UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

# A deep learning-based pipeline for visual geolocation in the urban environment

*Author:*
Volodymyr TSAPIV

*Supervisor:*
Ph.D. Taras FIRMAN

*A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science*

*in the*

Department of Computer Sciences and Information Technologies
Faculty of Applied Sciences


Lviv 2023

# Declaration of Authorship

I, Volodymyr TSAPIV, declare that this thesis titled, "A deep learning-based pipeline for visual geolocation in the urban environment" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

*"The whole point of taking pictures is so that you don't have to explain things with words."*

Elliott Erwitt

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

**A deep learning-based pipeline for visual geolocation in the urban environment**

by Volodymyr TSAPIV

# *Abstract*

The precise identification of location in urban areas is a challenging problem for Global Navigation Satellite Systems (GNSS), such as GPS, because of obstacles that include signal blockage, multipath interference, and urban canyons, among other factors. This thesis proposes a structure-based visual localization pipeline that uses a combination of Deep Neural Networks (DNNs) and traditional computer vision algorithms to perform accurate localization by an image. Additionally, we provide a collection of helpful tools for constructing a reference database for visual localization that can be used with any city found on Google Maps. The proposed method was evaluated on established visual localization benchmarks and produced competitive outcomes.

# *Acknowledgements*

I am grateful to my supervisor, Taras Firman, for his guidance, support, and feedback throughout my research. I appreciate my family's unwavering love and support, especially my younger siblings, who provided invaluable advice. I am grateful to the faculty, ML lab, and fellow students at Ukrainian Catholic University for their contributions to my academic and personal growth. I also thank the benefactors who provided financial support for my studies, enabling me to focus on my research without financial stress.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **GNSS** | **G**lobal **N**avigation **S**atellite **S**ystem |
| **GPS** | **G**lobal **P**ositioning **S**ystem |
| **EXIF** | **EX**changeable **I**mage **F**ile |
| **DNN** | **D**eep **N**eural **N**etwork |
| **PnP** | **P**erspective-**n**-**P**oint |
| **DOF** | **D**egree **O**f **F**reedom |
| **FOV** | **F**ield **O**f **V**iew |
| **DLT** | **D**irect **L**inear **T**ransformation |
| **RANSAC** | **RAN**dom **SA**mple **C**onsensus |
| **SfM** | **S**tructure **f**rom **M**otion |
| **BA** | **B**undle **A**djustment |
| **LSTM** | **L**ong **S**hort-**T**erm **M**emory |
| **CNN** | **C**onvolutional **N**eural **N**etwork |
| **GNN** | **G**raph **N**eural **N**etwork |
| **FC** | **F**ully **C**onnected (Layer) |
| **MVS** | **M**ulti-**V**iew **S**tereo |
| **AP** | **A**verage **P**recision |
| **MAP** | **M**ean **A**verage **P**recision |
| **PCA** | **P**rincipal **C**omponent **A**nalysis |

# Chapter 1

# Introduction

The capability to localize accurately is essential for many applications, such as autonomous cars, delivery robots, and augmented and mixed reality. GPS, among other GNSSs, such as Galileo, QZSS, or BeiDou, is the most adopted tool for accurate localization. The working principle of GPS involves using signals from at least three satellites to determine the receiver's location on the Earth's surface through the triangulation procedure. Many factors negatively influence GPS's accuracy, namely atmospheric conditions, signal interference, and the insufficient number or bad positioning of satellites in the sky. Even though GPS has been shown to achieve high levels of accuracy, with the typical horizontal error of 2–5 meters and vertical error of 5–10 meters in open-spaced environments, its accuracy is significantly impacted by urban environments due to the presence of tall buildings, narrow streets, and other obstructions that can block signals or cause multipath interference. The experimental data on GPS localization errors in urban environments vary significantly depending on particular research. Still, even the most optimistic estimates state horizontal error to be around 7–13 meters [34].

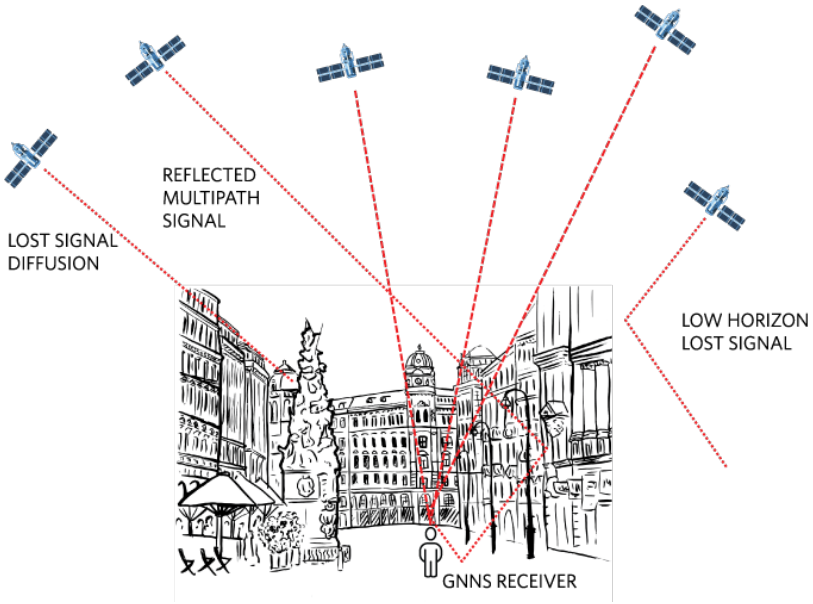



FIGURE 1.1: Factors of GPS localization error

One of the possible solutions is to use visual localization pipelines that can leverage urban environment complexity to identify and track unique features and landmarks that can be used to determine location accurately. Visual localization has the potential to provide both outdoor and indoor localization that's accurate up to a few centimeters and degrees.

In the case of (semi-)autonomous cars, the visual localization pipeline supplied with the rich visual information provided by built-in cameras all around the vehicle can help with navigation within the narrow streets in the city center. In AR applications, the visual localization pipeline determines the user's position and orientation relative to the real-world environment. This information is then used to overlay digital content onto the real-world scene, creating an immersive AR experience.

In this work, we explore the most prominent approaches to visual localization and evaluate their pros and cons to build the most suitable city-scale visual localization pipeline. The main objective of this solution is to robustly and accurately estimate camera position based on a single image and camera parameters that are usually automatically attached to the image through EXIF. The proposed pipeline is designed to enable flexibility and extensibility so that the growth of the reference image database would not require a time- and cost-consuming retraining process. In addition, we provide handy tools for collecting and processing geotagged datasets for visual localization based on Google Street View API.

# Chapter 2

# Technical background

## 2.1 Pinhole camera model

The pinhole camera model explains how the location of a point in three-dimensional space relates mathematically to its projection onto the image plane of a pinhole camera with no lenses and a single-point aperture.

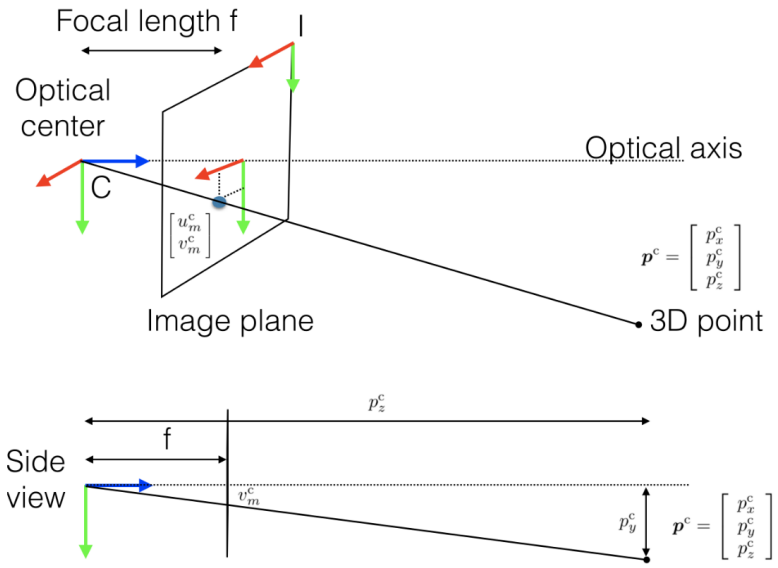



FIGURE 2.1: Coordinate system of pinhole camera model

The position $(u_m^c, v_m^c)$ of the projection of $p^c$ on the camera plane can be derived from similar triangles visible from the side view figure:

$$u_m^c = f\frac{p_x^c}{p_z^c} \tag{2.1}$$

$$v_m^c = f\frac{p_y^c}{p_z^c} \tag{2.2}$$

where $f$ is the focal length (in meters), and the subscript $m$ denotes that $(u_m^c, v_m^c)$ are still expressed in meters. But the image coordinate frame typically has its origin in the top-left corner of the image (see frame "I" in 2.1). Therefore, we can convert $(u_m^c, v_m^c)$ to pixels and change the reference frame as follows:

$$u^I = s_x u_m^c + o_x \tag{2.3}$$

$$v^I = s_y v_m^c + o_y \tag{2.4}$$

For convenience, the relation above can be written, by moving $p_z^c$ to the left-hand-side and using a matrix notation as:

$$p_z^c \begin{bmatrix} u^{\mathrm{I}} \\ v^{\mathrm{I}} \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{bmatrix}}^{K} \begin{bmatrix} p_x^{\mathrm{c}} \\ p_y^{\mathrm{c}} \\ p_z^{\mathrm{c}} \end{bmatrix} \tag{2.5}$$

where:

- $(o_x, o_y)$ is called the principal point;
- $s_x f$ is the focal length in horizontal pixels ($s_x$ number of horizontal pixels per meter);
- $s_y f$ is the focal length in vertical pixels ($s_y$ number of vertical pixels per meter);
- $s_\theta f$ is called the skew of the pixel (zero in most cases);
- $K$ is called the intrinsic (calibration) matrix.

There are a couple of ways to obtain an intrinsic matrix:

- calibrate camera using one of many methods;
- calculate $K$ based on the camera sensor information.

The first method is preferred whenever possible, but in many cases, one has no access to a camera that took the photo. Still, a relatively good approximation of $K$ can be obtained using camera sensor information:

- $o_x = w/2$;
- $o_y = h/2$;
- $s_\theta f = 0$;
- $s_x f = \frac{w}{2tan(FoV_x/2)}$;
- $s_y f = \frac{h}{2tan(FoV_y/2)}$.

When the point coordinates are given with respect to an external world frame $w$, i.e., $p^w$, it is straightforward to adapt 2.5 as:

$$p_z^{\mathrm{c}} \begin{bmatrix} u^{\mathrm{I}} \\ v^{\mathrm{I}} \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{bmatrix}}^{\text{intrinsic calibration}} \overbrace{\begin{bmatrix} R_{\mathrm{w}}^{\mathrm{c}} & t_{\mathrm{w}}^{\mathrm{c}} \end{bmatrix}}^{\text{extrinsic calibration}} \tilde{p}^{\mathrm{w}} \tag{2.6}$$

where:

- $\begin{bmatrix} R_{\mathrm{w}}^{\mathrm{c}} & t_{\mathrm{w}}^{\mathrm{c}} \end{bmatrix}$ is called the extrinsic (calibration) matrix;
- $R_{\mathrm{w}}^{\mathrm{c}}$ 3x3 rotation matrix;
- $t_{\mathrm{w}}^{\mathrm{c}}$ 3x1 translation vector in meters;

- $\tilde{p}^{w}$ is $p^{w}$ rewritten in homogeneous coordinates.

An important thing to note is that $t_{w}^{c}$ does not represent the camera's position in world coordinates, but rather describes the translation of the point in world coordinate system to camera coordinate system, accounting for rotation $R_{w}^{c}$ which is applied first. The following formulas define the relation between translation $t_{w}^{c}$ and camera position $c_{w}^{c}$:

$$t_{w}^{c} = -R_{w}^{c} c_{w}^{c} \tag{2.7}$$

$$c_{w}^{c} = -(R_{w}^{c})^{T} t_{w}^{c} \tag{2.8}$$

## 2.2 Triangulation

Triangulation is the process of finding the position of a point in space given its position in two or more images taken with cameras with known calibration and pose. This can be achieved by finding the intersection of at least two known rays. Those rays are defined by camera matrices and 2D projections of the common 3D point in each camera. In the absence of noise, this problem is trivial. Since it is not obvious how to estimate the 3D position of the point from two skew rays, different methods have been proposed.

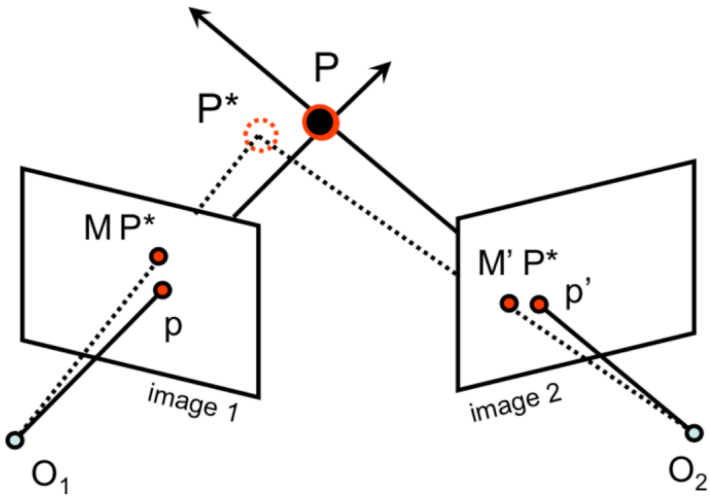



FIGURE 2.2: Noisy data triangulation

Mainly they can be classified into three types:

- midpoint methods [6, 7, 55] that find the (weighted) midpoint of the common perpendicular between the two rays;
- linear least squares methods [21];
- optimal methods that "minimally" correct the two rays to make them intersect [21, 25].

Note that all these three types of methods produce solutions that minimize some cost function; the (weighted) midpoint minimizes the (weighted) sum of squared distances to each ray, linear least squares methods minimize the algebraic errors,

and optimal methods minimize a cost function based on either image reprojection errors or angular reprojection errors. The most common cost functions are the L1 norm (sum of magnitude), L2 norm (sum of squares), and L∞ norm (maximum) of the reprojection errors [28].

## 2.3 PnP problem

Proposed by Fischler in the early 1980s, the Perspective-n-Point (PnP) problem had the goal of estimating the position and orientation of a calibrated camera from known 3D-to-2D point correspondences between a 3D model and their image projection [20]. Essentially this means the estimation of 3D rotation $R_w^c$ (roll, pitch, and yaw) and 3D translation $t_w^c$ that sums up to 6 DOF.

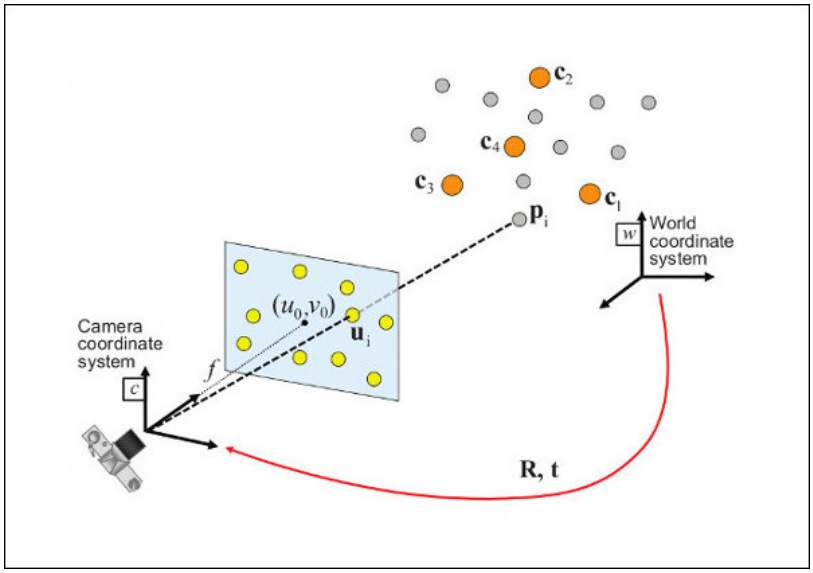



FIGURE 2.3: Camera pose estimation with PnP

There are several assumptions under which the problem is solved:

- camera intrinsic properties are already known (for most approaches);
- we cannot choose coplanar point correspondences;

Throughout all these years, many approaches and their variations have been proposed. Those could be divided into the following groups:

- iterative methods formulate the PnP problem into a non-linear least-squares problem [32] and then solve it using iterative optimization methods, i.e., Gauss–Newton and Levenberg–Marquardt method;
- non-iterative methods, the traditional methods apply linear operations to obtain solutions, i.e., the DLT, EPnP, RPnP, OPnP. Recently, the PnPf [37, 60] and PnPfr [36] methods were proposed to solve the pose estimation problem in the case of an uncalibrated camera. However, the accuracy of these methods is usually lower than the classical PnP method.

More or less, all the PnP approaches are prone to errors if there are considerable amounts of outliers in the set of point correspondences. In order to enhance the solution in terms of outlier resilience without making the underlying algorithms

more complex, the RANSAC procedure is to be incorporated. Recognizing this, we should seek not only the most accurate method but enough computationally efficient one since the underlying PnP solver is executed many times following the RANSAC procedure.

# Chapter 3

# Related Works

The estimation of camera pose using images by means of visual localization pipeline is associated with many challenges, namely:

- changes in illumination;
- dynamic scenes with moving objects that might not have been present;
- drastic changes in the scene's appearance;
- occlusion of the scene by an object or person;
- significant viewpoint changes.

Over the years, to address these challenges, the researchers' community proposed many approaches to visual localization. By ignoring minor differences and implementation details, most of the methods fall under one of the following categories:

- structure-based;
- structure-based with image retrieval;
- scene point regression;
- absolute pose regression;
- pose interpolation;
- relative pose estimation.

The fact that the single problem has several solutions suggests that each method has its pros and cons and should be applied after careful consideration of the particular problem.

## 3.1 Structure-based approach

The structure-based approach is a traditional camera pose estimation method that leverages a 3D reconstruction of a given environment. An image is localized within an area using structural information of the 3D reconstruction.

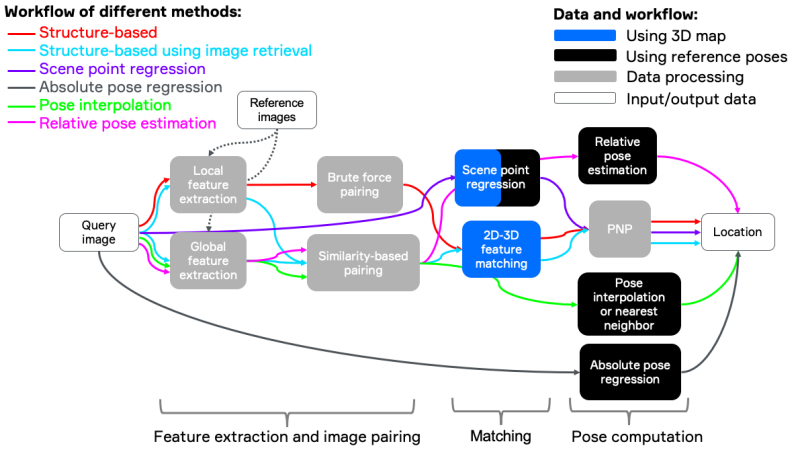



FIGURE 3.1: Overview of different methods of visual localization

### 3.1.1 Pixel correspondence estimation

The very first step of constructing a 3D map is finding the pixel-wise correspondences across multiple available reference images. The process of generating those correspondences consists of 3 stages:

- detection: identify the points of interest;
- description: the depiction of the nearby visual characteristics of each feature point should (ideally) possess invariance to changes in illumination, translation, scale, and in-plane rotation.;
- matching: descriptors are compared across the images to identify similar features.

Depending on the method, these stages can be isolated from each other or partially/fully fused. Early methods used handcrafted local feature extractors such as Harris Corner Detection or Shi-Tomasi Corner Detector and feature descriptors such as HoG, BRIEF, and ORB. The next-generation algorithms like SIFT[31] and its variations, SURF[5], KAZE[2], AKAZE[1] mainly represent a fusion of detection and description stages. Those methods significantly improve robustness and remain baseline options for many pipelines that rely on correspondences between images. Still, such keypoint extractors-descriptors have several drawbacks, including that they might be challenging to tune to the particularities of the target task. In particular, changing conditions such as day-night and seasons of the year are such stumbling blocks. Not to mention, the computation efficiency of SIFT-like algorithms often is insufficient for the tasks it tries to solve. Therefore, several data-driven learned representations were recently proposed, such as D2-Net[16] and SuperPoint[15]. They can learn local features in an end-to-end fashion.

Local feature matching is generally performed by matching keypoints (detected and described in the previous two stages) with a Nearest Neighbor (NN) search, filtering incorrect matches with Lowe's ratio test, the mutual check, and heuristics such as neighborhood consensus. Even though this approach is simple to understand and implement, it suffers from two major problems: computational complexity and lack of robustness to repeated patterns, which are common in human-built things such as buildings. Recent works on deep learning for matching propose learning to filter matches by classifying them into inliers and outliers [56, 57, 11]. These operate on

sets of matches, still estimated by NN search. Other works, such as SuperGlue[44] and SGMNet[13], completely eliminate the NN search step. Instead, they estimate matches by solving a differentiable optimal transport problem, whose costs are predicted by a graph neural network. Finally, there are end-to-end solutions that take pair of images as input and return the correspondences, effectively fusing all three stages(detection, description, and matching) into a single forward pass of the neural network [30, 41, 51, 61].

### 3.1.2 3D map construction

Building a map is called structure from motion (SfM), as spatial information about the environment is obtained due to the disparity of cameras' viewpoints. Usually, the SfM procedure is applied to a large set of unordered photos with unknown relative positions and (un)known intrinsic parameters of the cameras. Depending on how the reconstruction phase is approached, incremental and global SfM are distinguished.

In the case of global SfM[35], the whole procedure is performed in 2 global stages: absolute rotation estimation and absolute translation estimation, followed by bundle adjustment (BA) for further absolute pose refinement. The first stage contains relative rotation and translation estimation based on image pair (or triples) matches through essential matrix calculation. Further, the obtained input epipolar graph is filtered from outliers and false matches, and individual global rotations are computed with a least-square minimization. Then global translations are recovered from relative translations and global rotations. The global SfM, contrary to incremental methods, is less subjected to external calibration drift, and residual errors distribute evenly.

In contrast to global SfM, Incremental SfM[49] is a sequential processing pipeline with an iterative reconstruction component. It commonly starts with feature extraction and matching, similar to global SfM, followed by geometric verification. The resulting scene graph serves as the foundation for the reconstruction stage, which seeds the model with a carefully selected two-view reconstruction before incrementally registering new images, triangulating scene points, filtering outliers, and refining the reconstruction using bundle adjustment (BA).

### 3.1.3 Camera pose estimation

The way the 3D map is generated causes each 3D point to have at least two corresponding 2D keypoints in images of the reference image database. Using this fact, one can find 2D-3D correspondences between the 2D keypoints of the query image and the 3D points of the map by matching the 2D keypoints of the query image with 2D keypoints of images in the reference database. Alternatively, a more optimized algorithm can be applied that is in detail described here[45]. Based on the 2D-3D matches obtained in the previous step, the camera pose is computed using a so-called PnP solver within a RANSAC loop to increase robustness.

### 3.1.4 Limitations of the structure-based approach

The resulting maps can be quite large when mapping large areas like entire buildings or cities. For example, the 3D reconstruction of the Aachen-Day-Night visual localization dataset[46], which covers only the old inner city of Aachen, Germany,

contains between 700,000 and 2.5 million 3D points depending on the number of image pairs used. However, when the map grows even larger, it becomes impossible to efficiently analyze the correspondences between keypoints in the query image and all 3D points in the map. To address this issue, image retrieval methods (discussed in the next section) can be used. We can effectively reduce the search range by retrieving the most relevant images from the map. This allows for exploring keypoint correspondences only in the area defined by those images [23].

A slightly modified option would be to use retrieved images to create an ad hoc local map while maintaining the localization step instead of generating a global map of the environment. The advantage of this approach is that a global map is not required. However, constructing a local map using the retrieved images may not always be possible if there are not enough images showing the same scene as the query image.

Another way to establish correspondences in structure-based methods is to use scene point regression, where 2D-3D correspondences are estimated directly using techniques such as a DNN [9, 12, 29] or a random forest[50].

## 3.2 Image retrieval-based methods

Visual localization heavily relies on image retrieval as it assists in efficient structure-based localization methods, particularly in large-scale areas such as airports, commercial centers, and city regions, thereby enhancing both mapping and localization robustness. Additionally, image retrieval can be used as an alternative to structure-based methods, bypassing the need for pre-computed 3D structures. To perform image retrieval, the database of images must either have approximate location information like GPS (geo-localization) or exact poses with six degrees of freedom (DOF) within a designated world coordinate framework (visual localization). A pose of the query image is inferred by the pose of the nearest neighbor image or by interpolating the poses of top $k$ retrieved images [48, 54]. Note that these simple methods use only the poses of the retrieved images.

In both cases, the main objective is to choose a set of most similar images based on a chosen representation, followed by an optional re-ranking step that utilizes query expansion or filtering techniques.

In recent years, the predominant approach in image retrieval has been the utilization of different versions and expansions of the 'bag of visual words' technique. However, a more recent development has emerged where deep neural network activation features have demonstrated their capability to encode advanced semantic information, making them well-suited for image retrieval purposes. Furthermore, by training the network specifically for the retrieval task using ranking losses, the performance of such systems can be significantly enhanced [39]. Usually, fully-convolution networks such as VGG, AlexNet, or ResNet are fine-tuned for image retrieval on a large collection of unordered images in a fully automated manner. Reconstructed 3D models obtained by the state-of-the-art retrieval and structure-from-motion methods guide the training data selection. The training itself happens in metric learning fashion using triplet loss [22]. Recent approaches adopt learnable aggregation layers to boost retrieval performance even further. The goal of those layers is to transform features extracted by CNNs into compact descriptors. NetVLAD is a generalized VLAD layer that mimics the VLAD pooling procedure but keeps it differentiable in order to enable end-to-end training [3]. Alternatively, a pooling

layer called Generalized-Mean (GeM) generalizes max and average pooling by letting the network learn which is better for individual feature maps [38].

## 3.3 Pose regression-based methods

This approach involves utilizing a CNN network to learn how to regress the camera pose in an end-to-end manner from an RGB image by leveraging transfer learning from pre-trained deep CNNs for pose estimation. Specifically, PoseNet[26] modifies the CNN architecture used for image classification, such as VGGNet or ResNet, by replacing softmax layers with fully connected layers to regress both the 3D location and the orientation of the camera.

Directly regressing the absolute pose with end-to-end learning offers several benefits, including not requiring any feature engineering, relying on deep network encodings that are more resilient to challenging changes in the scene, and requiring less memory and constant inference time compared to 3D-based methods. However, the localization error achieved with PoseNet[26] is an order of magnitude larger than the error attained with state-of-the-art structure-based methods. In particular, the absolute camera pose regression approach suffers from two main issues[48]:

- It is sensitive to the scale of the training data: Since absolute camera pose regression directly predicts the camera pose in absolute coordinates, it requires a large and diverse dataset that covers a wide range of scales and viewpoints. However, collecting such a dataset is challenging and expensive.

- It is prone to overfit: Since the absolute camera pose regression approach has to predict the camera pose in absolute coordinates, it has a higher number of degrees of freedom compared to the relative camera pose regression approach. This makes it more prone to overfitting, especially when the training data is limited.

To address these issues, several improvements to the PoseNet[26] architecture have been proposed, including new loss functions, adding LSTM layers, or relying on additional data sources and sensor measurements to further enhance performance.

Despite these advances, learning an entire pipeline can result in suboptimal performance due to the tight coupling with the scene coordinates, limiting the generalization power of the network. Hybrid pose learning methods have been developed to address this issue by focusing on local or related problems and combining them with traditional image retrieval and structure-based pipelines, such as DSAC[12], which relies on geometrical constraints and is focused on 2D-3D matches. However, the learned models remain scene-dependent and do not generalize to new scenes.

To address the generalization problem, the recent SANet[17] proposes a scene-agnostic neural architecture for camera localization, where model parameters and scenes are independent of each other. The model leverages geometry through 3D point clouds obtained as dense MVS reconstructions from the top retrieved images and learns query-scene registration, as well as a camera pose regression.

# Chapter 4

# Methodology

In order to satisfy flexibility and extensibility requirements, we propose a structure-based approach that uses the database of reference images with known intrinsic and extrinsic parameters. The method utilizes image retrieval with multiple global image representations to select relevant images for further stages of visual localization. The request processing is divided into stages. During the very first stage, the input image's metadata is parsed, and feature enhancement algorithms are applied to the image itself. In the second stage, the input image is used to retrieve visually similar reference images from the database. The usual number of retrieved images is around 10-20, but it can be adjusted depending on the images' spatial disparity. In the next stage, we generate local 3D scene reconstruction by establishing 2D-2D local feature correspondences between the query image and the database images. One the final stage, the 6DOF camera pose is estimated by solving a PnP problem robustly inside a RANSAC with 2D-3D matches between keypoints on the query image and 3D points of the map.

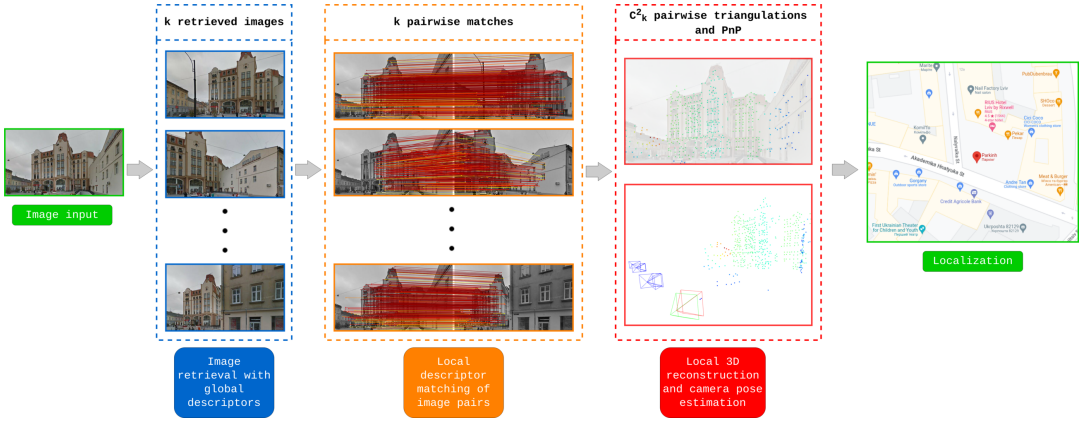



FIGURE 4.1: Key steps of visual localization

## 4.1 Preprocessing stage

The parameters of the camera and GPS are of the greatest interest to us when parsing image metadata. Specifically, we need the camera's parameters that would allow us to calculate the intrinsic matrix, which is required by the majority of PnP solvers. For example, if the metadata contains FOV, then we can approximate the camera intrinsic matrix as shown in 2.1. The GPS coordinates give us a good first approximation of the camera position, and if present, they are used to limit the search area during database retrieval.

The image itself is processed with a color equalization algorithm. This method usually increases the global contrast of the image, especially when the image is represented by a narrow range of intensity values which allows for finding better keypoint during the third stage. In addition, it partially mitigates changes in lighting conditions which can possibly lead to better retrieval results during the second stage.

Another optional step is a segmentation of all non-static objects, such as people, bicycles and cars, etc., followed by blending them with surroundings. It is important to note that usually masking is not enough. Masking, which introduces black blobs on the images, produces worse results during retrieval and local feature estimation than skipping this step entirely. The segmentation is performed by the DeepLabV3Plus-ResNet101[19] model pre-trained on the Cityscapes dataset[14]. Considering that the task this model solves in the pipeline is not critical, it might be replaced with another one offering lower computation costs as well as segmentation quality.

## 4.2 Retrieval stage

The retrieval step is essential for scalability, as regardless of the database's size, the computational complexity of the next pipeline's stages remains fixed. Namely, one can set hard and soft limits on the number of retrieved images. On the other hand, the next stages become heavily dependent on the retrieval step, as the quality (coverage of the scene) of selected images determines the quality of overall localization.

As discussed in 3.2, the main idea of the retrieval step is to find similar images in the reference database. Comparison is performed on images in some space which ideally is invariant to illumination changes, different seasons or weather conditions, viewpoint changes, etc. The function that maps images in such space is usually approximated with DNN. There are a couple of criteria when selecting a retrieval model. It is important to carefully investigate what datasets have been used for model training, namely the size, and presence of hard samples. Validation datasets should be investigated as well. The next criterion is speed which directly relates to model architecture. Most of the models have a similar structure: CNN features extraction block represented by popular backbones such as ResNet-18/50/101 or VGG followed by aggregation block, namely SPOC[4], MAC[53], R-MAC[33], GeM[38], NetVLAD[3] or CRN[27]. Optionally, an aggregation block can be followed by a dimensionality reduction block that might use FC or PCA. Surprisingly, the dimensionality reduction block might have the biggest influence on the pipeline's overall performance. The latency of retrieval model inference is constant since each query input image requires a single forward pass to generate a descriptor. However, the time of finding top k images in the reference database based on descriptors directly depends on the size of the database as well as the dimensionality of a descriptor. For the proper model selection, we used visual geo-localization benchmark infrastructure[8]. More specifically, we use two implementations of ResNet-101 + GeM + FC trained with triplet[22] and AP[39] losses, respectively.

It is possible to end up in a situation where the combination of database size and descriptor size would not allow all descriptors to fit into RAM. To overcome this, a specialized library such as Faiss[24] can be used. It is a library for efficient similarity search and clustering of dense vectors. It contains algorithms that search in sets of vectors of any size, up to ones that possibly do not fit in RAM.

In order to increase retrieval quality, a few additional steps are introduced:

- reranking;

- geometric expansion;
- geometric validation.

### 4.2.1 Reranking

Re-ranking techniques are categorized into two types based on similarity criteria, namely feature similarity and neighbor similarity. Evaluating feature similarity involves measuring the Euclidean distance in the feature space between a pair of images. On the other hand, the neighbor similarity is determined by the number of shared neighbors between images. Typically, neighbor-based methods outperform feature-based methods as they are robust to hard negatives or outliers that may differ by only one neighbor from true matches. The major challenge of neighbor-based re-ranking methods such as the k-reciprocal re-ranking method is their high computational complexity which makes them impractical for our application. We tackle this problem by using a highly parallelizable GNN-powered image retrieval re-ranking approach[58]. In order to eliminate redundant computations, we introduce partial caching of the initial k-NN graph, which is built in the first stage of the algorithm.

### 4.2.2 Geometric expansion

Geometric expansion is a rather simple technique to enrich the number of viewpoints on the scene. For each retrieved image, its neighbors within a radius of $r$ are added to the set of retrieved images.

### 4.2.3 Geometric validation

Geometric validation is a step that ensures that all of the retrieved images describe the same scene. First, perform cluster filtering based on the image position: we build a graph by connecting images if the Euclidean distance between their positions is less than $d_{thr}$. Then we look for the largest connected sub-graph in the graph and ignore all the other images. This way, we effectively ensure view overlap of the neighboring cameras by selecting optimal $d_{thr}$. For the remaining images, the dominant view direction is calculated. If an image view direction deviates from the dominant view direction by more than 90 degrees, then this image will be discarded as well.

## 4.3 Scene reconstruction and camera pose estimation stage

The local scene 3D reconstruction is a step that enables accurate position estimation. This step is subdivided into 3 phases: keypoints generation, matching, and triangulation. Keypoints generation methods have a long history of evolution from sets of handcrafted techniques to single end-to-end DNN that are able to find and describe thousands of points robustly. Most of the experiments were inducted with SuperPoint[15], D2-Net[16], and R2D2[40]. Those models feature in many benchmarks and sustainably get into the top 5. For the matching phase, we use a state-of-the-art model called SuperGlue[44]. It uses a graph neural network and attention to solve an assignment optimization problem, and handles partial point visibility and occlusion elegantly, producing a partial assignment. Other matching approaches that took into consideration either perform worse or are the variation of the SuperGlue

approach with little to no improvements in benchmark metrics. Alternatively, there are quite a few end-to-end approaches, such as DualRC-Net[30], LoFTR[51], and Patch2Pix[61], that incorporate keypoints generation as well as matching. We rejected those as they were incompatible with our 3D scene reconstruction procedure which we will discuss later.

The reconstruction algorithm consists of the following steps:

1. precompute keypoints for every image in the reference database (offline);
2. compute keypoints for query image with one of the keypoints generating models (SuperPoint by default);
3. establish keypoints correspondence of every retrieved image keypoints to query image keypoints with SuperGlue;
4. generate every possible pair of retrieved images;
5. for each such pair $(a, b)$ use matches from query keypoints to $a$'s keypoints and to $b$'s keypoints in order to establish correspondences between $a$ and $b$ keypoints;
6. use one of the triangulation algorithms (we use OpenCV implementation) to obtain 3D points corresponding to those keypoints;
7. apply *Expansion* or *Displacement* policy when populating shared 3D points pool with points from each individual pair.

The *Expansion* policy orders to append new points to the shared pool. As a result, a single 2D point on the query image might correspond to multiple 3D points.

The *Displacement* policy orders to replace a 3D point with another corresponding to the same 2D point if the confidence of the newer point is greater than the former one has. The confidence of the 3D point is calculated as a product of matching confidences of the keypoints that was used to calculate that 3D point. Alternatively, one can use non-linear triangulation of all points together. However, optimizing reprojection errors when data contains a high percentage of bad matches leads to lower robustness. It is better to have fewer points but to be more confident about them.

By the end of this procedure, we obtain well-established correspondences between 2D points on the query image and 3D points in reference database coordinates. After obtaining these 2D-3D matches, perspective-n-point (PnP) solvers are utilized for computing the camera pose. However, to improve the robustness and compensate for any outliers present in the 2D-3D matches, the PnP computation is performed within a RANSAC loop. In the pipeline, we use OpenCV implementation of PnP solver inside RANSAC since it delivers great performance.

## 4.4 Parallelization opportunities

Apart from the accuracy of the localization, latency is the second most important objective for the pipeline. One way of approaching this is to reduce the latency on each individual step. We can do this by using smaller networks or setting the number of retrieved images lower. However, it comes at the cost of accuracy. We suggest exploring data dependencies between each individual step before performing drastic optimization influencing the accuracy.

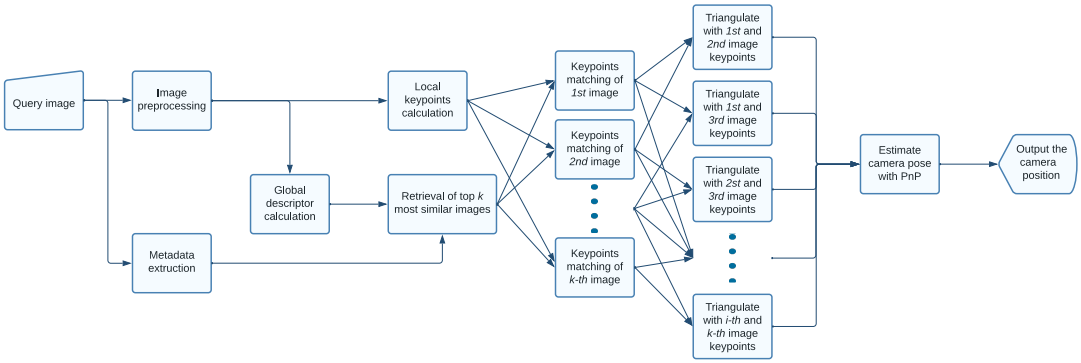



FIGURE 4.2: Dependencies between stages

As the diagram 4.2 suggests, the global descriptor calculation and image retrieval can be performed in parallel with local keypoints calculation. Considering the computation complexity of them both, we can conclude that in most cases, local keypoints calculation would not introduce any additional latency.

The matching of query image keypoints with keypoints of k retrieved images can be completed in parallel. The only limitation is the computation capacity.

The triangulation step is also completely parallelizable. Even though $C_k^2$ pairs might seem like a lot, considering that $k$ is usually between 10 and 20, the total number of triangulation pairs does not exceed 190. Furthermore, the majority of those pairs do not have any matching keypoints, meaning that only a small part of them triggers actual computation.

# Chapter 5

# Dataset

In addition to presenting the localization pipeline, we introduce a tool designed to facilitate the collection of datasets. The primary objective of this visual localization pipeline is to achieve universality, with the reference image database being the only element that requires modification, providing considerable flexibility and expandability. The tools for database collection are powered with OpenStreetMap API and Google Street View Static API.

The tools implement an interface to allow the following use case:

- user specifies what area needs to be covered with the database;
- user examines the preview map depicting points where images will be taken;
- user stops the process and goes to the first step if not satisfied else continues;
- user starts the database population process.

User can specify the coverage area in a couple of ways:

- supply the point with the latitude, longitude, and size of the bounding box around it in meters;
- supply the city name.

Additionally, the user can specify the distance between neighboring points. The supplied value is treated as an upper bound, meaning the maximum distance between 2 neighboring points is less or equal to that value. At the intersections and steep turns, the distance between points naturally decreases, which helps to cope with occlusions and limited views.

There are some more advanced parameters that have default values:

- camera FOV (90° by default);
- camera pitch (0° by default);
- number of images taken at each point (8 by default).

The default parameters listed above enable the creation of a 360-degree panorama 5.2 where each object in the scene is captured at least twice. This is crucial for image retrieval, as it prevents the objects from being split between separate views. Thus, making the retrieval process more robust to occlusions.

There might be situations where the other set of parameters would be more suitable. For example, if a user wants to capture more details at the cost of global context and structure, one can try to reduce FOV to 60-70 degrees and slightly increase pitch to capture more of building facades rather than sidewalks and roads. The number of photos taken at each point can be left as is or slightly increased.

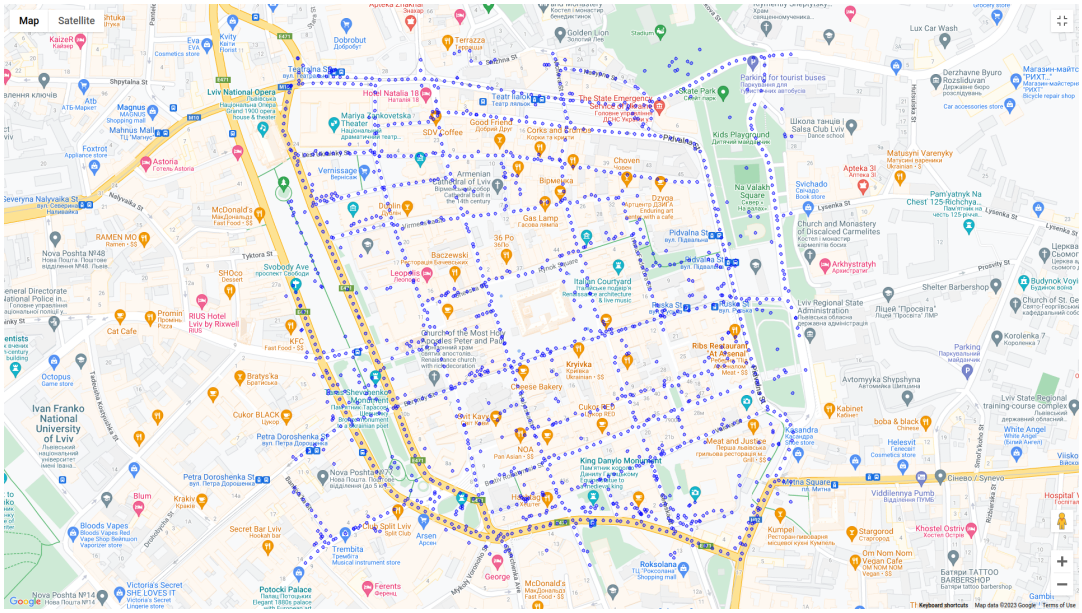
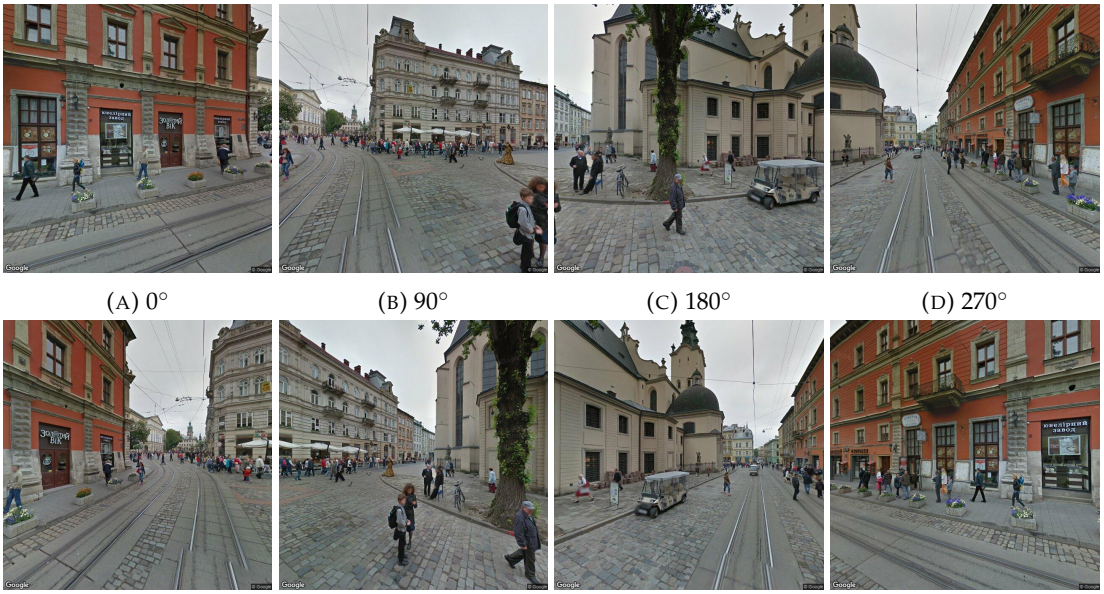



FIGURE 5.1: Preview map around Lviv City Council with 350-meter bbox



(A) 0° (B) 90° (C) 180° (D) 270°

(E) 45° (F) 135° (G) 225° (H) 315°

FIGURE 5.2: A panorama taken with default parameters

# Chapter 6

# Experiments

## 6.1 Dataset and Preprocessing

The experiments were conducted on several datasets, including the Cambridge Landmarks dataset[26], Aachen Day-Night datasets[47, 46, 59], and 2 datasets depicting the old city of Lviv that were collected with the tools discussed in the previous section.

Cambridge Landmarks, a large-scale outdoor visual relocalization dataset taken around Cambridge University, contains 6 scenes, namely Old Hospital, Kings College, St. Mary's Church, Great Court, Shop Facade, and Street, that total up to 8380 reference images and 4841 query images.

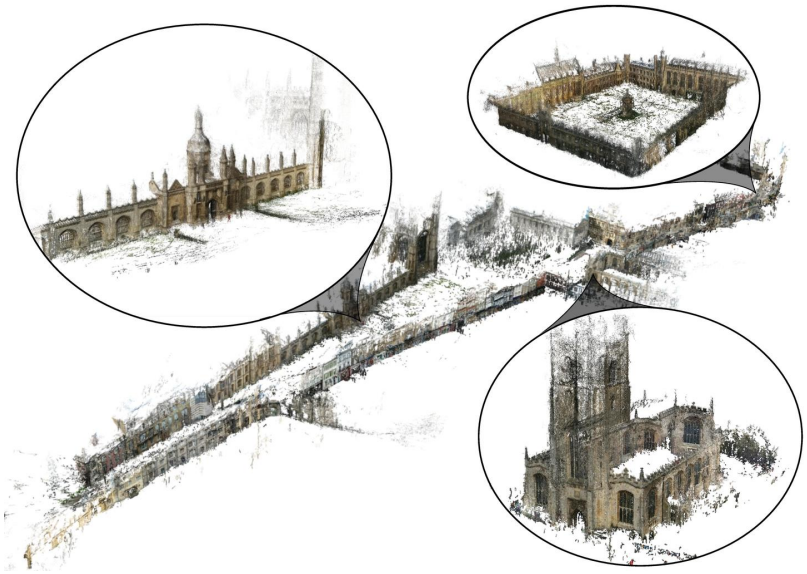


FIGURE 6.1: The 3D map of Cambridge Landmarks dataset

The Aachen Day-Night datasets, which are based on the original Aachen dataset, depict the old inner city of Aachen, Germany. The database images used to build the reference scene representations were all taken during the daytime with hand-held cameras over a period of about two years. The datasets offer query images taken during the daytime and at nighttime. All query images were taken with mobile phone cameras. Aachen Day-Night dataset features 4328 reference images and 922 (824 daytime, 98 nighttime) query images. Aachen Day-Night v1.1 dataset extends the Aachen Day-Night dataset with additional 93 nighttime query images and 2369 reference images.

FIGURE 6.2: The 3D map of Aachen Day-Night v1.1 dataset

Images of both Cambridge Landmarks and Aachen Day-Night datasets were downsized with OpenCV cubic interpolation to fit into GPU during evaluation. The intrinsic matrices were rescaled respectively.

The Lviv1 dataset consists of 5389 reference images and 1347 query images. The dataset covers a 500x500 meters square area around Lviv Opera. The images are taken exclusively at driveways with approximately 20-meter intervals in between.

The Lviv2 dataset consists of 9690 reference images and 2422 query images. The dataset covers a 350x350 meters square area around Lviv City Council. The images are taken at driveways as well as pedestrian zones with an approximately 10-meter interval in between. Unlike Lviv1, Lviv2 contains many images taken by mobile devices rather than Street View car equipment and images depicting scenes during nighttime.

## 6.2 Evaluation Metrics

There are a set of evaluation metrics used during the evaluation and parameter tuning of the visual localization pipeline. To measure the quality of descriptor vectors produced by DNNs, we use Mean Average Precision (MAP). MAP is a measure of how well a system or algorithm is able to retrieve relevant information from a dataset. One can think of MAP as an average of the area under the precision-recall curve for all possible retrieval cutoff values. The higher the MAP score, the better the algorithm is performing in terms of retrieving relevant information from the dataset. The trainset of each dataset was evaluated using MAP in order to determine which of [22] and [39] is more suited.

For the end-to-end evaluation of the Cambridge Landmarks dataset, we used the median position error in meters and median rotation error in degrees since most of the papers featuring this dataset use these metrics.

For the end-to-end evaluation of the Aachen Day-Night datasets as well as the Lviv1 and Lviv2 datasets, we used the accuracy metric, which was defined as the percentage of queries localized correctly within the predefined position errors and rotation errors.

## 6.3 Parameter settings

The pipeline has 6 major hyperparameters:

- **k** specifies how many images are retrieved from the reference database. The value of **k** was set to 10 for all the experiments.
- **cameras_distance_threshold** is used to select the largest set of images that are close enough to each other to ensure that they have enough view overlap. Each image within the set is not further than **cameras_distance_threshold** meters from the rest of the images in the set. The value of **cameras_distance_threshold** was set to 30 meters for all the experiments.
- **confidence_threshold** is used to filter out weak indirect matches. It can take values in the range from 0 to 1. The value of the **confidence_threshold** was set to 0.4 for all the experiments.
- **points_distance_threshold** is used to filter out 3D points that are too far from the cameras. Usually, incorrect matches can generate that kind of outlier. However, sometimes perfectly valid points can be quite distant. For example, the tops of churches or other tall buildings can be clearly visible from great distances. As a compromise, the value of **points_distance_threshold** was set to 300 meters for all the experiments.
- **reranking** is a boolean parameter that determines whether additional **reranking** will be applied after the retrieval step. This parameter wasn't fixed during the experiments.
- **policy** is a parameter that refers to the *Expansion* and *Displacement* policies discussed in the 4. This parameter wasn't fixed during the experiments.

## 6.4 Evaluation results

### 6.4.1 Cambridge Landmarks

The evaluation of the Cambridge Landmarks datasets shows the pipeline performance in case localization guided by GPS attached to the photo. The area which is covered by each scene except for *Street* is approximately 30x30 meters, which allows mimicking cases when reference images from the database are filtered by keeping only those within 30 meters around the GPS coordinates.

| **Cambridge Landmarks** | Median 6D Localization Errors | | | | | |
|---|---|---|---|---|---|---|
| Method | Great Court | K. College | Old Hospital | Shop Facade | St M. Church | Street |
| PoseNet [26] | 7.00 m, 3.7° | 0.99 m, 1.1° | 2.17 m, 2.9° | 1.05 m, 4.0° | 1.49 m, 3.4° | 20.7 m, 25.7° |
| ActiveSearch [45] | 0.24 m, 0.1° | 0.13 m, 0.2° | 0.20 m, 0.4° | 0.04 m, 0.2° | 0.08 m, 0.3° | N/A |
| InLoc [52] | 1.20 m, 0.6° | 0.46 m, 0.8° | 0.48 m, 1.0° | 0.11 m, 0.5° | 0.18 m0.6° | N/A |
| DSAC++ [10] | 0.40 m, 0.2° | 0.18 m, 0.3° | 0.20 m, 0.3° | 0.06 m, 0.3° | 0.13 m, 0.4° | N/A |
| DSAC* [12] | 0.49 m, 0.3° | 0.15 m, 0.3° | 0.21 m, 0.4° | 0.05 m, 0.3° | 0.13 m, 0.4° | N/A |
| HACNet [29] | 0.28 m, 0.2° | 0.18 m, 0.3° | 0.19 m, 0.3° | 0.06 m, 0.3° | 0.09 m, 0.3° | N/A |
| PixLoc [42] | 0.30 m, 0.1° | 0.14 m, 0.2° | 0.16 m, 0.3° | 0.05 m, 0.2° | 0.10 m, 0.3° | N/A |
| HLoc [43] + SuperGlue [44] | 0.16 m, 0.1° | 0.12 m, 0.2° | 0.15 m, 0.3° | 0.04 m, 0.2° | 0.07 m, 0.2° | N/A |
| NAVER [23] | 0.10 m, 0.0° | 0.05 m, 0.1° | 0.09 m, 0.2° | 0.02 m, 0.1° | 0.03 m, 0.1° | 0.10 m, 0.3° |
| Our(*Displacement*, w/o rerank) | 0.47 m, 0.2° | 0.22 m, 0.3° | 0.25 m, 0.5° | 0.07 m, 0.3° | 0.14 m, 0.4° | 0.27 m, 0.9° |

TABLE 6.1: The Cambridge Landmarks dataset evaluation results

The parameter settings for the experiment are reflected in 6.3. In addition, the parameter **reranking** was set to false since retrieval as a step wasn't necessary at all

due to the size of the scenes. The parameter **policy** was set to *Displacement* because the number of matches was high due to the size of the scenes. Filtering out weak matches during triangulation is beneficial for pose estimation later on.

The results suggest that our pipeline has average performance in most of the scenes. There are a couple of reasons for that:

- Unlike most of the approaches in the table, our method doesn't leverage 3D reconstruction, which is part of the dataset. Our pipeline is purely relying on the images, camera intrinsics, and camera extrinsics for reference images.
- The resolution of the images has been reduced by more than two times in order to fit in GPU constraints.

Besides, the last column of the table 6.1 reveals that, unlike most approaches, our pipeline keeps the error level low when the scene size increases significantly.

### 6.4.2 Aachen Day-Night v1.1

The Aachen Day-Night v1.1 dataset is much closer to the conditions our localization pipeline has been designed to work in. This dataset covers a much greater area compared to individual scenes of the previous dataset, as well as features different weather and lighting conditions.

| Aachen Day-Night v1.1 | Accuracy thresholds high: 0.25 m, 2° mid: 0.5 m, 5° low: 5 m, 10° | | | | | |
|---|---|---|---|---|---|---|
| | daytime | | | nighttime | | |
| Method | high | mid | low | high | mid | low |
| ActiveSearch [45] | 57.3 | 83.7 | 96.6 | 28.6 | 37.8 | 51.0 |
| HLoc [43] + SuperGlue [44] | 89.8 | 96.1 | 99.4 | 77.0 | 90.6 | 100 |
| NAVER [23] | 90.9 | 96.7 | 99.5 | 78.5 | 91.1 | 98.0 |
| RLOCS v1.0 [18] | 86.0 | 94.8 | 98.8 | 72.3 | 88.5 | 99.0 |
| Our(*Displacement*, w/o rerank) | 75.7 | 88.1 | 97.3 | 59.2 | 77.0 | 94.8 |
| Our(*Expansion*, w/o rerank) | 70.0 | 85.1 | 97.2 | 60.2 | 79.1 | 94.8 |
| Our(*Displacement*, w/ rerank) | 67.5 | 82.5 | 95.5 | 60.7 | 77.5 | 92.7 |
| Our(*Expansion*, w/ rerank) | 66.5 | 81.1 | 95.0 | 57.6 | 77.0 | 92.7 |

TABLE 6.2: The Aachen Day-Night v1.1 dataset evaluation results

Similar to the previous evaluation, the result table states that our pipeline has an average performance. The factors that might have influenced the metrics are the same as in 6.4.1. Overall, we see that presence of reranking step has negative effect on the performance. The possible reason is that reranking can often result in a bias towards a specific set of images, which could reduce the diversity of retrieved images. This could negatively impact the accuracy of the visual localization pipeline, particularly in cases where the initial set of retrieved images contains important images that were not included in the reranked set. Another observation about table data is that *Displacement* policy works better than *Expansion* policy. The possible explanation is that the images in the Aachen Day-Night v1.1 dataset have a lot of overlap and thus generate a lot of matches for 3D reconstruction. By applying *Expansion* policy, we extend the number of 3D points even further by introducing points based on weak matches. In the next step, the PnP solver in the RANSAC loop is prone

to exit before fitting the best subset of points due to max iteration constrain. Basically, the less filtering is done before pose estimation, the more challenging the pose estimation problem is.

### 6.4.3 Lviv1/2

The Lviv1/2 datasets encompass a much broader area compared to the Cambridge Landmarks or Aachen Day-Night v1.1 datasets. The panoramas are distributed evenly throughout the region as per the limits of the Google Street View database. The purpose of Lviv1/2 datasets is to be reference databases for visual localization in the corresponding areas. Therefore, splitting these datasets for evaluation negatively effects the performance, as query samples effectively become blind spots of the reference databases. On the other hand, we cannot create dataset of queries based on geotagged photos from the internet, as the source of their geodata is GPS in the vast majority of cases. Benchmarking the visual localization pipeline with GPS data does not make sense, considering that the research's primary purpose is to outmatch the GPS performance in the urban environment. Therefore, we need to stick to the first option, recognizing possible pitfalls.

| Lviv1 | Accuracy thresholds high: 0.5 m, 2° mid: 1 m, 5° low: 5 m, 10° | | | Median 6D localization error |
|---|---|---|---|---|
| Method | high | mid | low | |
| Our(*Displacement*, w/o rerank) | 0.5 | 0.69 | 0.86 | 0.43 m, 1.0° |
| Our(*Expansion*, w/o rerank) | 0.48 | 0.69 | 0.86 | 0.46 m, 1.0° |
| Our(*Displacement*, w/ rerank) | 0.42 | 0.61 | 0.81 | 0.58 m, 1.1° |
| Our(*Expansion*, w/ rerank) | 0.4 | 0.57 | 0.75 | 0.66 m, 1.2° |

TABLE 6.3: The Lviv1 dataset evaluation results

| Lviv2 | Accuracy thresholds high: 0.5 m, 2° mid: 1 m, 5° low: 5 m, 10° | | | Median 6D localization error |
|---|---|---|---|---|
| Method | high | mid | low | |
| Our(*Displacement*, w/o rerank) | 0.43 | 0.58 | 0.75 | 0.55 m, 1.2° |
| Our(*Expansion*, w/o rerank) | 0.39 | 0.54 | 0.71 | 0.66 m, 1.4° |
| Our(*Displacement*, w/ rerank) | 0.36 | 0.51 | 0.70 | 0.85 m, 1.6° |
| Our(*Expansion*, w/ rerank) | 0.32 | 0.45 | 0.65 | 1.19 m, 1.8° |

TABLE 6.4: The Lviv2 dataset evaluation results

The results slightly exceed the expectations, considering the intrinsically negative effect of splitting the datasets for evaluation. The tables 6.3 and 6.4 suggest that 60-70% of all queries have been localized within 1 meter around the ground truth, which is a decent result. Furthermore, a considerable portion of the query images depicts challenging scenes that make it tough or almost impossible to identify their location accurately:

- Some scenes are only visible from a single panorama. In this case, the proper multi-view 3D reconstruction is impossible; the only option is to estimate the camera pose based on matches from the single most similar view or use the camera pose of that view as a rough approximation itself.
- Scenes depicting an extensive amount of vegetation are challenging for the retrieval step of the pipeline. In most cases, such queries do not contain unique

features that would allow distinguishing between them and others found in the reference database.

- Scenes containing repeating patterns might cause a failure of localization during local keypoints matching. This problem is directly related to the method of matching, which, in this case, is SuperGlue [44].

(A) Single panorama view (B) Trees occluding a building (C) Repeating wall pattern

FIGURE 6.3: Problematic cases for visual localization pipeline

Another observation is that the performance of the pipeline with different parameters is consistent with previous experiments 6.2. We have also tried to benchmark the state-of-the-art approaches such as NAVER [23] and HLoc [43] + SuperGlue [44] on the Lviv1/2. Unfortunately, we were unable to conduct these experiments because the computational complexity required exceeded the capabilities of our hardware.

# Chapter 7

# Conclusions

In this work, we have built an end-to-end visual localization pipeline that is meant to increase localization accuracy in the urban environment. We analyzed various approaches, from structural-based to DNNs and everything in between. We came up with an efficient, flexible, and extensible solution that delivers decent results on many benchmarks without fine-tuning. Furthermore, we analyzed the scenarios in which the localization process might fail and found countermeasures for some of them. In addition, we developed a set of tools for collecting and extending a reference database by samples from a given area.

# Chapter 8

# Future work

We consider several possible directions for improvement and development. We plan on conducting more experiments with other localization datasets, as well as performing the evaluation of state-of-the-art approaches on Lviv1/2 datasets or their expanded versions. In addition, we are going to add filtering steps to our database acquisition tools in order to discard panoramas that are not helpful for localization.

Another possible track is to turn the research into a product. The backend, represented by the pipeline itself, must be deployed onto the cloud infrastructure. In addition to that, the REST API will be set up to allow other services to access the localization capabilities we are to offer. On top of that, the frontend app is to be created based on that API, allowing ordinary users to add or refine geotag information to their photos easily.

# Appendix A

The code for this project is available in our GitHub repository https://github.com/Tsapiv/visual-geolocation-pipeline. The repository contains the source code for the project, along with any necessary documentation or setup instructions.

# Bibliography


[1] Pablo Alcantarilla, Jesus Nuevo, and Adrien Bartoli. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". In: *Procedings of the British Machine Vision Conference 2013*. British Machine Vision Association, 2013. DOI: 10.5244/c.27.13. URL: https://doi.org/10.5244/c.27.13.

[2] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. "KAZE Features". In: *Computer Vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 214–227. DOI: 10.1007/978-3-642-33783-3_16. URL: https://doi.org/10.1007/978-3-642-33783-3_16.

[3] Relja Arandjelović et al. *NetVLAD: CNN architecture for weakly supervised place recognition*. 2016. arXiv: 1511.07247 [cs.CV].

[4] Artem Babenko and Victor Lempitsky. *Aggregating Deep Convolutional Features for Image Retrieval*. 2015. arXiv: 1510.07493 [cs.CV].

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 404–417. DOI: 10.1007/11744023_32. URL: https://doi.org/10.1007/11744023_32.

[6] P. A. Beardsley, A. Zisserman, and D. W. Murray. "Navigation using affine structure from motion". In: *Computer Vision — ECCV '94*. Springer Berlin Heidelberg, 1994, pp. 85–96. DOI: 10.1007/bfb0028337. URL: https://doi.org/10.1007/bfb0028337.

[7] P.A. Beardsley, A. Zisserman, and D.W. Murray. In: *International Journal of Computer Vision* 23.3 (1997), pp. 235–259. DOI: 10.1023/a:1007923216416. URL: https://doi.org/10.1023/a:1007923216416.

[8] Gabriele Berton et al. *Deep Visual Geo-localization Benchmark*. 2022. arXiv: 2204.03444 [cs.CV].

[9] Eric Brachmann and Carsten Rother. *Expert Sample Consensus Applied to Camera Re-Localization*. 2019. arXiv: 1908.02484 [cs.CV].

[10] Eric Brachmann and Carsten Rother. *Learning Less is More - 6D Camera Localization via 3D Surface Regression*. 2018. arXiv: 1711.10228 [cs.CV].

[11] Eric Brachmann and Carsten Rother. *Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses*. 2019. arXiv: 1905.04132 [cs.CV].

[12] Eric Brachmann and Carsten Rother. *Visual Camera Re-Localization from RGB and RGB-D Images Using DSAC*. 2020. arXiv: 2002.12324 [cs.CV].

[13] Hongkai Chen et al. *Learning to Match Features with Seeded Graph Matching Network*. 2021. arXiv: 2108.08771 [cs.CV].

[14] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[15] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. *SuperPoint: Self-Supervised Interest Point Detection and Description*. 2018. arXiv: 1712.07629 [cs.CV].

[16] Mihai Dusmanu et al. *D2-Net: A Trainable CNN for Joint Detection and Description of Local Features*. 2019. arXiv: 1905.03561 [cs.CV].

[17] Heng Fan and Haibin Ling. *SANet: Structure-Aware Network for Visual Tracking*. 2017. arXiv: 1611.06878 [cs.CV].

[18] Huanhuan Fan et al. *Visual Localization Using Semantic Segmentation and Depth Prediction*. 2020. arXiv: 2005.11922 [cs.CV].

[19] Gongfan Fang. *DeepLabV3Plus-Pytorch*. https://github.com/VainF/DeepLabV3Plus-Pytorch. 2019.

[20] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24 (1981), pp. 381–395.

[21] Richard I. Hartley and Peter Sturm. "Triangulation". In: *Computer Vision and Image Understanding* 68.2 (Nov. 1997), pp. 146–157. DOI: 10.1006/cviu.1997.0547. URL: https://doi.org/10.1006/cviu.1997.0547.

[22] Elad Hoffer and Nir Ailon. *Deep metric learning using Triplet network*. 2018. arXiv: 1412.6622 [cs.LG].

[23] Martin Humenberger et al. *Robust Image Retrieval-based Visual Localization using Kapture*. 2022. arXiv: 2007.13867 [cs.CV].

[24] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* 7.3 (2019), pp. 535–547.

[25] K. Kanatani, Y. Sugaya, and H. Niitsuma. "Triangulation from Two Views Revisited: Hartley-Sturm vs. Optimal Correction". In: *Procedings of the British Machine Vision Conference 2008*. British Machine Vision Association, 2008. DOI: 10.5244/c.22.18. URL: https://doi.org/10.5244/c.22.18.

[26] Alex Kendall, Matthew Grimes, and Roberto Cipolla. *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization*. 2016. arXiv: 1505.07427 [cs.CV].

[27] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. "Learned Contextual Feature Reweighting for Image Geo-Localization". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.346. URL: https://doi.org/10.1109/cvpr.2017.346.

[28] Seong Hun Lee and Javier Civera. *Triangulation: Why Optimize?* 2019. arXiv: 1907.11917 [cs.CV].

[29] Xiaotian Li et al. *Hierarchical Scene Coordinate Classification and Regression for Visual Localization*. 2020. arXiv: 1909.06216 [cs.CV].

[30] Xinghui Li et al. "Dual-Resolution Correspondence Networks". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2020.

[31] David G Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[32] C.-P. Lu, G.D. Hager, and E. Mjolsness. "Fast and globally convergent pose estimation from video images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.6 (June 2000), pp. 610–622. DOI: 10.1109/34.862199. URL: https://doi.org/10.1109/34.862199.

[33] Federico Magliani and Andrea Prati. *An accurate retrieval through R-MAC+ descriptors for landmark recognition*. 2018. arXiv: 1806.08565 [cs.CV].

[34] Krista Merry and Pete Bettinger. "Smartphone GPS accuracy study in an urban environment". en. In: *PLoS One* 14.7 (July 2019), e0219890.

[35] Pierre Moulon, Pascal Monasse, and Renaud Marlet. "Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion". In: *2013 IEEE International Conference on Computer Vision*. IEEE, Dec. 2013. DOI: 10.1109/iccv.2013.403. URL: https://doi.org/10.1109/iccv.2013.403.

[36] Gaku Nakano. "A versatile approach for solving PnP, PnPf, and PnPfr problems". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 338–352.

[37] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer. "Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.10 (Oct. 2013), pp. 2387–2400. DOI: 10.1109/tpami.2013.36. URL: https://doi.org/10.1109/tpami.2013.36.

[38] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. "Fine-Tuning CNN Image Retrieval with No Human Annotation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.7 (July 2019), pp. 1655–1668. DOI: 10.1109/tpami.2018.2846566. URL: https://doi.org/10.1109/tpami.2018.2846566.

[39] Jerome Revaud et al. *Learning with Average Precision: Training Image Retrieval with a Listwise Loss*. 2019. arXiv: 1906.07589 [cs.CV].

[40] Jerome Revaud et al. *R2D2: Repeatable and Reliable Detector and Descriptor*. 2019. arXiv: 1906.06195 [cs.CV].

[41] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. *Efficient Neighbourhood Consensus Networks via Submanifold Sparse Convolutions*. 2020. arXiv: 2004.10566 [cs.CV].

[42] Paul-Edouard Sarlin et al. *Back to the Feature: Learning Robust Camera Localization from Pixels to Pose*. 2021. arXiv: 2103.09213 [cs.CV].

[43] Paul-Edouard Sarlin et al. *From Coarse to Fine: Robust Hierarchical Localization at Large Scale*. 2019. arXiv: 1812.03506 [cs.CV].

[44] Paul-Edouard Sarlin et al. "SuperGlue: Learning Feature Matching With Graph Neural Networks". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.00499. URL: https://doi.org/10.1109/cvpr42600.2020.00499.

[45] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. "Efficient and Effective Prioritized Matching for Large-Scale Image-Based Localization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.9 (Sept. 2017), pp. 1744–1756. DOI: 10.1109/tpami.2016.2611662. URL: https://doi.org/10.1109/tpami.2016.2611662.

[46] Torsten Sattler et al. *Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions*. 2018. arXiv: 1707.09092 [cs.CV].

[47] Torsten Sattler et al. "Image Retrieval for Image-Based Localization Revisited". In: *British Machine Vision Conference (BMCV)*. 2012.

[48] Torsten Sattler et al. *Understanding the Limitations of CNN-based Absolute Camera Pose Regression*. 2019. arXiv: 1903.07504 [cs.CV].

[49] Johannes L. Schonberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.445. URL: https://doi.org/10.1109/cvpr.2016.445.

[50] Jamie Shotton et al. "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2013. DOI: 10.1109/cvpr.2013.377. URL: https://doi.org/10.1109/cvpr.2013.377.

[51] Jiaming Sun et al. *LoFTR: Detector-Free Local Feature Matching with Transformers*. 2021. arXiv: 2104.00680 [cs.CV].

[52] Hajime Taira et al. *InLoc: Indoor Visual Localization with Dense Matching and View Synthesis*. 2018. arXiv: 1803.10368 [cs.CV].

[53] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. *Particular object retrieval with integral max-pooling of CNN activations*. 2016. arXiv: 1511.05879 [cs.CV].

[54] Akihiko Torii, Josef Sivic, and Tomas Pajdla. "Visual localization by linear combination of image descriptors". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, Nov. 2011. DOI: 10.1109/iccvw.2011.6130230. URL: https://doi.org/10.1109/iccvw.2011.6130230.

[55] Kui Yang et al. "Iteratively Reweighted Midpoint Method for Fast Multiple View Triangulation". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 708–715. DOI: 10.1109/lra.2019.2893022. URL: https://doi.org/10.1109/lra.2019.2893022.

[56] Kwang Moo Yi et al. *Learning to Find Good Correspondences*. 2018. arXiv: 1711.05971 [cs.CV].

[57] Jiahui Zhang et al. *Learning Two-View Correspondences and Geometry Using Order-Aware Network*. 2019. arXiv: 1908.04964 [cs.CV].

[58] Xuanmeng Zhang et al. *Understanding Image Retrieval Re-Ranking: A Graph Neural Network Perspective*. 2020. arXiv: 2012.07620 [cs.CV].

[59] Zichao Zhang, Torsten Sattler, and Davide Scaramuzza. "Reference Pose Generation for Visual Localization via Learned Features and View Synthesis". In: *arXiv* 2005.05179 (2020).

[60] Yinqiang Zheng and Laurent Kneip. "A Direct Least-Squares Solution to the PnP Problem with Unknown Focal Length". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.198. URL: https://doi.org/10.1109/cvpr.2016.198.

[61] Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixe. *Patch2Pix: Epipolar-Guided Pixel-Level Correspondences*. 2021. arXiv: 2012.01909 [cs.CV].