Tampere University

Luca Ferranti

CONFIDENCE ESTIMATION IN IMAGE-BASED LOCALIZATION

Faculty of Information Technology and Communication Sciences
Master's Thesis
November 2019

ABSTRACT

Luca Ferranti: Confidence Estimation in Image-Based Localization
Master's Thesis
Tampere University
Electrical Engineering, MSc
November 2019

Image-based localization aims at estimating the camera position and orientation, briefly referred as camera pose, from a given image. Estimating the camera pose is needed in several applications, such as augmented reality, odometry and self-driving cars. A main challenge is to develop an algorithm for large varying environments, such as buildings or whole cities. During the past decade several algorithms have tackled this challenge and, despite the promising results, the task is far from being solved. Several applications, however, need a reliable pose estimate; in odometry applications, for example, the camera pose is used to correct the drift error accumulated by inertial sensor measurements. Based on this, it is important to be able to assess the confidence of the estimated pose and manage to discriminate between correct and incorrect poses within a prefixed error threshold. A common approach is to use the number of inliers produced in the RANSAC loop to evaluate how good an estimate is. Particularly, this is used to choose the best pose from a given image from a set of candidates. This metric, however, is not very robust, especially for indoor scenes, presenting several repetitive patterns, such as long textureless walls or similar objects. Despite some other metrics have been proposed, they aim at improving the accuracy of the algorithm, by grading candidate poses referred to the same query image; they thus recognize the best pose among a given set but cannot be used to grade the overall confidence of the final pose. In this thesis, we formalize confidence estimation as a binary classification problem and investigate how to quantify the confidence of an estimated camera pose. Opposed to the previous work, this new research question takes place after the whole visual localization pipeline and is able to compare also poses from different query images. In addition to the number of inliers, other factors such as the spatial distributions of inliers, are considered. A neural network is then used to generate a novel robust metric, able to evaluate the confidence for different query images. The proposed method is benchmarked using InLoc, a challenging dataset for indoor pose estimation. It is also shown the proposed confidence metric is independent of the dataset used for training and can be applied to different datasets and pipelines.

Keywords: Confidence Estimation, Visual Localization, Computer Vision, InLoc, Neural Networks

TIIVISTELMÄ

Luca Ferranti: Luottamuksen Estimointi Kuvapohjaisessa Paikannuksessa
Diplomityö
Tampereen yliopisto
Sähkötekniikka, DI
Marraskuu 2019

Kuvapohjainen paikannus pyrkii estimoimaan kameran sijainnin ja asennon annetusta kuvasta. Tätä voidaan soveltaa esimerkiksi lisättyyn todellisuuteen, odometriaan tai automatisoituihin autoihin. Estimointi osoittautuu haastavaksi muuttuvissa ympäristöissä, kuten kaupungeissa tai suurissa sisätiloissa ja viime vuosien nopeasta kehittymisestä huolimatta kuvapohjainen paikannus on edelleen avoin tehtävä. Sovellukset vaativat kuitenkin luotettavaa estimointia, sillä esimerkiksi odometriassa estimoituja kameran sijaintia ja asentoa käytetään inertiasensoreiden mittauksista kasaantuneen virheen korjaamiseen. Tämän johdosta kyky arvioida estimoinnin luotettavuutta on olennainen tehtävä. Tähän mennessä mittarina on monesti käytetty RANSAC-algoritmissa esiintyvää malliin sopivien pisteiden (inliers) määrää. Tämä mittari ei kuitenkaan sovellu esimerkiksi sisätilapaikannukseen, jossa kuvioiden toistuvuuden takia epätarkalla estimoinnilla voi olla myös suuri määrä malliin sopivia pisteitä. Tässä opinnäytetyössä luottamuksen estimointia formuloidaan binäärisenä luokittelutehtävänä ja kehitetään neuroverkkopohjaista menetelmää, jolla kvantisoida estimoidun kameran sijainnin ja asennon luotettavuutta. Menetelmä opetetaan ja testataan InLoc-datasetin avulla, joka on haastava sisätilapaikannukseen tarkoitettu datasetti. Työssä osoitetaan myös, että kehitetty menetelmä on opetuksessa käytetystä datasetistä riippumaton ja sitä voidaan soveltaa myös muihin datasetteihin.

Avainsanat: Luottamuksen Estimointi, Kuvapohjainen Paikannus, Konenäkö, InLoc, Neuroverkot

PREFACE

This thesis was written during autumn 2019 in the computer vision research group in Aalto University. The supervisors were Jani Boutellier (Tampere) and Juho Kannala (Aalto). The examiners were Jani Boutellier and Esa Rahtu (Tampere).

First, I would like to thank my supervisors Jani and Juho for their guidance and especially for the precious feedback I have been constantly given. Without you, this thesis would not exist and even if it did, it would be much worse. I am also grateful for all the amazing colleagues I have been working with, both in Tampere and in Espoo. Having a comfortable work environment was definitely a necessary condition for the success of this thesis. I also want to thank my friends from Tampere University for the time spent studying together and, most importantly, for the time spent not studying. We really had a lot of fun during these years! Moreover, many thanks also to all my friends and relatives for the support and understanding during my whole life. Last, but definitely not least, I wish to thank my fiancée Lotta Tapanainen for her immeasurable love and support.

Tampere, 19th November 2019

Luca Ferranti

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

α, β, γ	angles
ANN	Artificial neural network
AUC	Area Under Curve
$\mathbf{C}$	Camera position in real world coordinates
CNN	Convolutional neural network
$\sim$	Equivalence relation
η	Inliers coverage score
f, f_x, f_y	Focal length
Inls	Inliers
K	Camera intrinsic parameters matrix
λ	Scalar parameter for scale invariance
$\mathcal{L}$	Loss function
NN	Nearest neighbor
$\| \cdot \|$	Euclidean norm
P	Projection matrix
$\mathbb{P}^n$	Projective space
PnP	Perspective-n-point
p	Precision
p_x, p_y	Principal point offset coordinates
$\mathbf{q}$	Quaternion
R	Rotation matrix
RANSAC	Random sample consensus
r	Recall
$\mathbb{R}^n$	Euclidean space
SIFT	Scale-invariant feature transform
$\mathbf{t}$	Translation vector
$\mathrm{tr}(\cdot)$	Trace of a matrix
w	Added coordinate when mapping from $\mathbb{R}^n$ to $\mathbb{P}^n$
$\mathbf{X}$	Real world point

$\mathbf{x}$	Image point
x, y	Coordinates in 2D vector space
X, Y, Z	Coordinates in 3D vector space

1 INTRODUCTION

We, human beings, are able to extract information about our environment simply relying on what we see. For centuries people have been able to orient themselves using just a map and their eyes. This leads to the question *Could machines do the same?* Given a camera and some information about the environment, can a computer orient itself? In the field of computer vision, i.e. the automatic analysis of images, this question is known as *image-based localization* (or visual localization, or camera pose estimation) [25]. In a more formal way, image-based localization aims at computing the camera position and orientation, together shortly referred as *camera pose*, from a given image when the environment is known. Mathematically, knowing the environment means to know the coordinates of the 3D points in it and computing the pose of a camera means to find a mapping between the 3D points of the environment and the 2D points in the image. In pose estimation problems, it is also assumed that the internal parameters of the camera are known.

An accurate camera pose estimation is necessary in several applications. In robotics, autonomous vehicles need to keep track of the path they have travelled. This motion tracking problem, known as odometry [51], is generally tackled with inertial sensor measurements. Nevertheless, inertial sensors accumulate measurement errors, which makes the estimated position too inaccurate after a few minutes of motion. To overcome this problem, a reliable external signal, measuring the position of the device, must be used to correct the accumulated error. In outdoor environment, this task is accomplished by GNSS/GPS [9]. As GNSS signals are not detectable inside buildings, new solutions are needed for indoor odometry. Together with RF based (e.g. WiFi) positioning [21, 55], visual localization offers appealing possibilities for inertial odometry, as some state-of-the-art algorithms can already be run in real-time on mobile devices [26, 27]. Augmented Reality (AR) also requires information about the camera pose [31]. As objects are perceived differently in shape and size depending on the point of view of the observer, adding realistic-looking objects requires knowledge about the camera position and orientation.

In a typical pose estimation pipeline, the environment is known to the machine through a large database of images, for which the original 3D coordinates of the pixels, and hence also the camera pose, are known. When a *query image* in fed into the algorithm, it will first retrieve the most similar images from the database, form correspondences between points in the query and database images and then use the correspondences to compute the camera pose. As multiple database images are retrieved for a single query image,

multiple candidate poses will be produced. Computing the camera pose is generally an ill-posed problem, meaning after the computation only some correspondences will agree with the proposed camera. These correspondences are called *inliers*. The pose with the highest number of inliers will be chosen as the best candidate and outputted as final result.

Despite the improvements achieved in the past years [35, 36, 48], image-based localization is still far from being solved. High accuracies are achieved for simple stationary cases [17, 23], where the environments are relatively small and do not undergo big changes. However, realistic applications, such as self-driving cars, require pose estimation in large varying environments, such as whole cities or buildings. Clearly, the same square will appear completely different in winter or in summer, or even during different hours of the same day. Consequently, retrieving similar database images and forming correspondences will be challenging. On the other side, large environments have generally certain redundancy. For example, the same windows may appear on opposite sides of a building, which may introduce some wrong correspondences between query and database image. Indeed, sometimes estimating the pose may not be even possible. Let us consider the case of indoor visual localization and suppose our query image shows only a light switch on a white wall, without any additional details. Clearly, this picture cannot be used to reliably estimate the pose. A building can have hundreds of identical light switches and knowing which one of them is in the picture is impossible without further details. Still, our algorithm will compute a pose and it may also return a high number of inliers if an identical light switch is found from a database image.

The previous paragraph opens the question *How reliable my estimate is?* Research so far has focused on finding the best pose from a set of candidates and several metrics, also more robust than the number of inliers, have been developed. Still, these metrics do not answer our question, as the best pose among a set of candidates can still be wrong. For this reason, *confidence estimation*, investigated in this thesis, is a relevant research question. Failure is just something to cope with and for an algorithm being able to critically analyze the results and recognize unreliable estimates is just as important as giving an accurate one. Our philosophy to tackle the confidence estimation problem can be divided into two steps: first, in the analysis step, we try to identify what factors particularly affect the success of pose estimation, starting from empirical observations from state-of-the-art datasets. Next, in the synthesis step, we seek for an algorithm that can combine the knowledge obtained from observations, producing a measure of the confidence of the pose.

This thesis is structured as follows: in Chapter 2 we review the mathematical foundations of geometric computer vision, i.e. how to model image formation and how to parametrize the camera, particularly focusing on how to represent its pose. In Chapter 3 the visual localization task is presented, as well as an analysis of the algorithms used to solve it. A typical visual localization pipeline is dissected, analyzing the implementations of its core elements, which are common to most state-of-the-art approaches. We also present

the main modern pipelines used to solve the problem. The pipelines are presented in a comparative way, focusing on the strengths and weaknesses of each of them. In Chapter 3 the Inloc dataset and algorithm are also introduced, as they are later used to benchmark the proposed confidence estimation algorithm. Chapter 4 is dedicated to confidence modelling. First, singular factors affecting the success of the estimate are presented, using images from InLoc to show why they are relevant from a confidence estimation point of view. Next, it is described how these factors can be brought together to obtain a final confidence estimation measure. In Chapter 5 the executed numerical experiments are described, together with the used evaluation metrics. In Chapter 6 the results are then presented and discussed. Finally, in Chapter 7 the results and the main points of the thesis are summarized and an overview of possible future work is given.

2 GEOMETRIC COMPUTER VISION

In this chapter we review the theoretical background behind geometric computer vision, needed to understand the key issues of this thesis. First, we introduce the projective space, which is the mathematical structure used to describe the geometrical properties of images. The description will be short and focused on the results needed later. The purpose is to make the reader understand *why* the abstract concepts of projective spaces capture the geometric essence of computer vision. Next, we will introduce the pinhole camera model, the simplest mathematical model describing image formation. Finally, we will extend this model to describe real world cameras, obtaining a complete framework to mathematically describe image formation, needed to understand later the problem of visual localization.

2.1 Projective space

The space we live in can be modelled as a vector space, called *Euclidean space* $\mathbb{R}^3$. This means the position of each point can be expressed by a vector of three real numbers $[X, Y, Z]^T$, called *Cartesian coordinates*. A picture taken with a camera is a 2D representation of 3D objects. In other words, taking a picture means to *project* points from a 3D space into a 2D space. This 3D-2D double nature of images is not described well by Euclidean spaces. We introduce thus a fundamental tool used in computer vision, the *projective space* [8].

Definition 2.1. Given an Euclidean space $\mathbb{R}^n$, the projective space $\mathbb{P}^n$ is $\mathbb{P}^n \subseteq \mathbb{R}^{n+1} \backslash \{0\}$ equipped with the equivalence relation $\sim$ so that $\mathbf{x} \sim \mathbf{y} \Leftrightarrow \exists \lambda \neq 0 : \mathbf{x} = \lambda \mathbf{y}$[1]. The coordinates of a point in the projective space are called *homogeneous coordinates*.

Let us now dissect this definition to understand its meaning. The first part tells us the dimension of the space, so a point in the 2D-projective space $\mathbb{P}^2$ will have three coordinates and a point in the 3D-projective space $\mathbb{P}^3$ will have four coordinates. Also, the point with all zeros coordinates is not allowed. The second part tells us that points in the projective space are *unique up to scale*, meaning in $\mathbb{P}^2$ the coordinates $[1, 2, 3]^T$ and $[2, 4, 6]^T$ represent the same point. Cartesian coordinates can be converted to homoge-

[1] This actually means that the projective space is a *quotient space*, and we could compactly write $\mathbb{P}^n = \left(\mathbb{R}^{n+1} \backslash \{0\}\right) / \sim$, with $\sim$ defined as before.

neous coordinates and vice versa with equations (2.1) and (2.2), respectively.

$$[x_1, x_2, \ldots, x_n]^T \mapsto [x_1, x_2, \ldots, x_n, 1]^T, \tag{2.1}$$

$$[x_1, x_2, \ldots, x_n, w]^T \mapsto \left[\frac{x_1}{w}, \frac{x_2}{w}, \cdots, \frac{x_n}{w}\right]^T. \tag{2.2}$$

We will now highlight some important geometrical properties of projective spaces, which will help understand later why projective geometry is a suitable model for computer vision. We will describe these properties using $\mathbb{P}^2$ as an example, but the same concepts can be extended to $\mathbb{P}^3$ as well.

First, we analyze the points in the form $[x, y, 0]^T$, with $x \neq 0$ or $y \neq 0$. According to Equation (2.2), this point will map to $\left[\frac{x}{0}, \frac{y}{0}\right]^T = [\infty, \infty]^T$. For this reason, points of the form $[x, y, 0]$ are called *points at infinity*. It can be proved that these points lie on the same line, which is called *line at infinity*. With this new concept the following fundamental result can be proved:

- Parallel lines in $\mathbb{R}^2$ are incident in $\mathbb{P}^2$ and their intersection point is on the line at infinity.

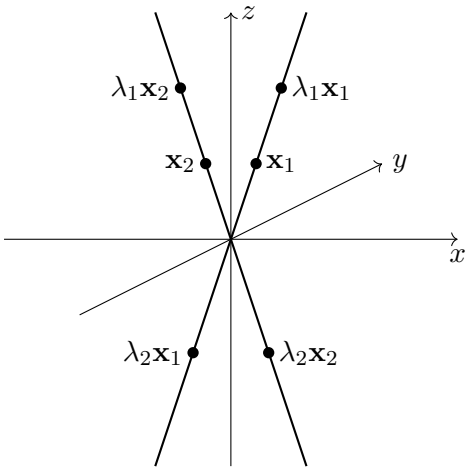


Figure 2.1. *Geometric interpretation of homogeneous coordinates.*

We give now a geometrical interpretation to the equivalence relation in the definition of projective space. Let $\mathbf{x} \in \mathbb{P}^2$, all points in the form $\lambda\mathbf{x},\ \lambda \neq 0$ will correspond to the same point. If we interpret the homogeneous coordinates in $\mathbb{P}^2$ as Cartesian coordinates in $\mathbb{R}^3$, the set of points $\lambda\mathbf{x}$ forms a line going through the origin, as depicted in Figure 2.1. This means that a point in $\mathbb{P}^2$ can be interpreted as a line through the origin in $\mathbb{R}^3$, noting that by definition of projective space the origin of $\mathbb{R}^3$ is excluded. This can be summarized as the following fundamental result:

- There is a one-to-one correspondence between points in $\mathbb{P}^2$ and lines passing through the origin in $\mathbb{R}^3 \backslash \{0\}$.

2.2 Pinhole camera model

Mathematically speaking, taking a picture means mapping points in the 3D space to points in a 2D plane, thus it can be described as an operator $P : \mathbb{R}^3 \to \mathbb{R}^2$. The coordinate system in the 3D space is called *real world coordinate system* and its points *real world points*. Similarly, the 2D plane where the image is formed is called *image plane*, its coordinate system *image coordinate system* and its points *image points*.

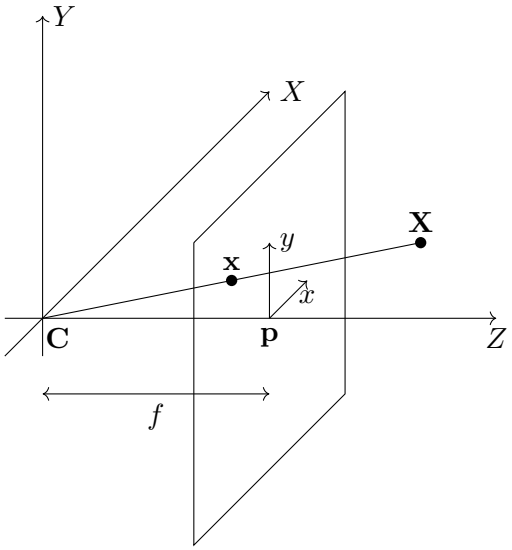


Figure 2.2. *Illustration of the pinhole camera model.*

The simplest model describing this operator is the *pinhole camera model* [15], illustrated in Figure 2.2. To be able to analyze this and further models, some terms need to be defined:

- Camera center $\mathbf{C}$: position of the camera in the real world coordinate system,
- Principal point $\mathbf{p}$: Center of the image plane,
- Focal length f: distance between principal point and camera center,
- Ray: line connecting a real-world point and the camera center,
- Principal axis: ray connecting camera center and principal point.

In this model, a point in the real world coordinate system $\mathbf{X}$ is mapped to a the point $\mathbf{x}$, obtained as the intersection between the image plane and the ray passing through $\mathbf{X}$. If the image plane were infinite, all points in the 3D space could be mapped to the image plane. However, cameras have a limited aperture, meaning the image plane will be finite.

The pinhole camera model models a camera under several strict assumptions. These assumptions can be divided into two groups: those regarding how the coordinate systems are defined and those regarding camera internal properties. The first set of assumptions requires that the origin of the real world coordinate system is at the camera center and the origin of the image coordinate system is at the principal point. Furthermore, the X and Y

axes of the real world coordinate system are aligned with the x and y axes of the image coordinate system. The second set of assumptions regards the technical properties of the camera, namely that pixels are squares, the image plane is rectangular and straight lines are not distorted when projected. If these assumptions hold, it can be derived that

$$P : \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \mapsto \begin{bmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \end{bmatrix}. \tag{2.3}$$

Equation (2.3) shows that P is a nonlinear operator. It can be noticed that this mapping introduces some loss of information. First, information about the 3D point depth (i.e. Z-coordinate) is lost. It can also be shown that ratios of lengths, angles between lines and parallelism is not preserved, as shown in Figure 2.3. Several tasks in computer vision consist in retrieving knowledge about 3D structures, such as depth, distances, shapes, from 2D images. Due to the loss of information, these tasks are usually *ill-posed*, meaning that the existence and uniqueness of the solution cannot be guaranteed.

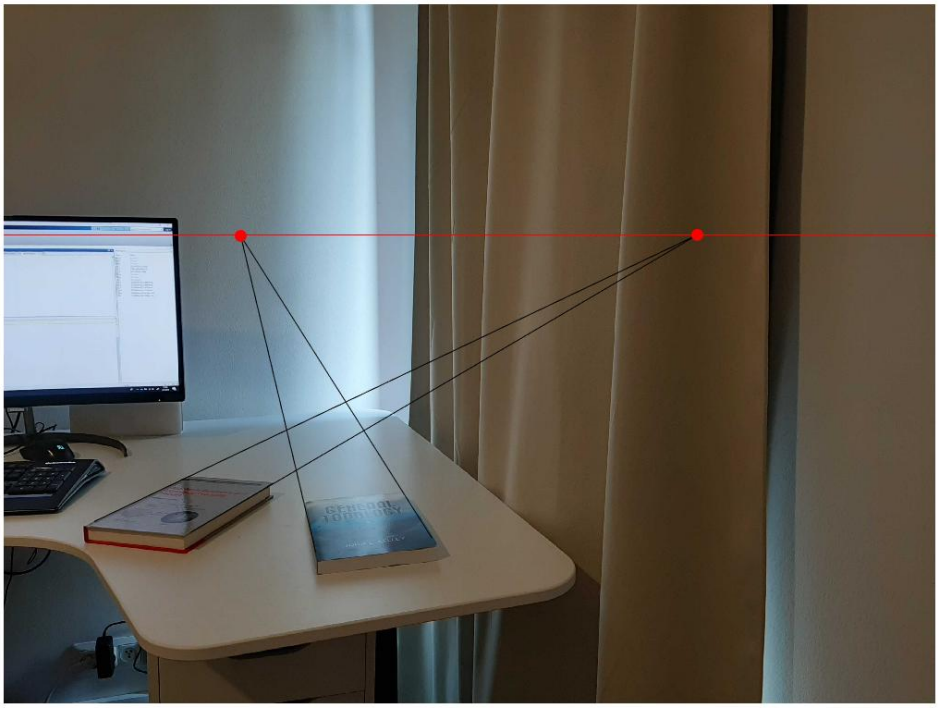


Figure 2.3. *Parallelism is not preserved during 3D-2D mapping.*

From Figures 2.2 and 2.3 two interesting properties can be noticed. First, each point in the image plane lies on only one ray, meaning there is a one-to-one correspondence between 2D points and 3D lines passing through the camera center. Second, parallel lines in the Euclidean space intersect each others when projected into the image plane. Fur-

thermore, the intersection points are all on the same line. Both of these properties were already encountered in the previous section, when the projective space was introduced. Based on this, the projective space seems to capture those geometrical properties that could not be formalized in Euclidean space. Rewriting the operator as a mapping between projective spaces, $P : \mathbb{P}^3 \to \mathbb{P}^2$ Equation (2.3) becomes

$$P : \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix}. \tag{2.4}$$

Equation (2.4) is indeed correct because points in homogeneous coordinates are unique up to scale, hence $[fX, fY, Z]^T \sim \left[\frac{fX}{Z}, \frac{fY}{Z}, 1\right]^T$, as in (2.3). Using homogeneous coordinates leads also to another important result. Let $\mathbf{X} = [X, Y, Z, 1]^T$ and $\mathbf{x} = [x, y, w]^T$ corresponding 3D and 2D points in homogeneous coordinates, now (2.4) becomes

$$\mathbf{x} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{=P} \mathbf{X} = P\mathbf{X}, \tag{2.5}$$

where the matrix P is called the *projection matrix*. Equation (2.5) shows that when using homogeneous coordinates the operator P is actually linear and can be defined as a matrix. This is a fundamental result, which allows us to exploit linear algebra tools in numerical applications. Problems involving linear operators are also numerically simpler to solve. Due to their advantages and wide use in computer vision, in this thesis we will assume all points are in homogeneous coordinates, unless explicitly stated differently. The most important exception being the camera center $\mathbf{C}$, which will always be assumed to be in euclidean coordinates.

2.3 Real camera model

So far we have analyzed the idealized pinhole camera model. As seen, it requires several strict assumptions, which hardly ever hold in real-life applications. In this section we will investigate how our model changes when these assumptions do not hold, until reaching a more general *real camera model*, which replaces the pinhole camera model in real life applications.

2.3.1 Intrinsic parameters

Intrinsic parameters describe camera internal properties. A first assumption says that pixels are square. Pixels shape needs to be taken into account because pixels are the unit of measure in image points coordinates. Practically this means that the focal length in (2.5), whose unit is naturally meters, needs to be converted to pixels. If the pixel is not square, the focal length will scale differently in horizontal and vertical directions, leading to two different focal lengths $f_x = \alpha_x f$ and $f_y = \alpha_y f$, where α_x and α_y are respectively the reciprocals of pixel width and height in meters.

Another assumption was the origin of the image coordinate system being on the principal point. As pixel indexing generally starts either from the left-most upper corner, with the y-axis pointing down or from the left-most lower corner, with the y-axis pointing up, this assumption does not hold. This means the principal point will not be at the origin of the image coordinate system, but it will have an offset (p_x, p_y), i.e. the projected point has to be translated by the offset to have the right coordinates. With these observations Equation (2.5) becomes

$$\mathbf{x} = \begin{bmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X}. \tag{2.6}$$

The last assumptions were the image plane being rectangular, meaning the image coordinate system is orthogonal, and straight lines being preserved when projected. If these do not hold, the model can be further generalized taking into account camera *skew* [13] and *lens distortion* [11]. For the case studies considered in this thesis, however, we can assume those assumptions to hold and hence we will not discuss them any further. Finally, the process of determining camera intrinsic parameters is known as *camera calibration* [16]. A camera is said to be *calibrated* if its intrinsic parameters are known.

2.3.2 Extrinsic parameters

Extrinsic parameters describe camera position and orientation, i.e. the camera pose, with respect to the real world coordinate system. In the pinhole model, it was assumed that real world and image coordinate systems were aligned and that the origin of the real world coordinate system was in the camera center. As the real world reference system a fixed inertial frame is generally used, breaking this assumption. To model this, let us define the *camera coordinate system* as the reference frame obeying the pinhole model, i.e. with origin on $\mathbf{C}$ and axes aligned with image axes, as shown in Figure 2.4. Given a real world point $\mathbf{X}$, a change of coordinates has to be performed before applying equation (2.6), i.e. we must first convert $\mathbf{X}$ to $\mathbf{X}_C$ whose coordinates are expressed in the camera coordinate system

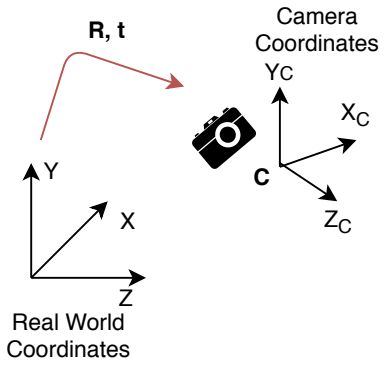


Figure 2.4. *Camera general pose.*

The camera coordinate system frame can be obtained from the original frame with a rotation to align the axes and a translation to align the origins. Thus, to obtain the point $\mathbf{X}_C$ in camera coordinates, we need to apply the euclidean transformation

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}, \tag{2.7}$$

where R is a rotation matrix and $\mathbf{t}$ is a translation vector. Equation (2.7) can be written in homogeneous coordinates as

$$\mathbf{X}_C = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^{3\times 1} & 1 \end{bmatrix} \mathbf{X}. \tag{2.8}$$

The rotation matrix R aligns the axes of the coordinate frames. Intuitively, axes alignment can be done aligning one axis at a time, i.e. by a combination of three rotations around axes X Y and Z of the real world coordinate systems with angles α, β and γ respectively. The rotation matrices around the axes of a given angle are defined as

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}, \quad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \quad R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.9}$$

where the notation $R_e(\theta)$ indicates a rotation around axis e of angle θ. The total rotation matrix can thus be written as

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma), \tag{2.10}$$

meaning we will first rotate around the z-axis, then around the y-axis and finally around the x-axis. Equation (2.10) reveals that the rotation matrix, despite having nine elements, has only three degrees of freedom, represented by the angles α, β and γ, which are called *Euler angles* [15].

Furthermore, it can be proved that a combination of rotations is still a rotation, meaning

that the rotation matrix R describes also a single rotation around the axis $\mathbf{e}$ of angle θ, i.e. the frames can be aligned with a single rotation. The existence of the rotation axis $\mathbf{e}$ implies

$$R\mathbf{e} = \mathbf{e}, \tag{2.11}$$

because the rotation axis does not change during the rotation. Equation (2.11) gives a way to compute the rotation axis, as it says $\mathbf{e}$ is the eigenvector of R corresponding to the eigenvalue $\lambda = 1$. The existence of $\mathbf{e}$ also implies we can construct a frame in which R is in one of the forms in equation (2.9), which leads to equation (2.12) to compute the rotation angle

$$\theta = \arccos\left(\frac{\mathrm{tr}(R) - 1}{2}\right). \tag{2.12}$$

Vice versa, given the rotation axis $\mathbf{e}$ and angle θ the rotation matrix can be computed with equation (2.13), which is called Rodrigues rotation formula

$$R = I + \sin\theta \begin{bmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix} + (1 - \cos\theta) \begin{bmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix}^2. \tag{2.13}$$

Based on the analysis so far, a 3D rotation can be described by a 3x3-matrix, by three euler angles or by an axis and an angle. This last result implies that rotations can be represented as a quadruple. Given the axis $\mathbf{e}$ and angle θ we define the associated quadruple

$$\mathbf{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & e_x \sin\left(\frac{\theta}{2}\right) & e_y \sin\left(\frac{\theta}{2}\right) & e_z \sin\left(\frac{\theta}{2}\right) \end{bmatrix}^T. \tag{2.14}$$

The quadruple in equation (2.14) is called *quaternion* [28]. The analysis of algebraic properties of quaternions is beyond the scope of this thesis. For our purposes, it is enough to define the norm of a quaternion as in equation (2.15)

$$\|\mathbf{q}\| = \sqrt{a^2 + b^2 + c^2 + d^2}, \tag{2.15}$$

where a, b, c, d are the components of the quaternion $\mathbf{q}$.

The axes being aligned, we want next to bring the origin of the rotated frame to the camera center. Let $\mathbf{C}$ the camera center euclidean coordinates in the real world coordinate system. After the rotation the new coordinates will be $R\mathbf{C}$. To bring this point to the origin we need to apply a translation with vector

$$\mathbf{t} = -R\mathbf{C}. \tag{2.16}$$

The vector in equation (2.16) is in euclidean coordinates and it has three degrees of freedom. It can be concluded that the camera pose has six degrees of freedom, three arising from the rotation matrix and three arising from the translation vector. This result is in accordance with classical mechanics, which says a rigid body in the 3D euclidean

space has six degrees of freedom in total. Finally, equations (2.6) and (2.8) can be combined obtaining

$$\begin{aligned}\mathbf{x} &= \begin{bmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^{3\times 1} & 1 \end{bmatrix} \mathbf{X} = \underbrace{\begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{:=K} \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \mathbf{X} \\ &= K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \mathbf{X} \\ &= K \begin{bmatrix} R & -R\mathbf{C} \end{bmatrix} \mathbf{X}. \end{aligned} \tag{2.17}$$

Equation (2.17) gives the real camera model. The matrix K models camera intrinsic parameters and hence it is called *intrinsic matrix* and matrix $\begin{bmatrix} R & \mathbf{t} \end{bmatrix}$ models camera extrinsic parameters, i.e. position and orientation and it is called *camera pose*.

3 IMAGE-BASED LOCALIZATION

So far we have developed a mathematical model to describe how a camera transforms 3D coordinates into 2D coordinates. This framework helps us understand the problem of *image-based localization* also known as *camera pose estimation* or *visual localization*. Formally, the visual localization problem can be expressed as follows *Given a query image, what was the pose of the camera?*, i.e. *what was its position and orientation, when the image was taken?* A visualization of this question is given in Figure 3.1: a query image is taken with a smartphone and the algorithm infers the position and orientation of the camera and is able to locate it in a map of the environment (indicated with a red pyramid).

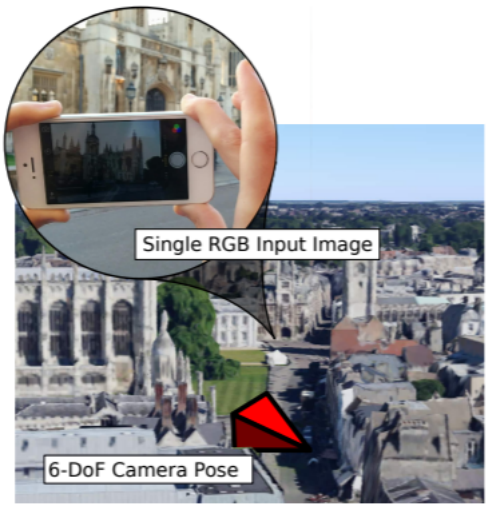


Figure 3.1. *Visualization of the pose estimation problem [19].*

Visual localization is a calibrated problem, meaning that the intrinsic parameters of the cameras used are known. Furthermore, visual localization algorithms require a "map" of the environment where the query image was taken. This map can be either a database of images for which the original 3D coordinates of most of the pixels are known or a 3D dense model of the environment. Further information about how these can be constructed can be found e.g. in [53] and [37]. In a traditional visual localization pipeline two main steps can be identified: formation of correspondences between 2D and 3D coordinates and pose estimation from points correspondences. In this chapter we first analyze these two phases, focusing on the methodologies used. Next, we present the structure of some typical state of the art approaches. Finally, we analyze more in details InLoc, a pipeline

for indoor visual localization, which will serve as case study in this thesis.

3.1 Pose Estimation from points correspondences

Let the correspondence $(\mathbf{x}, \mathbf{X})$ and intrinsic matrix K. Equation (2.17) can be simplified introducing the normalized coordinates corresponding to $\mathbf{x}$, defined as $\mathbf{u} = K^{-1}\mathbf{x} = [u, v, w]^T$. Thus, the correspondence equation becomes

$$\lambda \mathbf{u} = \begin{bmatrix} R & t \end{bmatrix} \mathbf{X}, \tag{3.1}$$

where the parameter $\lambda \neq 0$ was introduced because homogeneous coordinates are unique up to scale, meaning a pose satisfying the mapping $\mathbf{X} \mapsto \lambda\mathbf{u}$ is still a solution of the original problem. First, we investigate what is the smallest number of correspondences needed for the problem to have a finite number of solutions. Equation (3.1) can be written as

$$\begin{aligned} \lambda \begin{bmatrix} u \\ v \end{bmatrix} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\ \Leftrightarrow \begin{cases} \lambda u = r_{11}X + r_{12}Y + r_{13}Z + t_x \\ \lambda v = r_{21}X + r_{22}Y + r_{23}Z + t_y \\ \lambda w = r_{31}X + r_{32}Y + r_{33}Z + t_z \end{cases} &, \end{aligned} \tag{3.2}$$

we can now divide the first and second equation by the third one in (3.2) obtaining

$$\begin{cases} \frac{u}{w} = \frac{r_{11}X + r_{12}Y + r_{13}Z + t_x}{r_{31}X + r_{32}Y + r_{33}Z + t_z} \\ \frac{v}{w} = \frac{r_{21}X + r_{22}Y + r_{23}Z + t_y}{r_{31}X + r_{32}Y + r_{33}Z + t_z} \end{cases} . \tag{3.3}$$

Equation (3.3) shows that for each point correspondence we get two independent nonlinear equations. It was shown in the previous chapter that the camera pose has six degrees of freedom, hence at least three point correspondences are needed to solve the system in Equation (3.3).

3.1.1 Perspective-n-Point

The family of algorithms used to estimate the camera pose from n point correspondences are referred as *Perspective-n-Point* solvers (PnP), where the parameter n indicates the number of correspondences used. Based on the previous discussion, $n \geq 3$ is needed for the system to be determined. Next, we investigate the uniqueness of the solution. This is

addressed exploiting Gao's formulation of the P3P problem [54], which is widely used in several software implementations.

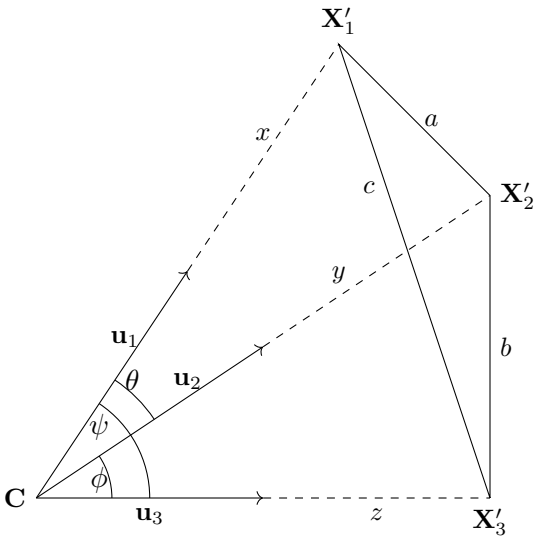


Figure 3.2. *Geometrical formulation of the P3P.*

The setup for Gao's solver is shown in Figure 3.2. Let the correspondences $(\mathbf{u}_1, \mathbf{X}'_1)$, $(\mathbf{u}_2, \mathbf{X}'_2)$ and $(\mathbf{u}_3, \mathbf{X}'_3)$ between image points in normalized homogeneous coordinates and real world points in Euclidean coordinates. By the law of cosines we obtain

$$\begin{cases} a^2 = x^2 + y^2 - 2xy\cos\theta \\ b^2 = y^2 + z^2 - 2yz\cos\phi \\ c^2 = x^2 + z^2 - 2xz\cos\psi \end{cases}, \tag{3.4}$$

where $x = \|\mathbf{X}_1 - \mathbf{C}\|$, $y = \|\mathbf{X}_2 - \mathbf{C}\|$ and $z = \|\mathbf{X}_1 - \mathbf{C}\|$. The cosines can be computed from the 2D points coordinates, e.g. $\cos\theta = \frac{\mathbf{u}_1 \cdot \mathbf{u}_2}{\|\mathbf{u}_1\| \cdot \|\mathbf{u}_2\|}$. The system (3.4) is the formulation of P3P algorithm used in Gao's solver, several algebraic techniques have been proposed to efficiently solve the system [56]. Once x, y and z have been solved, the camera position $\mathbf{C}$ can be found by trilateration, that is computing the intersection of the spheres centered on $\mathbf{X}'_1$, $\mathbf{X}'_2$ and $\mathbf{X}'_3$ and with radii x, y and z respectively. From Figure 3.2, it is clear that this intersection is $\mathbf{C}$. Next, equations (3.1) and (2.16) can be combined obtaining

$$\lambda_i \mathbf{u}_i = R(\mathbf{X}'_i - \mathbf{C}_i), \tag{3.5}$$

from which follows

$$|\lambda_i| \|\mathbf{u}_i\| = \|\mathbf{X}'_i - \mathbf{C}\|, \tag{3.6}$$

The value of λ can now be found from equation (3.6). We can safely drop the absolute value and assume λ is positive, as the sign can be corrected later. Finally, the rotation matrix can be computed from (3.5) – first we find the rotation axis $\mathbf{e} = \frac{\mathbf{u}_i \times (\mathbf{X}'_i - \mathbf{C})}{\|\mathbf{u}_i \times (\mathbf{X}'_i - \mathbf{C})\|}$ and the rotation angle θ as the angle between $\mathbf{u}_i$ and $(\mathbf{X'}_i - \mathbf{C})$. The rotation axis and angle can be determined by any of the three points correspondences. Finally, the Rodrigues

formula can be applied to obtain R. It should be noticed that for the obtained matrix holds $\det(R) = \pm 1$. The rotation matrix, however, requires $\det(R) = 1$, hence if $\det(R) = -1$, R and λ should be corrected changing their sign.

The system (3.4) has degree eight, meaning that it admits up to eight distinct solutions. The amount of solutions can be reduced introducing physical constraints. It can be proved that four of these solutions place the camera center on the wrong side of the image plane. After discarding physical unacceptable candidates the P3P leads to four possible camera poses. Further constraints can be generated requiring invariant properties of projective geometry to be preserved, leading finally to a unique solution. Alternatively, the ambiguity in the P3P algorithm can be removed introducing further point correspondences. In the P4P algorithm, for example, the solving system is derived from four points correspondences. This leads to a unique solution if the four 3D points lie on the same plane, otherwise it leads to five feasible solutions. The P6P algorithm leads to a unique solution, moreover P6P algorithm allows to solve the elements of the rotation matrix and translation vector directly from (3.2), making the problem linear.

In general, introducing further correspondences reduces the degree of the solving equation, reducing ambiguity. Also, a higher number of correspondences produces faster solvers. Despite maybe sounding controversial at the beginning, introducing more points correspondences produces more geometrical knowledge about the system, which can be exploited to reduce the final degree of the solving equation; for example, the P3P solver led to a system of degree eight, whereas in the P6P solver the solving system is linear. Furthermore, as more points correspondences reduce ambiguity, less postprocessing is needed to identify the correct solution. However, it should be beared in mind that our discussion so far assumed the points coordinates and the correspondences to be exact. In practical applications this does not hold. First, due to noise points coordinates themselves contain some uncertainty, hence the coordinates cannot be assumed to be exact. Second and more important, as we will discuss in the next section, forming points correspondences is the most challenging part in visual localization pipelines and a significant amount of wrong matches is introduced. Feeding bad matches to a PnP solver leads to absurd results. As discussed in the next subsection, solvers are generally iterated several times to deal with wrong matches and using a smaller number of correspondences increases the probability to pick only good matches at least once. The more correspondences are used, the smaller this probability and consequently, despite being the inner solver faster, the number of iterations needed increases, resulting in an overall slower algorithm.

3.1.2 RANSAC algorithm

The main problem with PnP solvers is their sensibility to input data. A small error in points correspondences can lead to a completely different camera pose. State-of-the-art approaches still introduce a significant amount of wrong correspondences, which are

called *outliers*. Consequently, pose estimation algorithms need to be robust to outliers. A popular choice to estimate parameters with outliers is the *Random Sample Consensus* (RANSAC) algorithm [10]. Given N correspondences $(\mathbf{u}, \mathbf{X})$, we randomly pick k of them and estimate the camera pose $\hat{P}$. Next, we compute the number of inliers, i.e the number of correspondences for which $d(\mathbf{u}, \hat{P}\mathbf{X}) \leq t$, where $d(\cdot, \cdot)$ and t are chosen distance measure and threshold. The process is iterated and at the end the model with the highest number of inliers is outputted. The parameters to be adjusted are the distance function, the threshold and the maximum number of iterations. In image-based localization applications a popular choice for the distance measure is the angular distance $d(\mathbf{u}, \hat{P}\mathbf{X}) = \angle(\mathbf{u}, \hat{P}\mathbf{X})$ with a threshold of a few degrees, e.g. $t = 1°$. The maximum number of iterations is chosen so that at least once the k randomly picked correspondences contain only inliers with probability p (e.g. $p = 0.99$). Let e the outliers ratio, the number of maximum iterations satisfying this property can be computed as $N_{max} = \frac{\log(1-p)}{\log(1-(1-e)^k)}$. In practical applications the outlier ratio is not known, hence it is estimated and updated during the RANSAC algorithm. The pseudocode of the RANSAC algorithm with P3P solver is shown in Figure 3.3.

Input: $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n\}$, $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n\}$, t, p
Result: estimated pose P, number of inliers $inls$
$N = \infty$ $i = 0$, $P_{best} = []$, $maxInls = 0$;
$e = 0.5$; /* or estimated ratio of outliers*/
while $N \geq i$ **do**
  randomly pick $(\mathbf{X}_1, \mathbf{x}_1)$, $(\mathbf{X}_2, \mathbf{x}_2)$, $(\mathbf{X}_3, \mathbf{x}_3)$;
  $\hat{P}$ =p3p_solver($(\mathbf{X}_1, \mathbf{x}_1)$, $(\mathbf{X}_2, \mathbf{x}_2)$, $(\mathbf{X}_3, \mathbf{x}_3)$);
  $inls = |\{\mathbf{X}_i | d(\mathbf{x}_i, \hat{P}\mathbf{X}_i) \leq t\}$;
  $e = 1 - inls/n$;
  $N = \frac{\log(1-p)}{\log(1-(1-e)^k)}$;
  $i += 1$;
  if $inls \geq maxInls$ **then**
    $P_{best} = \hat{P}$;
    $maxInls = inls$;
  end
end

Figure 3.3. *Pseudocode of RANSAC algorithm.*

During the years several modifications of the basic RANSAC algorithm have been proposed to improve its performances [33]. A popular variation used in visual localization is the *locally optimized-RANSAC* (LO-RANSAC) [6], which introduces a smaller inner RANSAC loop. Once a pose hypothesis and its inliers are computed (line 8 in Figure 3.3), a new RANSAC is started, sampling now only from the current set of inliers. Clearly, this reduces the probability of picking an outlier, hence in the inner RANSAC it is not strictly necessary to use the minimum amount of samples. The new hypothesis and the number of inliers are computed and the process is then iterated, keeping as best model the one with the highest number of inliers. Differently from the outer loop, the inner loop is generally iterated a fixed amount of times. At the end, the best model hypothesis, together with the new set of inliers are outputted from the inner loop, substituting the

original ones computed in the outer loop.

3.2 Forming points correspondences

So far we have discussed how to estimate the camera pose given 2D-3D correspondences, in this section we will analyze the problem of how to form these correspondences. Suppose we have a picture and a dense 3D model of the environment where the picture was taken, a point in the image will correspond to a point in the 3D model if they correspond to the same point in the real world. Ideally, each point in the image corresponds to exactly one point in the 3D model. In practical applications, however, this is not possible, nor useful. First of all, a query image taken with a common smartphone can easily have millions of pixels, which would make matching every point extremely time consuming. Second, some zones of the image carry almost no information and are not useful for finding correspondences, for examples pixels in a flat homogeneous zone. Based on this, we first want to identify *good points* from the image and then find their corresponding 3D coordinates. Once the set of good points has been identified, a method to compare points has to be defined. Intuitively, corresponding points will represent the same object, such as the same corner of the same monitor, or the same eye of the same person. Coordinates however only tell where the points are but have no information about their meaning. For this reason we need to create a semantical representation of the points, in which information about the content of the image is embedded. Practically, this means that a function $f : \mathbb{P}^n \to \mathbb{R}^k$ has to be designed. This function, called *descriptor*, associates to a given point a k-ary vector, called *feature vector* which models the semantic meaning of the point. As points are adimensional, descriptors generally consider the neighbourhood of a point. Finally, finding the corresponding point means to find the *most similar* vector, i.e. a distance measure to compare feature vectors is needed. Based on this discussion, the correspondences forming problem can be formulated as a Nearest Neighbour (NN) problem [7], as in Equation (3.7).

$$\mathbf{X} = \underset{\mathbf{X}_i}{\operatorname{argmin}}\, d(f(\mathbf{x}), g(\mathbf{X}_i)), \tag{3.7}$$

where $\mathbf{x}$ and $\mathbf{X}$ are corresponding 2D and 3D points, f and g are descriptors and $d(\cdot, \cdot)$ is a distance function. As feature vectors are vectors in the Euclidean space, a popular distance function is the Euclidean distance.

The described approach, despite being correct and giving state-of-the-art accuracies, is not very efficient, as 3D models are generally dense and hence have billions of points. Despite efficient search methods have been developed, solving the NN problem is still too time consuming, especially if the algorithm is required to run real time on a mobile device. For this reason an alternative approach is to perform 2D-2D matching. The dense model is replaced by a database of images for which the 3D coordinates of the pixels are known. For a given query image, the most similar database images are retrieved and

points correspondences are formed between pixels in the query and database images. The NN problem in (3.7) is hence modified as in (3.8)

$$\mathbf{x}_{db} = \underset{\mathbf{x}_i}{\operatorname{argmin}}\ d(f(\mathbf{x}_q), f(\mathbf{x}_{db})), \tag{3.8}$$

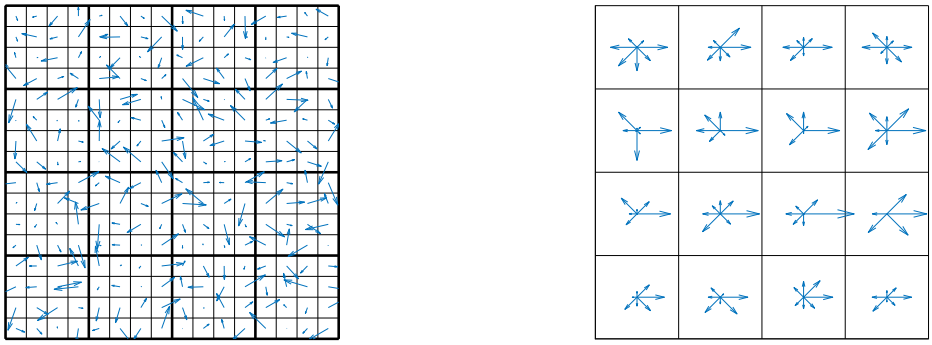
where $\mathbf{x}_q$ and $\mathbf{x}_{db}$ are now points in the query and database image, respectively. Since we are now comparing two 2D images, the same descriptor f is generally used for both. As this approach has become far more popular and can be also applied to a several range to tasks, beyond visual localization, we will describe here some popular descriptors and distance functions for 2D-2D matching. A brief overview of descriptors for 2D-3D matching will given in 3.3.1 where an example of direct matching pipeline is analyzed. Here we assume that relevant database images have already been retrieved, how the retrieval process works is discussed in 3.3.2.

3.2.1 Traditional descriptors

In order to solve the NN problem of Equation (3.8) first the interest points need to be found and then the descriptor f as well as the distance measure d need to be defined. Ideally, $f(\mathbf{x}_1) = f(\mathbf{x}_2)$ if and only if $\mathbf{x}_1$ and $\mathbf{x}_2$ correspond to each others. The semantic content of the image is taken into account by observing the neighbourhood of the pixels, which allows to capture local properties of points. Particularly, relevant semantic information can be detected by computing the gradients of the pixel. Suppose to have a flat black zone in the image, all pixels in that area will be indistinguishable from each others and hence they will not be good feature points. Mathematically, these pixels have zero directional derivatives in each direction. On the other hand, pixels with accentuated changes in their neighbourhoods will contain more information and hence work nicely as feature points. Interest points can thus be found by looking for pixels with strong intensity gradients. Based on this analysis, the following problem arises: the interest points detector depends on the scale of the image, as zooming closer would attenuate the gradients. To overcome this problem, the search is done at different resolutions, i.e. consecutively downsampling the image and performing the search at each resolution level [4]. A common algorithm to detect interest points is the Harris corner detector and its implementation details can be found in [47].

Once the interest points have been detected, a meaningful descriptor has to be applied to it. A popular choice is the Scale-Invariant Feature Transform (SIFT) descriptor [29]. Similarly to features detection, SIFT descriptor relies on the assumption that gradients can be used to model semantic meaning. The SIFT descriptor can be computed as follows: for a given feature point, a 16x16 neighbourhood around it is formed. This neighbourhood is then divided into sixteen 4x4 smaller cells. In each cell, the gradient for each pixel is computed. We then consider the eight main directions, spaced by 45°, i.e. similarly to the directions of a compass. Each computed gradient is then aligned to its closest main

direction and aligned gradients are summed, obtaining eight gradients for each cell, as depicted in Figure 3.4. The magnitudes of these form an 8-ary array called *histogram of oriented gradients* for each cell. Each so obtained array is circularly shifted so that the strongest gradient is the first element of the array. As the original neighbourhood was divided into sixteen cells and eight gradients were computed in each cell, the final result is a 128-ary array, which is the final descriptor associated with the considered interest point. The SIFT descriptor is translation invariant, as gradients are computed and scale invariant, as interest points were detected at different resolutions. Furthermore, circularly shifting the arrays in each cell introduces also rotation invariance.



(a) *Original gradients.* ***(b)*** *Clustered gradients.*

Figure 3.4. *Visualization of SIFT descriptor.*

Once SIFT descriptors have been computed for both query and database images, a distance measure has to be defined to match points from query and database images. As SIFT descriptor is a vector in the $\mathbb{R}^{128}$ space, vectors can be compared by simply using the euclidean distance as distance measure. It can be proved, however, that better performances are achieved using the Hellinger distance defined as

$$H(\mathbf{v}, \mathbf{u}) = \|\sqrt{\mathbf{v}} - \sqrt{\mathbf{u}}\|_2, \tag{3.9}$$

where $\sqrt{\mathbf{v}}$ indicates the element-wise square root of the array. The popularity of Hellinger distance led to a modified version of SIFT, known as rootSIFT, which is obtained as follows: given a SIF descriptor $\mathbf{v}$, it is first L1-normalized, i.e. each element is divided by the sum of the absolute values of the elements, then the square root of each element is taken and finally the so obtained array is normalized by its euclidean norm. The so obtained rootSIFT descriptors can then be compared by the normal Euclidean distance, i.e. rootSIFT descriptors with Euclidean distance is equivalent to using SIFT descriptors with Hellinger distance in (3.9) [3]. We have now a recipe to solve the NN problem in (3.8), once all descriptors have been computed they can be stored in appropriate data structures for efficient search algorithms, such as kd trees [43].

The presence of redundancy and noise in both database and query images can lead to wrong matches when applying the NN algorithm. Particularly, if several similar descriptors

are found, the nearest is not necessarily the correct one. To improve the robustness of the algorithm, given a feature vector in the query image $\mathbf{v}$, the two closest neighbours $\mathbf{u}_1$ and $\mathbf{u}_2$ are retrieved from the database image, with $d(\mathbf{v}, \mathbf{u}_1) < d(\mathbf{v}, \mathbf{u}_2)$, i.e. $\mathbf{u}_1$ is the closest match. Next, the correspondence $(\mathbf{v}, \mathbf{u}_1)$ is accepted only if

$$\frac{d(\mathbf{v}, \mathbf{u}_1)}{d(\mathbf{v}, \mathbf{u}_2)} < \gamma, \tag{3.10}$$

where γ is a predetermined threshold, generally around 0.7 or 0.8. Equation (3.10) is known as *Lowe's ratio test* [30] at it makes sure that only the most confident inliers are kept.

Despite Lowe's ratio test improves the robustness, a relatively high number of wrong matches is generally still returned. A further improvement is to use a verification step, given a set of points $\{\mathbf{x}_{q1}, \mathbf{x}_{q2}, \dots, \mathbf{x}_{qN}\}$ from the query image and the corresponding points $\{\mathbf{x}_{db1}, \mathbf{x}_{db2}, \dots, \mathbf{x}_{dbN}\}$ from the database image, all in homogeneous coordinates, we try to fit a transformation $H : \mathbf{x}_{qi} \mapsto \mathbf{x}_{dbi}, \quad \lambda \mathbf{x}_{dbi} = H\mathbf{x}_{qi}$. The transformation H between projective planes is called *homography* and it can be computed with algorithms similar to pose estimation within a RANSAC loop [15]. Once the best homography is computed, the inliers outputted from RANSAC form the final set of correspondences.

3.2.2 Deep learning based descriptors

The approaches described so far for points matching were human-reasoning based, meaning first the algorithm designer fixes what properties a good interest point and a good descriptor should have and then derives a mathematical formulation to compute them. Feature vectors have thus a physical interpretation, understandable to humans. For example, the SIFT descriptor extracts the gradients in a neighbourhood of an interest point, because gradients characterize the shape of the neighbourhood and hence capture its semantical meaning. Deep learning methods, on the other hand, let the computer "learn" by itself what a good descriptor should look like and how to compute it. This is implemented exploiting neural networks, which can be regarded as black boxes, taking images as input and outputting interest points and the corresponding feature vectors. Opposed to traditional methods, a human-understandable explanation of *why* those are considered good points and why descriptors are computed that way cannot generally be given. The interest for neural networks has exponentially risen in the past decade, reaching state-of-the-art solutions for several artificial intelligence problems, also beyond computer vision.

In the context of computer vision, and image processing in general, Convolutional Neural Networks (CNN) are particularly attractive [42], as the mathematical definition of 2D convolution allows to capture local and global properties of images. The constitutive unit of a convolutional neural network is generally composed by three layers: convolutional, activation and pooling. The convolutional layer consists of a set of K 2-dimensional filters,

each applied to the input image. If the input size is $H \times W \times 3$ (hence a RGB image), the output will have size $H \times W \times K$, such multidimensional structures are generally called *tensors*. The activation step applies a nonlinear function to each element of the previous tensor. Popular activation functions are, for example, $\arctanh$ or the Rectified Linear Unit (ReLU) $ReLU(x) = \max(0, x)$, the activation layer does not change the size of the tensor. The pooling layer performs downsampling in the first two dimensions, for example keeping every other element, outputting a tensor of size $H/2 \times W/2 \times K$. A CNN is obtained stacking several times the described units. The tensors outputted after each unit are called *feature maps* and in visual localization pipelines they are used to perform points matching. Before it can be used, the network needs to *learn* to produce relevant feature maps. Practically, this means that the coefficients of the filters in the convolutional layers need to be determined. To train a neural network, the last feature map is first flattened into a N-ary array which is, possibly after some intermediate layers, mapped to a scalar through a function $\mathcal{L}$, called *loss function*. The parameters of the networks are determined minimizing the loss function. Designing and training neural networks is a field of its own, for our purpose it is enough to understand the basic structure of a CNN and how it relates to visual localization. There is plenty of literature about neural networks and further information about their structure and training algorithms can be found e.g. in [14].

In the context of features matching, the feature map of size $H \times W \times K$ can be interpreted as a set of $H \cdot K$ interest points, each associated with a K-ary feature vector. Matching can now be done with a NN algorithm, i.e. for each feature vector in the query feature map the most similar is found from the database image feature map. Points matching with feature maps is generally referred as *dense matching*. As discussed, several feature maps are produced by the intermediate layers of a CNN. As they are consequently downsampled, they become sparser and sparser as they propagate through the network. This rises the question *What feature map should be used?* Using a too sparse feature map will weaken the accuracy of interest points coordinates and using a very fine feature map will, on the other side, result into a significantly time consuming process, unacceptable for e.g., real-time applications. To solve this, *coarse-to-fine matching* is used, where two feature maps of sizes $H_1 \times W_1 \times K_1$ and $H_2 \times W_2 \times K_2$ are used, with $K_1 > K_2$ and $W_1 > W_2$, i.e. the first one is the fine feature map and the second one is coarse feature map. Point matching is first done with the coarse feature maps. Once the correspondences are found, the corresponding points in the fine maps are taken and the matching accuracy is improved performing NN search in their neighbourhoods.

As a final remark, deep learning based descriptors generally produce more robust correspondences compared to traditional descriptors. Nevertheless, the final result may still and probably will contain several wrong matches, due to the challenging and ill-posed nature of the problem. From an accuracy point of view, forming correspondences is generally the bottle-neck of visual localization pipelines.

3.3 Typical image-based localization pipelines

As shown in the previous section, designing a pose estimation pipeline leaves the programmer some freedom, such as choosing the matching strategy or the kind of the descriptors. To concretize the previous discussion, we present here some popular state of the art pipelines, that have been recently developed to address long-term visual localization.

3.3.1 Direct matching

In direct matching approaches, points correspondences are directly formed between the 2D query image and the 3D model. Several approaches have been proposed to compute descriptors of 3D point clouds. For example, it is possible to generalize the SIFT descriptor to the 3D case considering a volumetric neighbourhood of a given interest point and considering the gradients orientation in the 3D space [41]. However, 3D models are generally computed using *Structure From Motion* (SfM) techniques, where the 3D coordinates of the point cloud are computed from a large database of images [20, 40]. A more popular approach is thus to associate 3D points to corresponding 2D descriptors. Suppose the point $\mathbf{X}$ in the 3D space corresponds to the points $\mathbf{x}_1, \mathbf{x}_2$ in some images of the database used to compute the point cloud. SIFT descriptors $\mathbf{d}_1, \mathbf{d}_2$ can now be computed as described before and the point $\mathbf{X}$ will be associated with the descriptors $\{\mathbf{d}_1, \mathbf{d}_2\}$.

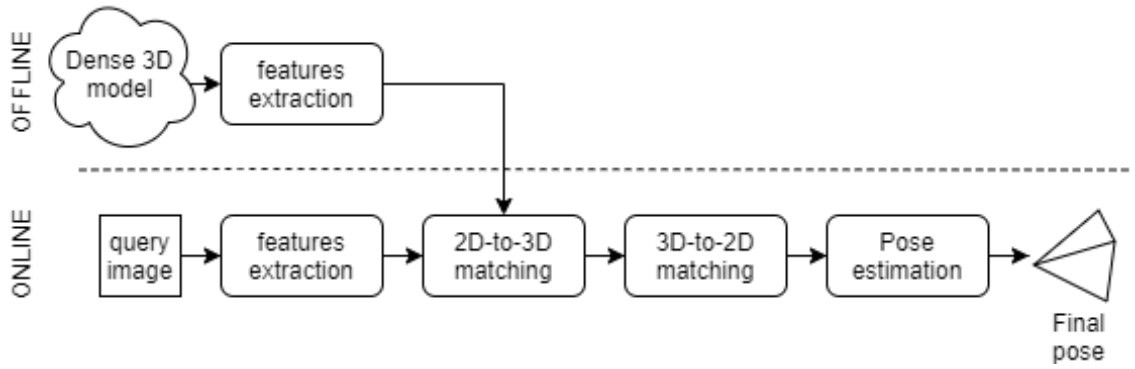


Figure 3.5. *Active search pipeline.*

Theoretically, given the descriptors of the interest points in the query image $\{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$, points correspondences can be formed with a NN-search. However, a point clouds can have millions of points and hence an exhaustive search would be too time consuming. An efficient search strategy, called *active search*, was proposed in [36]. The pipeline of active search is shown in Figure 3.5. The idea is to perform NN-search in a hierarchical way, saving significantly computation time. In the offline phase, descriptors associated with 3D points are computed. These descriptors are then clustered and each cluster

is associated with a *global* descriptor, called *visual word* [45]. For example, if k-means clustering is performed, the mean of each cluster will represent a visual word. In the online phase, given a descriptor in the query image $\mathbf{d}$, the corresponding visual word $\mathbf{D}$ is computed. Next, the most similar descriptor $\mathbf{d}^{3d}$ is retrieved only from those descriptors corresponding to the same visual word $\mathbf{D}$. Iterating this for each interest point in the query image, a set of correspondences $\{(\mathbf{d}_1, \mathbf{d}_1^{3d}), \dots (\mathbf{d}_N, \mathbf{d}_N^{3d}\}$ is obtained.

The correspondences were obtained performing 2D-to-3D matching, meaning interest points were detected from the 2D image and matched to the points in the 3D model. To refine the correspondences, a 3D-to-2D matching is introduced. In this step, descriptors for all pixels in the query image are computed and eacg previously retrieved descriptor from the 3D model $\mathbf{d}^{3d}$ is matched to the most similar descriptor from the query image. Again, this NN-search can be sped up considering only those descriptors in the image belonging to the same visual word of $\mathbf{d}^{3d}$. This 3D-to-2D matching is performed for each 3D-descriptor computed in the 2D-to-3D matching, obtaining the final correspondences $\{(\mathbf{d}'_1, \mathbf{d}_1^{3d}), \dots (\mathbf{d}'_N, \mathbf{d}_N^{3d}\}$. Finally, the camera pose can be computed using PnP-RANSAC, as described before.

Direct matching approaches allow to reach highly accurate camera poses. Particularly, it was shown in [36] that the introduced 3D-to-2D step can improve the accuracy. As a drawback, these approaches are slow and in most cases not suitable for real-time applications.

3.3.2 Retrieval based approach

As stated before, the greatest weakness of direct matching is the high computation time required. Furthermore, in some situations a dense 3D model is not available and the environment is described only by a large database of images. These lead to retrieval based approaches, where the direct 2D-3D matching is replaced by 2D-2D matching. A typical pipeline of retrieval based approaches is shown in Figure 3.6.

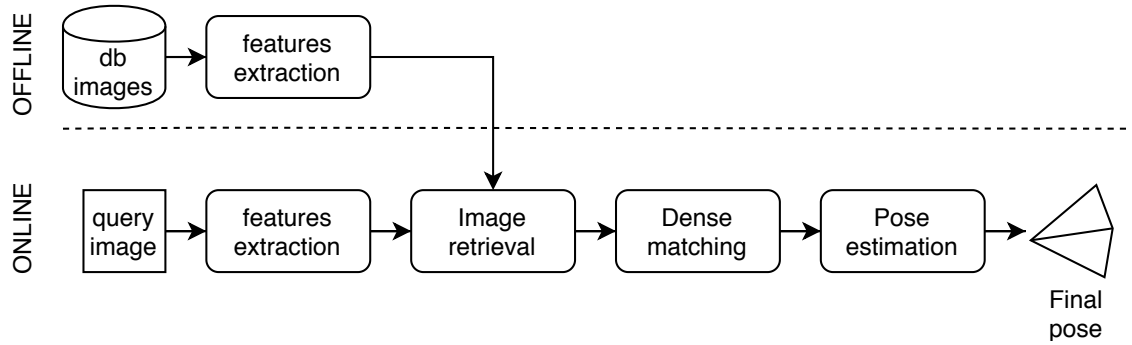


Figure 3.6. *Typical pipeline of image retrieval based localization.*

First, features from the query image are detected and extracted. In retrieval based approaches we have two kinds of features: those for image retrieval and those for points

matching. In image retrieval, given a query image we retrieve the N most similar images from the database. While in points matching we compare neighbourhoods of pixels, in image retrieval whole pictures are compared. For this reasons the features extracted for points matching are generally not directly suitable, as they describe *local* properties of some pixels, whereas in image retrieval we want *global* descriptors, in which information about a wider area is embedded. Similarly to active search, local descriptors can be clustered into visual words. In the context of image retrieval visual words are generally computed as follows: in the offline phase local features, i.e. the same features used for points matching, are computed for all database images. The local features are then clustered and each cluster represents a visual word. Each local feature in the original images can now be mapped onto a visual word, i.e. more features can correspond to the same word. If after clustering m distinct visual words are identified, the feature vector associated to an image I will be $\mathbf{d} = [d_1, d_2, \ldots, d_m]$, where each element d_i tells how many times the ith visual word occurs in the image. To speed up the retrieval process, in the offline phase we compute also a look-up table which, for each visual words, stores in what images the current visual word is present.

In the retrieval step, given a query image I_q, we compute the corresponding global descriptor $\mathbf{d}_q$. Next, for each visual word in I_q the corresponding database images are retrieved from the reversed index, obtaining the relevant database images. Finally, we need to measure how similar the query image I_q and the relevant database images are. A popular metric to measure similarity is the *cosine similarity*, defined as

$$sim(I_q, I_{db}) = \frac{\mathbf{d}_q \cdot \mathbf{d}_{db}}{\|\mathbf{d}_q\| \|\mathbf{d}_{db}\|}, \tag{3.11}$$

where $\mathbf{d}_{db}$ is the feature vector corresponding to a relevant database image I_{db}. It is good to remark that the formula in (3.11) is a similarity measure and not a distance measure, meaning a higher value means a higher similarity. Having computed the similarity between the query image and each relevant database image, the N best matches can be retrieved. The next steps, dense matching and pose estimation, are performed as described in the previous section.

Retrieval based approaches are considered state-of-the-art at the moment. First, they are significantly faster than Direct Matching approaches, being able to meet also real-time applications requirements on mobile devices. Furthermore, recent developments in deep learning for features extraction, as long as more sophisticated post-processing techniques to choose the best candidate pose have allowed retrieval-based approaches to reach promising results in challenging visual localization tasks, such as localizing the camera in a whole city.

3.3.3 Learning based approaches

In the methods analyzed so far neural networks were utilized only to extract features from images and point clouds and then points matching and pose estimation was performed with classical approaches, such as NN and PnP. Motivated by the increasing success of neural networks, learning based approaches replace one or both or the classical algorithms by a neural network. Here we will briefly discuss two cases: *Coordinates regression* approach, where points matching is performed through deep learning and *Pose regression* approach, where the whole end-to-end pipeline is carried out by a neural network.

In coordinate regression methods, instead of matching points from query and database images or from query image and 3D model by NN, a network is trained to directly compute the 3D coordinates of the points from a given query image. Finally, the pose is estimated from the computed correspondences by traditional PnP-RANSAC. An example of coordinates regression algorithms is the full-frame coordinates regression pipeline proposed in [23], whose architecture is depicted in Figure 3.7.

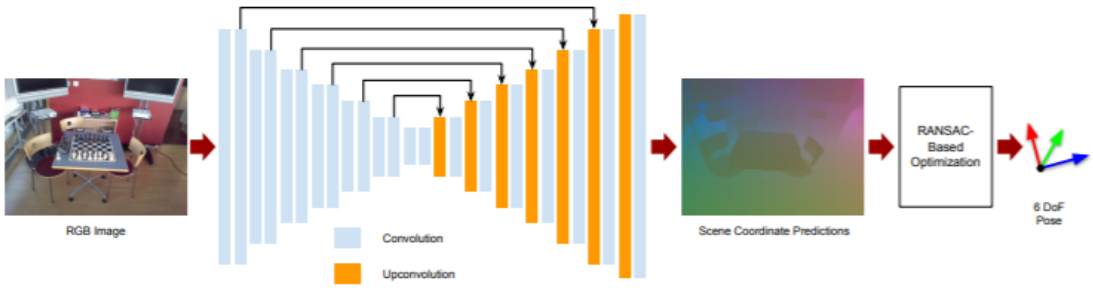


Figure 3.7. *Full-frame coordinate regression [23].*

In this architecture, a convolution-deconvolution network is used to predict the coordinates of a given query image. The first part of the network is similar to traditional CNN, combining convolution with downsampling. The second half performs then *deconvolution* followed by upsampling multiple times. Deconvolution is the inverse operation of convolution, i.e. given an image I and filter K, the output O of a convolutional layer will be $K * I$, where $*$ denotes the convolution operation. In a deconvolutional layer, the output is instead solved from the equation $K * O = I$. In order to train such architectures, a database of images with annotated 3D coordinates is needed, similar to retrieval based approaches. The choice of the loss function is also particularly critical [24], a common choice is to minimize the sum of distances between ground truth coordinates and predicted coordinates.

In pose regression approaches the whole end-to-end pipeline is replaced by a deep-learning algorithm, which is trained to directly predict the pose from a given query image [17]. Here, the pose is generally parametrized as the 7-ary array $\begin{bmatrix}\mathbf{t} & \mathbf{q}\end{bmatrix}$, where $\mathbf{t}$ is the

translation vector of the camera pose and $\mathbf{q}$ is the quaternion corresponding to the rotation matrix. The loss function for pose regression networks is generally in the form $\mathcal{L} = \|\mathbf{t} - \hat{\mathbf{t}}\| + \beta\|\mathbf{q} - \hat{\mathbf{q}}\|$, where $\begin{bmatrix}\mathbf{t} & \mathbf{q}\end{bmatrix}$ is the ground truth pose and $\begin{bmatrix}\hat{\mathbf{t}} & \hat{\mathbf{q}}\end{bmatrix}$ is the pose predicted by the network. Different network architectures, such as CNN or convolution-deconvolution networks have been experimented, with the latter giving slightly better results. Still, coordinate regression approaches outperform pose regression networks [32]. Overall, learning based approaches, despite being generally faster, cannot reach the same accuracy of retrieval based approaches and they are not yet suitable for visual localization in challenging environments [38].

3.4 Case study: InLoc

To conclude this chapter we analyze more in depth a retrieval based pipeline for indoor visual localization: InLoc. This pipeline and the corresponding dataset are also used to benchmark the confidence estimation algorithm proposed in the next chapter.

3.4.1 InLoc dataset

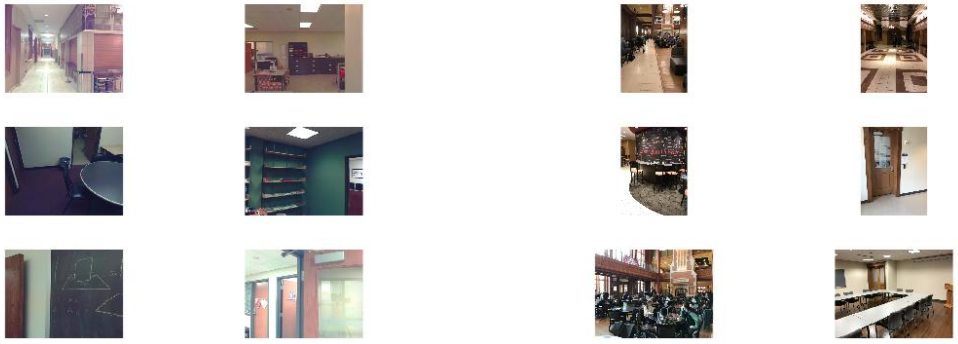
The InLoc dataset was designed to benchmark indoor long-term visual localization algorithms, it consists of a set of database images, a set of query images and a dense 3D model. Database images were collected from two different buildings, from now on referred as DUC and CSE, from Washington university. The images from the DUC building were taken from two different floors and the images from the CSE building from three different floors and the original 3D coordinates are known for almost every pixel. Further information about how the 3D coordinates were obtained for the database images can be found in [53]. Query image were collected with an iPhone only from the DUC building. The dense 3D model of both buildings was obtained with a laser scanner. Tables 3.1 and 3.2 present further technical details. Some examples of query and database images from the dataset are shown in Figure 3.8.

Table 3.1. *Dataset properties per image group.*

	# images	Resolution (pixels)	focal length (pixels)
Database images	9972	1600×1200	1430
Query images	329	4032×3024	1320

Table 3.2. *Dataset properties per floor.*

Floor ID	# query images	# database images	3D model coverage
DUC1	198	1800	98%
DUC2	131	2880	100%
CSE3	0	252	85%
CSE4	0	2700	93%
CSE5	0	2340	66%



(a) *Database images.* ***(b)*** *Query images.*

Figure 3.8. *Example pictures from the dataset.*

This dataset presents several challenges for indoor visual localization. Both database and query images are taken under different environment conditions, such as different illumination and different disposition of objects or people. Also, query images were taken only from the DUC building, the data from the CSE building are included to increase the challenge in the image retrieval step.

3.4.2 InLoc algorithm

The InLoc algorithm was proposed by Taira et al. in 2018 [48] and it addresses indoor visual localization under changing conditions. The salient points of the algorithm are the use of dense matching instead of sparse matching and the addition of a pose verification step at the end of the process. The pipeline of InLoc is depicted in Figure 3.9.

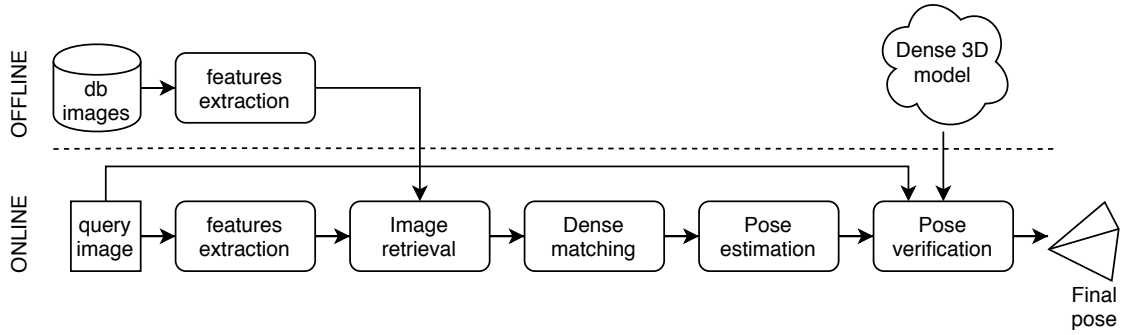


Figure 3.9. *Pipeline of InLoc algorithm.*

Features extraction for image retrieval is executed through the NetVLAD network [1], which is composed by the VGG network [44], computing features map, on top a which an additional layer, called VLAD layer [2], is added. This last layer aggregates local descriptors of feature maps into global descriptors to perform image retrieval, as described in Chapter 3. In this step, for each query image the 100 best matching database images are retrieved. In the next step, 2D-2D correspondences between query image and database images are formed using dense matching in a coarse-to-fine manner. Feature maps are obtained from the VGG network, particularly the coarse map is obtained after the fifth convolutional layer and the fine map after the third convolutional layer. As VGG is part of NetVLAD, features computation can be done only once for the whole pipeline. The computed correspondences are verified fitting a homography between query and database images. The 100 candidates database images are ranked based on the number of inliers and only the ten best candidates are kept. Pose estimation is then performed using P3P-LO-RANSAC and the candidate poses are ordered in decreasing number of inliers.

The final step in the Inloc pipeline is *pose verification* (PV), whose purpose is to choose the best pose among the ten previously computed ones. In this phase, a view is synthesized from the point cloud data using an estimated pose. RootSIFT descriptors are then extracted from both synthesized views and original query image. Finally a *verification score* is computed as

$$ver.score = \frac{1}{\text{median}(\|\mathbf{d}_1^q - \mathbf{d}_1^s\|, \dots \|\mathbf{d}_n^q - \mathbf{d}_n^s\|)}, \tag{3.12}$$

where $\mathbf{d}_1^q, \dots, \mathbf{d}_n^q$ and $\mathbf{d}_1^s, \dots, \mathbf{d}_n^s$ are RootSIFT descriptors from the query and synthesized image, respectively. The verification score in Equation (3.12) is compute for each candidate pose and used to rerank in descending order the ten candidate poses and the pose with the highest verification score is outputted as final result.

As an evaluation metric, translation and rotation error are used, which can be computed

with equation (3.13).

$$\epsilon_{tr} = \|\mathbf{c} - \hat{\mathbf{c}}\|,$$
$$\epsilon_{rot} = \arccos\left(\frac{\mathrm{tr}(\hat{R}R^T) - 1}{2}\right), \tag{3.13}$$

where $\begin{bmatrix}\hat{R} & -\hat{R}\hat{\mathbf{c}}\end{bmatrix}$ is the estimated camera pose and $\begin{bmatrix}R & -R\mathbf{c}\end{bmatrix}$ the corresponding ground truth. A query image is considered correctly localized if both errors are below a predetermined threshold.

Figure 3.10 shows the accuracy, i.e. the percentage of correctly localized query images, as a function of the translation threshold keeping the angular threshold at $10°$. Table 3.3 shows the measured computation times of the pipeline using two GeForce GTX Titan X GPU. The authors of InLoc provided only the results of the image retrieval step, but not the codes to perform it. For this reason, the image retrieval step could not be profiled.

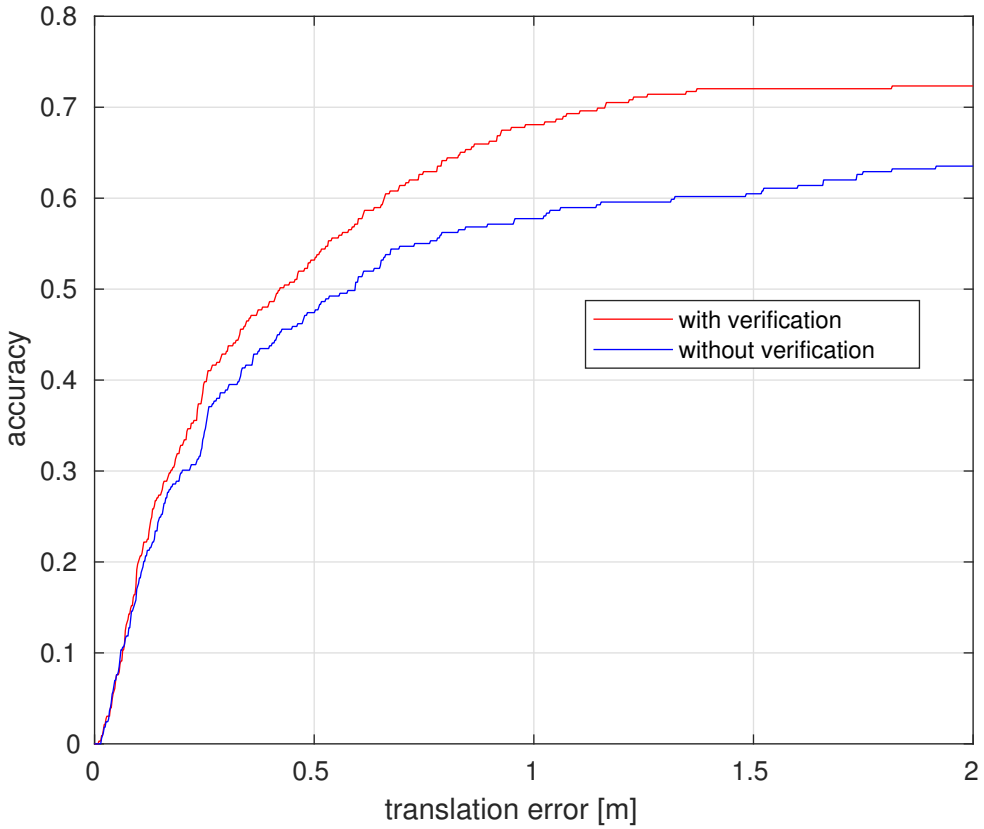


Figure 3.10. *Accuracy as a function of translation threshold, angular threshold being* $10°$*.*

Table 3.3. *Computation times of different steps for a single query image.*

Step	Computation Time
Features extraction	1 s
Dense Matching	100 s
Pose Estimation	5 s
Pose Verification	47 s

As can be noticed from Figure 3.10, pose verification improves the performance of the algorithm. However, the overall accuracy is still low, if e.g. compared to outdoor localization applications [35]. Using as thresholds 1 m and 10° the algorithm localized correctly only 70% of the images. Still, this is a state-of-the-art result for the proposed dataset. Furthermore, from Table 3.3 it can be noted that InLoc is still not suitable for real-time or mobile applications. Particularly, dense matching is the most time-consuming step in the pipeline. This same step, however, is crucial for the pipeline to work and the authors prove in their paper that sparse matching leads to a significantly lower accuracy [48]. The whole pipeline for a single query image takes approximately 2.5 minutes (excluding the retrieval step), indicating the InLoc pipeline is still not suitable for real-time or mobile applications.

4 CONFIDENCE MODELLING IN VISUAL LOCALIZATION

In this chapter the confidence estimation problem for visual localization is formalized and the proposed solution is described. In the presented approach, we first analyze several factors affecting the success of pose estimation and then propose an algorithm to merge the different parameters into a final confidence measure, regarded as the probability of the pose to be correct.

4.1 Related work

The number of inliers outputted by RANSAC is the most popular confidence estimator in pose estimation [22, 35], as well as in other model estimation problems in computer vision [39, 52]. A deeper analysis of the limitations of the number of inliers is given in the Section 3.3. In [50], an alternative version of RANSAC, called MLESAC is proposed. In MLESAC, the best estimate is chosen maximizing a likelihood function, instead of the mere number of inliers. Some alternative RANSAC implementations [5, 34] model how the error propagates during the RANSAC algorithm and propose a modified version and use this information to compute a confidence interval of the estimated parameters.

In [12] it was shown that dropout layers [46] can be exploited to modify a neural network into a *Bayesian neural network*, which estimates a confidence interval instead of giving a point estimate. In the context of visual localization, Bayesian neural networks were applied in [18] to compute a probabilistic distribution of the camera pose parameters, which was used to achieve better accuracies. The InLoc pose verification step itself can be considered a confidence estimation algorithm, as the verification score quantifies which candidate pose is most likely to be correct.

As a final remark, the confidence estimation approaches described take place in intermediate steps of the pipeline, either inside RANSAC or immediately after before outputting the final result. Thus these approaches focus on a single query image and aim at estimating a more accurate camera pose. In this work, a wider problem is considered. The proposed confidence estimator takes place *after* the final result is computed and aims at quantifying how reliable it is. The proposed confidence measure is thus used to compare the confidence of poses from different query images.

4.2 Problem formulation

The confidence estimation problem can be formalized as a binary classification problem, i.e. given an estimated camera pose $\hat{P}$, we want to classify it as correct (output 1) or incorrect (output 0). Ideally, the classifier $f : \hat{P} \mapsto y \in \{0, 1\}$ will have the properties $f(\hat{P}) = 1$ if and only if $\hat{P}$ is correct and $f(\hat{P}) = 0$ if and only if $\hat{P}$ is incorrect. Clearly, a perfect classifier is generally not achievable. A classifier is said to be linear if it has the form

$$y = \begin{cases} 1, & \gamma \geq \gamma_0 \\ 0, & \gamma < \gamma_0 \end{cases}, \tag{4.1}$$

where γ is a called *discriminator* and γ_0 is a predetermined threshold. Geometrically, a problem is solvable with a linear classifier if the boundary between the classes can be represented as a linear function. The choice of the threshold γ_0 depends on the specific application. In some applications, such as self-driving cars, it is crucial to be able to trust the estimated pose and hence a higher threshold may be set. Some other applications on the other hand may have more relaxed requirements and a lower threshold may be accepted. The discriminator γ can be regarded as a confidence measure, the higher it is, the more likely the estimated pose will be correct. Independently from the value of γ_0, a good discriminator should be able to properly rank several estimated poses. The rest of this chapter is dedicated to the computation of γ.

4.3 Number of inliers

The number of inliers obtained during the RANSAC-PnP algorithm has been the standard metric to quantify the confidence of the pose in most visual localization algorithms. In the RANSAC loop we hypothesize a pose from 2D-3D or 2D-2D correspondences, next we compute how many correspondences can be modelled by the proposed pose, obtaining the number of inliers. The number of inliers hence measures how many evidences we have to sustain our hypothesis and is used to determine the best pose among a set of candidates. However, a higher number of inliers does not always imply a better pose, as InLoc and others show in their experiments. Furthermore, a high number of inliers does not imply a correct estimation either. An example of this is shown in Figure 4.1, where a pair of query and database images from the InLoc dataset are reported, presenting 1921 inliers in total but still leading to an incorrect pose using $1\,\mathrm{m}$ and 10° as error threshold.

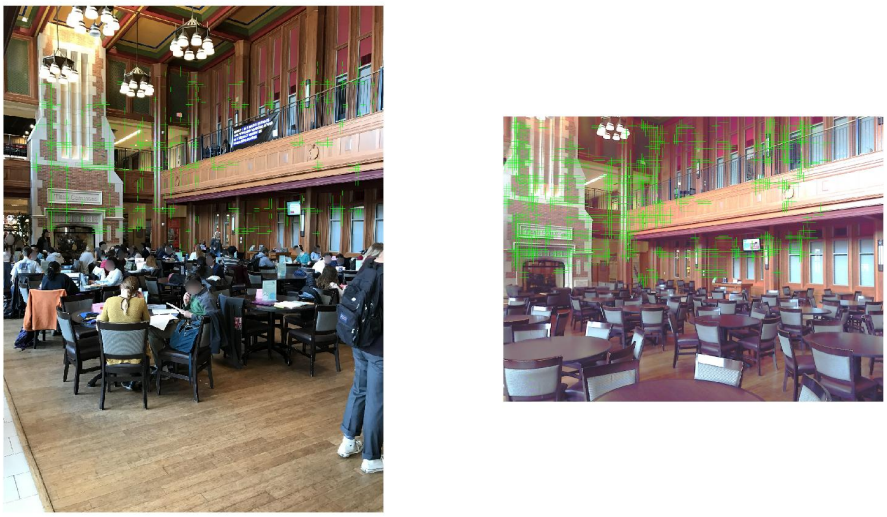


Figure 4.1. *Example of an inaccurate estimate with a high number of inliers.*

The main factor leading to a wrong estimate with a high number of inliers is the presence of repetitive patterns in the images. Referring to Figure 4.1, both images present, e.g. multiple times the same windows, several identical chairs and, most importantly, walls present a significant amount of redundancy, as they are made of identical bricks. Thus, despite the overall clusters of inliers in query and database image appear to be in the same area, the matches between single inliers are wrong, leading to an inaccurate pose.

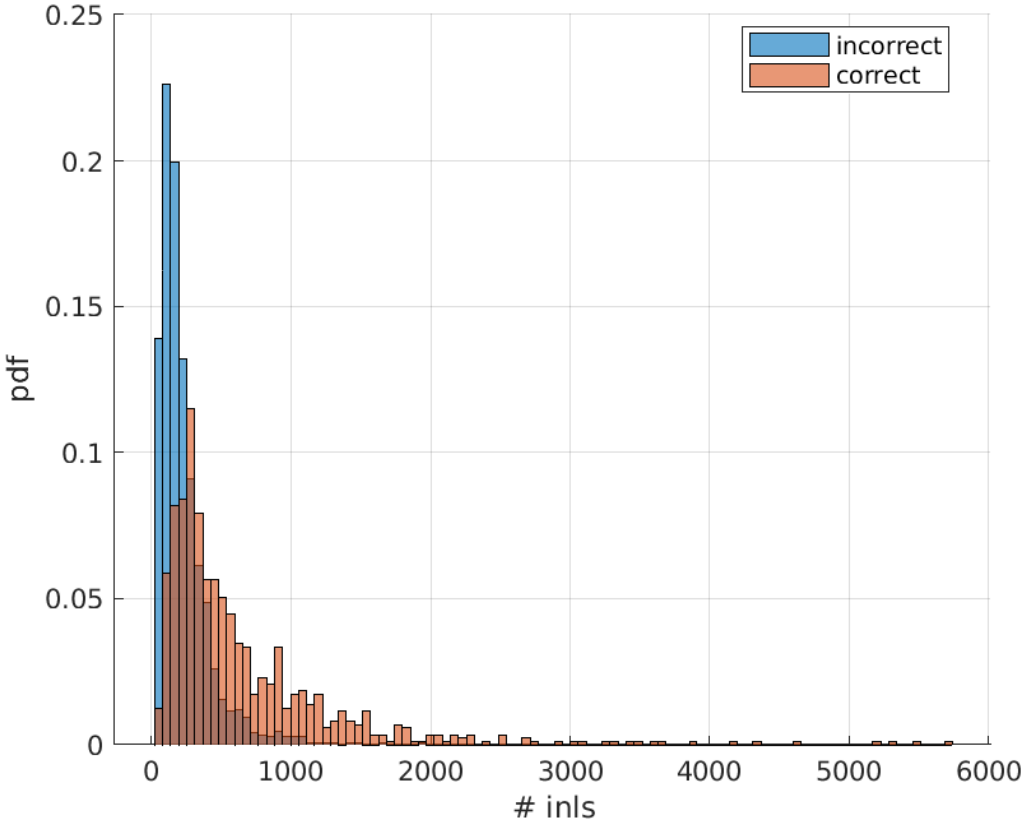


Figure 4.2. *Number of inliers distributions.*

The histograms reported in Figure 4.2 help visualizing how well a parameter can be used to discriminate between successful and unsuccessful camera pose estimates. The poses are divided into two classes: correct and incorrect. For both classes we plot the histograms approximating the probability distributions of the number of inliers. As can be noticed, the class of correct poses has a longer tail, meaning if the number of inliers is very high (over 2000) then it is highly probable that the pose is correct. This easy cases, however, appear to be relatively rare, as the mass of most classes lies between zero and 1000, where the distributions present a significant overlapping, leading to the conclusion that the number of inliers alone is not a robust metric to discriminate between correct and incorrect poses.

4.4 Inliers distribution

A main challenge in visual localization is the presence of recurrent patterns in the environment. As an example, the same window could be present at the opposite sides of the same building. In database-based pose estimation algorithms, this means the same pattern can be present in both database and query image, even though they represent completely different places. An example of this is demonstrated in Figure 4.3, taken from

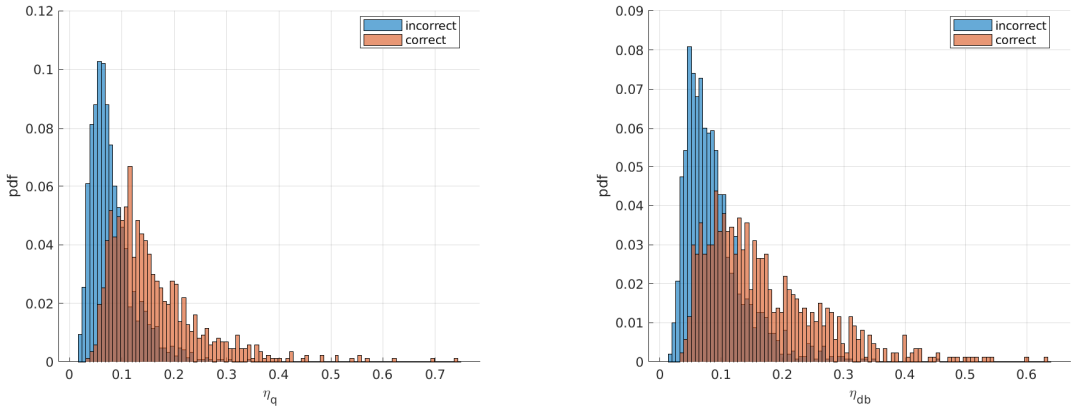
the Aachen dataset [37]. The two images (query image on the left and database image on the right) present two different scenes with a similar car in both. The algorithm seems to conclude the images represent the same scene and computes the camera posed relying only on the car and ignoring the background, as can be noticed by the inliers being concentrated on a small zone. This leads us to investigate whether the inliers spatial distribution is relevant.

Figure 4.3. *Failure due to a repetitive pattern [35].*

To take into account inliers spatial distribution, we propose to compute a *coverage map* of the image. We say a pixel is *covered* if there is at least one inlier in its neighborhood. With coverage maps, we can quantify the inliers distribution using a *coverage score*, defined as the number of covered pixels divided by the total number of pixels in the image. To compute coverage maps, one first needs to fix the definition of neighborhood, namely its shape (square, circle) and size. Particularly crucial is the choice of the size of the neighborhood, as using a size to small would be equivalent to counting the number of inliers; on the other side a neighbourhood an excessively big neighborhood would give a too optimistic coverage score. Here, we simply define the neighbourhood of a pixel as a rectangle, centered on the pixel, whose dimensions are 1/15 of the dimensions of the image. In other words, we divide our picture in a grid with twenty cells per side and mark all pixels in the cells that contain at least one inlier. The size of the neighbourhood was chosen by visual inspection of the obtained coverage maps. Despite it could have been optimized by grid search, numerical experiments revealed small variations in the neighbourhood size did not affect significantly the final results. The coverage score is then computed dividing the number of marked pixels by the total number of pixels in the image. This is formalized in Equation (4.2). Let I be the image, p a pixel and J the set of inliers and $B(p)$ the neighborhood of p as defined above, then the coverage score η is

$$\eta = \frac{|\{p \in I | \exists j \in J : j \in B(p)\}|}{|I|}, \tag{4.2}$$

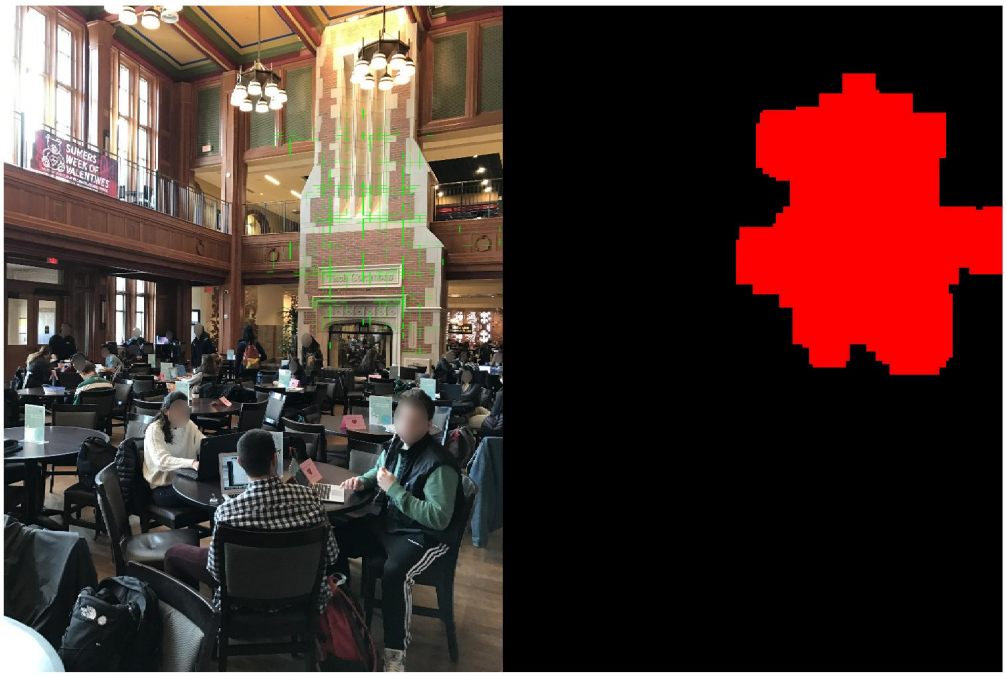
where score$|\cdot|$ denotes the cardinality of a set and hence $|I|$ denotes the number of pixels in the image.



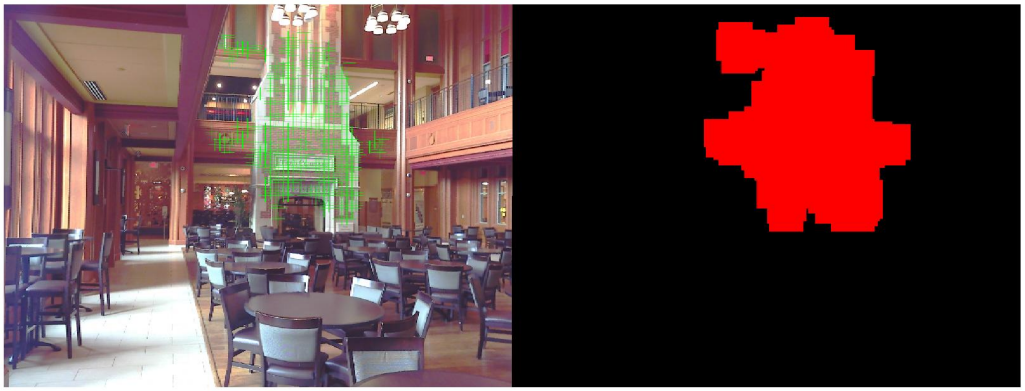
(a) *Query images.* ***(b)*** *Database images.*

Figure 4.4. *Coverage scores distributions.*

To investigate how well the coverage score performs as discriminator, we plot in Figure 4.4 the classes distribution as a function of η. As the histograms reveal, the coverage scores seem to distinguish the classes better than the number of inliers, despite some significant overlapping is still present. A small coverage score seems to be a necessary but not sufficient condition for incorrect poses, as correct estimations can be achieved also with relatively small coverage scores. These occur mainly when a correct estimation is achieved with a small number of inliers if, for example, only few good points correspondences have been formed.



(a) *Query image and coverage map,* $\eta = 15.5\%$.



(b) *Database image and coverage map,* $\eta = 15.3\%$.

Figure 4.5. *Incorrect estimate with many inliers covering a small area.*

Figure 4.5 presents an example of a query-database image from the InLoc dataset and the corresponding coverage maps. The pair contains 1084 inliers, which are however all focused in a small area, covering only 15% of the pictures. By Figure 4.2, 1084 inliers strongly suggests that the pose is correct. Figure 4.4, however, shows that $\eta = 15\%$ is at the boundary between the two distributions and hence is more likely to be classified as incorrect.

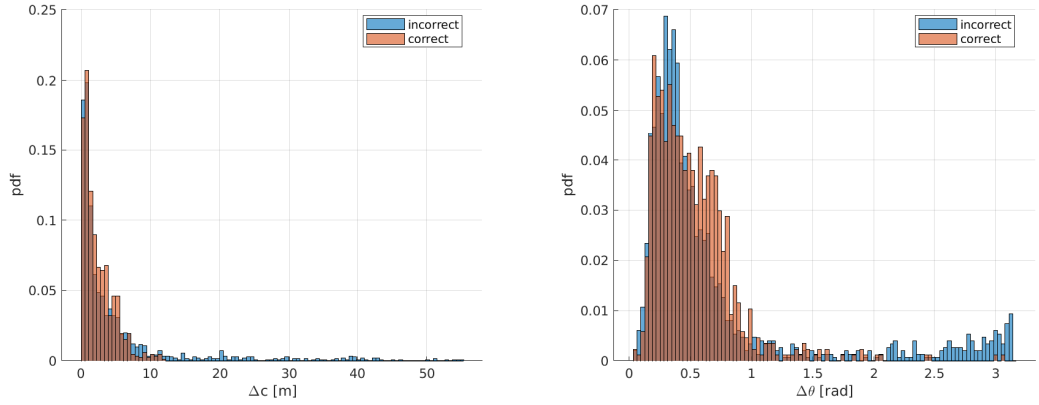
4.5 Cameras similarities

In retrieval based approaches we first extract the most similar database images to a given query image. Intuitively, if two images are similar, the two cameras are close to each others and point in the same directions as it is quite unlikely that cameras pointing in opposite directions will see the same scene, as well as cameras being close to each others are more likely to see the same scene. Motivated by this intuition, we investigate whether taking into account the difference between camera poses can help the confidence estimation task.

First, a mathematical formalization of the "pose difference" is needed. Particularly, we want to model both the difference of positions and the difference of viewpoints (orientations) of the cameras. Similarly to (3.13), the position difference Δc and angular difference $\Delta\theta$ can be computed as

$$\begin{aligned} \Delta c &= \|\mathbf{c}_q - \mathbf{c}_{db}\|, \\ \Delta\theta &= \arccos\left(\frac{\mathrm{tr}(R_q R_{db}^T) - 1}{2}\right), \end{aligned} \tag{4.3}$$

where $\begin{bmatrix} R_q & -R_q\mathbf{c}_q \end{bmatrix}$ is the query image camera pose and $\begin{bmatrix} R_{db} & -R_{db}\mathbf{c}_{db} \end{bmatrix}$ is the database image camera pose. Figure 4.6 shows how the two classes depend on the poses differences.



(a) *Translation difference.* ***(b)*** *Angular difference.*

Figure 4.6. *Cameras similarities distributions.*

Figure 4.6 confirms that correct estimates tend to have database and query image cameras close to each others, as their distance is at maximum 10 m. However, the assumption about angular distance does not hold as strongly. It is shown, indeed, that despite correct estimates tend to have a smaller angular distances, $90°$ at most, there can also be few cases where a good estimate is achieved when cameras are pointing in nearly opposite directions. This can be understood thinking of cameras at opposite sides of a room, seeing the same scene in the middle. Also, a small pose difference, especially for

translation difference, appears to be a necessary but not sufficient condition for a correct estimate as wrong estimates span a wide range of both translation and angular distance.

4.6 Neural networks for confidence estimation

Based on the previous discussion, none of the parameters alone can be used to discriminate between successful and unsuccessful estimates. Nevertheless, the histograms in the previous sections show a correlation between the success of the estimate and the parameters analyzed. The next step is to learn how to weight those parameters to obtain the final confidence measure γ.

In the proposed method, we let an Artificial Neural Network (ANN) learn itself the pattern. During the experiments, other approaches, such as decision trees and support vector machines were also considered, however they gave inferior performances and hence ANN was chosen. The network, given a pair of query image with estimated pose and database image with corresponding pose, will predict the probability that the estimated pose is correct. As network input parameters we use the number of inliers obtained after the PnP-RANSAC, the coverage scores of query and database images and the cameras translation and angular differences. Figures 4.7 shows the architecture of the network.

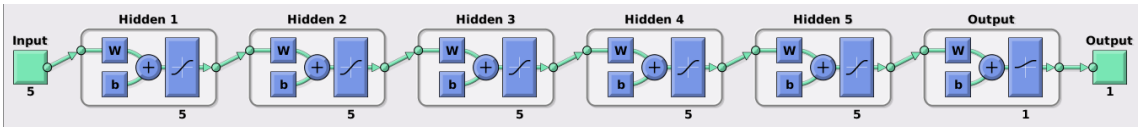


Figure 4.7. *Proposed network architecture.*

As we use a relatively small amount of input parameters (5 scalars) and the amount of data available is relatively small, we choose to keep the network architecture small, namely we use five fully connected layers with five neurons each. As nonlinear activation function, $\tanh$ is used in the hidden layers and the logistic function is used in the output layer. The internal parameters of the network are learnt minimizing the cross-entropy $\mathcal{L}$, computed with Equation (4.4)

$$\mathcal{L} = -y \log(p) - (1 - y) \log(1 - p), \tag{4.4}$$

where y is the label (0 or 1) and p is the probability of the estimate to be correct.

5 EXPERIMENTS

In this chapter we describe the experimental setup used to benchmark our algorithm. Particularly, we describe how the InLoc dataset is used to train and test our neural network. The metrics used to evaluate the performances of our confidence estimator are also presented.

5.1 Network training

Our confidence estimation algorithm can be applied *after* the camera pose has been estimated, meaning the 329 InLoc query images have to be used to both train and test our method. To overcome the problems stemming from the small amount of training data, we produce an *extended dataset* as follows: for each query image, all 10 candidate database images provided by the InLoc pipeline are taken into account, obtaining 3290 query and database image pairs, each associated with an estimated camera pose. Out of these, we remove those pairs that have less than 3 point correspondences, as they represent a trivial case of failed estimation. For each pair, the coverage scores can be computed from the inlier positions produced by RANSAC. Each estimated pose is labelled as correct (1) if the error is below the threshold of $1\,\mathrm{m}, 10^\circ$ and as incorrect (0) otherwise. As a result, our extended dataset presents 2368 estimated poses, out of which 75% are used as training data and the rest as test data. The training-test splitting is done so that all the image pairs related to the same query image are either in the training set or test set, but not in both. Overall, our training dataset contains 1765 query-database images pairs, related to 230 distinct query images and the test dataset contains 603 query-database image pairs, corresponding to 99 distinct query images.

5.2 Evaluation metrics

In this section the metrics used to evaluate our algorithm are presented. As discussed before, the confidence estimation problem is formulated as a binary classification problem, given a query image with the corresponding estimated pose, classify whether the pose is correct or incorrect. We call *true positives* (TP) those samples whose pose is correct and that are classified as correct. Similarly, those samples whose pose is not correctly estimated but are still classified as correct are called *false positives* (FP) and those which

have a correct pose but are classified as incorrect are referred as *false negatives* (FN). This terminology is helpful to define two widely used metrics in classification problems: *precision* and *recall*, whose definition are given in Equations (5.1) and (5.2)

$$p = \frac{TP}{TP + FP}, \tag{5.1}$$

$$r = \frac{TP}{TP + FN}. \tag{5.2}$$

Using only one of the above metrics to asset the performance of a classification algorithm is not very informative, as it is possible to have a high precision and poor recall and vice versa. Moreover, both metrics depend on the chosen value of the threshold γ_0 in Equation (4.1) and hence they can be improved arbitrarily. As an example, setting a very low threshold the algorithm will classify all samples as correct and hence it will have $r = 1$, as $FN = 0$, but the precision will be low, due to the high number of false positives. On the other side, setting a very high threshold the algorithm will classify most samples as incorrect having a high precision but a poor recall.

To asset the performance of a classification algorithm, we want a measure taking into account both precision and recall and independent of the threshold value. A way to achieve this is to compute both precision and recall for several values of γ_0 and plot the precision as a function of the recall. The obtained curve is called *precision-recall curve* (PR-curve). An evaluation metric for the classification problem can now be obtained taking the *Area Under Curve* (AUC), which can be formally defined as

$$AUC = \int_0^1 p(r)dr, \tag{5.3}$$

where $p(r)$ is the precision as a function of the recall.

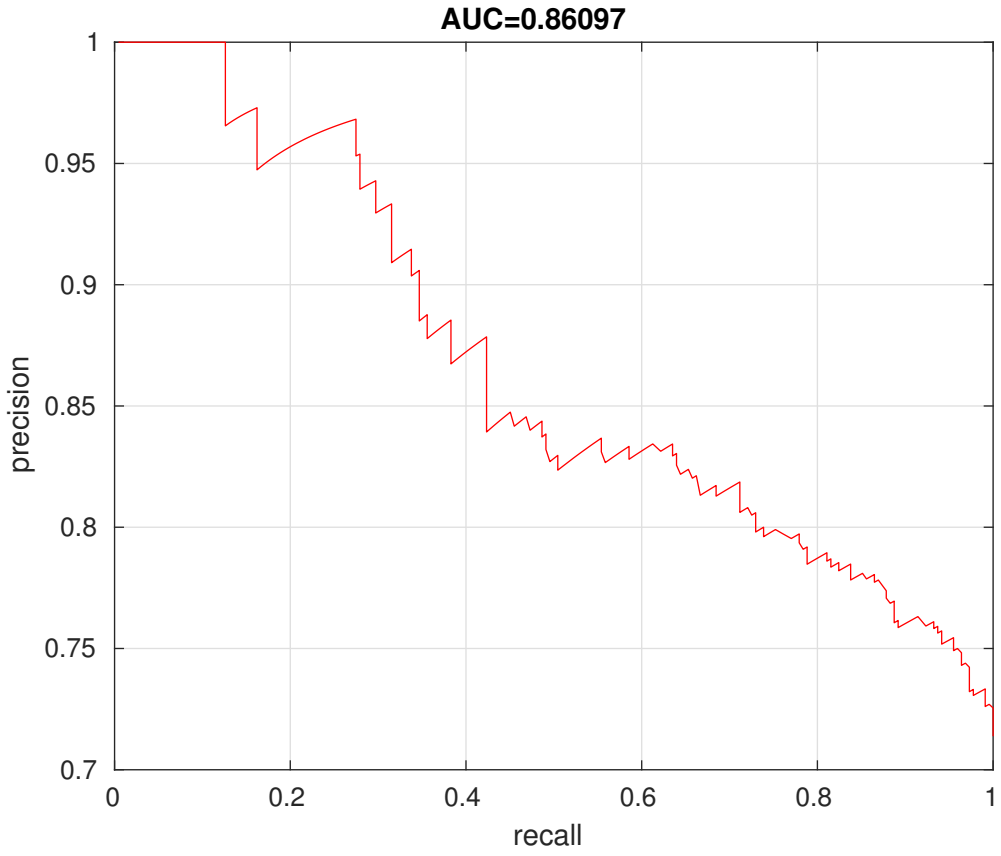


Figure 5.1. *PR-curve of InLoc using the number of inliers as discriminator.*

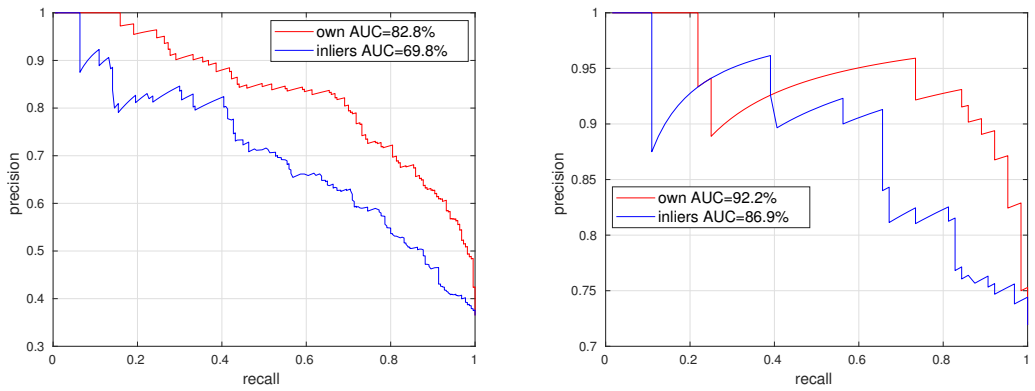
An example of Precision-Recall curves is shown in Figure 5.1, obtained from using the number of inliers as discriminator and $1\,\mathrm{m}$, 10° as error threshold for correct poses. The left-most part of the curve, having high precision and low recall, corresponds to a high γ_0, i.e. only those poses with a very high number of inliers are accepted as correct. Decreasing the threshold the recall improves but the precision gets worse. Finally, having recall one means that all poses are accepted as correct and in that case the precision is equivalent to the accuracy of the algorithm. From Equation (5.3) it can be noticed that the AUC value can be interpreted as the average precision. Hence, AUC=86% means that on average 86% of the poses classified as correct are actually correct.

6 RESULTS

In this paper we present the results of the quantitative experiments to evaluate the performances of our model. The inclusion of InLoc verification score into our confidence estimator is also discussed. Next, we show how our model can also be included inside the InLoc pipeline to achieve more accurate camera poses. Finally, we show how our model, despite being trained with InLoc, can be applied also to different datasets.

6.1 Confidence estimation in InLoc

In this first set of experiments, we feed to our neural network the number of inliers, cameras differences and the coverage scores of query and database images. The parameters were obtained as discussed in the previous chapter. The network is trained using the training data of the extended dataset. To evaluate the functionality of our model, the confidence measures for the test data of the extended dataset are computed and the PR curve is plotted. The PR-curve is then compared to the one obtained using the number of inliers as discriminator. The results are shown in Figure 6.1a. The figure shows that our confidence measure helps indeed in classifying better if an estimate is correct or not. Particularly, we can notice an improvement in the AUC of roughly 12%.



(a) *Extended dataset test data.* ***(b)*** *Only best candidates.*

Figure 6.1. *Precision-recall curves obtained with our method.*

We also investigate how our algorithm performs when tested only on the best candidate pose for each query image. The results are shown in Figure 6.1b. The figure shows

that also in this case our algorithm produces a better PR-curve, improving the AUC from 86.9% to 92.2%. This shows that the algorithm works also for realistic pipelines, where the easier true negatives are already filtered out when choosing the best candidate.

Table 6.1. *Performances of the proposed method at different error threshold.*

Threshold	AUC (inls)	AUC (our)
$1.5\,\mathrm{m}, 10^\circ$	87.2%	94.4%
$1.0\,\mathrm{m}, 10^\circ$	86.9%	92.2%
$0.5\,\mathrm{m}, 10^\circ$	68.7%	77.4%

During the training process the error threshold was fixed to $1\,\mathrm{m}, 10^\circ$. In Table 6.1 we show that the proposed method does not depend from the threshold used for training. The error threshold used to determine if the pose is correct was changed, without changing the threshold used to train the model. The experiments were done considering only the best candidate for each query image. As can be seen in Table 6.1 a stricter threshold makes the classification problem harder in general, but the proposed method is still more robust than the inliers count alone.

6.1.1 Ablation study

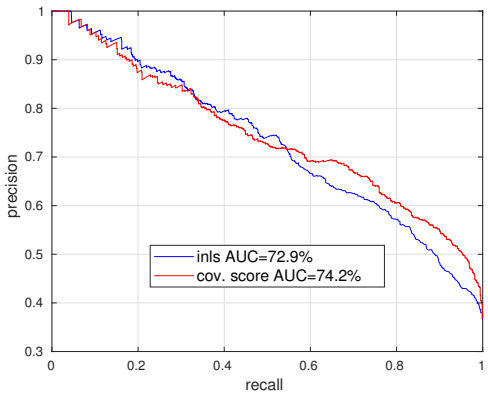
An interesting question to address is *what is the contribution of each single input?* or in other words, *Can we obtain a similar PR-curve with less parameters?* To answer these questions, we performs an *ablation study* on the network, meaning we retrain the network leaving out one input parameter. This is repeated for each input parameter, to quantify how much each feature affects the confidence measure. The results of the ablation study are reported in Table 6.2.

Table 6.2. *Ablation study of the proposed model. The full network contains inliers, cameras differences and cov. scores.*

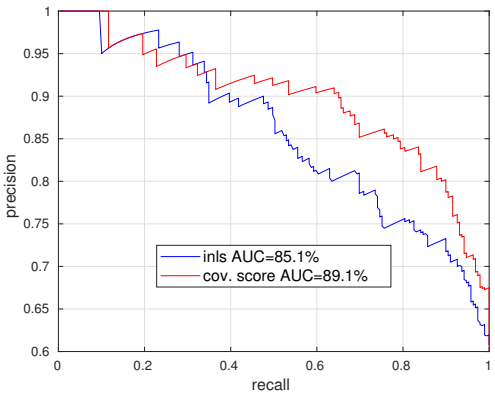
Case	AUC
full network	**82.8%**
only inliers	69.8%
without inliers	76.3%
without cameras difference	77.7%
without translation distance	80.7%
without angular distance	79.8%
without coverage scores	73.3%
without database image coverage score	81.0%
without query image coverage score	75.8%

From Table 6.2 some insights can be extracted. First, it seems that the coverage score has a particularly important role in the confidence estimation task, as without it the AUC drops 10%. Particularly, the coverage score of the query image seems to give a major contribution than the coverage score of the database image. Camera poses difference has a smaller contribution to the value of the AUC. Particularly, it appears that the poses difference can be modelled by either translation or angular distance, as removing only one of them does not affect significantly the AUC.

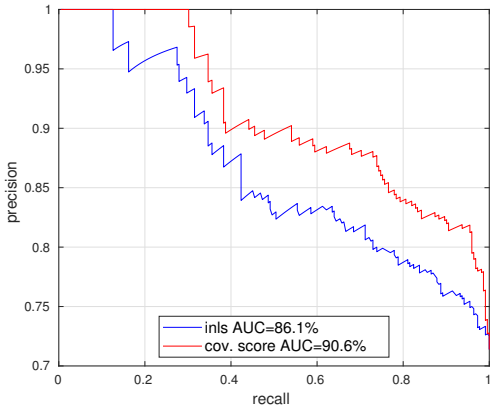
As shown in Table 6.2, the coverage score, modelling the inliers spatial distribution, improved significantly the AUC of the algorithm. To emphasize the importance of inliers spatial distribution, we try to classify correct and incorrect poses using only the coverage score of the query image as discriminator. Figure 6.2 reports the results in three different cases: using the whole extended dataset (a), using only the best candidate according to the number of inliers for each query image (b) and using the best candidate according to the verification score (c). As the coverage score alone does not need to be trained, the evaluation can be performed on the whole InLoc dataset, meaning 2368 estimated poses for case (a) and 311 estimated poses for cases (b) and (c).



(a) *All pictures.*



(b) *Best candidate w.r.t. inliers score.*



(c) *Best candidate w.r.t. verification score.*

Figure 6.2. *Performances of coverage score as discriminator.*

As Figure 6.2 reveals, the coverage score and the number of inliers have a similar performance for the extended dataset. When only the best candidates are taken into account, however, the coverage score produces a better PR-curve. When taking the best candidate (using the number of inliers or verification score), we already filter out some incorrect poses. Among the best matches, those which are truly correct tend to cover a higher area, making the coverage score a good tie-breaker to classify which among the best candidates are actually correct.

Table 6.3. *Coverage score improving InLoc accuracies.*

Sorting Method	Accuracy
Inliers	57%
Weighted Inliers	61%
Verification Score	69%

To further highlight the importance of the coverage score, we show how it can be used to improve the accuracy of the algorithm. For a candidate pose, we call *weighted number of*

inliers the number of inliers multiplied by the coverage score of the query image. In Table 6.3, we show how sorting the candidates by the weighted number of inliers affects the accuracy. Despite not reaching the results of the verification score, the weighted number of inliers gives an improvement compared to the number of inliers alone. This can be particularly useful for cases where 3D models are not available.

6.1.2 Pose verification for confidence estimation

The analysis so far has been carried out without using the InLoc verification score. Here, we conduct similar experiments as before, taking into account also the InLoc verification score. The purpose of this analysis is to both to benchmark the performances of the verification score itself and to investigate how it compares to our confidence estimation framework and how it can be integrated into it.

The first natural question is *can the verification score alone be used as a confidence measure?* In other words, can we use the verification score as parameter to discriminate between successful and unsuccessful estimates? As Figure 6.3 shows, the answer is no. Despite improving the accuracy of the algorithm, the verification score alone is not a good confidence estimator. This can be understood as follows: as described in the previous chapter, for each query image the ten candidates are sorted based on their verification score, i.e. the comparison is done among images related to the same query image, hence somehow similar to each others. When assessing the parameter performance as confidence estimator, different query images are compared. The verification score is hence designed to compare similar images, but fails when applied to different query images.

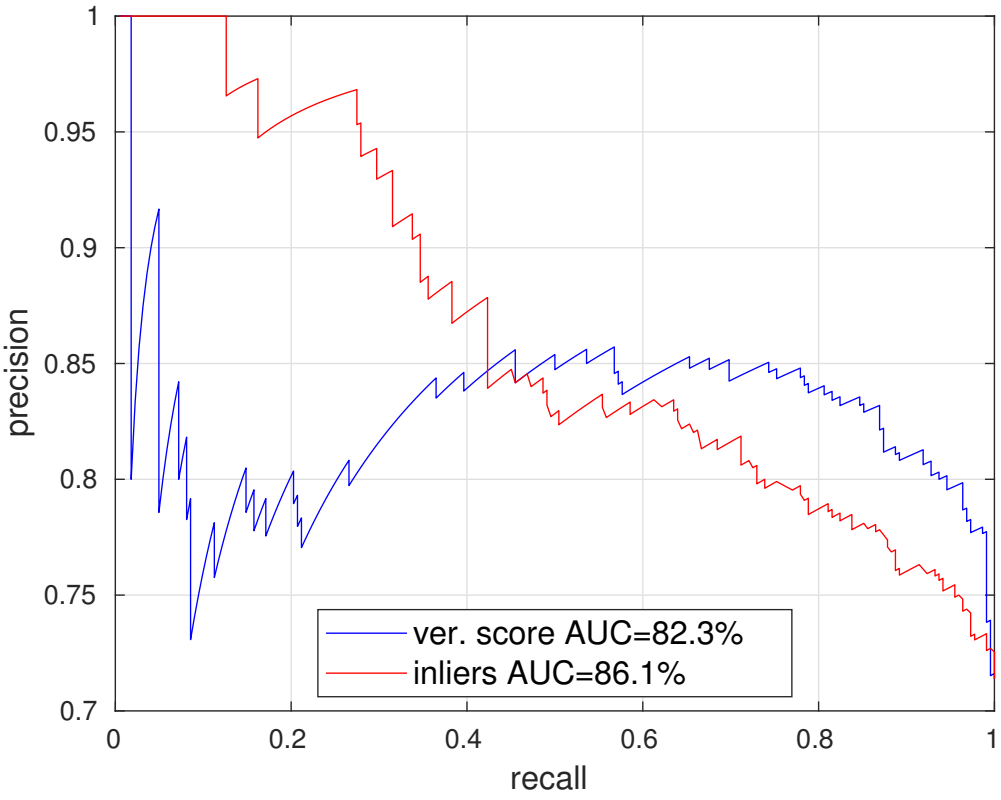
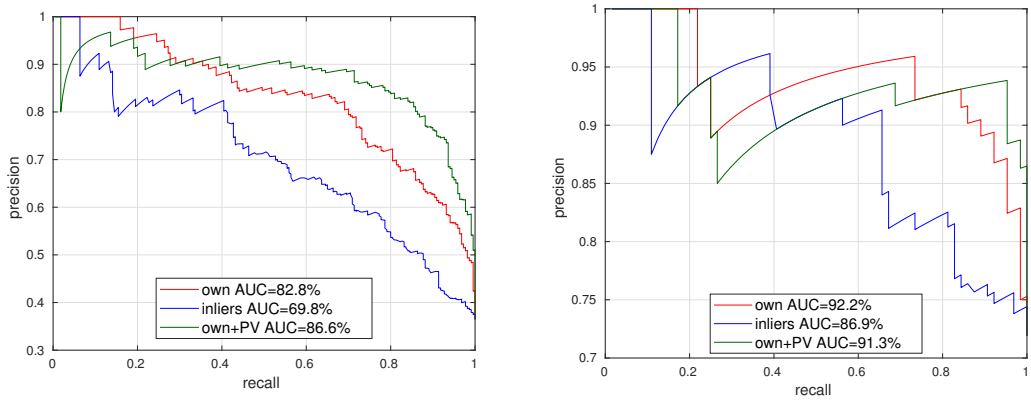


Figure 6.3. *Verification score performances as confidence estimator.*

As the verification score alone cannot be used as a discriminator alone, we investigate how it can be integrated in the previously developed confidence estimation model. In this new version the network will also take the verification score as input parameter, in addition to the previous ones. The results for the test data of the extended dataset are depicted in Figure 6.4a. Overall, the AUC improves from 83% to 87%, showing that the model seems to benefit from the verification score.

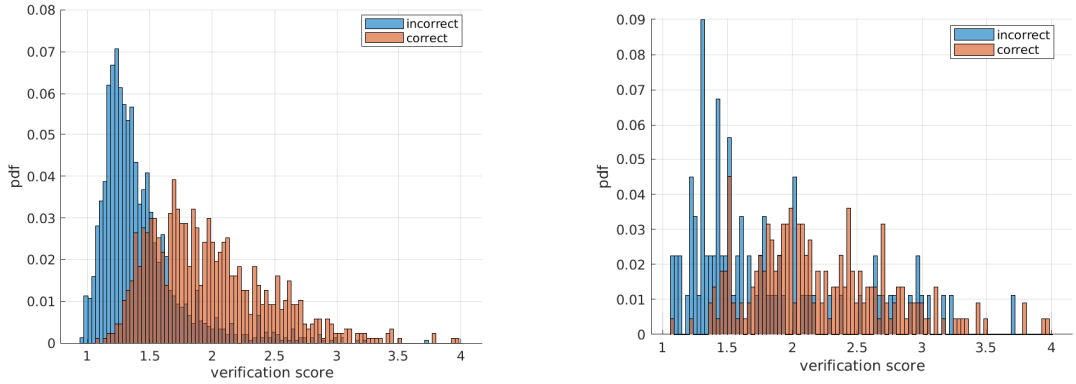


(a) *Extended dataset.*

(b) *PR-Curves using only the best candidates.*

Figure 6.4. *Inclusion of InLoc verification score to our method.*

Again, we also investigate how our model performs when tested only on the best candidates for each query image. The results can be seen in Figure 6.4b. Despite the lack of smoothness due to the few number of test data, the figure shows both model perform better than the number of inliers. This time, however, the addition of the verification score does not produce any significant improvement. To investigate what causes this discrepancy between extended and original dataset, we plot the histograms of the verification score for both datasets, as shown in Figure 6.5.



(a) *Extended dataset.*

(b) *Only best candidates.*

Figure 6.5. *Verification score distributions.*

Observing Figure 6.5, in the extended dataset correct and incorrect poses present two clearly distinct distributions, with some small overlapping. However, when only the best candidates for each query image are chosen, sampling is done preferring those points with a higher verification score. This does not preserve the shape of the distributions, leading to two new indistinguishable histograms.

6.2 Confidence estimation for better accuracies

Our algorithm estimates the probability of the pose to be correct. So far it was applied at the very end of the pipeline, to classify the final pose of different query images as correct or incorrect. However, it is also natural to wonder whether our confidence estimator can be used inside the pipeline to chose the best candidate pose for a given query image. To have a fair comparison, the accuracies are computed using only those query images that were not used to train the network. The results are shown in Table 6.4 using $1\,\mathrm{m}$ and 10° as error threshold. In Figure 6.6 the accuracies as a function of the translation threshold are plotted, keeping the angular threshold fixed to 10°. Again, the error threshold was changed only during evaluation, but not during training. The accuracies for InLoc differ by the ones in the original paper [48] as now only 99 query images are considered.

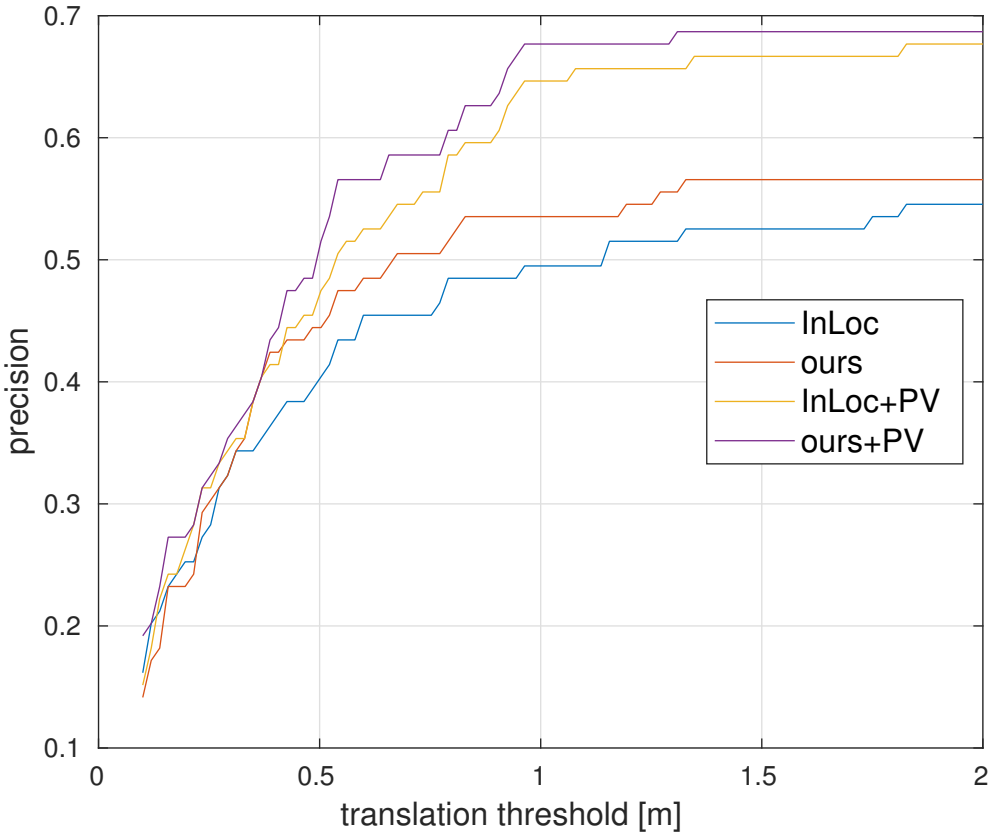


Figure 6.6. *Comparison of accuracies with our method and with InLoc.*

Table 6.4. *Accuracies with different candidates sorting methods.*

	InLoc	our
without PV	49%	54%
with PV	65%	68%

Based on the empirical results, our model allows to improve the accuracy in both with and without the verification score. As Figure 6.6 confirms, our model is not limited by the threshold used during the training process and higher accuracies are achieved at different thresholds. Furthermore, our model without verification score, despite not reaching InLoc+PV accuracy, gives a significant improvement to InLoc without pose verification, indicating that it can be used to replace the latter. Computing the verification score requires a dense 3D model of the environment and is relatively time consuming, as was shown in Table 3.3. Our model is thus an appealing alternative for datasets without 3D dense models or use case scenarios where a faster computation is required.

In Figure 6.7 we show a couple of examples where choosing the best candidate with InLoc leads to an incorrect pose, but our method leads to a correct pose. For fairness, the figure also shows an example where InLoc succeeds in computing the pose but our

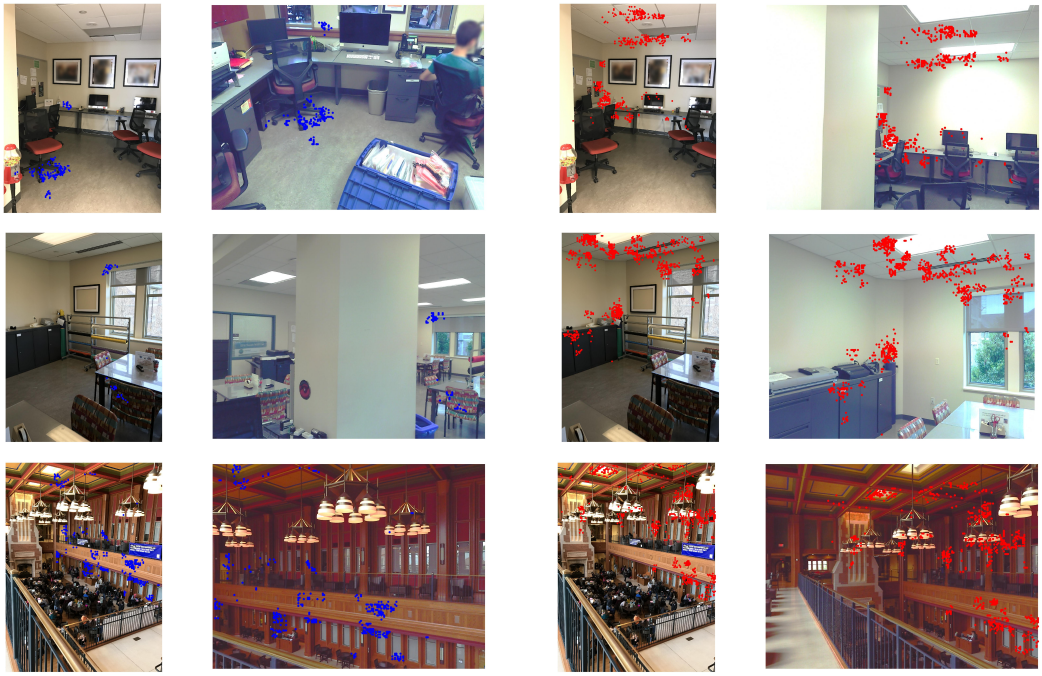


Figure 6.7. *Qualitative comparison of best candidate poses with InLoc (left, inliers highlighted in blue) and our method (right, inliers highlighted in red), both including PV.* ***First row:*** *InLoc error:* $32\,\mathrm{m}$, $3.9°$. *Our error:* $0.09\,\mathrm{m}$, $3.7°$. ***Second row:*** *InLoc error:* $5.3\,\mathrm{m}$, $14.3°$. *Our error:* $0.24\,\mathrm{m}$, $2.9°$. ***Third row:*** *InLoc error:* $0.56\,\mathrm{m}$, $2.1°$. *Our error:* $4.4\,\mathrm{m}$, $2°$.

model fails. Observing the pictures, it can be noticed that in those cases where InLoc pose verification fails the inliers are condensed into a small portion of the image. For example, in the first row all inliers are placed on a single chair. As the chair has been moved from query to database image, the final pose leads to a translation error of $32\,\mathrm{m}$. In our approach, on the other hand, the area covered by the inliers is also taken into account. This penalizes the candidate proposed by InLoc and a better one is chosen instead, leading to final translation error of only 9 cm.

6.3 Generalization to different datasets

In the experiments so far InLoc dataset was used for both training and testing. A good confidence estimator should however be independent of the dataset, algorithm or error threshold used for training. For this reason, we investigate here how our approach, trained on the InLoc dataset and algorithm with $1\,\mathrm{m}$ and $10°$ error threshold, performs when applied to unforeseen datasets and pipelines. Particularly, we consider two outdoor visual localization datasets: Cambridge Landmarks [17] and Aachen day/night [37] and the coordinates regression based pipeline presented in [22]. Since our confidence estimator trained with InLoc is now required to work on unforeseen datasets, a smaller network architecture is used, to improve its generality.

6.3.1 Cambridge Landmarks

Cambridge Landmarks dataset contains several scenes from around Cambridge University. This dataset is not particularly challenging, as query and database images were taken in similar condition, lacking e.g. strong differences in viewpoints or illumination. In these experiments, we consider the regression based pipeline presented in [22], where 3D coordinates of the pixels of the query image are computed and the so obtained 2D-3D correspondences are used to compute the camera pose with a P3P-RANSAC loop. As there is no image retrieval step, the only parameters available are the number of inliers and the coverage score of the query image. Given the reduced number of input parameters, a tiny neural network, with a single layer and two neurons, is trained on the InLoc dataset as described before. This network is then used to compute the confidence score of the estimated poses in Cambridge Landmarks, where the pose is considered correct using an error threshold of 0.35 m and 5°. Cambridge Landmark is a fairly simple dataset and most competitive pipelines can achieve accuracies over 90% [23, 24, 48]. Clearly, if in a dataset most poses can be assumed to be correct, the importance of the confidence estimation decreases. The purpose of this experiment is thus not to outperform the inliers count, but to show that our method is independent of the dataset used for training and can still be applied to unforeseen data.

Table 6.5. *Performances of our InLoc trained algorithm in Cambridge Landmarks dataset.*

Scene	AUC (inls count)	AUC (our)
Great Court	80.08%	83.41%
King's College	94.88%	95.30%
Old Hospital	89.64%	88.25%
Shop Facade	98.95%	98.96%
St. Mary's Church	99.80%	99.81%
Total	**93.79%**	**94.75%**

Table 6.5 reports the AUC values obtained when classifying poses as incorrect or incorrect with the number of inliers alone and with our algorithm. As can be noticed, our confidence estimator gives good performances, even outperforming the standard inliers count for most scenes.

6.3.2 Aachen

Aachen day/night dataset presents several images from the Aachen city during day- and night-time. The pipeline used by the authors in [22] combines coordinates regression and image retrieval techniques. Particularly, for a given query image 10 candidate poses are

produced. In their original work, the authors choose the best pose based on the number of inliers. Ground truths for the Aachen dataset are not available and hence the PR-curve cannot be computed. However, an online evaluation tool [49], computing the accuracy for the estimated poses is available. The accuracies are computed at three different thresholds: 0.25 m, 2° / 0.5 m, 5° / 5m, 10° for day scenes and 0.5 m, 2° / 1 m, 5° / 5m, 10° for night scenes. In our experiments, we use our confidence estimation algorithm to rerank the candidates of each query image. As this approach is retrieval based, we can use our full model with all five input parameters. Again, we choose to use a smaller neural network, with two layers and two neurons on each, to improve generalization to different datasets. The network is trained with InLoc with the standard 1 m, 10° error threshold.

Table 6.6. *Improved accuracies on Aachen when adding our method to the pipeline of [22].*

Aachen	Baseline	our
Day	70.9 / 81.9 / 91.6	71.4 / 82.6 / 92.0
Night	32.7 / 43.9 / 64.3	33.7 / 44.9 / 65.3

Table 6.6 shows the accuracies obtained with the baseline algorithm and our method. Ranking candidate poses with our method reaches higher accuracies at all thresholds for both day and night scenes, indicating our method, despite being trained with InLoc at a specific error threshold, can be flexibly integrated into different visual localization pipelines for better results.

7 CONCLUSIONS AND FUTURE WORK

This thesis investigated how to asset the confidence of estimated poses in visual localization pipelines, using InLoc as case study. We showed that the number of inliers from the PnP-RANSAC solver, the standard confidence measure, is not particularly robust, especially for indoor scenes with significant redundancy. The confidence estimation problem was formalized as a binary classification problem and we proposed a novel confidence metric, computed by a neural network and taking into account different factors, such as the number of inliers, their spatial distribution in the cameras similarity. Our confidence metric outperformed the standard inliers count. Furthermore, we showed the proposed method can also be used to improve the accuracy of the algorithm, reaching state-of-the-art results. During the study, it was emphasized the importance of inliers spatial distribution in query and database images and it was showed how this information can be effectively exploited to improve both confidence estimation and pose estimation. Finally, it was shown that the proposed approach does not depend on the data used for training and can be flexibly applied to different datasets and pipelines.

The proposed confidence estimation algorithm was used as a postprocessing step, quantifying how reliable the final result is. For future development, it can be interesting to modify the algorithm to predict the achievable reliability of the pose before or at early stages of the pipeline, e.g. training a convolutional neural network to predict a confidence measure directly from the query image. This new version, in addition to improving the robustness of visual localization pipelines, would also allow to save computation time, as query images classified as too challenging could already be rejected before computing the pose. In mobile robot applications, where a camera films continuously the environment, this predictive confidence estimator could be used to choose what frames are more likely to produce an accurate camera pose. In future development, the proposed confidence measure could also replace the inliers count inside the PnP-RANSAC loop, to achieve more accurate camera poses.

REFERENCES

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[2] R. Arandjelovic and A. Zisserman. All About VLAD. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013.

[3] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2012.

[4] P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications* (Apr. 1983).

[5] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. June 2005.

[6] O. Chum, J. Matas and J. Kittler. Locally Optimized RANSAC. *Pattern Recognition*. Springer Berlin Heidelberg, 2003.

[7] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* (Jan. 1967).

[8] D. Cox, J. Little and D. O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Undergraduate texts in mathematics. Springer-Verlag, 1992.

[9] J. L. Crassidis. Sigma-point Kalman filtering for integrated GPS and inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems* (Apr. 2006).

[10] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM* (1981).

[11] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Dec. 2001.

[12] Y. Gal and Z. Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2015.

[13] C. Geyer and K. Daniilidis. Paracatadioptric camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (May 2002).

[14] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. MIT Press, 2016.

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[16] J. Kannala and S. S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Aug. 2006).

[17] A. Kendall, M. Grimes and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. *2015 IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[18] A. Kendall and R. Cipolla. *Modelling Uncertainty in Deep Learning for Camera Relocalization*. 2015.

[19] A. Kendall and R. Cipolla. Geometric Loss Functions for Camera Pose Regression With Deep Learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[20] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *J. Opt. Soc. Am. A* (Feb. 1991).

[21] N. Le Dortz, F. Gain and P. Zetterberg. WiFi fingerprint indoor positioning system using probability distribution comparison. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2012.

[22] X. Li, J. Verbeek and J. Kannala. *Hierarchical Joint Scene Coordinate Classification and Regression for Visual Localization*. 2019.

[23] X. Li, J. Ylioinas and J. Kannala. *Full-Frame Scene Coordinate Regression for Image-Based Localization*. 2018. arXiv: `1802.03237 [cs.CV]`.

[24] X. Li, J. Ylioinas, J. Verbeek and J. Kannala. Scene Coordinate Regression with Angle-Based Reprojection Loss for Camera Relocalization. *The European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

[25] J. Z. Liang, N. Corso, E. Turner and A. Zakhor. Image Based Localization in Indoor Environments. *2013 Fourth International Conference on Computing for Geospatial Research and Application*. July 2013.

[26] H. Lim, J. Lim and H. J. Kim. Real-time 6-DOF monocular visual SLAM in a large-scale environment. *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014.

[27] H. Lim, S. N. Sinha, M. F. Cohen and M. Uyttendaele. Real-time image-based 6-DOF localization in large-scale environments. *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012.

[28] P. S. W. R. H. LL.D. XXII. On quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1846).

[29] D. G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Sept. 1999.

[30] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* (Nov. 2004).

[31] E. Marchand, H. Uchiyama and F. Spindler. Pose Estimation for Augmented Reality: A Hands-On Survey. *IEEE Transactions on Visualization and Computer Graphics* (Dec. 2016).

[32] I. Melekhov, J. Ylioinas, J. Kannala and E. Rahtu. Image-based Localization using Hourglass Networks. IEEE International Conference on Computer Vision Workshops (ICCVW). 2017.

[33] R. Raguram, J.-M. Frahm and M. Pollefeys. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. *The European Conference on Computer Vision (ECCV) Workshops*. 2008.

[34] R. Raguram, J.-M. Frahm and M. Pollefeys. Exploiting uncertainty in random sample consensus. *2009 IEEE 12th International Conference on Computer Vision* (2009).

[35] P.-E. Sarlin, C. Cadena, R. Siegwart and M. Dymczyk. *From Coarse to Fine: Robust Hierarchical Localization at Large Scale*. 2018. arXiv: `1812.03506 [cs.CV]`.

[36] T. Sattler, B. Leibe and L. Kobbelt. Improving Image-Based Localization by Active Correspondence Search. *The European Conference on Computer Vision (ECCV) Workshops*. Ed. by A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato and C. Schmid. 2012.

[37] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl and T. Pajdla. Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[38] T. Sattler, Q. Zhou, M. Pollefeys and L. Leal-Taixe. Understanding the Limitations of CNN-Based Absolute Camera Pose Regression. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[39] D. Scaramuzza, F. Fraundorfer and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. *2009 IEEE International Conference on Robotics and Automation*. May 2009.

[40] J. L. Schonberger and J.-M. Frahm. Structure-From-Motion Revisited. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[41] P. Scovanner, S. Ali and M. Shah. A 3-dimensional Sift Descriptor and Its Application to Action Recognition. *Proceedings of the 15th ACM International Conference on Multimedia*. MM '07. 2007.

[42] A. Sharif Razavian, H. Azizpour, J. Sullivan and S. Carlsson. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2014.

[43] C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. *2008 IEEE Conference on Computer Vision and Pattern Recognition*. June 2008.

[44] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* (2014).

[45] Sivic and Zisserman. Video Google: a text retrieval approach to object matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*. Oct. 2003.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* (2014).

[47] R. Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Springer-Verlag, 2010.

[48] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla and A. Torii. InLoc: Indoor Visual Localization With Dense Matching and View Synthesis. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[49] *The Visual Localization Benchmark*. `https://www.visuallocalization.net/`. Accessed: 2019-11-15.

[50] P. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding* (Aug. 2000), 138–156.

[51] V. Usenko, J. Engel, J. Stückler and D. Cremers. Direct visual-inertial odometry with stereo cameras. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016.

[52] E. Vincent and R. Laganiere. Detecting planar homographies in an image pair. *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.* June 2001.

[53] E. Wijmans and Y. Furukawa. Exploiting 2D Floorplan for Building-Scale Panorama RGBD Alignment. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017).

[54] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Aug. 2003).

[55] C. Yang and H. Shao. WiFi-based indoor positioning. *IEEE Communications Magazine* (Mar. 2015).

[56] J. Zhou and D. Wang. Solving the perspective-three-point problem using comprehensive Gröbner systems. *Journal of Systems Science and Complexity* (Oct. 2016).