SPRING

Deliverable D2.4: Visual-based localisation in relevant environments

Due Date: 31/05/2022

Main Author: David Kunz, Rakshith Madhavan, Tomas Pajdla

Contributors: CVUT

Dissemination: Public Deliverable

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

DOCUMENT FACTSHEET

Deliverable	D2.4: Visual-based localisation in relevant environments
Responsible Partner	CVUT
Work Package	WP2: Environment Mapping, Self-localisation and Simulation
Task	T2.1: Robot Self-localisation from Images
Version & Date	31/05/2022
Dissemination	Public Deliverable

CONTRIBUTORS AND HISTORY

Version	**Editor**	**Date**	**Change Log**
1	CVUT	27/05/2019	First draft
2	CVUT	29/05/2019	First revision
3	CVUT	31/05/2019	Final version

APPROVALS

Authors/editors	David Kunz, Rakshith Madhavan, Tomas Pajdla
Task Leader	CVUT
WP Leader	CVUT

Contents

Executive Summary

This deliverable investigates how to do global robot localization in the conditions that are relevant to SPRING project that calls for building global maps using the sensors on the robot itself, rather than using extra mapping effort. We construct global visual maps by COLMAP global structure from motion pipeline and employ purely image based localization Hloc pipeline. We present a new localization pipeline that uses elements of Hloc and combine them with additional processing of ARI images to achieve robust localization. We show, that in the relevant BROCA hospital environment, we can achieve 90% consistent recall between the global and local mapping and localization at 1.5 meter tolerance.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

1 Introduction

Global localization aims at finding robot positions in a global map after waking up a robot as well as at providing a secondary mechanism for checking the performance of local mapping using, e.g., ORB-SLAM [8] and RTAB-Map [6] pipelines, and detecting their failures. This brings the necessary robustness for real world performance.

The main goal of this deliverable is to investigate how to do global robot localization in the conditions that are relevant to SPRING project. Earlier [13], we have investigated and evaluated InLoc [14] localization pipeline that uses a detailed 3D map of the environment. The map is obtained by 3D scanning using high-quality 3D scanners and a 3D map construction technology, e.g., from Matterport [1, 7]. Using this technology, we can localize robot accurately and reliably [13].

However, there are several challenges related to the global localization in SPRING project. First, it turns out that it is not viable to use 3D scanning campaigns because they are not scalable and can't react to changing environments. Secondly, it is necessary to interconnect global localization with local maps built via visual SLAM provided by, e.g., ORB-SLAM or RTAB-Map. Hence, we conclude that it is necessary to build global maps using the sensors on the robot itself, rather than using extra mapping effort.

ARI robots used in SPRING are equipped with two 180 degrees field of view fish-eye cameras providing omnidirectional vision sensing. Hence, it is convenient to construct global visual maps by global structure from motion pipelines, e.g., COLMAP [12] and employ purely image based localization pipelines. Recently, a very successful pipeline Hloc [10, 11] become the state of the art, see the comparison in [3, 4].

Thus, in this deliverable we investigate whether and how Hloc can be used for global ARI localization in relevant BROCA hospital environment. We present a new localization pipeline 6.2 that uses elements of Hloc and combine them with additional processing of ARI images to achieve robust localization. We show, Section 5, that we can achieve 90% consistent recall between the global and local mapping and localization at 1.5 meter tolerance.

The deliverable is structured as follows. In Section 2, we describe BROCA hospital data acquisition with ARI. In Section 3, we describe how ARI fish-eye cameras were calibrated. Then, in Section 4, we explain how we have constructed a global 3D map using COLMAP 3D reconstruction pipeline. In Section 5, we describe and evaluate the Hloc localization in the COLMAP model we have constructed. To be able to evaluate as well as to connect the global localization to the local mapping, we align the global and the local maps in Section 6. We evaluate the quality of the global localization by comparing its agreement with the local mapping based on visual odometry.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

2 BROCA hospital data collection with ARI

The data used was collected from the first floor in the BROCA hospital, Paris during the SPRING experimentation week in April 2022.

The first set of data collection was performed with the ARI robot set in the mapping mode. It was found that there was a significant accuracy increase in the internal mapping of ARI when the auto exposure was turned ON for the torso front camera, an Intel Realsense D435i.

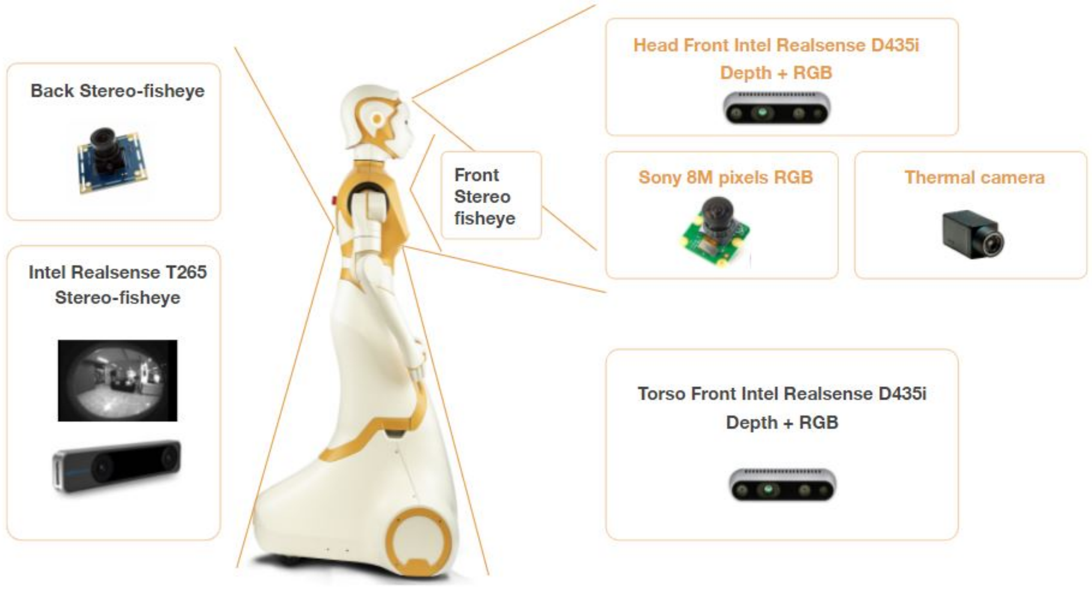


Figure 2.1: ARI Cameras Configuration. The Torso Front camera, an Intel Realsense D435i is used with ORB SLAM

The following topics were recorded in multiple rosbags for each session, since the recordings had to be stopped any time hospital staff crossed in the field of view of the robot:

- `/front_fisheye_camera/image_raw/compressed` with images from the front RGB fisheye camera positioned just above the touch-screen recorded at an average of 7 fps.
- `/rear_fisheye_camera/image_raw/compressed` with images the rear RGB fisheye camera above the emergency button recorded with with an average 7 fps.
- `/torso_front_camera/color/camera_info` with the camera calibration information for the Intel RealSense D435i RGBD camera.
- `/torso_front_camera/color/image_raw/compressed` with images from the RGB camera in the D435i module, recorded at 11 fps.
- `/torso_front_camera/aligned_depth_to_color/image_raw/compressed` with the depth information aligned to each pixel of the color image.
- `/torso_front_camera/infra1/image_rect_raw/compressed` and `/torso_front_camera/infra2/image_rect_raw/compressed` with images from the monochrome cameras in the D435i module, recorded at 11 fps each.
- `/torso_back_camera/fisheye1/image_raw/compressed` and `/torso_back_camera/fisheye2/image_raw/compressed` with the images from the fisheye cameras in the Intel T265 module in the rear of the robot.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

- Other topics with the tf transforms information, and the ARI internal SLAM outputs such as the estimated pose, and the point cloud.

As can be seen in the map in Figure 4.2c, the mapping was started with the robot in the room on the right, moved to the main hallway through the corridor performing a loop within the hallway, and returned back to the room again through the corridor. This ensured loop closure in the ARI internal SLAM, and also provides sufficient data to build a complete map, and localize from both sides (front and rear) of ARI.

Implementation Code and Model are available at: `https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

3 ARI Fish Eye Camera Calibration

The SLAM map is built with images from the front fisheye camera of ARI, which is a wide field-of-view fisheye (lens) camera. These cameras have significant distortion, especially towards the edges of the image, an example of which can be seen in Figure 3.1.

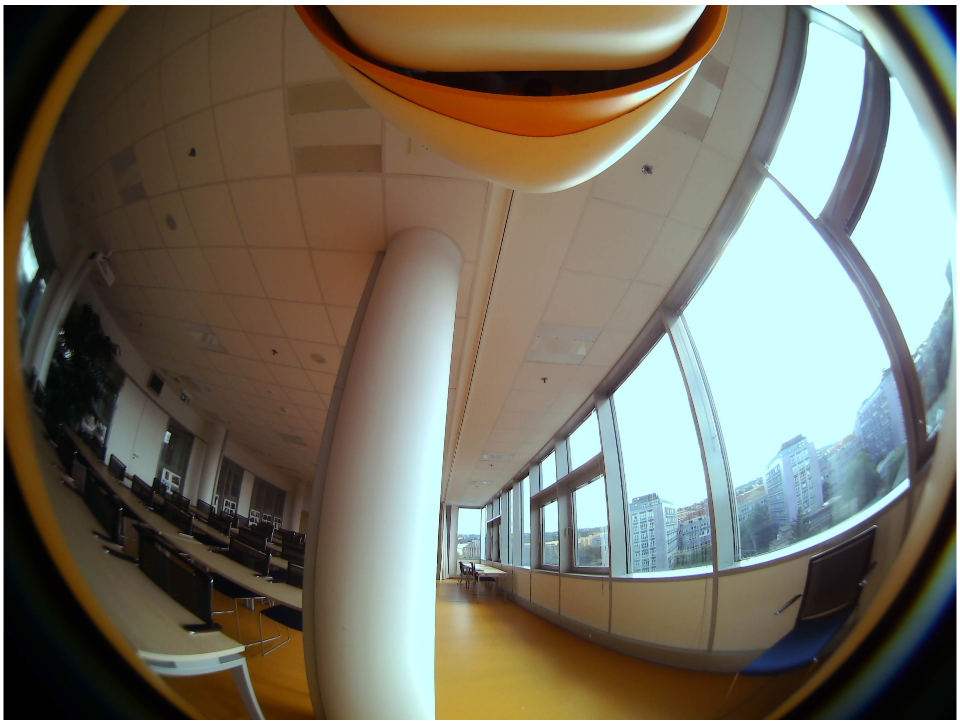


Figure 3.1: Example of image from front fisheye camera of ARI. The field of view is almost $180°$, but there is significant distortion.

For meaningfully working with these images, we need to model this distortion, and we do so with the OpenCV fisheye calibration package which uses the Kannala-Brandt model [5]. The model is briefly described below:
Consider a 3D point X in the world projected to image coordinates (a, b) by the Pinhole model without any distortion. For,

$$r^2 = a^2 + b^2$$

and

$$\theta = atan(r)$$

The fisheye distortion is modelled by a 8th order polynomial equation:

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \tag{3.1}$$

where the distorted point coordinates are:

$$\begin{aligned} x' &= (\theta_d/r)a \\ y' &= (\theta_d/r)b \end{aligned} \tag{3.2}$$

and the final pixel coordinates (u, v) we see in the actual image are:

$$\begin{aligned} u &= f_x(x' + \alpha y') + c_x \\ v &= f_y y' + c_y \end{aligned} \tag{3.3}$$

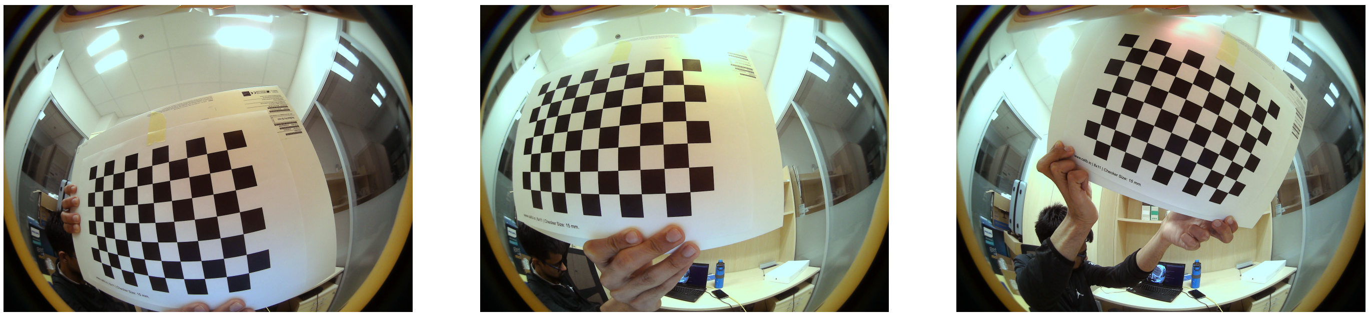


Figure 3.2: Examples of the checkerboard pattern in different configurations for the fisheye calibration task.

In the calibration task, we estimate the intrinsic parameters of the camera from the pinhole model, as well as the distortion parameters k_1, k_2, k_3, k_4 from (3.1)

The OpenCV fisheye camera module [2] is used for this task with a checkerboard pattern for the corner detections in various orientations and distances. Examples of images used for calibration are shown in Figure 3.2. Using the OpenCV functions, we compute the corners in the checkerboard, and with multiple 2D-3D correspondences we optimize for the best set of intrinsic and distortion parameters.

For the input images used, we obtain the following parameters for the front fisheye camera: Intrinsic camera matrix

$$K = \begin{bmatrix} 1.18869e03 & 0 & 1.631948e03 \\ 0 & 1.189301e03 & 1.242818e03 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

and the distortion parameters

$$\begin{matrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{matrix} = \begin{matrix} -0.0650748646548443144 \\ 0.022948182911307041 \\ -0.0039319897787193784 \\ -0.0037411424922587362 \end{matrix} \tag{3.5}$$

This gives an average reprojection error of $2\,px$ which is not insignificant, and so this is only used as an initialization input to the Structure-from-Motion (SfM) process.

Implementation Model, code and data are available at: `https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

4 3D Map Building from ARI images based on COLMAP

First of all, we tried to build a map using the COLMAP Sparse Reconstruction method on all of the gathered images. This did not work and resulted in unusable maps, as can be seen in Figrue 4.1.

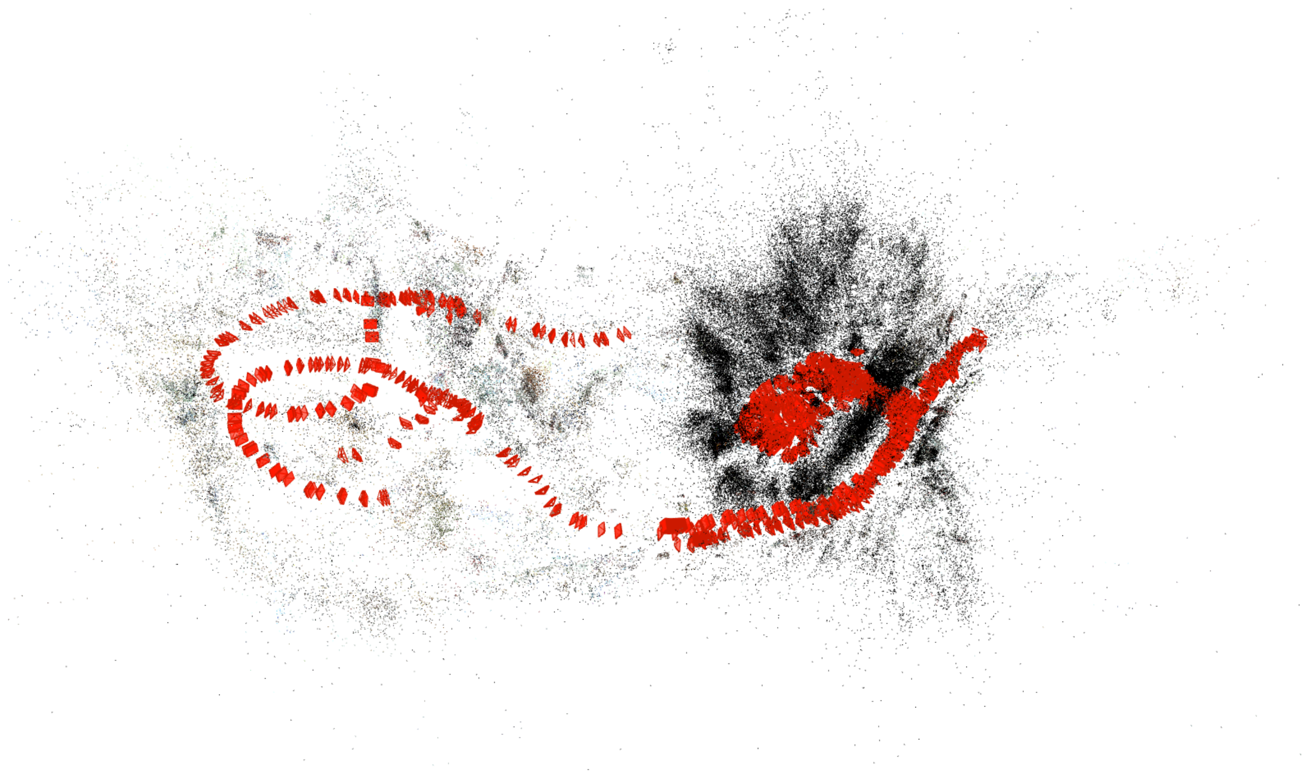
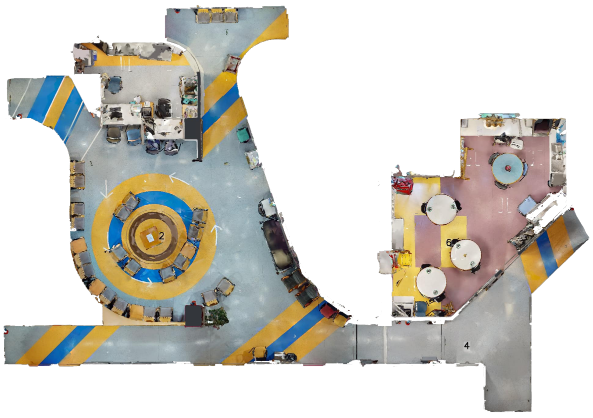


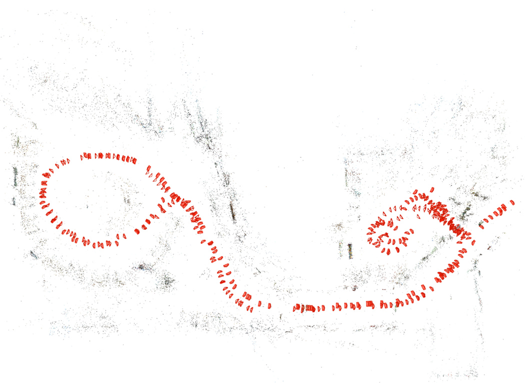
Figure 4.1: COLMAP sparse map reconstruction on the full dataset. This map doesn't represent the path of the robot. The images were not matched at all in some places creating a separate loop of corresponding images instead.

To build a good map from this dataset we had to take into account how the data was gathered and what quality of images it produced. We will discuss this in the next two sections; Section 4.0.1 and Section 4.0.2.

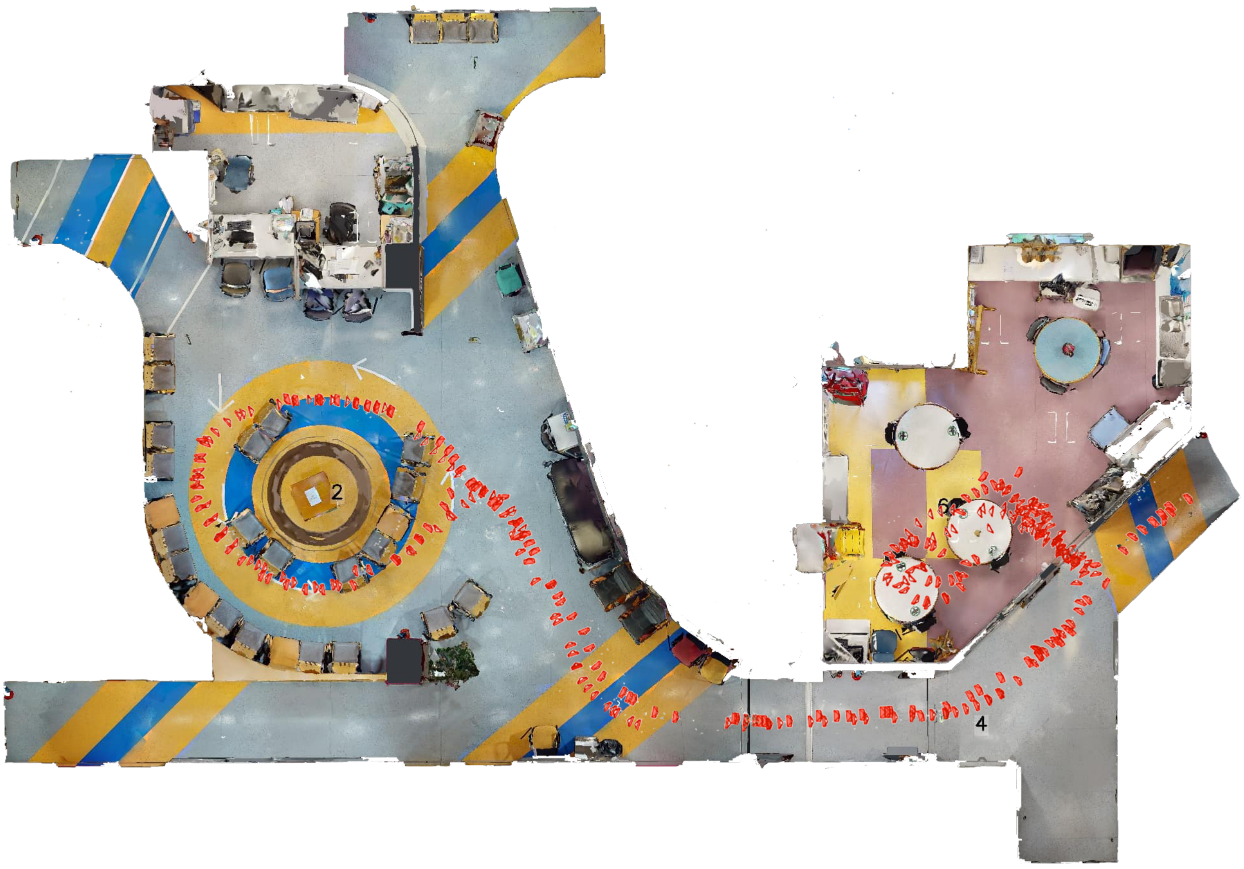
This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.



(a) Matterport map.



(b) COLMAP map reconstruction created from the blur and distance filtered data. The data filtering process in described in 4.0.2



(c) COLMAP reconstruction ma overlayed over the Matterport map.

Figure 4.2: Matterport map visualized as a floor plan and COLMAP map reconstruction.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

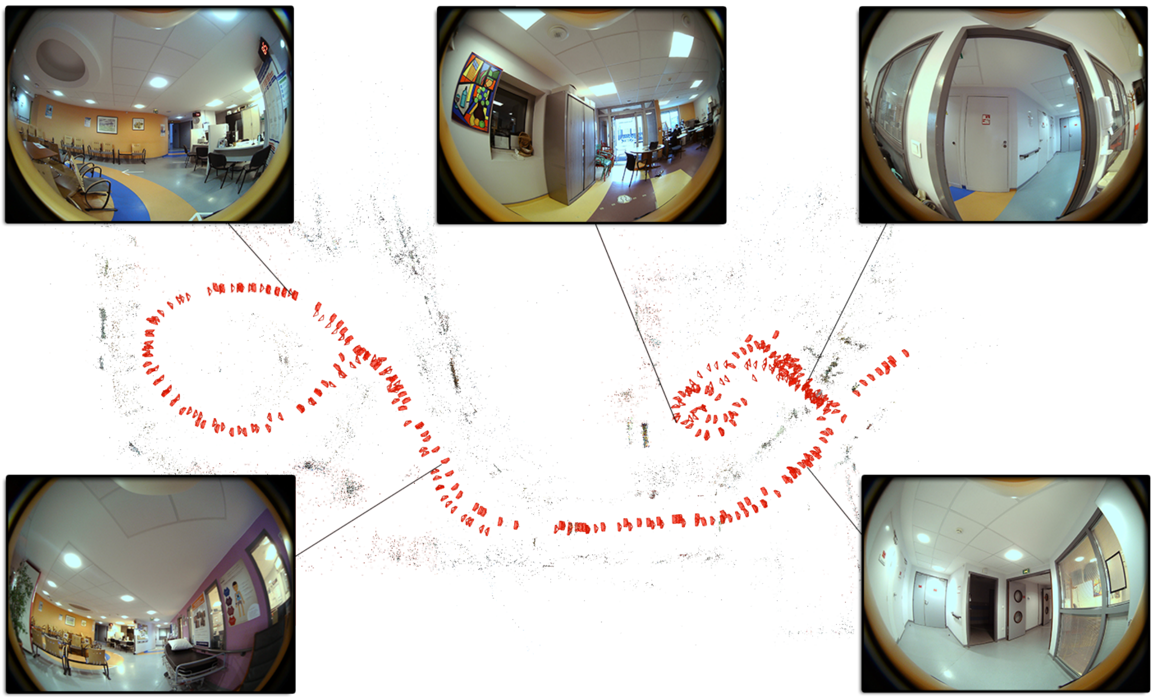


Figure 4.3: COLMAP map reconstruction with dataset image examples.

Implementation Code and Model are available at: `https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`

4.0.1 Dataset Nature

As the robot was moving, it captured images from two cameras – one on its chest and the other on its back, called front and rear cameras. The robot was also saving its pose (including its location and orientation) throughout the movement using ORB SLAM [8]. The robot sampled the frames of these cameras in real-time. Due to the limited bandwidth and computational load incurred by many processes running simultaneously, the sampling process is not perfect. Sometimes, the frames saved are mere hundredths of a second apart and, on other occasions, only seconds apart. Also, when collecting the data, the robot had to be often stopped, which led to multiple capturing of many places. This led to a couple of problems when creating the map, which we remedied by preprocessing the dataset.

The ORB SLAM poses, with clumped regions, can be seen in Figure 4.4.

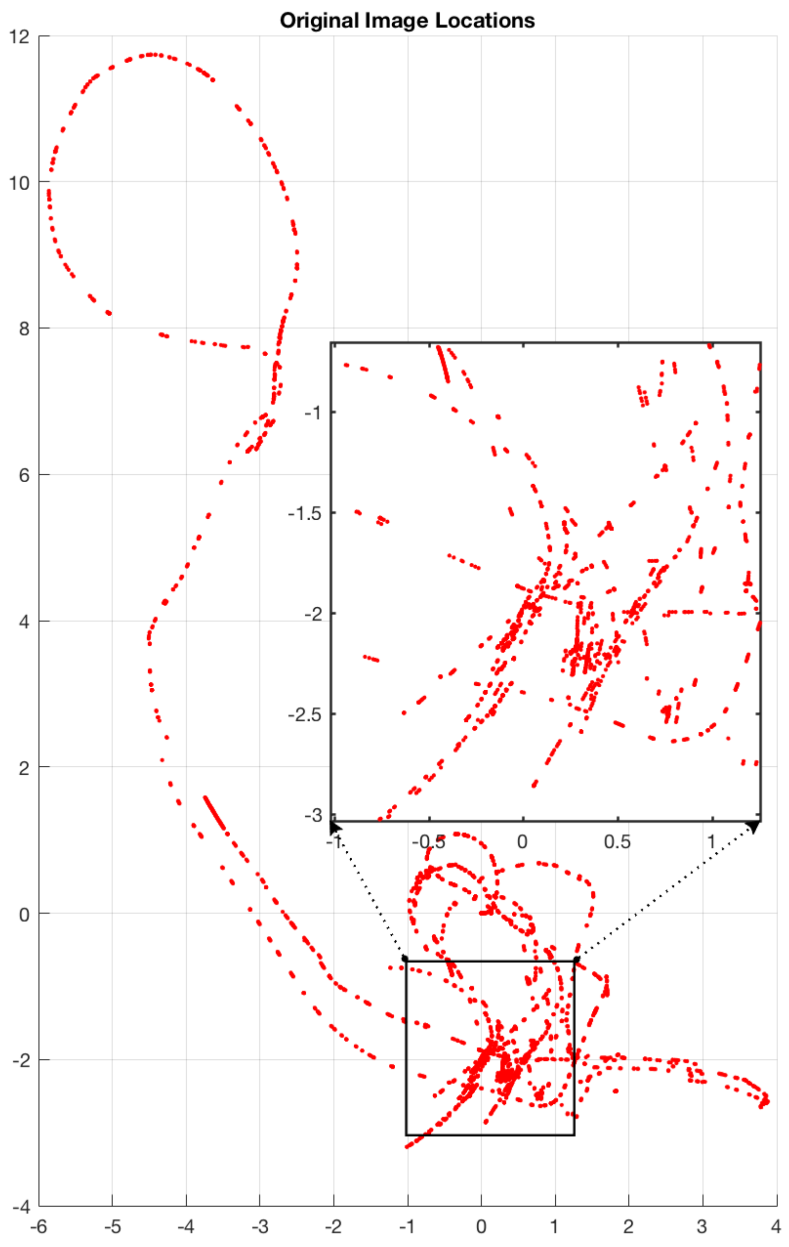
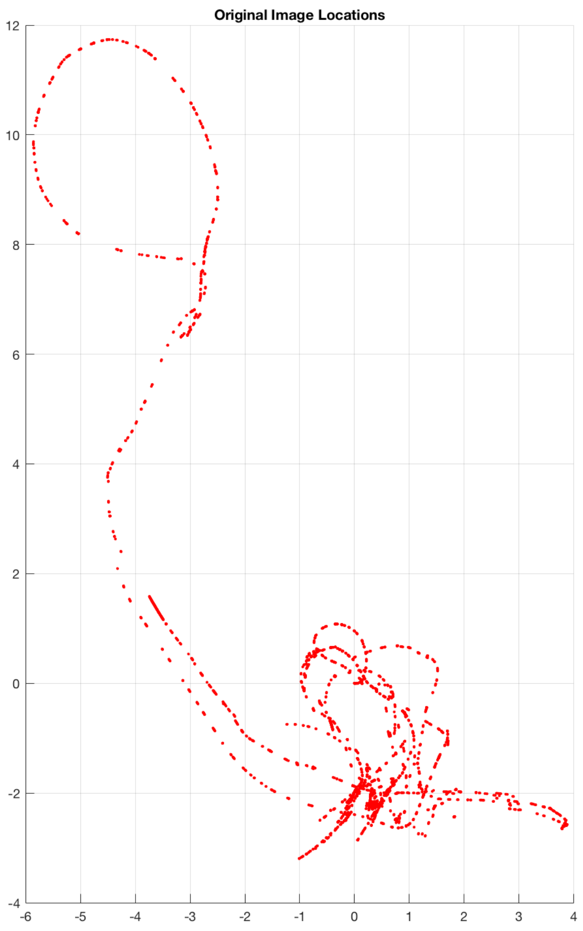


Figure 4.4: ORB SLAM image locations. Note that there are many places where the images were taken almost simultaneously, making for almost identical images. This can be seen in the magnified region.

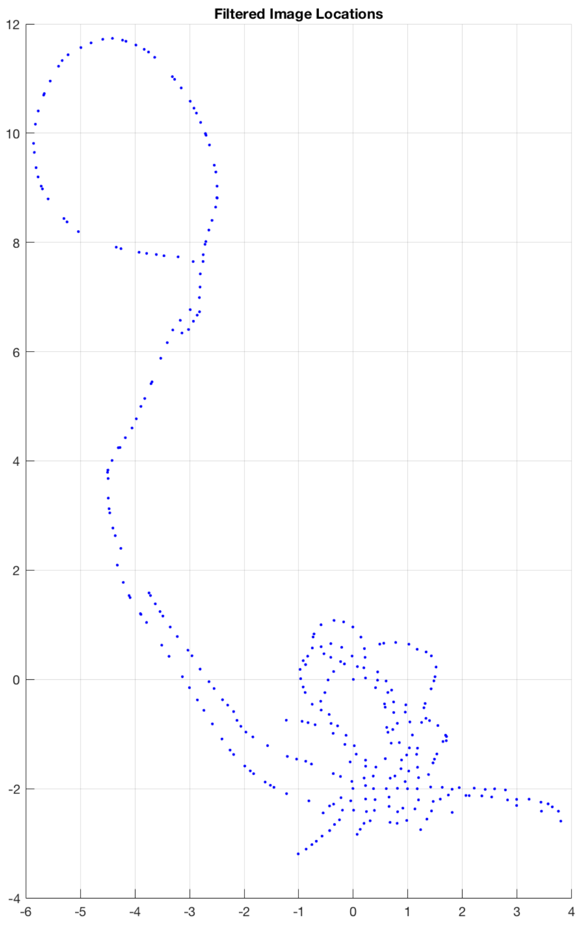
4.0.2 Dataset Preprocessing

First, we had to remove the blurred images caused by the robot's movement. Blurred images were filtered using a variation of Laplacian filtering [9].

Secondly, we had to filter out images taken at almost exactly the same positions, making them almost identical. We created a regular grid and picked a single image in each cell of the grid. The results of the pre-processing are shown in Figure 4.5.



(a) Original image locations. Clumped up in places.



(b) Filtered image locations.

Figure 4.5: Unfiltered ORB SLAM image locations on the left and images filtered by snapping to a grid on the right.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

5 Visual localization based on HLoc

To test the HLoc localization, we localized more than 500 test images that were not used to build the COLMAP map and plotted them onto a scatter graph together with the map. All but 16 localized query images were on the robot's trajectory. This can be seen in Figure 5.1.

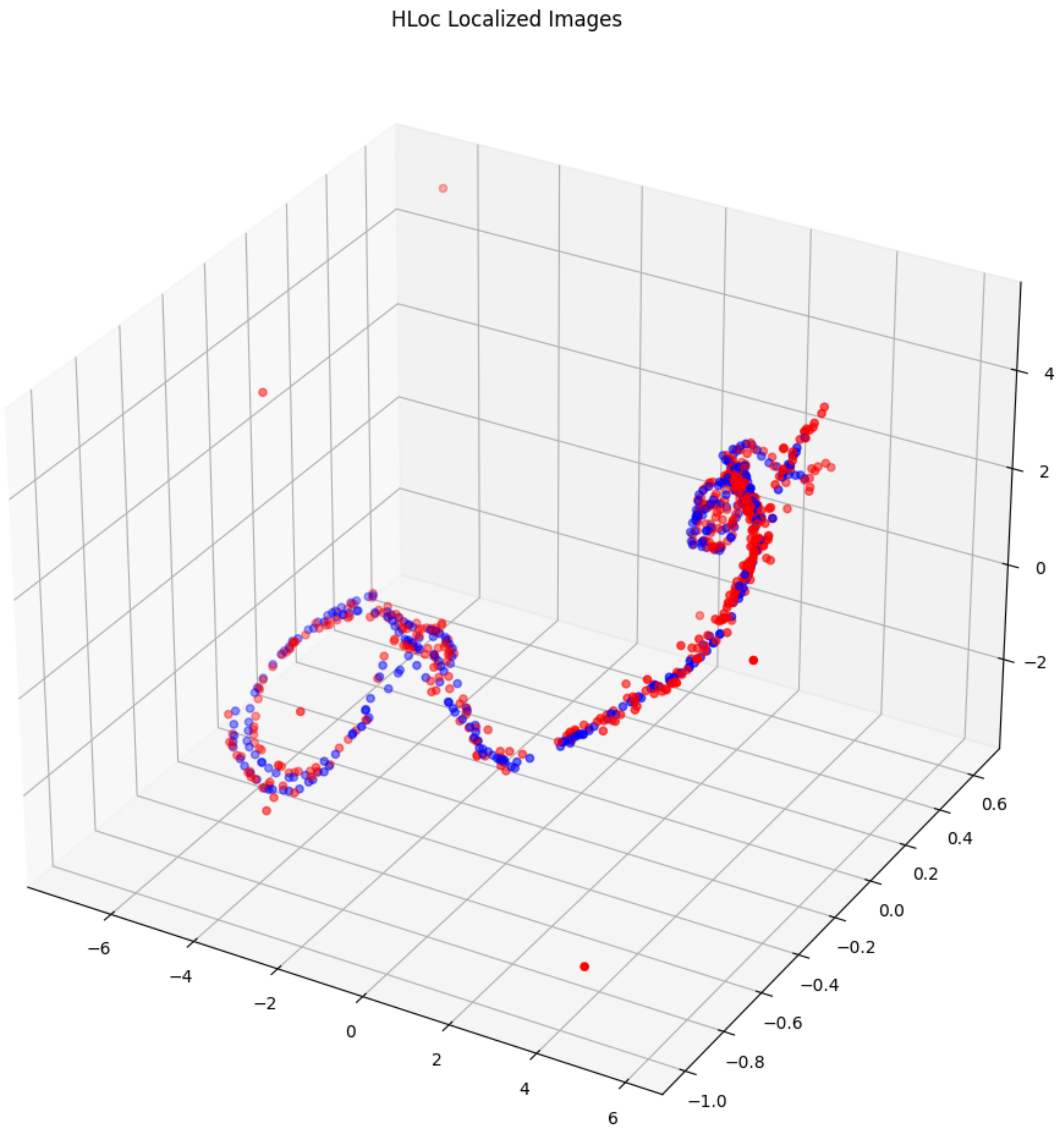


Figure 5.1: Localized images using HLoc in red and COLMAP map reconstruction in blue.

In Figure 5.2, we show that these points not only lie on the robot's trajectory, but 90%, resp. 75, of them are within a 1.5, resp. 1, meter distance from the locations obtained from aligned, Section 6, ORB-SLAM visual odometry. The error is measured by comparing the positions obtained by Hloc with those obtained by ORB-SLAM odometry. Therefore, our evaluation measures the agreement between the two methods, not the agreement with a correct ground truth.

Figure 5.2 shows the localization based on a single image from the front or rear ARI fish eye in red. ARI localization based on sequences of three consecutive images from either front or rear cameras is shown in blue. We see that the results are almost the same.

The results show that global localization is capable of reliably determining the starting point of the ARI robot on the level of rooms to retrieve the local map of a room and start ORB-SLAM localization in the room. However, it is still not accurate enough to provide the absolute pose of ARI within centimeters of the robot from a short sequence of images from a single camera.

We see several sources of errors that lead to our results. First, the 3D mapping of COLMAP is far from perfect.

First, we discovered that the trajectories reconstructed from front and rear ARI fish-eye cameras sometimes differ by tens of centimeters. This can be seen, for example, in Figure 4.2c in the corridor just in front of the larger room. Although the images were taken from approximately the same place (just on the other side of the robot), the cameras in the reconstruction were placed in different locations. Secondly, some parts of the scene are much less covered by the 3D map than others because of dropouts in image acquisition. To improve the quality of the 3D model, we will calibrate the two fish-eye cameras of ARI as a common rigid camera rig and improve the fish-eye image acquisition to fit the frame rate to the available bandwidth.

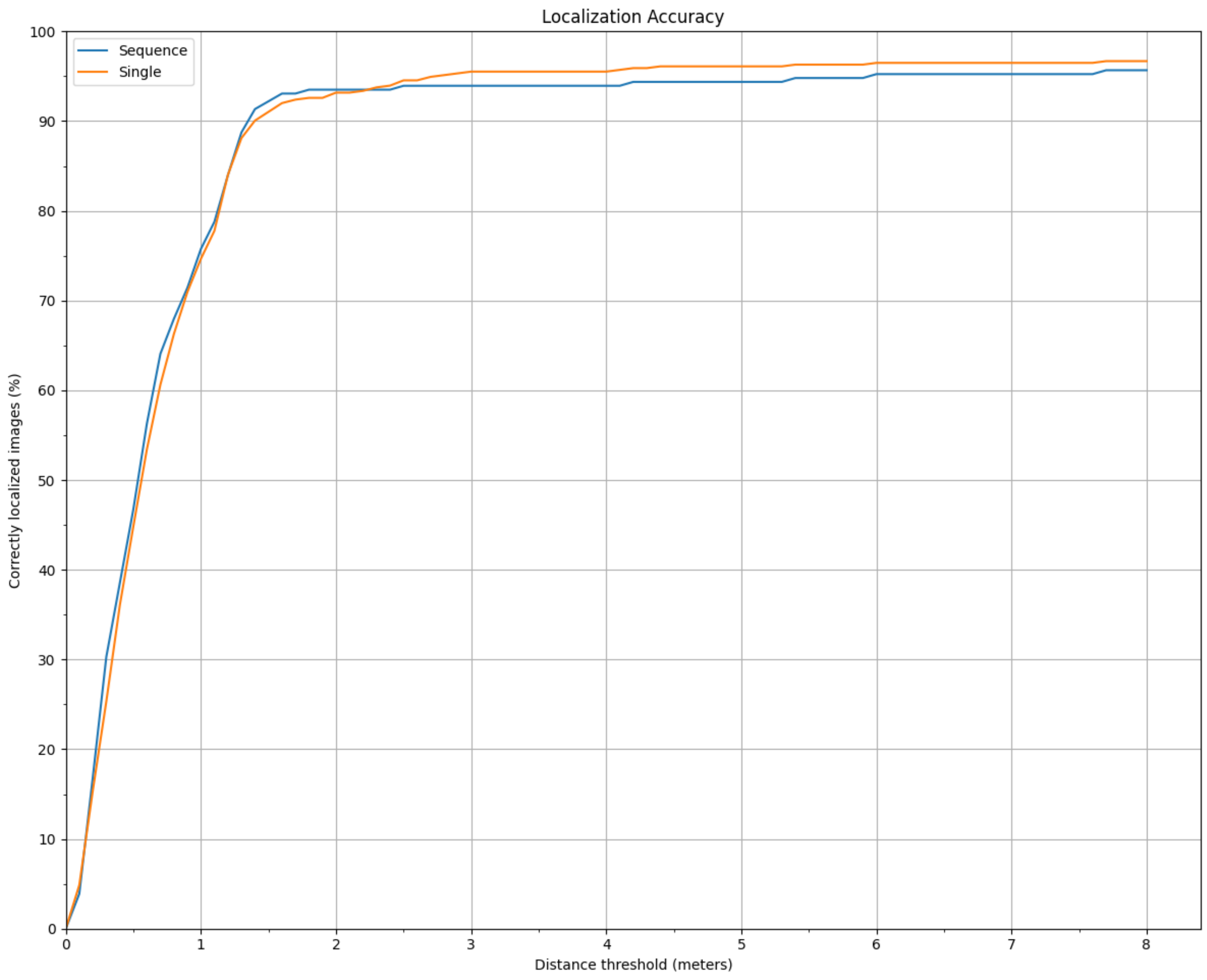


Figure 5.2: The graph shows the fraction of correctly localized queries (y-axis) within a certain distance (x-axis). In orange are images localized in groups of three when knowing their relative poses from ORB SLAM. In blue are images localized one at a time.

6 Aligning the global HLoc map with the local ORB SLAM map

The alignment of the global HLoc map with the local ORB SLAM map is done by Procrustes analysis, i.e., the transformation — rotation, translation, and scale — is obtained by minimizing the objective function

$$\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{y}_i - (cR\boldsymbol{x}_i + t)\|^2 \,, \tag{6.1}$$

where $\mathbf{y}_i$ and $\mathbf{x}_i$ are the corresponding 3D points and R, t and c represent the desired transformation consisting of rotation, translation, and scale, respectively [15].

The models aligned using this transformation can be seen in Figure 6.1

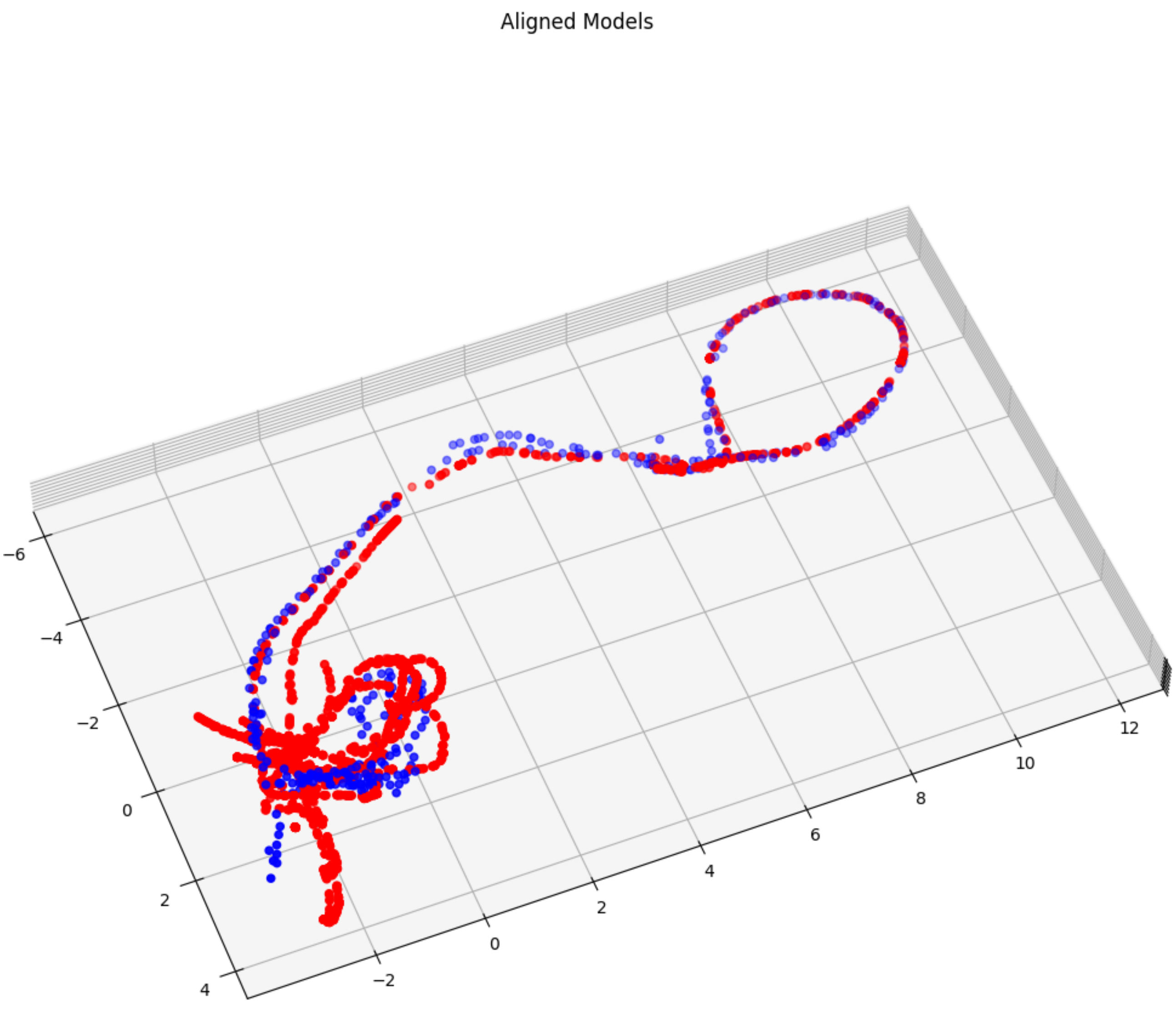


Figure 6.1: Aligned map reconstructions. HLoc/COLMAP in blue and ORB SLAM in red.

6.0.1 Pipeline

The whole process of preprocessing data, creating maps, localizing and aligning maps is carried out using the pipeline shown in Figure 6.2.

The pipeline consists of the following processing units:

- **KeyframeSelector** takes all the images and returns the sharp ones, as described in Section 4.0.2.
- **ImageSelector** takes images and image locations from ORB SLAM and returns a subset without those taken from the same location as described in Section 4.0.2.
- **KeypointsDetector** tinds key points in images and saves them to a database in COLMAP format.
- **Matcher** computes matches between all pairs of images.
- **Mapper** creates a COLMAP map from image key points and matching image pairs.
- **HLocMapCreator** creates an HLoc map from the images and the COLMAP map.
- **BrocaHLocQueryComposer** creates testing data for localization using distance-filtered images from ImageSelector and omitting images used by the Mapper.
- **HLocLocalizer** localizes query images in a HLoc map.
- **IOConverter** converts the ORB SLAM model into COLMAP format.
- **ModelsAligner** cAlignes two models as described in Section 6.
- **ModelsVisualizer** visualizes the aligned models.

Implementation Code and Model are available at: `https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

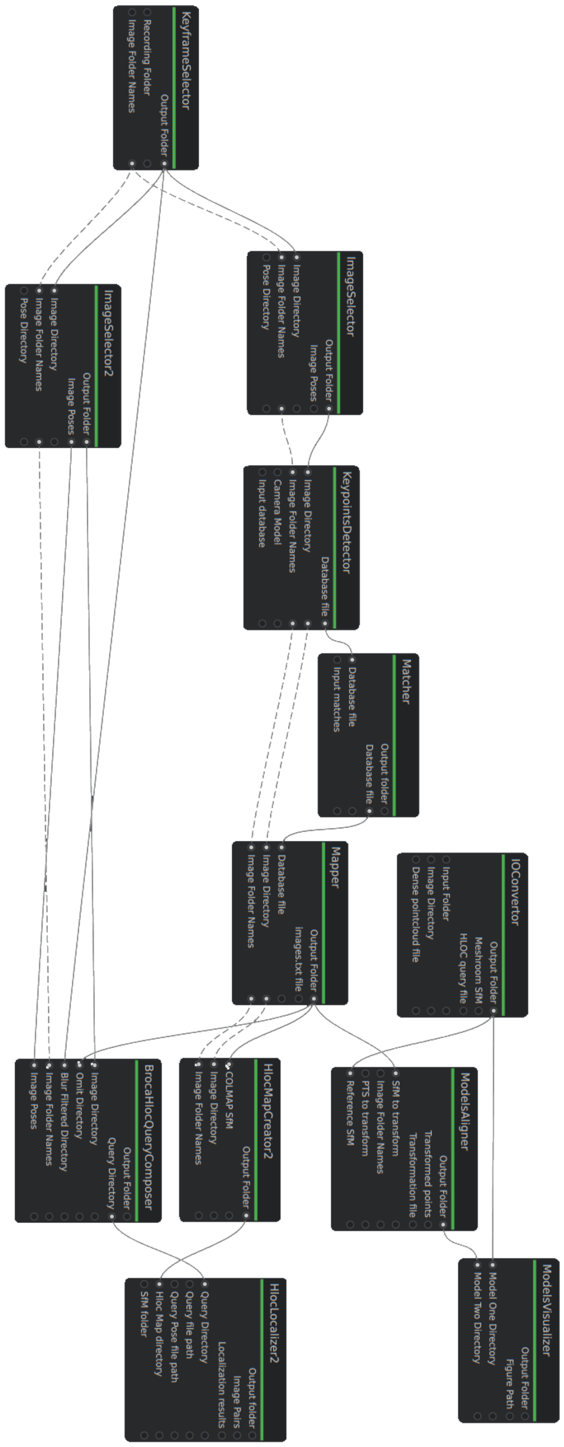


Figure 6.2: Pipeline used to preprocess and filter images, find key points in images, create COLMAP and HLoc map reconstructions, run images on the HLoc localizer and align models.

7 Conclusions

This deliverable demonstrated that our localization pipeline, based on Hloc, can (i) build global 3D maps from ARI sensors and (ii) achieve 90% consistent recall between global and local mapping and localization with 1.5 meters tolerance. This is sufficient for global localization. We suggest further improvements to reach higher accuracy in Section 5.

All code and data are available in the project gitlab repositories:
`https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`
`https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`
`https://gitlab.inria.fr/spring/wp2_mapping_localization/mapping_localization`
where it will be available for at least 4 years after the end of the project.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

Bibliography

[1] 3d models by matterport. https://matterport.com/.

[2] Opencv fisheye camera module: https://docs.opencv.org/4.x/db/d58/group__calib3d__fisheye.html.

[3] Cvpr 2020 joint workshop on long-term visual localization, visual odometry and geometric and learning-based slam, 2020. https://sites.google.com/view/vislocslamcvpr2020/home.

[4] Eccv 2020 long-term visual localization under changing conditions, 2020. https://sites.google.com/view/ltvl2020/.

[5] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE TRANS. PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 28:1335–1340, 2006.

[6] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *J. Field Robotics*, 36(2):416–446, 2019.

[7] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[8] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[9] J.L. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia. Diatom autofocusing in bright-field microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 314–317 vol.3, 2000.

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

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

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

[13] Stanislav Steidl and Tomas Pajdla. D2.1: isual-based localisation in realistic environments, 2021. EU H2020 SPRING Project No. 871245.

[14] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018.

[15] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.