_{\text{CAD cam}}$, which is the current pose of the camera in the CAD frame. Expressed in a global frame, the scene coordinates are consistent across the dataset. See Figure 4.3(c) for a visualization of the generated dense scene coordinates.

$$ {}_{\text{cam}}\mathbf{t}_{\text{cam point}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \left(\frac{u-c_x}{f_x}\right) \cdot z \\ \left(\frac{v-c_y}{f_y}\right) \cdot z \\ z \end{bmatrix} \tag{4.2} $$

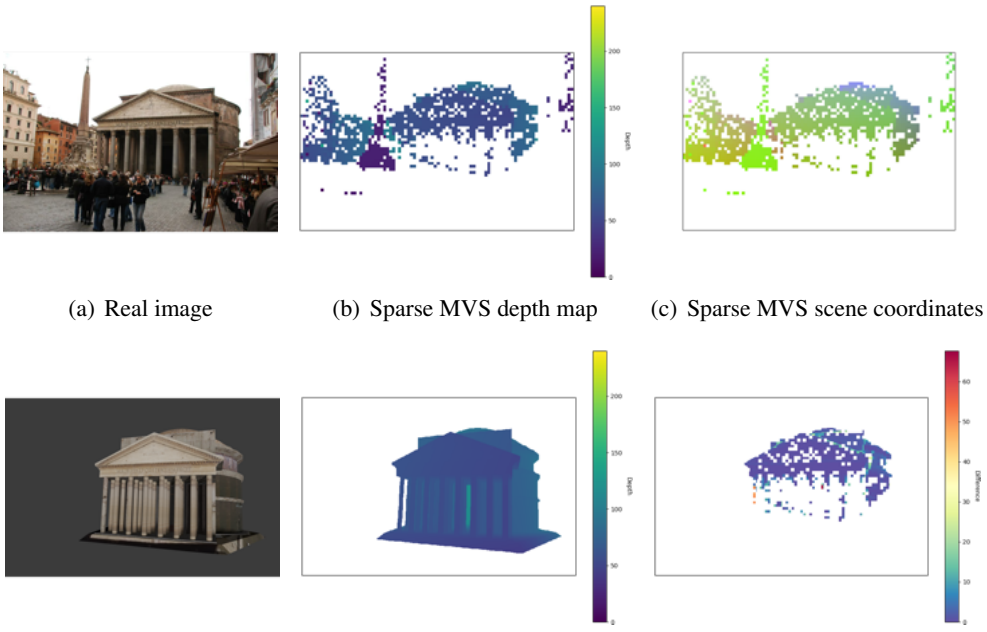
$$ \begin{bmatrix} {}_{\text{CAD}}\mathbf{t}_{\text{CAD point}} \\ 1 \end{bmatrix} = T_{\text{CAD cam}} \cdot \begin{bmatrix} {}_{\text{cam}}\mathbf{t}_{\text{cam point}} \\ 1 \end{bmatrix} \tag{4.3} $$

Real Scene Coordinates

For real-world data, the scene coordinates are generated from the MVS point clouds. Here, all the 3D coordinates are already known, but need to be allocated to the corresponding image pixels.

The steps are as follows. Firstly, the 3D points are loaded in the global frame and converted to the camera frame using the ground truth pose. Next, the 3D points are projected to the image, and points outside the image are ignored. Then, points that are too far away are filtered out. Finally, a scene coordinate is assigned to a pixel if that pixel is not yet filled or the point is closer than the current scene coordinate. Meanwhile, the depth map is also filled with the corresponding depth values. See Figure 4.6 for the results of this process and a comparison with synthetic data.

Subsampling to patches instead of individual pixels can be directly integrated into this process, without the need for a dense intermediate step. This method is adapted and generalized from GLACE's [34] Cambridge dataset setup.



(a) Real image (b) Sparse MVS depth map (c) Sparse MVS scene coordinates

(d) Rendered image (e) Dense synthetic depth map (f) Depth difference

Figure 4.6: MVS scene coordinates and synthetic data compared

The figure shows (a) the real image, its generated (b) sparse MVS depth map, and (c) scene coordinates. The corresponding synthetic data includes (d) the rendered image, (e) its generated dense depth map, and (f) the depth difference between the real and synthetic data, which verifies the alignment of the scene coordinates and the correctness of the synthetic data generation process.

4.2 Supervised Training using Scene Coordinates

This section covers supervised training of Scene Coordinate Regression, particularly GLACE [34], using ground truth scene coordinates to improve training efficiency by leveraging existing 3D information. We compare reprojection loss with 3D loss and a mixed loss switching between the two during training.

GLACE's training pipeline is shown in Figure 4.7, including global feature extraction, buffer generation, and network training processes. Global features are extracted from the database images for storage in a feature-lookup table. Meanwhile, a training buffer is generated with local features, ground truth data, and other necessary information. The network is then trained on shuffled patches from the buffer and their corresponding global features against a loss function using the ground truth data. The output is a scene-specific neural network.

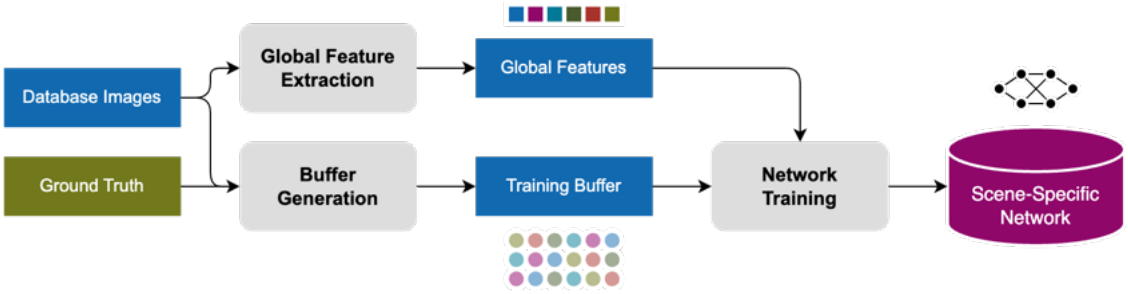


Figure 4.7: GLACE [34] training pipeline

ACE [5] and GLACE [34] are limited to unsupervised training using reprojection loss via the ground truth poses – a great approach for real-world data where ground truth scene coordinates are usually not available or expensive to reconstruct. The results using reprojection loss are accurate, but it takes some time to converge to reasonable values during training. To improve training efficiency and accelerate convergence, we can use the privileged information generated from 3D models to train the network directly against a supervised 3D loss. We can also start training with the 3D loss and switch to the reprojection loss to combine the potential benefits of both loss functions.

Note that the loss is not actually computed for individual pixels in the image, but for each patch (default: 8×8 pixels). ACE and GLACE extract a feature vector for each patch, which is then used to predict the scene coordinate for that patch, i.e. its center. This approach is more efficient than predicting scene coordinates for each pixel individually.

For the 3D loss to be able to access the ground truth scene coordinates, the dataloader needs to include them in the training buffer and shuffle them to correspond with the features and other data. During loss computation, the invalid coordinates are masked out according to the defined ground truth coordinates.

Loss Functions

Reprojection loss computes the 2D distance between the original pixel location (x_i) and its predicted 3D scene coordinate (y_i) reprojected back into image space using the camera intrinsics (K) and ground truth pose (H) available during training. See Equation (4.4) for the loss function and Figure 4.8(a) for a visual representation.

$$l_\pi(x_i, y_i) = \|x_i - \pi(K, H, y_i)\|_2 \tag{4.4}$$

The 3D loss directly computes the 3D Euclidean distance between the predicted scene coordinate (y_i) and the ground truth 3D scene coordinate (y_i^*) that is generated from the 3D model. See Equation (4.5) for the loss function and Figure 4.8(b) for a visual representation.

$$l_{3D}(y_i,\ y_i^*) = \|y_i - y_i^*\|_2 \tag{4.5}$$

(a) Reprojection loss

(b) 3D distance loss

Figure 4.8: Reprojection and 3D loss functions visualized

4.3 Transfer Learning for Domain Adaptation

This section covers transfer learning to bridge the domain gap between synthetic and real-world data. We focus on adapting the encoder of the Scene Coordinate Regression network to synthetic data using different training settings and loss functions.

As shown in Figure 4.9, the regressor network consists of a scene-agnostic feature encoder and a scene-specific prediction head. The encoder was pre-trained by ACE [5] and is fixed during training, while only the prediction head is updated during training. GLACE [34] concatenates global features from R2Former [38] with local features from the encoder to predict scene coordinates.

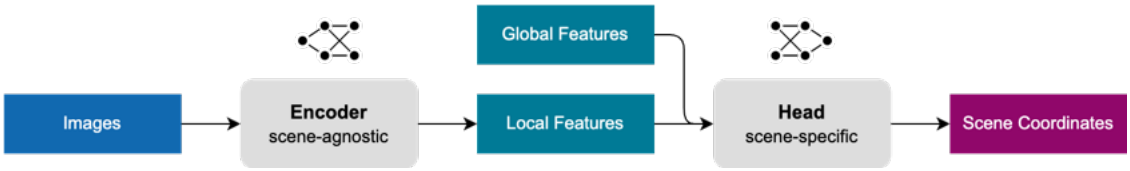


Figure 4.9: Regressor consisting of scene-agnostic feature encoder and scene-specific prediction head.

The problem is that the pre-trained encoder was learned on real images only and does not perform well on synthetic data. By using transfer learning, we aim to adapt the encoder to generalize across the domain gap between synthetic and real-world data.

We compare two main training settings: (1) a separate encoder for synthetic images and (2) a combined encoder for both real and synthetic images. The separate encoder is pre-trained on real-world data and fine-tuned for synthetic data, whereas the combined encoder is trained on both datasets simultaneously to be compatible with both. For each setting, we explore different methods for encoder fine-tuning, including feature similarity, contrastive learning, and scene coordinates.

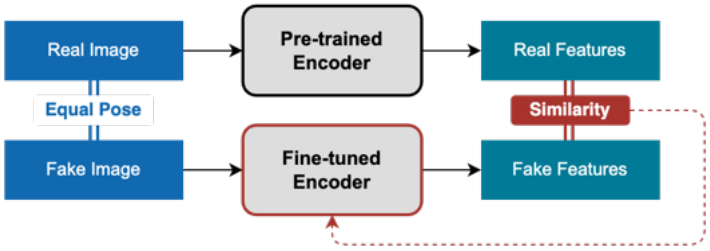
Data Loading

A new data loader is implemented to output real and synthetic images, their validity masks (from depth maps), ground truth scene coordinates, and global features. It handles multiple datasets simultaneously, such that the fine-tuned encoder remains scene-agnostic. For training by contrasting features (Section 4.3.2), a far-away negative sample (calculated using the ground truth pose) from the same dataset is included in the batch, randomized for each epoch. For training against scene coordinates (Section 4.3.3), it also tracks the dataset name to choose the correct scene-specific prediction head.

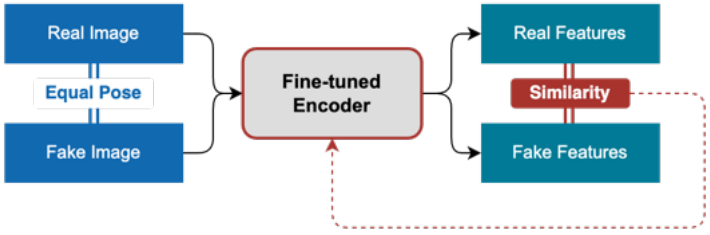
4.3.1 Feature Similarity

The simplest approach is to fine-tune the encoder such that the features of synthetic images become more similar to the features of corresponding real images. Similarity is measured using cosine loss and a magnitude term is added to avoid degenerate solutions.

In Figure 4.10 the training process is shown, where a real image and a corresponding synthetic image of the same pose are passed through the encoder. In the separate setting, the real image is passed through the pre-trained encoder, while the synthetic image is passed through the encoder to be fine-tuned. In the combined setting, both images are passed through the same encoder that is fine-tuned for compatibility with both datasets.



(a) Separate encoder for synthetic images



(b) Combined encoder for real and synthetic images

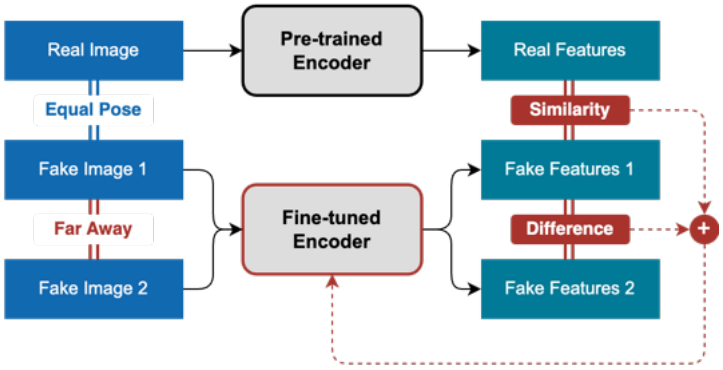
Figure 4.10: Encoder fine-tuning via feature similarity.

The similarity loss function is a cosine loss with target 1.0 for maximum similarity and margin 0.1 since the features don't need to be exactly the same. The magnitude term uses target 1.0 and margin 0.15, approximately maintaining the feature norms from the pre-trained encoder.

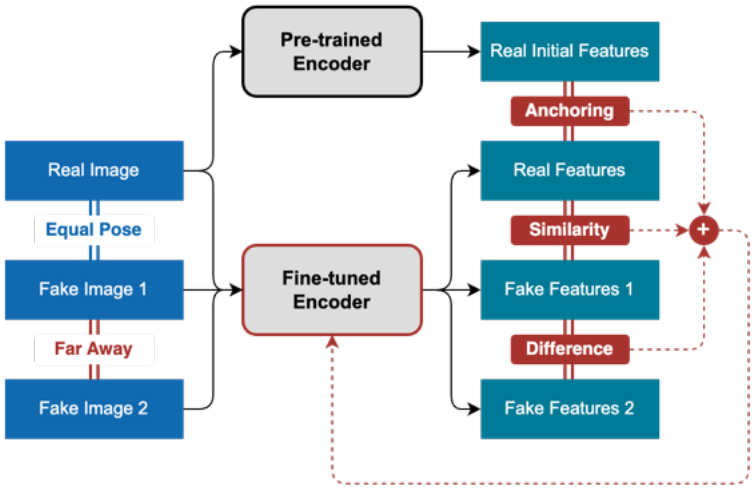
4.3.2 Feature Contrasting

A more sophisticated approach is to train the encoder to make features of synthetic images similar to features of corresponding real images, while maximizing distinctiveness to far-away synthetic views from the same dataset. An additional anchoring term is used in the combined setting.

In Figure 4.11 the training process is shown, where a real image, a corresponding synthetic image, and a far-away synthetic image from the same dataset are passed through the encoder. In the separate setting, the real image is passed through the pre-trained encoder, and the synthetic images are passed through the encoder to be fine-tuned, whereas in the combined setting, all images are passed through the same encoder. An anchoring term is also computed from the real image being passed through the pre-trained encoder. Therefore, the loss is a combination of two or three terms, depending on the setting.



(a) Separate encoder for synthetic images



(b) Combined encoder for real and synthetic images

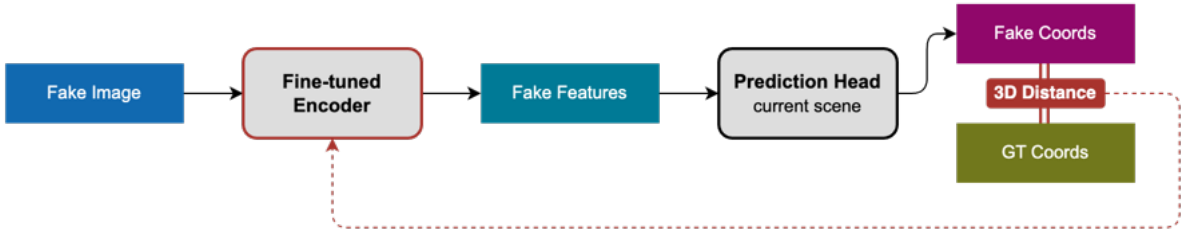
Figure 4.11: Encoder fine-tuning via feature contrasting.

The similarity loss function is the same as in Section 4.3.1 with target 1.0 and margin 0.1, while the difference loss function uses target -1.0 and margin 0.3, as the features should be distinct but don't need to be exactly opposite. The anchoring term uses a cosine loss with target 1.0 and margin 0.2, also promoting similarity but with a higher margin. The magnitude term is the same as in Section 4.3.1.

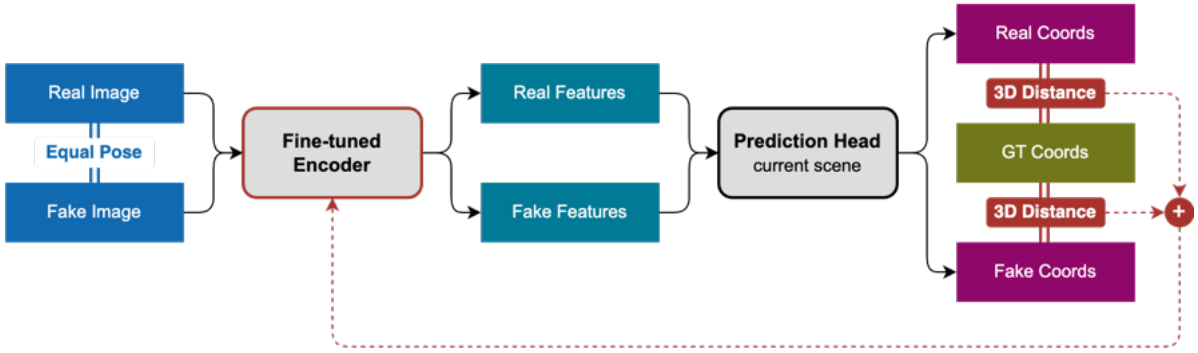
4.3.3 Against Scene Coordinates

An alternative approach to ensure the features become/remain useful for Scene Coordinate Regression, is to train the encoder by directly evaluating predicted scene coordinates against ground truth scene coordinates. The predicted scene coordinates are obtained using the scene-specific prediction heads, which are pre-trained with real data for each scene, but not updated during fine-tuning; only the shared encoder weights are updated. The same prediction head is used for real and synthetic images from the same scene. The loss function is the mean Euclidean distance between the predicted and ground truth scene coordinates.

In Figure 4.12 the training process is shown. In the separate setting, the real image is passed through the fine-tuned encoder, followed by the prediction head, such that the predicted scene coordinates can be compared to the ground truth. In the combined setting, both real and synthetic images from the same scene are passed through the fine-tuned encoder, followed by the prediction head. Here, the 3D distance loss for both real and synthetic images is combined into a single loss function.



(a) Separate encoder for synthetic images



(b) Combined encoder for real and synthetic images

Figure 4.12: Encoder fine-tuning against scene coordinates from scene-specific prediction heads.

Chapter 5

Experiments

This chapter describes the experiments conducted, presents their results, and discusses the findings individually. The goal is to evaluate the effectiveness of our proposed methods in adapting Scene Coordinate Regression for training on synthetic data. Section 5.1 focuses on supervised training using scene coordinates, while Section 5.2 covers the results of transfer learning for domain adaptation. These sections are analogous to the Method sections 4.2 and 4.3, respectively. Note that synthetic data generation does not require experiments and is therefore not covered in this chapter; the outputs have been presented in the Method chapter, Section 4.1.

5.1 Supervised Training using Scene Coordinates

This section covers the results of supervised training using scene coordinates, as described in Method, Section 4.2. The expected benefits of this approach are improved training efficiency and convergence compared to the reprojection loss. We evaluate localization performance with different loss configurations and maximum training iterations, followed by taking a closer look at the loss convergence during training. These configurations include the reprojection loss, the 3D distance loss, and a switching strategy that starts with the 3D distance loss and switches to the reprojection loss at a specified iteration.

Localization Results Comparison

In Table 5.1, the localization results for different loss functions are shown, specifically the median rotation error, median translation error, and average processing time. 30K iterations are the default setting, whereas lower numbers that reflect faster training times are evaluated to challenge the reprojection loss and demonstrate potential benefits of using 3D loss for faster training times.

Iterations	Loss Function	Rotation [deg]	Translation [cm]	Time [ms]
30K	Reprojection	0.2	16.4	58.4
	3D Distance	0.6	42.3	64.6
	Switch at 10K	0.6	55.4	61.9
	Switch at 5K	0.2	18.7	58.5
15K	Reprojection	0.2	18.7	54.9
	3D Distance	0.6	49.4	61.0
	Switch at 5K	0.3	26.3	60.3
	Switch at 1.5K	0.3	18.7	54.3
8K	Reprojection	0.3	22.9	59.4
	3D Distance	0.6	53.3	65.5
	Switch at 3K	1.3	118.4	72.9
4K	Reprojection	0.4	26.0	58.0
	3D Distance	0.7	61.0	64.7
	Switch at 1K	1.0	87.9	66.8

Table 5.1: Localization results for different loss functions (dataset: Pantheon)

The results show that the 3D distance loss works correctly but does not perform as well as the reprojection loss in the current configuration. Reprojection loss consistently outperforms 3D distance loss in terms of localization accuracy. The switching strategy provides a slight improvement over the 3D distance loss, but pure reprojection loss remains superior.

Convergence Comparison

In Table 5.2, the convergence of the different loss functions is compared over 15K iterations. The units for the reprojection loss are in pixels, and for the 3D distance loss in meters, indicated by [pix] and [m], respectively. Note that the reprojection loss runs on a *tanh* learning schedule with a dynamic threshold, so individual values are not directly comparable to the 3D distance loss.

Iteration	Loss Function							
	Reprojection		3D Distance		Switch at 5K		Switch at 1.5K	
0	36.0	[pix]	13.2	[m]	13.2	[m]	13.2	[m]
500	40.7		1.8		1.7		1.8	
1000	35.5		1.1		1.3		1.1	
1500	33.2		1.1		1.1		38.8	[pix]
2000	31.1		1.0		0.9		33.9	
2500	30.1		0.8		0.8		31.0	
3000	29.1		0.8		0.8		29.6	
3500	29.0		0.7		0.7		29.3	
4000	29.9		0.7		0.6		28.2	
4500	27.1		0.6		0.6		27.7	
5000	27.1		0.6		32.3	[pix]	27.3	
5500	26.8		0.6		37.3		26.6	
6000	25.8		0.6		31.5		25.7	
6500	25.3		0.6		29.3		25.2	
7000	25.0		0.5		28.1		24.7	
7500	24.3		0.5		26.9		24.1	
...	...		...		...		...	
12500	15.4		0.2		17.1		15.4	
13000	13.5		0.2		15.8		14.2	
13500	12.2		0.2		14.3		12.8	
14000	10.6		0.2		12.4		11.2	
14500	8.4		0.2		9.8		8.8	
15000	5.0	[pix]	0.2	[m]	5.7	[m]	5.2	[m]

Table 5.2: Convergence compared between loss functions at 15K iterations (dataset: Pantheon)

At the beginning, the 3D distance loss converges very fast, while the reprojection loss takes longer to converge and leads to more accurate results, as shown in the previous section. The switching strategies provide a better starting point for the reprojection loss, but the final loss is still lower for the reprojection loss compared to the switching strategies. In principle, the switching strategy should be beneficial to speed up training time while maintaining accuracy, but this is not the case here.

5.2 Transfer Learning for Domain Adaptation

This section covers the results of transfer learning with the goal to bridge the domain gap between real and synthetic data. It is divided into the same subsections as in Method, Section 4.3: feature similarity, feature contrasting, and scene coordinates.

Training Settings

For training the GLACE prediction head, the same settings are used in all following experiments. The Cambridge dataset configuration is used, since its scene size is similar to our models, and it is a good compromise between performance and training time. Particularly, the number of head blocks is set to 2, and the maximum number of iterations is set to 30,000. We also use reprojection loss, train on datasets with 1000+ images, and test on 100 independent images from the same dataset. However, we reduce the training buffer size from 16M to 4M to fit the model on our GPU memory, not affecting the results, as verified by testing the Cambridge dataset locally.

When it comes to fine-tuning the encoder, training is done on multiple datasets and validated on an additional independent dataset. Due to a limited quantity of datasets, the same dataset is used for validation and testing, but with independent samples. We validate a range of different settings, including the number of iterations, batch size / gradient accumulation, data augmentation, learning rate scheduler, and optimizer across datasets in order to determine the best configuration.

Original Results without Domain Gap

The results using the original GLACE network serve as a baseline for the following experiments. In particular, we need the results for the original GLACE network without domain gap, as well as with domain gap, to evaluate the performance of the model in the different scenarios. In Table 5.3, the localization results for the original GLACE network are shown for different datasets without domain gap, i.e. trained and tested on the same data domain. Moreover, the results are compared to a reference dataset used in the GLACE paper [34] to verify local performance.

Dataset	Description	Rotation [deg]	Translation [cm]	Time [ms]
Cambridge King's College [12]	Reference	0.31	20.4	61.1
Notre Dame	Real	1.24	259.8	67.3
	Real (closest 50%)	0.72	122.6	61.3
	Renders	0.73	133.3	60.1
Pantheon	Real	0.26	17.6	60.2
	Renders	0.17	14.3	54.7
Brandenburg Gate	Real	0.36	57.4	66.5
	Renders	0.31	50.2	57.3

Table 5.3: Original localization results with same data domain used for training and testing.

Compared to the reference dataset, the results are on the same order of magnitude, but slightly worse in most cases. In general, it can be observed that GLACE performs similarly well for real and rendered images when trained and tested on the same domain. The slightly better performance of the rendered images can be attributed to more consistent features and less noise in the data, compared to real images.

Looking at the different datasets, the results vary depending on the scene. The Pantheon dataset in particular shows very good results, very similar to the reference dataset, for both real and rendered images. Brandenburg Gate results are slightly worse, but still good, with a similar rotation error but a 3x higher translation error compared to the reference dataset. Notre Dame results are worse, with an acceptable rotation error around 1 degree (3x higher than the reference) but a 10x higher translation error.

These differences can be attributed to the scale of the scenes and the camera positions. Pantheon poses are relatively close together and have very diverse camera angles, which makes the scene easier to localize. In comparison, Brandenburg Gate poses are further apart. Notre Dame has the largest scale, poses are both further apart and more limited in perspective, which makes the scene the hardest to localize. Therefore, given the same settings, the results are not surprising and still relatively good considering the real-world scale of the landmarks.

When only using the closest 50% of images of the Notre Dame dataset, the median translation error is reduced by a factor of two. Effectively, this reduces the distance from the model and the variance in camera positions, proving that these two factors have a significant impact on the localization performance. Firstly, closer to the model, the patches used for feature extraction and predictions represent a smaller 3D space, which makes individual predictions more accurate. Secondly, a smaller variance in camera positions makes the parametrization of the position decoder more precise.

Original Results with Domain Gap

To evaluate the performance of the original GLACE network with a domain gap, the models trained before are now tested on the opposite data domain, i.e. networks trained on real images are tested on rendered images and vice versa. In Table 5.4, the new localization results for different datasets are shown.

Dataset	Train	Test	Rotation [deg]	Translation [cm]	Time [ms]
Notre Dame	Real	Renders	4.90	1393.3	100.4
	Renders	Real	35.50	3550.7	123.2
Pantheon	Real	Renders	6.53	478.3	90.2
	Renders	Real	16.68	1235.9	106.5
Brandenburg Gate	Real	Renders	39.95	4574.6	121.7
	Renders	Real	81.10	6309.2	122.6

Table 5.4: Original localization results with domain gap between data used for training and testing.

The results are significantly worse, indicating a large domain gap between real and rendered images. However, note that training on real images and testing on rendered images performs considerably better than the other way around, i.e. training on rendered images and testing on real images. This is likely due to the fact that real training images are more diverse and contain more noise, which makes the model more robust to the simpler rendered test images.

There are also differences between the datasets, with Notre Dame and Pantheon trained on real images and tested on rendered images showing better results than Brandenburg Gate. In these cases, the median rotation errors are acceptable, but the translation errors are very high. Brandenburg Gate and all configurations with training on renders and testing on real images show very high rotation errors, indicating that the model is not able to generalize to the real domain. Moreover, the high average processing times that are about 2x higher than without domain gap indicate that many RANSAC iterations are performed, which means that the scene coordinate predictions are inaccurate and contain many outliers.

5.2.1 Feature Similarity

The feature similarity approach attempts to fine-tune the encoder simply by minimizing the difference between real and fake features. The process and results are a simplified version of Feature Contrasting in the next subsection, equivalent to a weight of 1.0 on the first loss term and 0.0 on the other terms.

Separate Encoder

The training progression and results are shown in Figure 5.1-5.2 and 5.6 for the configuration with weights 1.0/0.0 (gray). When exclusively looking at the fake-real feature similarity, it shows a steady decrease in the training loss. The validation loss on the other dataset also decreases, indicating that the updated model generalizes well to unseen data. However, tracking the difference between far-apart fake image features shows an increase, implying that the features become less spatially distinct.

In terms of localization results, the performance is worse than with the pre-trained encoder. This suggests that the encoder loses information during fine-tuning, making fake features more similar and less useful for localization. Therefore, the next approach adds a loss term on the difference between far-apart fake image features, in an attempt to promote spatial distinctiveness and prevent the loss of information during fine-tuning.

Combined Encoder

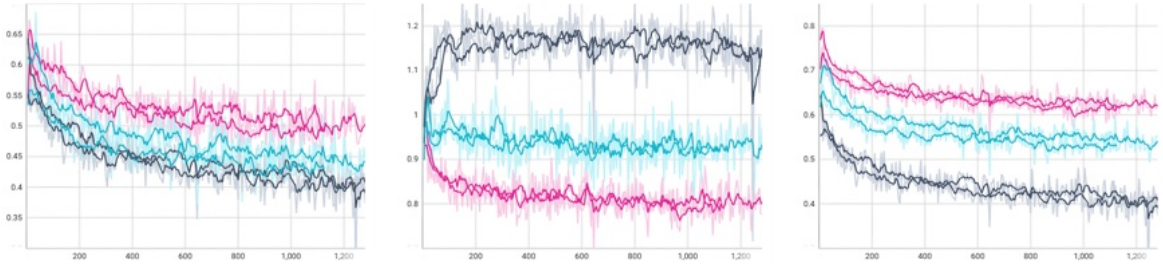
For the combined approach, as expected, the results are similar. In addition, here it is observable that not only the fake features become less spatially distinct, but also the real features become worse compared to the initial features. Therefore, the next approach adds a loss term for anchoring the real features to the pre-trained encoder, in an attempt to limit deviation of the real features. Note that the separate encoder does not need anchoring, as the real images use the pre-trained encoder directly.

5.2.2 Feature Contrasting

This approach implements additional loss terms to fine-tune the encoder, aiming to balance the similarity between real and fake features with the spatial distinctiveness of the fake features. Different weights for the loss terms are evaluated to determine the performance compared to the original network.

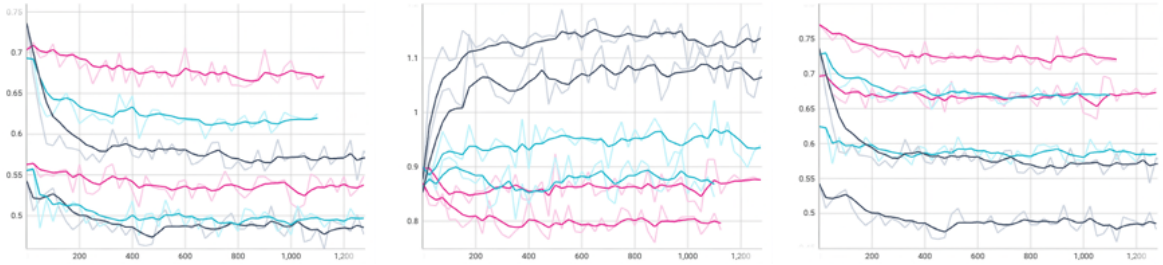
Separate Encoder

In Figure 5.1 and 5.2, the training and validation loss terms are shown as a function of training progress. The colors represent different weights for the loss terms F-Ri (similarity between fake features and real initial features) and F1-F2 (difference between far-apart fake image features), see Table 5.5. Graphs of the same color represent different datasets combinations, where two datasets are used for training and the third for validation each. Note that the loss terms are defined such that lower values always reflect better performance.



(a) Similarity F-Ri (b) Difference F1-F2 (c) Total

Figure 5.1: Training loss terms for separate feature contrast training.



(a) Similarity F-Ri (b) Difference F1-F2 (c) Total

Figure 5.2: Validation loss terms for separate feature contrast training.

Color	F-Ri	F1-F2
Pink	0.6	0.4
Blue	0.8	0.2
Gray	1.0	0.0

Table 5.5: Weights for loss terms in separate feature contrast training.

Across all loss terms and datasets, it can be observed that the training and validation loss terms decrease together, implying that the training generalizes well to other datasets. The absolute values of the loss terms differ between datasets, but the relative trends are similar. This indicates that the loss terms have a similar effect on different datasets, despite the high variance in the absolute values, which is likely due to the diversity in 3D model appearance.

When analyzing the different weight configurations, 0.6/0.4 (pink) only slightly improves fake-real similarity by about 0.1, but is able to enhance the distinctiveness of the fake features by about 0.2. Yet, these changes are only partially reflected in the validation loss. The 0.8/0.2 (blue) configuration shows an improvement in fake-real similarity by about 0.2, while the distinctiveness remains roughly constant, which is also reflected in the validation loss. The 1.0/0.0 (gray) configuration improves the fake-real similarity slightly more, but reduces the distinctiveness of the fake features by about 0.1-0.2. Here, the validation loss significantly changes for both loss terms. While this is beneficial for the fake-real similarity, it comes at the cost of the distinctiveness of the fake features.

Overall, compared to the previous approach, the addition of the F1-F2 loss term leads to a better balance between the similarity of real and fake features and the distinctiveness of the fake features. However, there is a conflict between the two loss terms, as it is not possible to minimize both at the same time; there is always a compromise between the two.

In Table 5.6, the preliminary localization results are shown with different fine-tuned encoders, obtained from the above training, using the original prediction head. In Table 5.7 and 5.8, the localization results for the configuration 0.8/0.2 (blue) with a new prediction head trained using the fine-tuned encoder are shown for fake and real data, respectively. In each case, the results are compared to the original results on the same data domains.

Encoder	**Head**	**Notre Dame**		**Pantheon**		**Brandenburg Gate**	
		[deg]	**[cm]**	**[deg]**	**[cm]**	**[deg]**	**[cm]**
Fine-tuned 0.6 / 0.4	Original Real	24.9	4157.5	24.2	1669.4	102.9	6000.8
Fine-tuned 0.8 / 0.2	Original Real	21.8	2774.4	12.5	1022.4	63.6	6521.9
Fine-tuned 1.0 / 0.0	Original Real	45.7	7087.4	22.6	1523.0	112.8	7279.5
Pre-trained	Original Real	4.9	1393.3	6.53	478.3	39.95	4574.6

Table 5.6: Localization results on fake data with fine-tuned encoder and original head for different weights

Despite the slight improvements in the loss terms, the localization results for all configurations are worse than with the pre-trained encoder. This is surprising, as the training and validation loss terms seemed to generalize well to other datasets. In theory, assuming the loss terms are representative of the localization performance, the improvement in the loss terms should lead to better localization results, but this is not the case here. Nevertheless, as expected, the 0.8/0.2 configuration performs best, demonstrating that its balance between feature similarity and distinctiveness yields some benefit.

Encoder	Head	Notre Dame		Pantheon		Brandenburg Gate	
		[deg]	[cm]	[deg]	[cm]	[deg]	[cm]
Fine-tuned	New	1.10	241.3	0.24	20.6	0.30	42.3
Pre-trained	Original Renders	0.73	133.3	0.17	14.3	0.31	50.2

Table 5.7: Localization results on fake data with fine-tuned encoder and updated head for 0.8/0.2

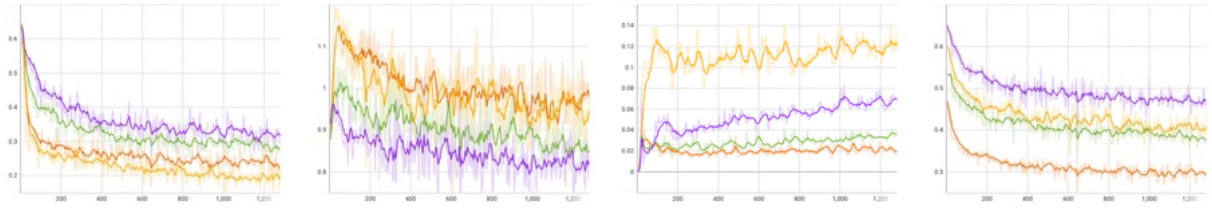
Encoder	Head	Notre Dame		Pantheon		Brandenburg Gate	
		[deg]	[cm]	[deg]	[cm]	[deg]	[cm]
Pre-trained	New	29.72	4290.5	23.39	1564.9	90.65	7074.5
Pre-trained	Original Renders	35.50	3550.7	16.68	1235.9	81.10	6309.2

Table 5.8: Localization results on real data with pre-trained encoder and updated head for 0.8/0.2

After training the new prediction head on the fine-tuned encoder with fake data, the results for fake data are better, but still slightly worse than the baseline results with identical data domains (training and testing on rendered images). The results for real data are comparable to the baseline results with identical domain gap (training on renders and testing on real images); some are slightly better, others slightly worse. Overall, there is no clear improvement in the localization results, despite the improved loss terms.

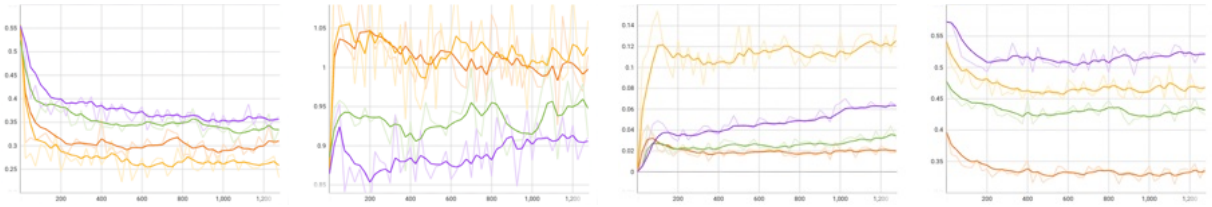
Combined Encoder

In Figure 5.3 and 5.4, the training and validation loss terms are shown as a function of training progress. The colors represent different weights for the loss terms F-R (similarity between fake and real features), F1-F2 (difference between far-apart fake image features), and R-Ri (anchoring between real and real initial features), see Table 5.9. Note that the anchoring term starts at zero by definition, so a slight increase is expected.



(a) Similarity F-R (b) Difference F1-F2 (c) Anchoring R-Ri (d) Total

Figure 5.3: Training loss terms for combined feature contrast training



(a) Similarity F-R (b) Difference F1-F2 (c) Anchoring R-Ri (d) Total

Figure 5.4: Validation loss terms for combined feature contrast training

Color	F-R	F1-F2	R-Ri
Orange	0.4	0.2	0.4
Green	0.4	0.3	0.3
Purple	0.4	0.4	0.2
Yellow	0.5	0.3	0.2

Table 5.9: Weights for loss terms in combined feature contrast training

Similar to the separate approach, the training and validation loss terms decrease together, suggesting that the training generalizes well to other datasets. The fake-real similarity improves significantly for all configurations, by about 0.3-0.4. The other two loss terms, F1-F2 and R-Ri, appear to be in conflict, where one improves at the cost of the other, or both worsen in favor of the fake-real similarity (yellow configuration: 0.5/0.3/0.2). In other words, anchoring the real features to the pre-trained encoder comes at the cost of the distinctiveness of the fake features, and vice versa. Only the green (04/0.3/0.3) and purple (0.4/0.4/0.2) configurations show a slight improvement in the distinctiveness of the fake features.

The conflict between the loss terms demonstrates that it is not possible to anchor real features to the pre-trained encoder and still perform well on fake features. This implies that anchoring is not very useful to improve the encoder, however, some loss term is needed to keep the features useful for localization. When ignoring the anchoring term, the conflict between the first two loss terms is also visible.

In Table 5.10 and 5.11, the localization results for the configuration 0.4/0.3/0.3 (green) with a new prediction head trained using the fine-tuned encoder are shown for fake and real data, respectively. Both results are compared to the original results on the same data domains.

Encoder	**Head**	**Brandenburg Gate**	
		[deg]	**[cm]**
Fine-tuned	New	0.29	51.6
Pre-trained	Original Renders	0.31	50.2

Table 5.10: Localization results on fake data with fine-tuned encoder and updated head for 0.4/0.3/0.3

Encoder	**Head**	**Brandenburg Gate**	
		[deg]	**[cm]**
Fine-tuned	New	64.97	6240.7
Pre-trained	Original Renders	81.10	6309.2

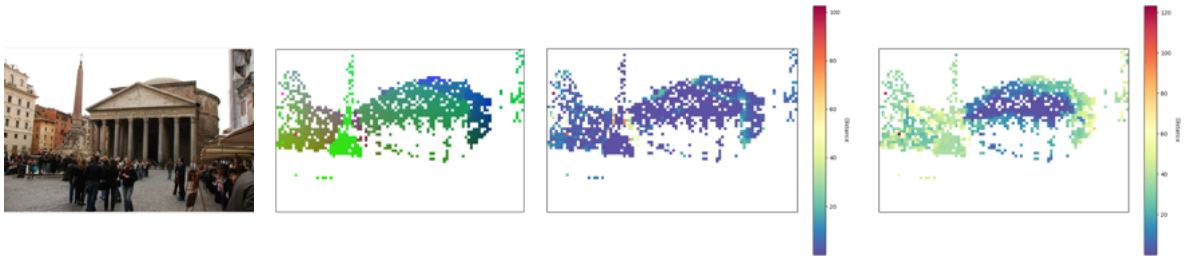
Table 5.11: Localization results on real data with pre-trained encoder and updated head for 0.4/0.3/0.3

The localization results show a minimal improvement, but no specific conclusion can be drawn from this. Testing on fake data is still quite accurate with the new encoder, but testing on real data is also similarly bad as before. Moreover, even with this slight improvement, the loss graphs indicate that very little further improvement can be expected.

5.2.3 Against Scene Coordinates

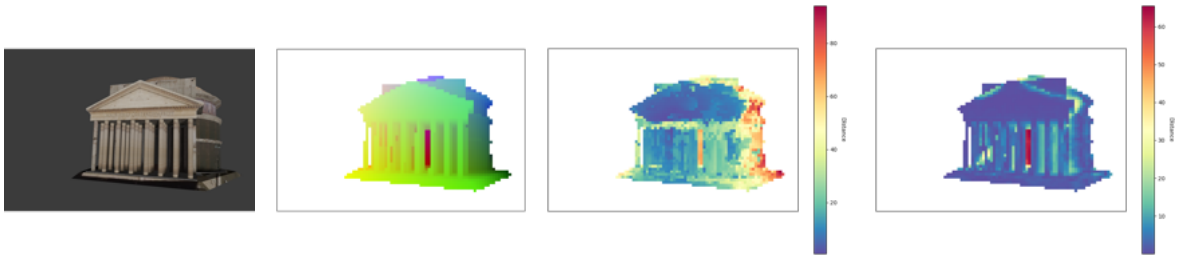
This approach uses a completely different evaluation metric, the distance between scene coordinates, which is a good indicator for localization performance. The idea is that by being able to better assess the fine-tuned encoder during training, one can optimize more effectively and have a better feedback mechanism than with the previous loss terms on features.

The initial evaluation is shown in Figure 5.5 and Figure 5.6, which compares predictions to ground truth scene coordinates. For (a) real and fake images, respectively, (b) ground truth scene coordinates are visualized, as well as the prediction errors using the pre-trained encoder and the original head trained on (c) real or (d) rendered images.



(a) Real image (b) MVS ground truth (c) Prediction error real head (d) Prediction error fake head

Figure 5.5: Scene coordinate evaluation of real image using pre-trained encoder (dataset: Pantheon)



(a) Fake image (b) CAD ground truth (c) Prediction error real head (d) Prediction error fake head

Figure 5.6: Scene coordinate evaluation of fake image using pre-trained encoder (dataset: Pantheon)

The visualization shows that the predictions are good without a domain gap, i.e. in Figure 5.5(c) and Figure 5.6(d), but worse with a domain gap, i.e. in Figure 5.5(d) and Figure 5.6(c). This difference is expected, but it is interesting to point out that with a domain gap, some coordinates are very close, while others are very far off, particularly at the edges of the structure.

The results for the entire dataset, containing the image shown above, are summarized in Table 5.12. It also includes a comparison to the results obtained with the fine-tuned encoder from the previous separate feature contrast training (0.8/0.2 weights) on fake images. The average distance is computed for all images and the median over the test dataset is displayed.

Encoder	Head	Train=Test Dataset	Images	Median Error [m]
Pre-trained	Real	No	Real	3.9
			Fake	16.8
	Fake		Real	12.9
			Fake	4.7
Fine-tuned	Real	No	Fake	17.9
		Yes		17.0

Table 5.12: Scene coordinate evaluation results (real head training, dataset: Pantheon)

Similar to the localization results with the fine-tuned encoder in the previous subsection, the scene coordinate results are slightly worse than with the pre-trained encoder. This demonstrates that scene coordinate evaluation is a good indicator for localization performance, as the results are consistent with the localization results.

When fine-tuning an encoder purely based on a scene-coordinate loss, the loss progression is random even over thousands of iterations, not resulting in any consistent improvement. Despite verification that the loss is computed correctly, flows through from the head to the encoder, and the weights are updated accordingly, it appears that the weight updates are rather random. This implies that no specific direction is learned to improve the scene coordinates. Gradient clipping and other techniques have been implemented to stabilize the training and avoid exploding gradients, but this still does not lead to any consistent improvement.

For further analysis, a simplified approach has been tested where the same dataset was used for training and testing, with independent samples. While not being able to generalize to other datasets, this could serve as a proof-of-concept for the approach. However, the results are similarly random, not leading to any consistent improvement.

Chapter 6

Discussion & Conclusion

In this chapter, we provide a general discussion of the results obtained in the previous chapters, interpreting their findings, discussing their implications, and suggesting potential improvements. We also analyze several limitations and make recommendations for future research.

6.1 Key Findings and Interpretations

6.1.1 Supervised Training using Scene Coordinates

Supervised training using scene coordinates as privileged information shows promise for enhancing visual localization training performance. This method is computationally efficient, requiring only 3 values per patch compared to 512 features. While buffer creation time increases slightly, training time decreases as reprojection computations are not required. Experiments demonstrate faster convergence but no improvement in localization performance. Even when switching from 3D distance loss to reprojection loss, the results are slightly worse than reprojection loss alone.

Reprojection loss outperforms 3D distance loss likely due to its more sophisticated implementation. Key features include dynamic thresholding to ignore large reprojection errors, filtering of invalid depth values, and considering only visible points in the image. These constraints effectively ignore outliers and difficult-to-predict points, which are taken care of by the pose solver in the next step. Despite longer convergence times, reprojection loss achieves state-of-the-art localization accuracy.

In contrast, 3D distance loss is a simpler function that lacks these advanced constraints. It uses the average distance to valid ground truth scene coordinates, allowing outlier predictions to have an effect. While median distance could theoretically perform better, its non-differentiable nature makes it unsuitable for backpropagation. To improve supervised training with scene coordinates, we suggest implementing dynamic thresholding for large distances and applying camera-based constraints using the ground truth pose and intrinsics to filter predicted scene coordinates based on depth and visibility. These enhancements could potentially surpass reprojection loss in training speed while maintaining accuracy.

In conclusion, while reprojection loss remains a powerful, optimized tool for state-of-the-art localization accuracy, supervised training using scene coordinates shows potential. With the suggested improvements, it could become a valuable addition to the training pipeline, offering faster convergence without compromising accuracy.

6.1.2 Transfer Learning for Domain Adaptation

Transfer learning was explored as a method to adapt the pre-trained encoder to synthetic data, bridging the domain gap without training from scratch. Various approaches were tested, including fine-tuning separate encoders for synthetic and combined real/synthetic data, using both feature and scene coordinate loss functions.

Experiments revealed that fine-tuning the pre-trained encoder did not consistently improve localization performance. While feature-based loss functions showed improvements in individual terms and overall losses, this did not translate to better scene coordinate regression performance. Even sophisticated feature loss functions incorporating similarity between real and synthetic images, differences between far-away images, and anchoring to the pre-trained encoder for real features failed to enhance localization performance. There was a trade-off between different loss terms, limiting potential improvements. The conflict between anchoring and domain invariance suggests that the encoder must learn entirely new feature representations more suited to the task, which is difficult to achieve with feature fine-tuning.

Alternative experiments using scene coordinate loss, which directly represents localization performance, also failed to yield consistent improvements. Training across multiple scenes and even within single scenes did not reveal generalizable patterns, suggesting a lack of useful information for encoder updates. The absence of small improvements, even with shorter training times, indicates that the current method is ineffective for domain adaptation to synthetic data.

Potential enhancements to the fine-tuning process include improving the scene coordinate loss function, similar to what has been suggested for supervised training, and implementing advanced training principles introduced in ACE [5], particularly gradient decorrelation. This principle, a key innovation in ACE for faster training times, combines random patches from across the training dataset into new images, effectively decorrelating their gradients to provide a better training signal and more stable convergence.

However, given the variety of fine-tuning approaches tested, it appears that fine-tuning the pre-trained encoder alone is insufficient for domain adaptation to synthetic data. The experiments indicated that the encoder needs to learn entirely different feature representations suited for the new task. While adjusting other settings, such as adding encoder layers or modifying RANSAC parameters, might help bridge the domain gap and improve localization performance despite suboptimal scene coordinate predictions, finding generalizable patterns across multiple datasets may ultimately require a more comprehensive approach.

These improvements and adjustments may likely need more data and longer training times. Achieving effective domain adaptation might only be possible with end-to-end training over significantly longer periods, allowing the model to learn task-specific features that generalize across both synthetic and real domains.

6.1.3 End-to-End Training

End-to-end training optimizes the entire network simultaneously, including the encoder and prediction heads for multiple datasets, to generate a new pre-trained encoder. This approach, similar to ACE's original pre-training process (not publicly released) on 100 datasets in parallel using 100 prediction heads, allows both the encoder and heads to be updated during training. While implementing and evaluating a from-scratch pre-training pipeline is beyond this work's scope due to time and resource constraints, we experimented with an end-to-end version of fine-tuning. However, this limited approach, updating the head for a single dataset, did not yield noticeable improvements in scene coordinate loss or localization performance

6.2 Limitations

Several limitations should be considered when interpreting the results:

- Registration accuracy: Slight misalignments between CAD and MVS models due to model inaccuracies are passed onto the images and ground truth poses when comparing across the domain gap. This has a small effect on reported localization accuracy, and a larger impact on fine-tuning when comparing features between real and synthetic images. Fine-tuning via a coordinate loss that implicitly bridges the domain gap can avoid this issue by comparing predicted real and synthetic scene coordinates separately to their respective MVS and CAD ground truth coordinates.
- CAD model quality: Varying complexity and appearance of CAD models impact results. Experiments focused on textured models, but raw models would likely yield worse results and might require separate fine-tuning.
- Dataset availability: Limited high-quality CAD datasets with corresponding real-world images restrict result generalizability. Experiments were conducted only on outdoor landmarks, which may not fully reflect indoor AR application characteristics. While results are likely transferable to indoor datasets, as is the case with other localization methods, this remains untested.

6.3 Future Directions

Based on our findings, we propose several directions for future research to enhance the training process and achieve better results:

- End-to-end training: Implement a comprehensive end-to-end approach from scratch across multiple datasets to train an encoder that can learn new, task-specific features more suited to both synthetic and real data.
- Simultaneous real and synthetic training: Process real and synthetic data concurrently, utilizing a shared head for each dataset, employing scene coordinate loss that implicitly bridges the domain gap.

- Enhanced scene coordinate loss: Improve the scene coordinate loss function by incorporating dynamic thresholding and camera constraints, similar to the optimizations found in reprojection loss. This could lead to better training performance and more accurate localization.
- Gradient decorrelation: Implement advanced training principles introduced in ACE, particularly gradient decorrelation. This technique, which combines random patches from across the training dataset into new images, could provide better training signals and more stable convergence.
- Other Scene Coordinate Regression settings: Experiment with additional encoder layers to capture added complexity and/or different RANSAC options (e.g., number of iterations, inlier threshold) to optimize localization performance despite worse coordinate predictions.
- Dataset expansion: Broaden the range of datasets used in training, including more varied scenes and particularly indoor environments. This will improve the generalizability of the results and better reflect diverse real-world applications, including AR scenarios.
- Extended training periods: Once consistent relative improvements are demonstrated, explore longer training times to further reduce the domain gap and enhance overall performance.

By pursuing these research directions, future work can build on the foundations of this study, potentially leading to advancements in visual localization using synthetic data for real-world applications.

6.4 Concluding Remarks

This report has investigated the feasibility of adapting Scene Coordinate Regression for training on synthetic data, specifically CAD models, to enable localization in real-world images. We developed a synthetic data generation pipeline and explored various training strategies. Supervised training using scene coordinates showed faster convergence but did not improve localization performance in its current form. Our experiments with transfer learning for domain adaptation, despite testing multiple approaches, did not yield consistent improvements. This suggests that bridging the gap between synthetic and real data requires more comprehensive methods. The limited exploration of end-to-end training provided insights into potential future directions but did not result in significant improvements.

While our current approaches did not surpass existing methods, this work has identified several promising directions for future research. Our findings highlight both the challenges and opportunities in using CAD models for real-world localization tasks. The results underscore the complexity of the problem and the need for further investigation into leveraging synthetic data for visual localization.

Bibliography

[1] Cloudcompare (version 2.13). GPL software, 2024. Retrieved from http://www.cloudcompare.org/.

[2] Dror Aiger, Andre Araujo, and Simon Lynen. Yes, we CANN: Constrained Approximate Nearest Neighbors for local feature-based visual localization. In *ICCV*, 2023.

[3] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2024.

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

[6] Eric Brachmann and Carsten Rother. Learning less is more - 6D camera localization via 3D surface regression. In *CVPR*, 2018.

[7] Eric Brachmann and Carsten Rother. Visual camera re-localization from RGB and RGB-D images using DSAC. *TPAMI*, 2021.

[8] Bingyi Cao, Andre Araujo, and Jack Sim. Unifying deep local and global features for image search, 2020.

[9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, 2017.

[10] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[11] Jared Heinly, Johannes Lutz Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization, 2016.

[13] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023.

[14] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. IEEE, 1999.

[15] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. MeshLoc: Mesh-Based Visual Localization. In *ECCV*, 2022.

[16] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. Visual Localization using Imperfect 3D Models from the Internet. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

[17] Jerome Revaud, Jon Almazan, Rafael Sampaio de Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019.

[18] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

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

[20] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[21] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L. Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR: Benchmarking Localization and Mapping for Augmented Reality. In *ECCV*, 2022.

[22] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Victor 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.

[23] 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*, 2017.

[24] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions, 2018.

[25] Torsten Sattler, Tobias Weyand, B. Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *British Machine Vision Conference*, 2012.

[26] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[28] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[29] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021.

[30] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018.

[31] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: the new data in multimedia research. *Commun. ACM*, 59(2):64–73, January 2016.

[32] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape, 2016.

[34] Fangjinhua Wang, Xudong Jiang, Silvano Galliani, Christoph Vogel, and Marc Pollefeys. Glace: Global local accelerated coordinate encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024.

[35] Guangming Wang, Yu Zheng, Yanfeng Guo, Zhe Liu, Yixiang Zhu, Wolfram Burgard, and Hesheng Wang. End-to-end 2d-3d registration between image and lidar point cloud for vehicle localization, 2023.

[36] Qunjie Zhou, Sérgio Agostinho, Aljosa Osep, and Laura Leal-Taixé. Is geometry enough for matching in visual localization?, 2022.

[37] Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixe. Patch2pix: Epipolar-guided pixel-level correspondences. In *CVPR*, 2021.

[38] Sijie Zhu, Linjie Yang, Chen Chen, Mubarak Shah, Xiaohui Shen, and Heng Wang. R2 former: Unified retrieval and reranking transformer for place recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.