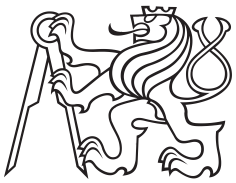
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Visual Localization of Mobile Robot

Vojtěch Pánek

Supervisor: Ing. Vladimír Smutný, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: Robotics
August 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Pánek Vojtěch** Personal ID number: **457211**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Master's thesis details

Master's thesis title in English:

Visual Localization of Mobile Robot

Master's thesis title in Czech:

Určení polohy mobilního robotu z kamerových dat

Guidelines:

1. Study literature dealing with the indoor visual localization.
2. Propose/select algorithm for indoor localization using upward looking fish-eye camera.
3. Implement and test the algorithm on real data.
4. Evaluate experimental results and make conclusion.

Bibliography / sources:

[1] H Taira, M Okutomi, T Sattler, M Cimpoi, M Pollefeys, J Sivic, T Pajdla, A Torii: InLoc: Indoor Visual Localization with Dense Matching and View Synthesis, CVPR 2018
[2] O. Chum, M. Perdoch, J. Matas: Geometric min-Hashing: Finding a (Thick) Needle in a Haystack CVPR 2009
[3] Luca Mainetti ; Luigi Patrono ; Ilaria Sergi: A survey on indoor positioning systems, 2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)

Name and workplace of master's thesis supervisor:

Ing. Vladimír Smutný, Ph.D., Robotic Perception, CIIRC

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **13.12.2019** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

Ing. Vladimír Smutný, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to express my deep gratitude to my supervisor Vladimír Smutný for his patience, guidance, and numerous pieces of advice throughout the writing of the thesis. I would also like to thank Tomáš Pajdla for consultations on the topic of the state of the art papers. Finally, I wish to thank my family and girlfriend for their support.

Declaration

I declare that the presented work was developed independently and that I have listed all source of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 14. 8. 2020
Vojtěch Pánek

. . . . . .

Abstract

This work aims to examine the current state of the art in visual localization and find a suitable solution for an indoor mobile robotic platform equipped with a single upward-looking RGB camera and fisheye lens. The system should be able to perform longterm global localization in changing indoor industrial or office environment. A dataset of localized omnidirectional images was captured and used for evaluation of the performance of selected methods. VLAD and NetVLAD descriptors were tested in combination with tiled panorama representation. A simple localization method based on taking the position of the most similar database image is proposed as the solution.

Keywords: visual localization, mobile robotics, fisheye, panorama, VLAD, NetVLAD, tiled panorama descriptor

Supervisor: Ing. Vladimír Smutný, Ph.D.

Abstrakt

Tato práce se zaměřuje na prozkoumání současné situace na poli určování polohy z kamerových dat a na navržení vhodného řešení pro mobilní robotickou platformu vybavenou vertikálně orientovanou RGB kamerou s fisheye čočkou. Navržený systém by měl být schopen dlouhodobě vykonávat globální lokalizaci v měnícím se vnitřním prostředí výrobních závodů a kancelářských budov. Pro ověření funkčnosti vybraných metod byl nasnímán dataset fisheye obrazů spolu s jejich polohou. VLAD a NetVLAD deskriptory byly otestovány v kombinaci s dlaždicovou reprezentací panoramat. Jako řešení byla navržena jednoduchá metoda, určující aktuální polohu na základě polohy nejpodobnějšího obrazu z databáze.

Klíčová slova: vizuální lokalizace, mobilní robotika, fisheye, panorama, VLAD, NetVLAD, dlaždicová reprezentace panorama

Překlad názvu: Určení polohy mobilního robotu z kamerových dat

Contents

Figures

Tables

Chapter 1

Introduction

This thesis is part of the Hermes project ongoing in the RMP-ROP group at CIIRC CTU in Prague. The project aims to develop a prototype of an autonomous mobile transportation platform for indoor environments. The cornerstone of the system is NDT SLAM, which needs a coarse pose estimate for initialization. The initial pose should be provided by the visual localization approach presented in this thesis.

The rest of this chapter provides an overview of the task, used hardware and the target environment. Chapter 2 contains short survey of the visual localization field and applicable technologies. Chapter 3 describes the process of selecting suitable technologies, discusses the selected methods and proposes a possible task solution. The implementation and programming point of view is summarized in Chapter 4, and finally, the description of the captured dataset and undertaken experiments is present in Chapter 5.

Specification of the problem

The implemented system should be capable of global localization with previous knowledge of the environment using fisheye camera data. The targeted use case is the estimation of position after system startup, where the position could change during the period when the system was turned off (similar to the kidnapped robot problem). The time demands are not as strict as during robot motion, where the localization process should have a sub-second duration. The environment should not be adjusted in any way for the robot operation, which demands the localization method to work markerless.

Experimental platform

We use the Clearpath Jackal four-wheel holonomic robot with ROS (Robot Operating System). The robot provides wheel odometry and IMU data. Moreover, the platform is equipped with a single lidar (Sick TiM561) and a single RGB camera (Basler daA2500-14uc) with a fisheye lens (Sunex PN DSL215). The lidar scans in a horizontal plane approximately 30 *cm* above the ground.

Camera pose is assumed to be restricted to 3 DOF (Degrees of Freedom), fixedly mounted to the mobile robot performing a planar motion. Height above the floor is constant, and the camera optical axis is assumed to be perpendicular to the floor plane (roll and pitch rotations of the robot are zero).

Environment and possible challenges

We assume that the robot will operate in an indoor environment, specifically interiors of office buildings or production halls, transporting production material, goods or paper documents.

Outdoor global localization can be today solved mainly by the use of GNSS (Global Navigation Satellite System), which are able to achieve decimeter position precision with publicly available services and centimeter precision with commercial services [1]. Problems start to occur in urban areas, where buildings can cause attenuation of signal from satellites, significantly reducing the precision. Indoor environments are an extreme case, often almost blocking the signal and making the satellite navigation unusable at all. That brings an excellent opportunity to solve the localization e.g., by visual algorithms. However, the indoor environment brings many challenges, which will be briefly enumerated in the following paragraphs.

Lighting conditions change throughout the day with natural illumination cycles, weather, and seasonal variations, and artificial light sources. Highly reflective surfaces changing appearance with view angle can be present. The physical structures (parts of buildings, furniture) and texture patterns can be repetitive. Walls and other surfaces are commonly textureless.

We cannot assume the ceiling to be in constant height above the camera. The ceiling height in manufacturing plants can vary from tens of centimeters in low underpasses to tens of meters in production halls. The suspended ceilings in office buildings can change levels between rooms.

The environment contains dynamic elements as people, forklifts, cranes, doors, which cannot be used for localization and which can cause camera occlusion. The localization should be able to function in the long run, so we have to also count on changes of more static elements such as furniture and production machines.

On the other hand, the indoor environment is relatively stable in the means of seasonal or weather changes such as snow cover, fog, and also vegetation growth and its seasonal cycles.

Chapter 2

State of the art

Robot localization using a camera fixed on the robot is equivalent to the camera pose estimation task. There are two major approaches to pose estimation in the computer vision field. The first one is the visual place recognition task, which focuses on coarser localization e.g., identification of room in a building. The second approach is the visual localization task aiming to retrieve precise camera coordinates.

Localizing a robot is different from the localization of a single photo because the camera is mobile and can produce a video feed or at least a sequence of images from multiple places. This feature allows to exploit not only "discrete" visual localization methods, but also "continuous" ones, such as visual odometry or visual SLAM. This thesis concluded into the implementation of system prototype used for single-time-per-run location initialization, where continuous methods could not be appropriately utilized. However, the foregoing visual localization field survey counted also on the possibility of continuous localization so that the methods will be briefly mentioned in Section 2.2.

The field of visual localization and computer vision generally is being explored since the invention of the digital camera. The advances in digital imagery and computational power performed in the last decade resulted in much faster progress and growth of papers published on the topic. As a consequence, no survey can fully cover the whole field, and only selected works, which came across during the thesis preparation, are mentioned in this paper.

2.1 A survey of surveys on visual localization

We should start with available surveys of the available methods to see all possible ways, which can be followed. A good introduction could be the survey from Zhou [2], briefly mentioning all aspects of visual localization without going into details of particular papers.

The survey from Piasco et al. [3] goes much deeper, providing a comprehensive overview of available visual-based localization methods. It introduces

a division of methods into two main categories: direct and indirect. Direct methods estimate the camera pose straight away from the image data, and the survey mentions e.g., PoseNet from Kendall et al. [4] as an example of such a method. Indirect methods employ an intermediate step, most often in the form of most similar database image retrieval, and InLoc from Taira et al. [5] can be taken as an example. In addition to the localization methods overview, the survey also provides an overview of possible image representations and available datasets.

The paper by Morar et al. [6] is similarly broad as the previous one, however, it specializes in the indoor environment and classifies methods by different criteria. The first divider is an environment representation that can be a 3D model, image or feature database, saved locations of static cameras, or markers placed in the environment. The second employed criterion is a type of used cameras, which can be mobile or statically placed and 2D or 3D. The last measures are whether the method uses artificial markers, or uses the natural appearance of the environment and whether some kind of AI system is employed. The survey divides research papers into groups based on the criteria and mentions available information about used datasets, performance results, and computation time. This approach allows us to quickly find several papers relevant to our hardware setup and desired system concept.

2.2 Local visual localization methods

The task of camera pose transformation estimation from a sequence of images is called Visual Odometry and was introduced in the work of Nistér et al. [7]. The original algorithm first detects Harris corners, then matches the features in consecutive image frames by simple correlation over windows around the detected corners. The matched features are triangulated, and obtained 3D points are used for camera pose estimation. The work provides algorithms for both the monocular and the stereo system. The main difference of algorithm for the monocular system is the use of SfM (Structure from Motion) over MVS (Multi-View Stereo) 3D point triangulation.

An example of visual odometry use is motion estimation of MER (Mars Exploration Rover) robots described in the paper of Cheng et al. in [8] and later evaluated in [9]. The papers are interesting mainly because of the reliability requirements on the rovers, operating several years predominantly in autonomous mode. The system uses the NAVCAM stereo camera and employs an error model for greater robustness and precision. The difficulties experienced during MER missions are, in many ways, similar to our own. Both systems operate in GNSS denied environments, solve unreliable odometry measurements caused by wheel slippering, and inaccurate IMU (Inertial Measurement Unit) pose estimation.

Another example is the work of Scaramuzza et al. [10]. The camera setup is

very similar to our own, which is an omnidirectional camera placed at the roof of a vehicle. The work is based on the fact that the points in the ground plane observed from different views are related by a homographic transformation. The homography is estimated from matched SIFT features in the RANSAC scheme and decomposed to obtain a camera pose transformation between the frames. Our camera does not see the majority of the ground plane, which could, therefore, be replaced by a ceiling in our case.

Of course, an end-to-end neural network solution also exists, e.g., DeepVO from Wang et al. [11]. It uses CNN (Convolutional Neural Network) for the extraction of features on the concatenation of a pair of subsequent images. The output of the last CNN layer is passed to the RNN (Recurrent Neural Network) part of the system, which cares about motion modeling.

Some papers deal with the detection and tracking of geometric features specific to the environment. The paper of Xu et al. [12] extracts lines from dropped ceiling tiles employing Hough transform. Known dimensions of dropped ceiling tiles can be used for initial camera pose estimation with the PnP algorithm. Position estimation during motion is provided by tracking of detected line or point features. In the paper of Krotkov [13], the vertically oriented lines in the environment are being detected and taken as landmarks. The camera-landmark rays are computed, and an ideal matching between the rays and prepared landmark map is found, resulting in simple localization within the map.

2.3 Global visual localization methods

The paper of Lowe et al. [14] is today one of rather older works in the field of visual localization. This method performs 3D reconstruction of SIFT features detected by a stereo camera and incrementally generates a 3D landmark map. All the reconstructed landmarks also keep information about positional uncertainty. The paper proposes to track the landmarks in a SLAM like system or to match all visible landmarks to map and localize a robot globally.

Paper of Torii et al. [15] deals with localization in urban environments under significant changes. The used database is prepared from cutouts of Google Street View panoramas. Description of the images is done by a dense sampling of RootSIFT features in multiple scale levels and subsequent aggregation by VLAD. The dense sampling makes the image representation less invariant to viewpoint change, which is being overcame by virtual viewpoints synthesis. The virtual panoramas are computed in a regular grid around streets thanks to depth maps provided by Street View.

InLoc is an indoor localization system invented by Taira et al. [5]. The used database consists of cutouts from panoramas generated by an RGBD scanner. The system uses NetVLAD [17] image descriptor for the first retrieval

stage. Pixel-to-pixel matching is done for the retrieved images, and candidate query positions are computed by P3P-RANSAC. This first stage is called DensePE (Dense Pose Estimation). The second stage is DensePV (Dense Pose Verification). Virtual viewpoints at the places of estimated camera positions are generated from the nearest panorama scan, and a comparison of densely sampled RootSIFT patches between virtual viewpoints and query image is applied to confirm the match.

The pose verification, which is the last step of the InLoc system, was further extended in the paper of Taira et al. [40]. The first improvement is the introduction of a scan-graph, which allows to generate virtual viewpoints from multiple near database scans. The other novelty is DenseNV (Dense Normal Verification), which is a similar approach to DensePV, but the surface normals are taken instead of feature descriptors. Combined system DensePNV takes normals consistency as a weight for descriptor comparison. Semantics is another modality that can be taken into account. 3D point clouds and query images are being semantically labeled, and the semantic classes are divided into superclasses depending on the reliability of the object position. The pixels belonging to unreliable (easily movable) objects are then ignored during the dense verification steps.

Another work of Torii et al. [42] deals with localization of panoramas. A fixed number of vertical tiles is cut out from the panorama and described with BoW or VLAD descriptor, generating an ordered list of tile descriptors. The retrieval step is performed with the circular ordering of the tiles in mind. An optimal circular shift between the tiled description of query and database image is sought during retrieval.

One of the recent works dealing with visual place recognition is the paper of Camara et al. [16]. It uses VGG-16 CNN proposed by Simonyan et al. [44] and extracts outputs of two convolutional layers. The outputs of those two layers are spatially aggregated into a fixed number of vectors, which are similar to descriptors of densely sampled features. The later layer with less spatial, but more semantical information is used in the filtering stage, preselecting similar images while ignoring the geometrical arrangement of the features. The output of the earlier layer with more spatial information is used for re-ranking, where corresponding descriptor patches and other patches in the same row and column are being checked, emphasizing their geometrical order. The paper has been revisited in another work of Camara et al. [43], further polishing parameters of spatial aggregation and changing the re-ranking step, so it checks not only the similarity of rows and columns of the current patch but the similarity of features in the square area around the patch.

2.4 Image representations overview

In the image retrieval task, we need some efficient image representation, which allows us to compare the images and find the ones capturing the same scene. There is a vast number of ways, how to describe an image. This section tries to briefly introduce and classify the ones, which were encountered along the way of writing this thesis.

The first divider of image representation methods could be whether we use a single description of the whole image, or we find some interesting local image features, which are stable despite scene changes and describe them instead. Both of these approaches are covered in the following subsections.

Other approaches exist, some exploiting geometric features in the image or geometric relations between local features, such as in papers from Chum et al. [48, 49]. These methods seem very attractive, mainly because they could largely improve the robustness of the local features, however, none of them was tested because of the unavailability of any public implementation and lack of time for writing our own implementation.

Many methods use 3D data, either in the form of matching 2D image features to 3D map or in the form of reconstruction of the image features and matching 3D point clouds on each other. The advantages and disadvantages of these approaches are well described in the paper by Sattler et al. [39]. The necessity of consistent 3D map maintenance of a dynamic environment is the major reason why this thesis tries to perform the localization without using any kind of 3D map.

To have a complete list of all possible image representations, semantical approaches should also be mentioned. Utilization of outputs or semi-products of object recognition networks could provide a quiet robust system as described in the paper of Taira et al. [40], which is further explained in Section 2.3.

2.4.1 Local image features

Local feature processing is standardly divided into two subsequent steps. Firstly, the features (sometimes called key points) have to be detected, that is, interesting patches of the image have to be localized. Then, the found patch (area of defined size around detected key point) is described, and a feature descriptor, which is usually a numerical array is generated. These two steps are done by the feature detector and feature descriptor, respectively. Some methods specify both the detector and the descriptor, and some specify only single of them and leave the choice of the other one on the user.

Algorithms for local features detection and description are being evaluated

on the basis of invariance to certain changes. Invariance to a change means that if we apply the change to an image, the features will be detected on the same (or corresponding) spots, and generated descriptors will be the same. Let us mention invariance to intensity change, rotation, scaling, and affine geometric transformation.

The process of feature detection on specific places in the image described above can sometimes be called sparse feature extraction. The opposite is the dense feature extraction, where the detection step is being replaced by sampling in a regular grid. This technique can be beneficial in environments with texture-less surfaces, where some detectors could not find any valid key points.

Harris corner detector introduced by Harris et al. in [24] is probably the most well-known feature detection algorithm. It searches for corners, which are patches unique in their vicinity. That is done by detecting local maximums of thresholded (corner) response function. The response function has high value at the points where the image function changes significantly in all directions. The problem comes with a lack of scale invariance.

FAST(Features from Accelerated Segment Test) feature detector presented by Rosten et al. in [25] generates a circle of specified radius around each pixel and sums the number of the pixels on the circle, which differ from the central one by the given threshold. If that is more than $\frac{3}{4}$ of the pixels on the circle than the central pixel is accepted as a feature. The detector is not scale invariant.

BRIEF (Binary Robust Independent Elementary Features) is a feature descriptor from Calonder et al. [26]. It takes a smoothed image patch and generates a binary vector of length n_d by performing n_d intensity comparison tests on sampled pairs of patch pixels. The paper proposes several sampling strategies, where the one using single Gaussian distribution performs the best.

ORB (Oriented FAST and Rotated BRIEF) by Rublee et al. in [27] is presented as an efficient alternative to the other existing systems. It uses the FAST feature detector on multiple scale pyramid levels. Specified constant number of features with the highest Harris corner response are accepted on each level. Orientations of features are computed using the intensity centroid method introduced by Rosin [28]. The resulting detector is named oFAST (Oriented FAST). BRIEF descriptor is applied on a rotationally normalized patch with pre-learned pairs sampling strategy to improve the variance of the tests. This alteration of BRIEF descriptor is called rBRIEF (Rotated BRIEF).

SIFT (Scale-Invariant Feature Transform) by Lowe [29] is probably the most used detector and descriptor system, with many variations aimed at better performance. The detector part searches for extrema of DoG (Difference of Gaussian) filtered images with multiple scale levels. The keypoint locations

are further filtered as described in another paper by Lowe [30]. The prevailing gradient orientations in feature patch are retrieved, which can be used for feature orientation normalization, providing rotational invariance. The patch around the key point is divided into 16 4x4 pixel blocks, and 8-bin HOG (Histogram of Oriented Gradients) is generated for each, generating 128 element descriptor vector.

Upright SIFT is a variant of SIFT, which omits feature orientation normalization, making the descriptor rotational invariant, but improving descriptiveness in applications with stable camera orientation as can be seen in paper of Baatz et al. [31].

RootSIFT described in the paper of Arandjelović et al. [32] is obtained by applying an element-wise square root on the L1 normalized SIFT descriptor. The matching procedure is done using the Euclidean distance, which is equivalent to the use of Hellinger distance on the original SIFT descriptors.

D2-net of Dusmanu et al. [41] is an approach using CNN for local features detection and description. The tensor output of $H \times W \times D$ dimensions, which is generated by a CNN, is interpreted in two ways. D layers of $H \times W$ size are taken as response maps similar to ones used for Harris corners search, and features are detected at local maximums. The other interpretation of the tensor is that each vector of length D in the $H \times W$ grid is taken as a descriptor for the corresponding image patch. Image pyramids are applied to ensure scale invariance.

2.4.2 Global image representations

Global image descriptors represent the whole image with a single vector, allowing straightforward similarity search with the possibility of a low memory footprint. The representation can be derived directly from an image, or aggregated from a set of described local features.

BoW (Bag of Words), BoVW (Bag of Visual Words) or BoF (Bag of Features) is an aggregation algorithm introduced by Csurka et al. [33] and later extended by Sivic et al. [34], following procedures from text analysis. It prepares a vocabulary by dividing descriptor space into clusters and then describes the image by numbers of local feature descriptors belonging to each of the clusters.

FV (Fisher Vector) is another aggregation approach used for the first time for image description by Sánchez et al. in their paper about image classification task [35]. The BoW descriptor space clustering is replaced by soft assignment to GMM (Gaussian Mixture Model), and the resulting score is computed as a gradient of log-likelihood of the feature descriptors on the model.

VLAD (Vector of Locally Aggregated Descriptors) is an aggregation technique by Jégou et al. [36], which can be taken as a simplification of FV. The

hard feature-cluster assignment from BoW is used, and pooling is done by summing the distance of all features corresponding to the cluster from the cluster centroid along each dimension. The experiments indicate an increase in retrieval performance compared to BoW and FV.

NetVLAD of Arandjelović et al. [17] is the only representative of direct image descriptors in this list. It uses an off-the-shelf CNN (Convolutional Neural Network) with a custom-designed aggregation layer, which imitates the VLAD descriptor pooled from densely extracted features. It is currently considered one of the best image representations for the place recognition task.

Chapter 3

Solution proposal

This chapter describes the course of solution development and discusses the used methods in more detail.

3.1 Dataset preparation

The robotic platform which has been used for dataset capturing runs ROS Kinetic on Ubuntu 16.04. The script provided by colleague Václav Plavec was used for publishing camera data to a ROS topic with a given period of driven distance measured by odometry. The topics containing captured images and the data needed for NDT SLAM were recorded into ROS bag files. NDT SLAM package by Nováček [18] was launched offline on the recorded data, with the possibility of slowing down the data playback in case of localization issues. The database processing script from implemented `visual_localization` package is being launched together with NDT SLAM and matches the results of localization with captured images, generating a text file with the position of each captured image.

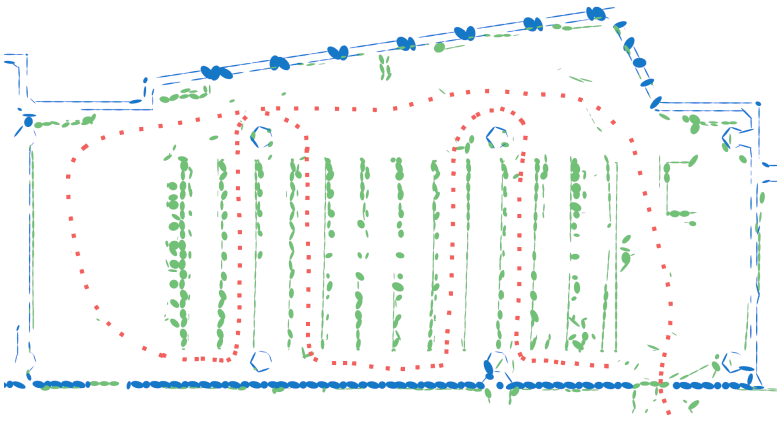


Figure 3.1: Dataset images position estimation performed by NDT SLAM and visualized by rviz. Estimated positions are marked by red color, blue part of the NDT map was generated from CAD building drawings, green part was generated dynamically by the SLAM system.

3.2 Early development stages and implemented ROS package

Our first idea of how to solve the problem of visual localization was to detect the local features in the fisheye image, find similar images by performing feature matching between images, apply triangulation, and estimate camera pose by PnP (Perspective-n-Point) algorithm. With this idea in mind, an initial part of the system was implemented for ROS in the `visual_localization` package. The package contains functions for detection and descriptions of local features, assignment of a pose to the database images based on localization output of NDT SLAM system, and matching of local features between described images. The implemented package was used through dataset collecting, to assign pose from NDT SLAM to the captured images. Unfortunately, there has not been enough time to integrate the following parts, evaluated in experiments, into the package, so these are separate pieces of software.

There have also been a few tries to use the COLMAP system from Schönberger et al. [51, 52]. COLMAP is SfM and MvS pipeline, not meant to work as a visual localization system. However, our idea was that the camera pose computation, which is already performed during the SfM process, could be used for robot localization. This approach was abandoned after several tests because the localization took too long, failed to converge most of the time, and the converged results were deformed in various ways. We later found out that none of the camera models implemented in COLMAP is able to model our camera lens with FoV larger than 180°.

3.3 InLoc

A literature survey was performed simultaneously with the implementation of the `visual_localization` package. It was clear early that many more advanced approaches exist. One of such systems is InLoc, briefly mentioned in 2.3, which had for us an advantage of the availability of personal consultations with the authors. The InLoc system has available Matlab code and also provides a testing dataset. That allowed us to test if we were able to install the system so that the results on the dataset match the results from the paper. The results of the localization can be evaluated to ground truth by The Visual Localization Benchmark [47]. The performance did not match the paper exactly, which was explained by the use of the P3P algorithm within RANSAC, producing slight variations in results.

InLoc uses precise RGBD panoramas for database and perspective images as queries. To provide the same query image type, for which the method was prepared, we studied, implemented, and tested fisheye to perspective image mapping. The datasheet of the used fisheye lens (Sunex PN DSL215) [19] mentions that the lens geometry can be modeled by equidistant (f-θ)

model with 6% distortion. The mapping can be done by taking a perspective camera of selected parameters with the same center as the real fisheye camera, reprojecting the perspective image's pixels to a unit sphere, and projecting these 3D points into the fisheye camera image using the equidistant model. Values at the projected points are retrieved using an interpolation.

Many other fisheye projection models exist, some of them with available implementation. The problem appears when we want to apply the projection on a lens with FoV (Field of View) greater than 180° (sometimes called super fisheye). Many projection algorithms perform division by Z coordinate of a 3D point, which for the rays with angles (from the optical axis) greater than 90° results in projection into the wrong image half-plane, if not handled correctly. The model from Scaramuzza et al. [21], which is used in Matlab Computer Vision Toolbox and the OpenCV fisheye camera model, contain this flaw and are not usable for our purposes. OpenCV public code repository already contains the improved code version [20], which is able to handle the super fisheye case, but the code was not published yet in main branch. Worth mentioning is the model from Mei et al. [22], which comes with the camera calibration toolbox for Matlab [23] and is able to perform the super fisheye projection.

The 3D data in the InLoc database is used for camera pose computation and virtual view synthesis. Our robotic platform has only an RGB fisheye camera and planar lidar, which cannot substitute the RGBD scanner used for generating InLoc database. Our approach also tends to eliminate the need for on-site preparations before starting the robot operation, which discourages us from manual scanning the environment. That was the leading cause, why we did not continue in testing this approach.

3.4 Tiled descriptors

We used the idea of describing a panorama by a set of tiles introduced in the paper of Torii et al. [42], adjusted it for our environment and tested in combination with VLAD and NetVLAD descriptors. The approach seems interesting for our application because generating a panorama from an upward-looking fisheye image is a simple operation.

Our use case differs in the target environment. Indoor localization meets different challenges than outdoor, where the main one for tiled descriptors is that the objects have a smaller distance to the camera, and consequently, the scene changes significantly even for small camera motions. An experiment, described in Section 5.3, was done to find out how should be the tile parameters adjusted for the target environment.

The paper performs tests on two datasets from the outdoor urban envi-

ronment. Pittsburgh dataset used Google Street View panoramas for both the database and query images. Their own dataset, called Shibuya, uses Google Street View for database and panoramas captured by various cellphone hardware and software for query images. The performance graphs indicate that the methods have much better results on the Pittsburgh dataset, but it is not clear if the cause is that the Shibuya dataset uses different image sources for database and query, or there exists any other cause. Our approach tends to use the same fisheye camera for capturing both the database and query images, which eliminates possible loss of performance while matching images captured by different methods.

VLAD and NetVLAD descriptors were selected for describing the tiles because of their performance and popularity for purposes of visual localization and visual place recognition tasks. The tiled representations are tested in detail in Section 5.4 along with VLAD and NetVLAD descriptors applied directly on fisheye images.

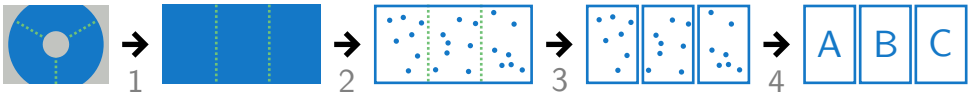


Figure 3.2: Fisheye image description procedure by Tiled VLAD: 1. the fisheye image is transformed into a panorama, 2. SIFT features are detected in the panorama and described, 3. local features are assigned to tiles, 4. local features are aggregated into VLAD descriptor for each tile

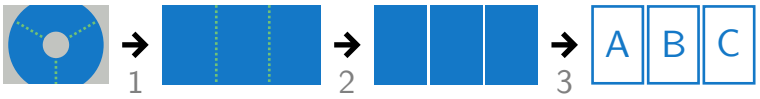


Figure 3.3: Fisheye image description procedure by Tiled NetVLAD: 1. the fisheye image is transformed into a panorama, 2. the panorama is divided into tiles, 3. individual tiles are described by NetVLAD

The procedure of description of fisheye image by Tiled VLAD descriptor is presented in Figure 3.2. We have to obtain the panorama from the fisheye image, as described in Section 4.1.1. The SIFT features are detected and described upright by RootSIFT descriptors. The descriptors are divided into tiles and aggregated into VLAD representations. In the case of NetVLAD, which can be seen in Figure 3.3, the panorama is divided into tiles first, and the descriptor is computed directly on the tile images. The result of image description phase is a set of N described tiles with circular ordering constraint.

The computation of the distance of two described images is shown in Figure 3.4. It starts with generating all possible circular shifts of the query image tiled descriptor. The Euclidean distance is used to compute individual tile

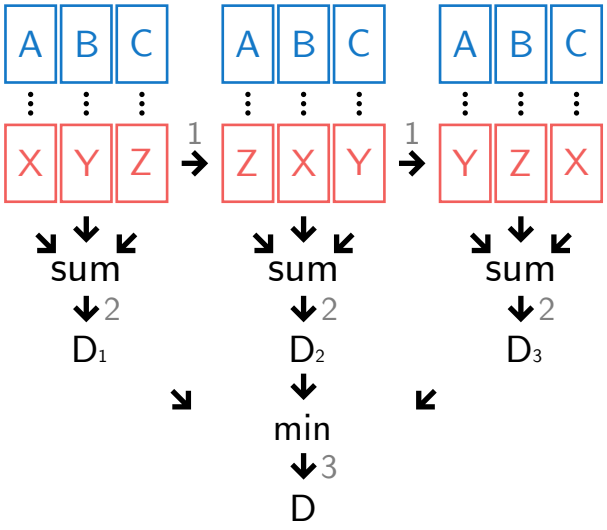


Figure 3.4: Computation of description distance of two images described by tiled descriptor: 1. all possible circular shifts of query image (red) tiles are generated, 2. descriptor distance between database image (blue) and query image (red) representations is computed for each shift by taking the sum of Euclidean distances of corresponding tile descriptors, 3. the minimal distance over all possible shifts is taken as the resulting distance value

descriptors' dissimilarity, and the distances of individual tiles are summed to obtain the distance between the tiled representations with particular shift. The minimal distance among the shifts is taken as the resulting distance of the tiled representations. The angle of the best circular shift is kept for orientation estimate. This procedure is applied to each database image, so all the distances are obtained, and the database can be ordered by similarity to the query image.

The position of the most similar database image is taken as the position estimate of the query image. This approach is possible only when the database images cover the working area with sufficient density. However, an interesting property of this approach is that the computations are kept strictly in the 2D domain.

The tiled descriptors have an advantage of a direct orientation estimate. The angle of the best circular shift can be summed with the orientation of the database image to obtain the orientation of the query image up to the tiles angular resolution. The global descriptors applied directly to the fisheye image can be, in theory, very similar to the representations of images of completely different orientations, making the orientation estimation quite difficult. However experimental results presented in Section 5.4.5 show, that the performance of orientation estimation is nearly equal for descriptors applied directly to fisheye images and tiled descriptors.

The used solution should provide invariance to basic transformations. Invariance to light changes should be ensured by the underlying SIFT descriptor in the case of Tiled VLAD, and by properly built training set in the case of Tiled NetVLAD. The real performance under changing lighting conditions is tested in Section 5.4.4. Rotational invariance should be ensured by the circular ordering constraint of the tiles. Invariance to camera translation is currently partly solved by the use of global descriptors and partly by densely captured database images, which is not an ideal solution.

There are many adjustments and improvements, which could be examined in the following works. One such improvement is to estimate the position from multiple retrieved images or use retrieval results of multiple subsequent query images. The latter idea has already been examined by Haraoka in [50].

Chapter 4

Implementation

This chapter deals with details of the implemented software. The implementation of experiments is described in the following Section 4.1. The second Section 4.2 introduces the implemented ROS package.

4.1 Matlab experimental scripts

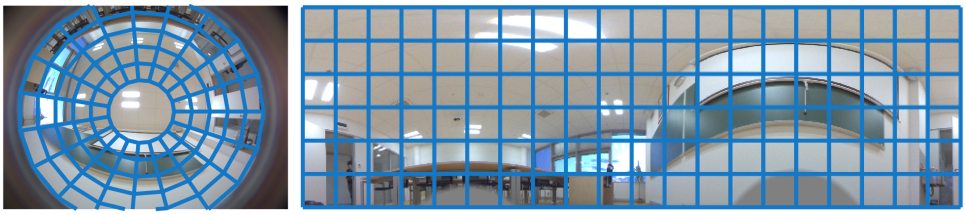
All experiments described in Chapter 5 were implemented in Matlab language, because of compatibility with available codes of state of the art approaches, such as NetVLAD. The code uses VLFeat library (`https://www.vlfeat.org/`), which has to be downloaded and installed separately. Also the NetVLAD code and one of the provided networks have to obtained separately (`http://github.com/Relja/netvlad`).

4.1.1 Image preprocessing

The tiled descriptors are computed from panoramas, which we have to generate from fisheye images. The transformation is performed by mapping the polar coordinate system of the fisheye image on the panorama's Cartesian coordinate system. The situation is depicted in Figure 4.1. The origin of the polar system was obtained from camera calibration, further described in Section 5.1. The whole omnidirectional image can be used to generate the panorama, however, the distortion of the mapping in the top part of the panorama (which is mapped from the central area of the omnidirectional image) is quite large, so this area is being excluded, and the mapping is performed only for the area bounded by two circles. The outer circle matches the circular edge of the fisheye image, and the inner circle has selected radius, which is $r_{in} = 200\ px$ in our case. The radius of the circular edge of the omnidirectional image was manually measured and is equal to $r_{out} = 1116\ px$ for our image of 2592×1944 resolution.

Height of the output panorama is equal directly to the difference of the radii of the outer and inner circles $h = r_{out} - r_{in} = 1116 - 200 = 916$. Width was selected as $w = 2\pi \left(\frac{r_{out} - r_{in}}{2}\right) \doteq 2877$, resulting in the panorama of 2877×916 resolution.

The mapping is done in a standard way. First, a regular grid that represents the pixels of the panoramic image of the precomputed size is generated. Then, the positions of corresponding points in the fisheye image are computed, and finally, quadratic interpolation is used to retrieve the values for the panorama pixels.



(a) : Fisheye image **(b) :** Panoramic image

Figure 4.1: The mapping from a fisheye image into a panorama

Proper masking should be applied to overcome the detection of local features in inappropriate places. The panorama areas, which are mapped from outside of the fisheye image, should be masked out, and the edges should be smoothed, so they do not attract local feature detectors. The same procedure should be applied to the rest of the fisheye image around the main circular area, so the features are not detected e.g., on the light reflections from optics.

4.1.2 SIFT

The SIFT features are detected and described using the VLFeat library. The feature orientation normalization is being done only for experiments on fisheye images. Tiled descriptors use Upright SIFT, where the detected orientation is omitted. The descriptors are RootSIFT normalized as suggested in the paper of Arandjelović et al. [32]. The peak threshold parameter was tuned for the detection of a sufficient number of features. The final value is set to 1 because many tiles did not contain any features while using higher thresholds. The rest of the VLFeat SIFT parameters is set to default values.

4.1.3 VLAD

The VLAD aggregation is performed using a function from VLFeat library and custom vocabulary with 256 centroids, trained on a random subset of database images. The dimensionality of the resulting representation is equal to the length of the underlying local descriptor times the number of centroids, which is $128 \cdot 256 = 32768$ for our case. Intranormalization of the VLAD representation from the paper of Arandjelović et al. [45] is used to reduce the influence of burstiness. PCA reduction is applied, so the final descriptor has a length of 128 elements. The initial vocabulary size and the

number of dimensions after PCA reduction was selected based on the results of experiments from the paper of Torii et al. [42]. Matching of the descriptors is done using Euclidean distance.

4.1.4 Tiled VLAD

Tiled VLAD image representation was implemented by the description in the paper of Torii et al. [42]. The omnidirectional images from our dataset are transformed into panoramas. The SIFT features are being described upright, which is made possible thanks to the geometry of panoramic images. Root-SIFT normalization is applied to the descriptors, and the features are divided into N tiles of defined parameters. Precomputation of the local features and their descriptors allows cheap changes of feature-to-tile assignments with adjustments of tile parameters. The VLAD descriptor is computed for each tile the same way, as mentioned in Section 4.1.3. The final representation of the image consists of N VLAD descriptors in a fixed order.

The representations are matched by computing sum of Euclidean distances over all tiles for each of the possible circular shifts. The minimal distance is taken as a result.

4.1.5 NetVLAD

The NetVLAD paper authors provide code [37] and also a trained network. The network version based on the convolutional part of the VGG-16 network trained on the Pitts250k dataset [46], connected with the NetVLAD layer and whitening, is marked as the best performing in the paper and so was used in this thesis. The resulting descriptor has 4096 dimensions. Only single function `computeRepresentation` for image description computation was used from the provided repository.

4.1.6 Tiled NetVLAD

Because Tiled NetVLAD produces the global description of the provided image, it lacks the possibility of dynamic assignment of local features to tiles as in Tiled VLAD. A workaround in the form of panorama image vertical cutouts is applied. Each of the image tiles is fed into the neural network with the NetVLAD layer, which generates its descriptor. In the case of tile parameters change, the tiling and description steps have to be rerun, which makes each tile parameter change computationally expensive. Image matching step is identical to Tiled VLAD descriptors matching, with the only difference of descriptor dimensions.

4.1.7 Software structure and usage

Each tested method (e.g., Tiled NetVLAD) has its own script for the description of images (`tiled_netvlad_describe.m`), which takes the images from a dataset and produces MAT file with descriptors into the corresponding

dataset directory. The expected structure of the dataset directories can be seen in Figure 4.2. In the case of VLAD and Tiled VLAD, MAT file with SIFT features is generated first, and the VLAD description is computed based on the prepared SIFT features. The next step is to run `load_positions.txt` script, which takes the positions file generated by `visual_localization` package and assigns the positional information to the described images.

The second source file present for each method is the retrieval script (`tiled_netvlad_retrieval_script.m`). It loads all existing description MAT files with assigned positions for both the database and query datasets and finds the most similar database images for each query image. Compressed results are written into another MAT files, which can be loaded by `plot_performance.m` script and the performance graphs used in Section 5.4 can be generated.

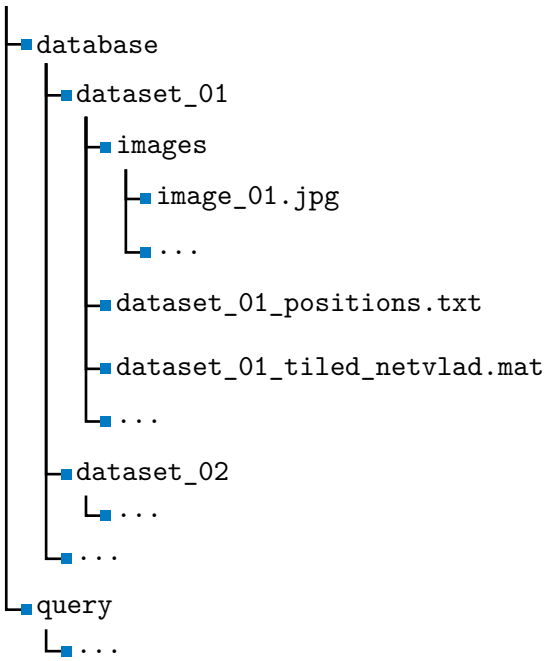


Figure 4.2: Example structure of the dataset directories and files

Initialization script `init.m` is used to set all parameters needed to run the experiments. Brief descriptions and examples accompany the parameters for better understanding. The user should set paths to VLFeat and NetVLAD libraries and to test data directories, so the software finds all necessary functions and inputs.

The individual scripts can be run manually, or by using automation script `auto_process.m`, which contains examples of whole experiments. Thanks to this approach, multiple experiments can run continuously without the need

for supervision.

There are other minor scripts and functions, which are not presented here. Their use and interface are described directly in the source code.

4.2 visual_localization ROS package

The package should be used to create a database of localized described images and subsequently to match newly coming captured images to the database. As the design of the solution at the time of implementation counted only with local image features, the whole system is adjusted for this purpose. There is prepared a parent class `dd_orb_cv.py` for local detect-and-describe methods, which can be exchanged as needed. Class for ORB detector and descriptor `dd_orb_cv.py` was implemented as an example. Both the Python and C++ classes are prepared so that a larger number of the available public implementations of local image feature methods can be used.

The database preparation script `desc_pose_generator.py` listens to a specified ROS topic with camera data and matches each new image with positional information of the camera. The source of the pose, in our case, is the NDT SLAM system. The script node runs in parallel with the selected local feature description node, takes the generated local features and descriptors, and integrates them into the data structure. The data structure of localized and described images can be exported and imported again by using a JSON format file, so the database collecting can be continued even after system rebooting. The script can also generate the TXT positions file, used in our Matlab experiments. The last possible output of the script is the images file, which is needed in the COLMAP system (`https://colmap.github.io/`).

Image retrieval is done by `desc_pose_user_single.py` and `desc_pose_user_live.py` scripts. The first one is used for testing the retrieval of individual images from files and communicates with the user through prepared CLI (Command-Line Interface). The second is intended to work with the image data stream from ROS topics, whether it comes from a rosbag, or live from a robot.

Each of the use cases has its own launch file. The image retrieval can be started by launching `match_individual_images.launch` or `match_live_images.launch`. The database preparation is controlled by `process_database.launch` file. It tries to run SLAM node from `ndt_ciirc_slam` package by Nováček [18], which has to be obtained separately. An important step is to provide the correct initial position for the SLAM system, so it can correctly localize in the prepared NDT map of the environment. The starting positions can be measured in the CAD drawing of the environment.

Chapter 5

Experimental results

5.1 Camera settings

The first thing that had to be done was camera preparation, especially choosing the right exposure settings. Auto-exposure was tested, but the results with optimally chosen constant value were evaluated as the better ones. An example of images captured by two different exposure settings can be seen in Figure 5.1. The exposure time should be a compromise between the minimization of saturated and black areas in the image. Because the camera uses a rolling shutter, interference with the blinking of artificial lighting had to be also considered. The interference causes light and dark strips visible in the left part of Figure 5.1. The final chosen value is 10 ms, with which the image is saturated only at the places with direct sunlight, the environment illuminated only with artificial lighting is clearly visible, and rolling shutter artifacts are minimized.

(a) : Automatic exposure **(b) :** 10 ms constant exposure

Figure 5.1: Influence of exposure settings on the image

The Basler camera provides drivers for use with ROS, so the parameters can be set using rosparam at system initialization. Changing other parameters than exposure time seems unimportant for our application.

The camera calibration was performed with possibilities of geometrical image transformations and 3D point cloud reconstruction in mind. We used

the fisheye camera calibration procedure from OpenCV, with integration of yet unsupported functions ([20]), to ensure the functionality with our super fisheye lens. Obtained principal point coordinates are used for fisheye to panoramic image conversion as the origin of the polar system.

5.2 Dataset

This chapter briefly presents the dataset prepared for testing of various approaches on our robotic platform. The dataset captures multiple rooms and the corridor on the 6th floor of CIIRC building B by a monocular vertically oriented fisheye camera. The images were captured with approximately 0.25 m travel distance period measured by robot odometry. All data needed for image positions retrieval has been saved into rosbag files, and the locations were obtained offline by the NDT SLAM system. Capturing was done in multiple runs to get images containing various lighting conditions and to reduce the SLAM localization error by shortening the driven distance from a starting position. The individual runs were performed over a period of four months, and therefore slight changes in the environment can be observed. Figure 5.2 contains examples of captured images at several places under different lighting conditions. The data from each run is kept in a separate subset named by daytime, day, and place of the run.

Figure 5.2: Examples of dataset images captured by fisheye camera - top and bottom rows depict same places in different lighting conditions

The localization error of the NDT SLAM at the time of capturing was quite high. The process of image positions retrieval demanded visual supervision because of the possibility of SLAM position estimate divergence. SLAM parameters had to be tunned for individual runs to obtain reasonable results. The especially challenging environment is corridors, which lack distinctive features for planar lidar. Localization along corridor length can have errors in the order of tens of percent of the driven distance. One of the used solutions was to install artificial obstacles that are nearly not visible in the captured images but make a significant change in the lidar scans and improve the

localization results. The other problem is that the lidar scans can match on the opposite side of the wall, shifting the location estimate. Because of the mentioned issues, the threshold between correctly and incorrectly localized image is set to 1 m distance between the image locations. For comparison, The Visual Localization Benchmark [47] uses three distance thresholds for methods evaluation, where the most strict is 0.25 m, which we are not able to achieve with our ground truth precision. The work of Boxan [38] has improved the system and should allow flawless processing of future datasets and eventually lowering our evaluation distance threshold.

(a) : Stairs and elevator

(b) : Printer corner

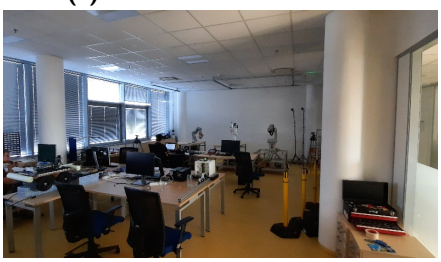
(c) : Southern part of the corridor

(d) : Northern part of the corridor

(e) : Southern kitchenette

(f) : Northern kitchenette

(g) : Lecture hall



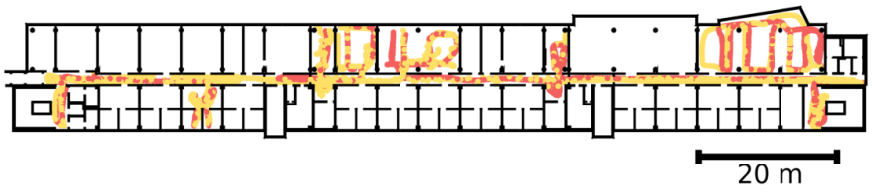
(h) : Laboratory

Figure 5.3: Examples of the environment (captured by cellphone)

Totally 6106 images in 34 subsets were localized successfully and have been divided into a database of 20 subsets with 3906 images and a query dataset of 14 subsets with 2200 images. The individual subsets correspond to the runs, where the robot was initialized at a position with accurately measured coordinates within a building CAD drawing and then driven along some path while periodically capturing images and saving the data into a rosbag file.

Both the database and query dataset contain at least one day and one dark subset for each place. The captured locations contain a corridor, one office, kitchenettes, printer corners, a laboratory, and a lecture hall. Examples of the captured environment are shown in Figure 5.3.

All the localized positions are in the coordinate system of the CAD architectural drawing, which is used for NDT SLAM initialization. The estimated image positions can be visualized inside the map, as shown in Figure 5.4.



(a) : Database image datasets

(b) : Query image datasets

Figure 5.4: Estimated image positions for a day (yellow) and dark (red) images

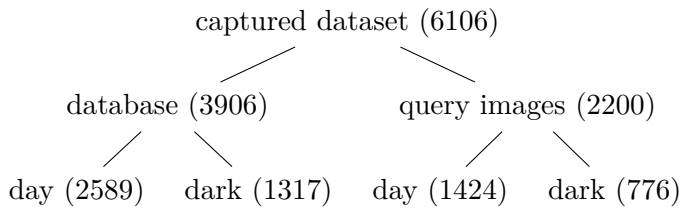
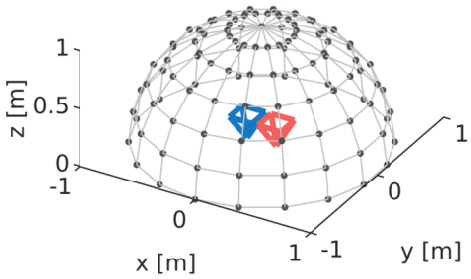


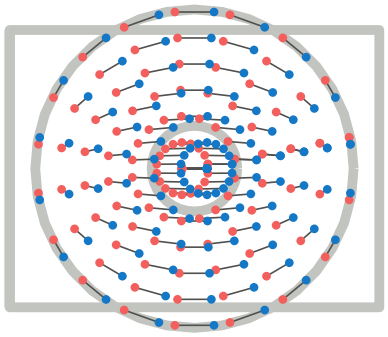
Figure 5.5: Division of the dataset into subsets used for experiments - the brackets contain the number of images in each set

5.3 Optimal tiled descriptor parameters

The tile width for Tiled VLAD and Tiled NetVLAD descriptors was estimated by the following experiment. 3D point grids were generated, simulating the environment in which the robot with the camera operates.

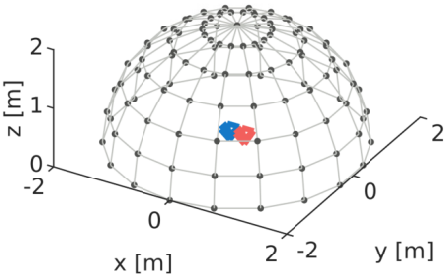


(a) : A 3D scene containing hemisphere of points connected by wire mesh for clarity. Omnidirectional cameras (A and B) are placed around the center of the hemisphere, 0.25 m from each other, and vertically oriented.

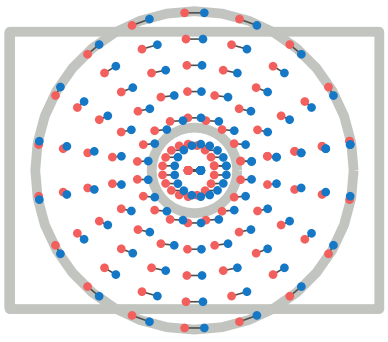


(b) : Merged projections of points from the cameras (A and B). The corresponding points are connected. The grey rectangle marks the limits of the fisheye image and the grey circles bound panorama mapping area.

Figure 5.6: Experiment for estimation of the optimal tile width - environment imitated by a hemisphere of 1 m radius.



(a) : A scene containing hemisphere of points and cameras (A and B)



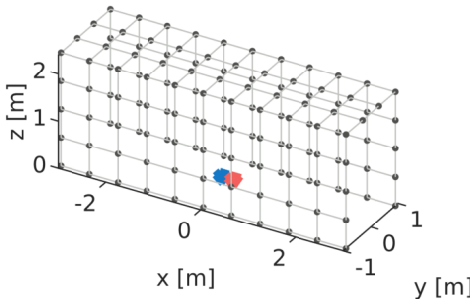
(b) : Merged projections of points from the cameras (A and B)

Figure 5.7: Experiment for estimation of the optimal tile width - environment imitated by a hemisphere of 2 m radius.

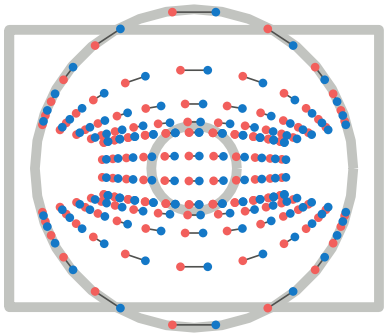
The projection of the environment has been computed for two camera positions with a distance of 0.25 m from each other. The selected camera distance is the same as the travel distance period with which was the dataset captured, as explained in Section 5.2. The scene can be seen on a simple case of a hemisphere with a radius of 1 m in Figure 5.6a.

The projections from both cameras were merged into a single image, and the corresponding points were connected by lines, as can be seen in Figure 5.6b. A simplified equidistant camera model was used for the projections. A grey rectangle marks the limits of the resulting fisheye image, and the two gray circles bound the area from which the panorama is generated.

If the objects are farther away from the camera, the resulting shift in the omnidirectional image is smaller, which is a natural behavior for all cameras using other than a telecentric lens. The effect is clearly visible while comparing Figures 5.6 and 5.7, where the cameras are placed inside the hemispheres of 1 m and 2 m radius respectively. A more realistic case is captured in Figure 5.8, trying to simulate a camera moving along a corridor. The corridor model dimensions were selected to be similar to the testing environment.



(a) : A scene containing a corridor model from points and cameras (A and B).



(b) : Merged projections of points from the cameras A and B

Figure 5.8: Experiment for estimation of the optimal tile width - the environment imitated by a corridor of 6.0 m x 1.8 m x 2.4 m dimensions

The tile width was selected to be more than double the maximum angular change between two corresponding projected points. The idea is that more than half of the content in the most changing tile will remain in the tile after the shift of the camera by the specified distance. That should allow successful matching of the corresponding panorama tiles. Both the maximum angular change of approximately 15.8° and the selected tile width of 40° are marked in Figure 5.9. The angle of the maximum change can be measured from the image, or computed by equation 5.1, where $\mathrm{d}(A, B)$ is the distance between the cameras, $\mathrm{d}(\overline{AB}, C)$ is the distance from the line between the cameras to

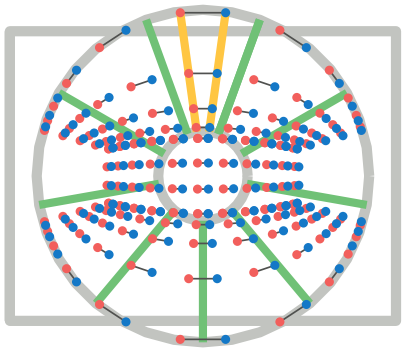


Figure 5.9: Figure 5.8b with marked angle of maximal point shift (yellow) of 15.8° and the selected angular width of tiles (green) of 40°. Only tiles with a shift of 40° are displayed for clarity. The final method uses tiles shifted by 10°.

the point of maximal change. The values in the equation are taken from the case in Figure 5.8.

$$\phi_{max} = 2 \cdot \tan^{-1}\left(\frac{\mathrm{d}(A,B)/2}{\mathrm{d}(\overline{AB},C)}\right) = 2 \cdot tan^{-1}\left(\frac{0.25/2}{0.9}\right) = 15.8° \qquad (5.1)$$

5.4 Descriptor tests

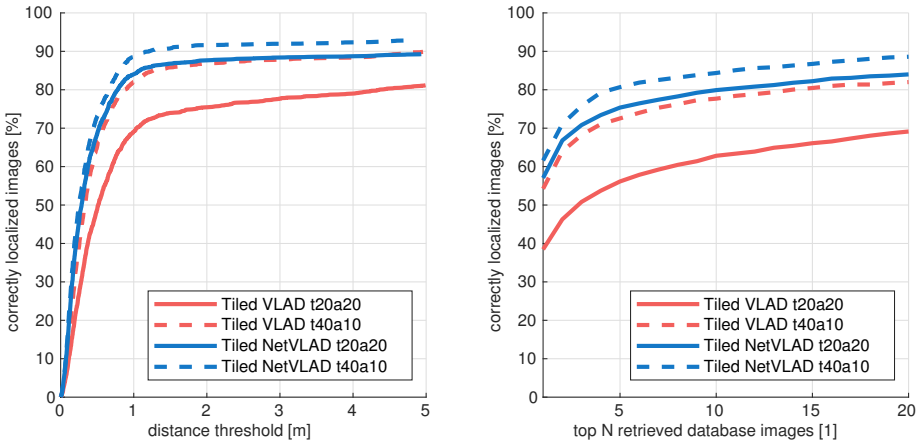
Multiple combinations of descriptors and their parameters were tested. Evaluation metrics were selected to comply with the standardly used methodology. Namely The Visual Localization Benchmark [47], InLoc [5] and D2-Net [41] were used as reference. All mentioned works employ the percentage of correctly localized images within a distance threshold as the metric. Two figures were generated for each test. The first captures the percentage of correctly localized queries within the constant distance threshold of 1 meter by taking the nearest database image out of N best retrieved, where N is tested for values from 1 (the single best) to 20. The second figure plots the percentage of correctly localized images for the nearest image out of $N = 20$ best retrieved and variable distance threshold from 0 m up to 5 m. The approach was selected to count with the possibility of employing multiple retrieved images into a final position estimate calculation.

5.4.1 Tiled descriptors parameters

The results of the first experiment in Figure 5.10 show the difference between Tiled VLAD and Tiled NetVLAD descriptors for two different sets of tile parameters. The first set produces tiles of 20° width with a 20° shift from each other. These values were an initial guess based on the values used in the original paper [42]. The second set uses 40° wide tiles with a 10° shift,

which were selected based on the experiment described in Section 5.3 and should correspond more to the indoor environment.

The later values definitely perform better, which is, on the other hand, compensated by higher memory, computational, and, consequently, also time requirements. All further experiments were performed with the later parameters. Tiled NetVLAD performs better than Tiled VLAD for both cases, which corresponds with results from the NetVLAD paper [17].



(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable distance threshold

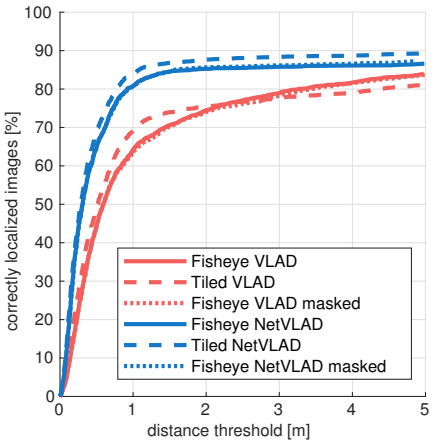
(b) : Percentage of correctly localized images from N best retrieved database images with a threshold of 1 m from the ground truth

Figure 5.10: Comparison of retrieval results between Tiled VLAD and Tiled NetVLAD and two different sets of tiling parameters - t20a20 (tiles of 20° width with 20° shift) and t40a10 (tiles of 40° width with 10° shift)

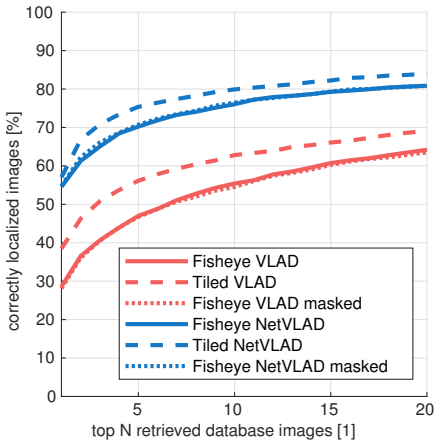
5.4.2 Fisheye image central area information gain

The second experiment deals with information gain from the central part of the fisheye image, which, in our case, captures the ceiling most of the time. The mentioned central part can be seen in Figure 5.9, bounded by the inner gray circle.

The first tested case is directly described fisheye image. The second is a tiled descriptor with the better performing parameters from the previous experiment. The last tested case uses descriptor applied on the fisheye image with masked central part, which is being omitted in the tiled descriptors.

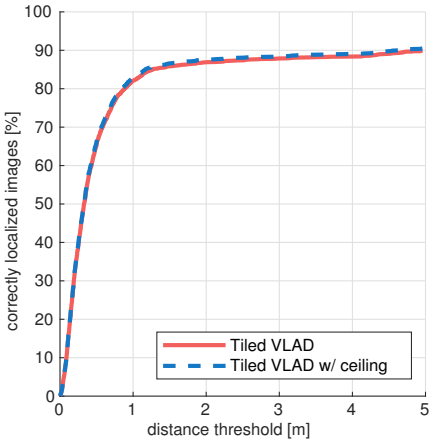


(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable distance threshold

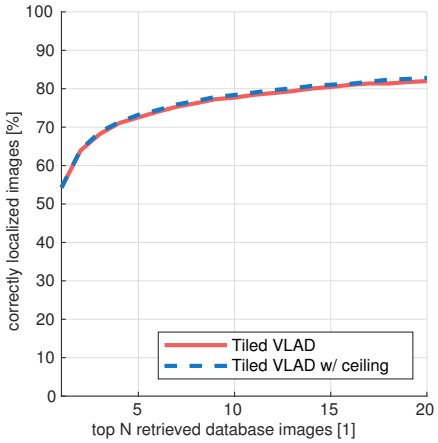


(b) : Percentage of correctly localized images from N best retrieved database images with a threshold of 1 m from the ground truth

Figure 5.11: Retrieval results for descriptor (VLAD and NetVLAD) applied directly on fisheye image, on tiled panorama and on fisheye image with masked center



(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable distance threshold



(b) : Percentage of correctly localized images from N best retrieved database images with a threshold of 1 m from the ground truth

Figure 5.12: Retrieval results for Tiled VLAD descriptor without and with appended center

The results (in Figure 5.11) of described fisheye images with and without masked central parts are nearly the same, which indicates that the center carries only minimal information usable by the global descriptors. The results of the tiled descriptors seem to be better for both, VLAD aggregation

and NetVLAD. It is probably caused by the circular shift constraint, where part of the features positional information is taken into account in contrast with directly described fisheye image, entirely omitting the keypoint positions.

Incorporating the central part of the fisheye image into a tiled descriptor as another tile, on which the circular shift constraint does not apply, was also tested. The results are shown in Figure 5.12. The performance is nearly the same for both cases, which confirms the results of the previous experiment and validates that incorporating the central part of the fisheye image to the retrieval does not pay off.

The results of the experiments suggest that the central part of the fisheye image carries a negligible amount of information, and integrating this part of the image into the description does not have any advantage. However, these conclusions are specific to our environment, where the central part captures a repetitive suspended ceiling with only a few distinct elements such as lights, sprinklers, or air conditioning exhausts.

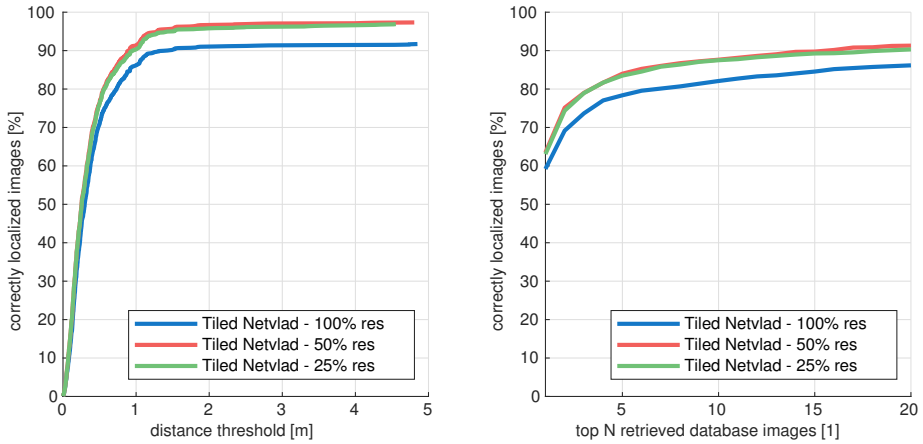
5.4.3 Influence of image resolution on NetVLAD

The next test deals with an influence of the panorama resolution on the Tiled NetVLAD descriptor. The tiles with the original, 50% and 25% resolution were tried, and the results can be seen in Figure 5.13. It is clear that the full resolution performs the worst. The lower resolution results are nearly equal, with 50% slightly the best. An explanation of the results could be that the used network was trained on images with the larger side size of 640 px, as mentioned in the NetVLAD paper [17], and it could perform better on the images of similar resolution.

Resolution [%]	Resolution [px]	Processing time [ms]
100 %	917×320	110
50 %	459×160	80
25 %	230×80	79

Table 5.1: Table of the tested tile resolutions with corresponding times for a single tile description by NetVLAD (`computeRepresentation` function) - the experiment was performed on tiles with 40° in width

The processing time declines with reducing resolution as expected. However for our Matlab implementation, where we load the fisheye image, transform it into a panorama, apply masking, divide into tiles, describe all the tiles and save the results into structure array, the description step itself takes only a little time fraction.



(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable distance threshold

(b) : Percentage of correctly localized images from N best retrieved database images with threshold of 1 m from ground truth

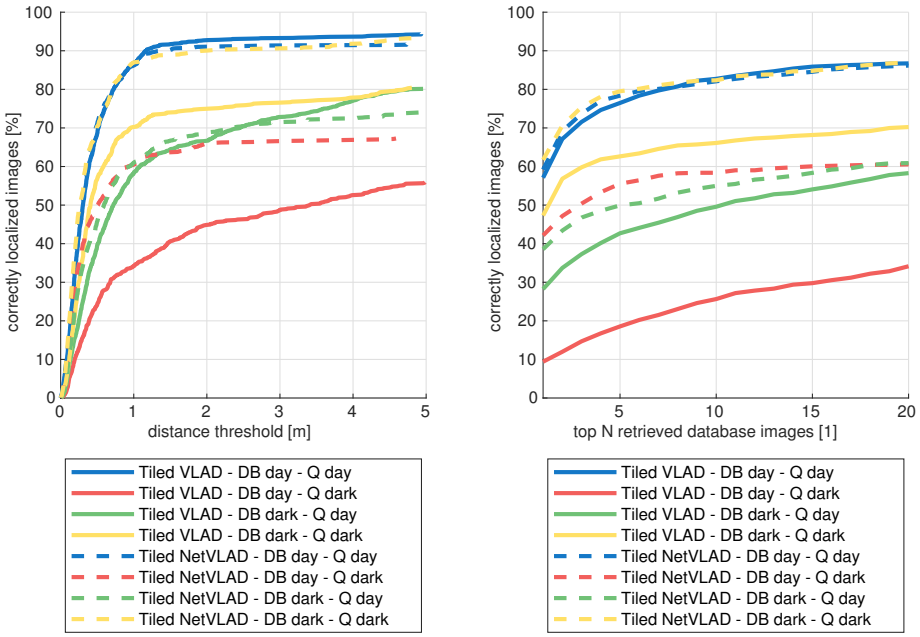
Figure 5.13: Comparison of NetVLAD performance on different tile resolutions

Only smaller subset of our dataset was used to perform this test to save time. All the other experiments use a 100% resolution, but the results from this test should be taken into account for the final system implementation. The same experiment for the Tiled VLAD descriptor should not be necessary due to the scale invariance of used SIFT features.

5.4.4 Influence of lighting conditions

The influence of lighting conditions was tested for both Tiled VLAD and Tiled NetVLAD. The database datasets and query datasets were divided into day and dark groups, based on the table in Figure 5.5, which were then tested separately. The results for Tiled VLAD and Tiled Netvlad can be seen in Figure 5.14. It is clear that Tiled VLAD performs the best for the day database in combination with day query images. Multiple tests were performed to find out why the other methods perform worse.

One of the hypotheses was that there are more detected SIFT features in day images than in the night ones, which could result in more descriptive VLAD representation. The average number of features per image in the datasets can be seen in Table 5.2. It shows that the average number of detected features is similar for all subsets and is not the cause of the inferior results while using the datasets from dark environments.



(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable distance threshold

(b) : Percentage of correctly localized images from N best retrieved database images with threshold of 1 m from ground truth

Figure 5.14: Retrieval results of Tiled VLAD and Tiled NetVLAD descriptors for different lighting conditions

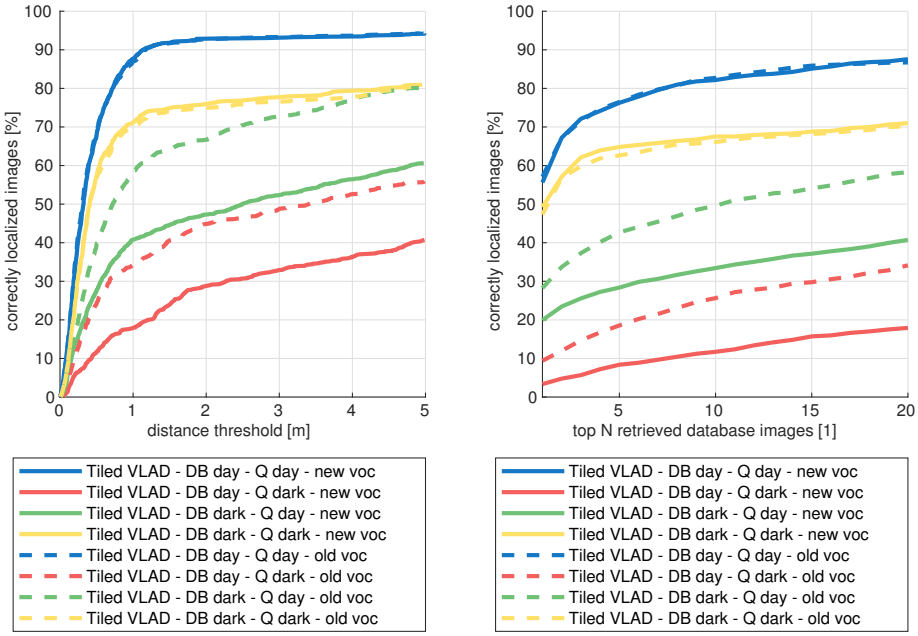
Dataset	Average number of SIFTs per image
Database - day	565
Database - night	602
Query - day	625
Query - night	726

Table 5.2: Average number of SIFT features per image from database and query datasets divided into day and dark subsets

The second hypothesis deals with the ratio of day and dark images in the VLAD vocabulary training dataset. The initial vocabulary, which is also used in all tests, was trained at an early stage of dataset capturing and contained only daylight images. A new training dataset with more dark images was generated, and the vocabulary was recomputed. The comparison of new and old VLAD vocabulary can be seen in Figure 5.15. The dark images in the training dataset caused that VLAD started to differentiate between SIFT features from the day and dark environment, causing lower performance in the matching across different lighting conditions. There is no significant

improvement for both the day-day and night-night tests while using the new vocabulary. It is not clear, why there is such performance difference between day-dark and dark-day experiments.

NetVLAD descriptor performs well when both the database and query dataset contain images with the same lighting conditions. The performance drops when the lighting conditions differ. The cause is probably similar to the newly trained VLAD vocabulary, where the descriptor successfully distinguishes between features captured in different lighting conditions. That can be overcome if the database is being built continually through robot operation because various lighting conditions are captured for the whole working area, and the correct place should be retrieved regardless of the daytime.



(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable distance threshold

(b) : Percentage of correctly localized images from N best retrieved database images with threshold of 1 m from ground truth

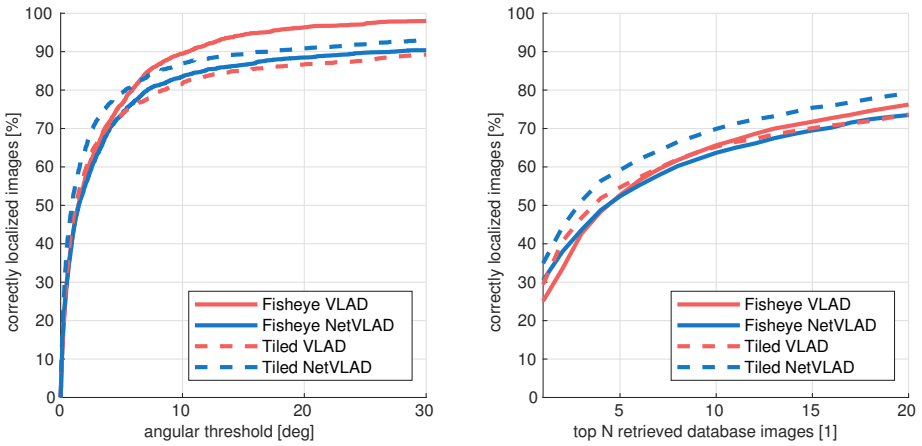
Figure 5.15: Retrieval results of Tiled VLAD descriptor for different lighting conditions and two vocabularies

5.4.5 Orientation estimation

The previous experiments tested only position estimation performance regardless of the orientation, so this section tries to correct this shortcoming

and deal with the testing of the orientation estimate. The testing procedure is the same as for the previous tests, but the distance threshold was replaced by an angular threshold of 5°. The threshold was selected so that the used SLAM system should be able to correct the remaining error by itself.

Expected results were that the descriptors applied on fisheye images would have quiet poor performance because the images captured with different camera orientations are except the top and bottom sensor crop very similar, and their description could be easily matched. The tiled descriptors have the ability to correct the orientation by using the rotation of the optimal tile circular shift, which should result in better performance. The rotation can be estimated up to the sliding interval between individual tiles α which is equal to $360/N$, where N is the number of tiles in the used representation.



(a) : Percentage of correctly localized images from $N = 20$ best retrieved database images and variable orientation threshold

(b) : Percentage of correctly localized images from N best retrieved database images with threshold of 5° from ground truth

Figure 5.16: Performance of orientation estimation for used descriptors

The results can be seen in Figure 5.16. Unexpectedly descriptors applied to fisheye images have very similar performance to the tiled descriptors. The possible cause is that there exists a database image of a similar orientation for nearly every query image, and there are not many similar images with different orientations. This cause is quite probable, because the dataset was captured by driving the mobile robot along very similar paths for both the database and query sets. However, this scenario can be common even in real applications, where the mobile robot drives between several stops along nearly identical paths.

5.4.6 Time demands

The experiments were performed on PC with Intel Core i7-9750H 2.60GHz CPU and Nvidia GeForce RTX 2070 Max-Q GPU. Most of the operations ran on a single CPU core. The parallelization was successfully applied on retrieval tests on precomputed representations so that multiple query images could be processed at once, however, RAM capacity had to be kept in mind. A single-core variant has been used for time demands tests. NetVLAD description generation function supports GPU computation, which was also used for the following experiments.

The experiments were performed on a sample of 100 query and 100 database images. Mean durations of individual operations over the sample are written in the Tables 5.3, 5.4, 5.5, 5.6. The measurements include all operations from loading the necessary data, applying computations, and storing the results. E.g., SIFT operation in Table 5.5 involves loading of the image, masking of the central and edge areas, transformation to a panorama, detection and description of local features by Upright RootSIFT and saving the results into MAT file.

The shorter time durations are rather approximative, because the measurements are influenced by used Matlab `tic toc` functions, adding some unspecified time instant. The influence is clearly visible on example in Table 5.5, where duration of score computation for single circular shift multiplied by the number of tiles is greater than the measured duration of score computation for the whole image ($36 \cdot 14$ µs $= 504$ µs > 400 µs).

VLAD in Table 5.3 and NetVLAD in Table 5.4, which are directly applied to fisheye images have nearly equally long description phase duration. The difference in the score computation time is caused by different descriptor dimensionality, where VLAD after PCA compression has 128 dimensions, and the output of the used NetVLAD network has 4096 dimensions.

Tiled VLAD and Tiled NetVLAD representations use the parameters proposed in Section 5.3, which means, that 36 tiles are generated. Tiled VLAD in Table 5.5 is more than twice faster in the description phase than Tiled NetVLAD with measurements in Table 5.6. The possible cause of poor NetVLAD results is that the used function `computeRepresentation` loads the network to the GPU before each new image, which is each tile in our case. Faster function, which loads the network only once, exists, however, it is not usable in our implementation. Score computation per image is approximately twenty times faster in the case of Tiled VLAD, which is again caused by the length of VLAD and NetVLAD descriptors, in addition, multiplied by the number of tiles.

Operation	Per data unit	Time
SIFT	image	1.8 s
VLAD + PCA	image	5.9 ms
retrieval	image to database (100)	0.8 ms
score computation	image to image	5.2 µs

Table 5.3: Table of fisheye VLAD time demands

Operation	Per data unit	Time
NetVLAD	image	1.7 s
retrieval	image to database (100)	1.4 ms
score computation	image to image	7.5 µs

Table 5.4: Table of fisheye NetVLAD time demands

Operation	Per data unit	Time
SIFT	image	2.0 s
Tiled VLAD + PCA	image	29.1 ms
Tiled VLAD + PCA	tile	1.0 ms
retrieval	image to database (100)	36.3 ms
score computation	image to image	0.4 ms
score computation	circular shift	14.0 µs

Table 5.5: Table of Tiled VLAD time demands

Operation	Per data unit	Time
Tiled NetVLAD	image	5.4 s
NetVLAD	tile	0.1 s
retrieval	image to database (100)	0.8 s
score computation	image to image	7.8 ms
score computation	circular shift	0.4 ms

Table 5.6: Table of Tiled NetVLAD time demands

The panorama has approximately 2.6 MP (megapixels), and the original fisheye image has about 5 MP. That should cause faster descriptions in the case of the panorama. However, our selected tiling parameters (40° width, 10° shift) cause, that each part of the panorama is present in four tiles, which are being described separately. The result is the same as describing a four times larger image. This does not apply to the local features used with VLAD,

which are detected and described only once and then assigned to the tiles.

We can easily estimate the duration of the retrieval operation for a database with more images because the time is approximately linear to the size of the database. For Tiled NetVLAD and our database with 3906 images, the length of the retrieval phase for a single query image is $3906/100 \cdot 0.8\ s = 31.2\ s$.

The presented measurements should be taken only as an approximate comparison between individual methods. The implementation was not done with any particular emphasis on time efficiency in mind. A different programming language such as C++ should be used for final implementation, and the efficiency of individual operations can change significantly.

Chapter 6

Conclusion

The main tasks of this thesis were to make a survey of available visual localization methods, select some of them, and evaluate the selected ones on real data. The used system should be able to provide a global initial pose guess for NDT SLAM system.

Dataset of fisheye images was captured using the target robotic platform. The dataset contains images of multiple rooms on the 6th floor of CTU CIIRC building. The images were captured for different lighting conditions. The ground truth positions were generated using NDT SLAM system.

ROS package was prepared for the purpose of database preparation and visual localization based on local image features. VLAD and NetVLAD global image descriptors with a combination of tiled panorama representation were selected as a foundation of the system. Description and retrieval functions using selected methods were implemented in Matlab language and thoroughly evaluated on the captured dataset. The results indicate that the tiled descriptors perform slightly better than the directly described fisheye images. However, better results are achieved at the cost of much higher computational demands. Localization procedure based on the location of the most similar image retrieved from a database was proposed.

The thesis should be a valuable source of information and cornerstone of visual localization system for anybody, who will continue in work on this topic. Future work should provide full integration into ROS, based on prepared `visual_localization` package. Aggregation of information from multiple retrieved database images and connection of results from several consequent query images could result in great advance in performance. Query expansion based on database images location graph could also have a great impact. Training NetVLAD network on images from the target environment could be worth a try.

Bibliography

[1] *Galileo GNSS: The path to high GNSS accuracy* [Online], Available: `https://galileognss.eu/the-path-to-high-gnss-accuracy/`.

[2] A. Zhou, *Survey on Visual-Based Localization.*

[3] N. Piasco, D. Sidibé, C. Demonceaux, V. Gouet-Brunet, *A survey on Visual-Based Localization: On the benefit of heterogeneous data*, Pattern Recognition, Volume 74, 2018, pp. 90-109.

[4] A. Kendall, M. Grimes, R. Cipolla, *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization*, 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 2938-2946.

[5] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, *InLoc: Indoor Visual Localization with Dense Matching and View Synthesis*, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 7199-7209.

[6] A. Morar, A. Moldoveanu, I. Mocanu, F. Moldoveanu, I. E. Radoi, V. Asavei, A. Gradinaru, A. Butean, *A Comprehensive Survey of Indoor Localization Methods Based on Computer Vision*, Sensors 20, no. 9, 2020, pp. 2641.

[7] D. Nister, O. Naroditsky and J. Bergen, *Visual odometry*, Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., Washington, DC, USA, 2004.

[8] Yang Cheng, Mark Maimone, Larry Matthies, *Visual Odometry on the Mars Exploration Rovers*, IEEE International Conference on Systems, Man and Cybernetics, 2005.

[9] Mark Maimone, Yang Cheng, Larry Matthies, *Two Years of Visual Odometry on the Mars Exploration Rovers*, J. Field Robotics. 24. 169-186. 10.1002/rob.20184, 2007.

[10] Davide Scaramuzza, Roland Siegwart, *Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles*, IEEE Transaction on Robotics, vol. 24, NO. 5, October 2008.

[11] Sen Wang, Ronald Clark, Hongkai Wen, Niki Trigoni, *DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks*, 2017 IEEE International Conference on Robotics and Automation (ICRA) (2017).

[12] D. Xu, L. Han, M. Tan, Y. F. Li, *Ceiling-Based Visual Positioning for an Indoor Mobile Robot With Monocular Vision*, IEEE Transactions on Industrial Electronics, vol. 56, no. 5, pp. 1617-1628, May 2009.

[13] E. Krotkov, *Mobile robot localization using a single image*, Proceedings, 1989 International Conference on Robotics and Automation, Scottsdale, AZ, 1989, pp. 978-983 vol.2.

[14] S. Se, D. Lowe, J. Little, *Local and global localization for mobile robots using visual landmarks*, Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180), Maui, HI, USA, 2001, pp. 414-420 vol.1.

[15] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi and T. Pajdla, *24/7 place recognition by view synthesis*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1808-1817.

[16] L. G. Camara and L. Přeučil, *Spatio-Semantic ConvNet-Based Visual Place Recognition*, 2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic, 2019, pp. 1-8.

[17] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla and J. Sivic, *NetVLAD: CNN Architecture for Weakly Supervised Place Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 6, pp. 1437-1451, 1 June 2018.

[18] David Nováček, *Localization of Mobile Robot Using Multiple Sensors*, diploma thesis, CTU, FEE, 2018.

[19] *Sunex: DSL215 Specification Sheet* [Online], Available: `www.optics-online.com/OOL/DSL/DSL215.PDF`.

[20] *GitHub: opencv repository pull request: Support for super-fisheye lenses*, [Online], Available: `https://github.com/opencv/opencv/pull/6801`.

[21] D. Scaramuzza, A. Martinelli, R. Siegwart. *A Toolbox for Easy Calibrating Omnidirectional Cameras*, Proceedings to IEEE International Conference on Intelligent Robots and Systems, (IROS). Beijing, China, October 7–15, 2006.

[22] Christopher Mei, Patrick Rives, *Single View Point Omnidirectional Camera Calibration from Planar Grids*, Proceedings to IEEE International Conference on Robotics and Automation, 2007.

[23] Christopher Mei, *Omnidirectional Calibration Toolbox*, [Online], Available: `https://www.robots.ox.ac.uk/~cmei/Toolbox.html`.

[24] C. Harris, M. Stephens, *A combined corner and edge detector*, In Proc. of Fourth Alvey Vision Conference, 1988, pp. 147-151.

[25] E. Rosten, T. Drummond, *Fusing points and lines for high performance tracking*, Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, 2005, pp. 1508-1515 Vol. 2.

[26] M. Calonder, V. Lepetit, C. Strecha, P. Fua, *BRIEF: Binary Robust Independent Elementary Features*, In: K. Daniilidis, P. Maragos, N. Paragios (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, 2010, vol 6314. Springer, Berlin, Heidelberg.

[27] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, *ORB: An efficient alternative to SIFT or SURF*, 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2564-2571.

[28] P. L. Rosin, *Measuring Corner Properties*, Comput. Vis. Image Underst. 73, 1997, 291-307.

[29] D. G. Lowe, *Object recognition from local scale-invariant features*, Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2.

[30] D.G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision 60, 91–110, 2004.

[31] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, M. Pollefeys, *Handling Urban Location Recognition as a 2D Homothetic Problem*, ECCV 2010, 6316, 266-279.

[32] R. Arandjelović and A. Zisserman, *Three things everyone should know to improve object retrieval*, 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, 2012, pp. 2911-2918.

[33] G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, *Visual categorization with bags of keypoints*, Work Stat Learn Comput Vision, ECCV. Vol. 1, 2004.

[34] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman and W. T. Freeman, *Discovering objects and their location in images*, Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, 2005, pp. 370-377 Vol. 1.

[35] J. Sánchez, F. Perronnin, T. Mensink, J. Verbeek, *Image Classification with the Fisher Vector: Theory and Practice*, Int J Comput Vis 105, 222–245, 2013.

[36] H. Jégou, M. Douze, C. Schmid and P. Pérez, *Aggregating local descriptors into a compact image representation*, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 2010, pp. 3304-3311.

[37] R. Arandjelović, *NetVLAD: CNN architecture for weakly supervised place recognition*, GitHub repository, [Online], Available: `https://github.com/Relja/netvlad`.

[38] M. Boxan, *NDT SLAM Respecting Visibility*, bachelor thesis, CTU, FEE, 2020.

[39] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, T. Pajdla, *Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 6175-6184.

[40] H. Taira, I. Rocco, J. Sedlar, M. Okutomi, J. Sivic, T. Pajdla, T. Sattler, A. Torii, *Is This The Right Place? Geometric-Semantic Pose Verification for Indoor Visual Localization*, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4372-4382, 2019.

[41] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, T. Sattler, *D2-Net: A Trainable CNN for Joint Description and Detection of Local Features*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 8084-8093.

[42] A. Torii, Y. Dong, M. Okutomi, J. Sivic, T. Pajdla, *Efficient Localization of Panoramic Images Using Tiled Image Descriptors*, IPSJ Transactions on Computer Vision and Applications, 6, 58-62, 2014.

[43] L.G. Camara and L. Přeučil, *Visual Place Recognition by Spatial Matching of High-level CNN Features*, 2020

[44] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, CoRR abs/1409.1556, 2015.

[45] R. Arandjelovic and A. Zisserman, *All About VLAD*, 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, 2013, pp. 1578-1585.

[46] A. Torii, J. Sivic, M. Okutomi, T. Pajdla, *Visual Place Recognition with Repetitive Structures*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 11, pp. 2346-2359, 1 Nov. 2015.

[47] L. Hammarstrand, F. Kahl, W. Maddern, T. Pajdla, M. Pollefeys, T. Sattler, J. Sivic, E. Stenborg, C. Toft, A. Torii, *The Visual Localization Benchmark*, [Online], Available: `https://www.visuallocalization.net`.

[48] O. Chum, J. Matas, *Geometric Hashing with Local Affine Frames*, 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 2006, pp. 879-884.

[49] O. Chum, M. Perďoch, J. Matas, *Geometric min-Hashing: Finding a (thick) needle in a haystack*, 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 17-24.

[50] S. Haraoka, *Retrieval for the best location from InLoc candidates using graphs*, internship report, CIIRC CTU, 2019.

[51] J. H. Schönberger, J-M. Frahm *Structure-from-Motion Revisited*, Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[52] J. H. Schönberger, E. Zheng, M. Pollefeys, J-M. Frahm *Pixelwise View Selection for Unstructured Multi-View Stereo*, European Conference on Computer Vision (ECCV), 2016.