GRANT AGREEMENT N. 871245

Deliverable D2.1

Visual-based Localisation in Realistic Environments

Due Date: 31/05/2021
Main Author: CVUT
Contributors: INRIA, PAL
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 no.	D2.1: Visual-based Localisation in Realistic Environments
Responsible Partner	CVUT
Work Package	WP2: Environment Mapping, Self-localisation and Simulation
Task	T2.1: Robot Self-localisation from Images
Version & Date	V2, 30/05/2021
Dissemination level	[X] PU (public) [ ] CO (confidential)

CONTRIBUTORS AND HISTORY

Version	Editor	Date	Change Log
1	CVUT	24/05/2021	First Draft
2	CVUT	31/05/2021	Final Draft including all partners comments

APPROVALS

Authors/editors	CVUT
Task Leader	CVUT
WP Leader	CVUT

D2.1: Visual-based Localisation in Realistic Environments

Leader: CVUT
Contributors: INRIA, PAL

May 2021

Contents

1 Introduction

One of the *strategic objectives* (**StO-1**) of **H2020 SPRING** is to enable robust robot perception in complex, unstructured and populated environments. A necessary component of it is a *specific objective* (**SpO-1.1**) which aims to perform self-localisation and tracking in cluttered and populated spaces. Given a 3D map of the environment, e.g. a hospital or a museum, visual localisation will rely on image-based place recognition and will be integrated into the robot navigation planners that are already available.

The work package two (**WP2**): *Environment Mapping, Self-localisation and Simulation* led by CVUT aims to provide solutions to this objective (among others). Specifically the task **T2.1** named *Robot Self-localisation from Images* is tailored to focus on the self-localisation which is the first and necessary step to accomplish the objective. In this task we develop visual localisation based on viewpoint visual similarity. This will be achieved by retrieval of candidate camera positions, followed by camera pose estimation via geometric matching and careful camera pose verification using camera pose computation in 3D or similar weaker geometrical verification techniques.

The outcome of **T2.1** is split into two parts (**D2.1 – Visual-based localisation in realistic environments** and **D2.4 – Visual-based localisation in relevant environments.**). The aim of **D2.1** is to put together the general self-localisation module with off-the-shelf components which will be further enhanced and the final version will be presented in the form of **D2.4**. This work presents the first part, the deliverable **D2.1**.

Our work builds on well known work of H. Taira et al. **InLoc**: *Indoor Visual Localization with Dense Matching and View Synthesis*[1]. We build localisation map from data gathered in Broca hospital where will take place the experimental evaluation of the project. Part of the data is used to evaluate precision of localisation.

2 Visual Self-Localisation

The task of visual self-localisation (localisation for further reference) can be defined as following task: *Given a photograph, find the precise position where it was taken.* We assume that we have access to a visual map of the environment where the image was captured and the result is a six degree-of-freedom pose (rotation and translation vector) with respect to the map. We aim for a robust system which can be applied even in environments which experience minor to moderate changes on a daily basis and are not reflected in the map, and thus the map cannot be fully relied upon. Second desired property is the ability to handle multiple distinct environments of arbitrary size at once. In case of SPRING this would include all places where we want to deploy our robot.

2.1 InLoc: Indoor Visual Localisation with Dense Matching and View Synthesis

We follow the approach proposed by Taira et al. in InLoc[1]. The visual map is constructed as a database of images with known poses and depth information. The database is accompanied by a mesh model of the environment. The localisation can be divided into four main steps:

1. In order to localize a query image in this map, we first search the database for images looking at the same scene, this is the **Image retrieval** step.

2. The second step is to extract and verify matching feature points between a retrieved image and the query. We select only those tentative matching feature points which are supported by a rigid geometrical transformation. **Geometrically verified** points are more likely to be true matches and can be used for pose estimation. After this step we re-rank the retrieved images and continue with only a few best.

3. The third step is to take the retrieved images from the second step and solve relative the **pose estimation** problem based on matched and verified features. The outcome of this step is a candidate camera pose per retrieved image.

4. The final step is a **pose verification** step where we project the mesh model into the candidate camera pose forming a synthetic image. The synthetic image is then compared via similarity measure to the original query image. Candidates are re-ranked accordingly and the pose with the highest score is deemed as the final pose estimate.

2.2 Image Retrieval

In order to efficiently retrieve related images, we follow the netVLAD idea of Arandjelovic et al. [2]. A modified VGG16 network[3], is used to estimate image embedding that is then comapred to retrieve most similar image.

The original VGG16 network is a deep convolutional neural network with approx 138 million parameters. The network is composed of convolution layers followed by ReLU units[4] and max-pooling layer as every fourth layer. The final two layers are fully connected and were originally used for image classification.

We use a pre-trained VGG16 network on Pittsburgh30K dataset[5] and the last two layers are removed and replaced with the **netVLAD** layer. This layer takes (in our case) 4096 dimensional image embedding which is further compared to 8 pretrained centroids and residuals are computed. The residuals are then L2 normalised and concatenated into a single 32768 dimensional vector – **netVLAD** vector. This vector is then computed for every database image and and stored.

To evaluate a query, we first compute the netVLAD vector of the query. The similarity of two images is then computed as a dot product of their embeddings. By comparing the query to all database images we get the list of candidate images for the second step. We use the comparison results to rank the database images and top hundred candidates are used in next steps. See Figure 1 for an example of results.

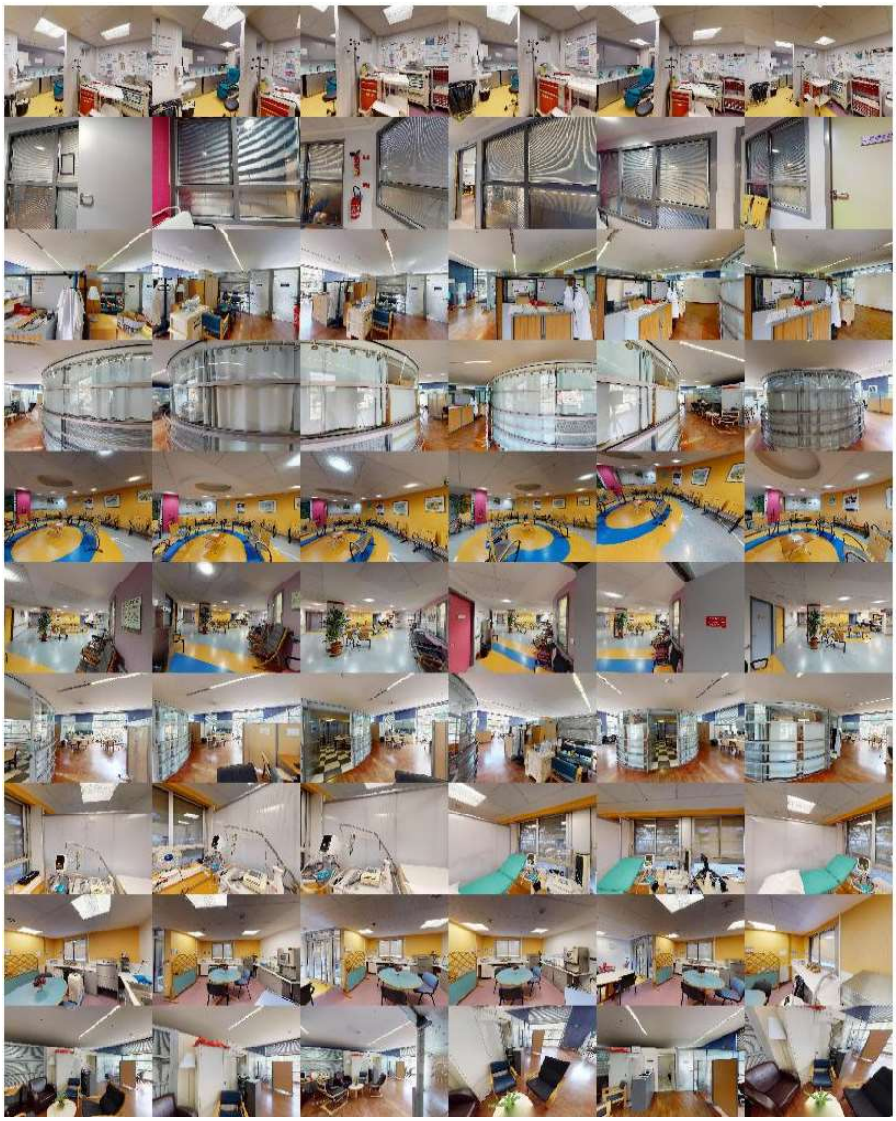

Figure 1: Image retrieval step results for 10 randomly sampled queries. Leftmost image is the query image followed by the top five retrieved images, sorted from left to right. $Ids = [158, 137, 275, 284, 67, 174, 360, 229, 250, 265]$

2.3 Geometrical Verification

At this stage we have a ranked list of images observing the same scene. First step of pose estimation is to find matching feature points in the query and a database image. Dense feature points are extracted from results of intermediate

layers of VGG16 used for netVLAD computation. The features are retrieved at the coarser layer and pruned by the finer layer using third and fifth layers respectively. Such tentatively matched feature points are visually similar. We compute homographies between feature points and image plane using well known random sample consensus method (**RANSAC**)[6] and select two with largest support.
RANSAC, in general, is a model fitting method. The algorithm iteratively takes random samples of data, estimates model parameters based on the sample and evaluates model support by number of data (inliers) within a specified error range. The model with highest support is then selected.

The supporting feature points are then considered as weak inliers – verified, yet tentative matches for further steps. We re-rank the candidate reference images by amount of these inliers and select only ten best for next steps. See Figure 2 for closer inspection of feature points.

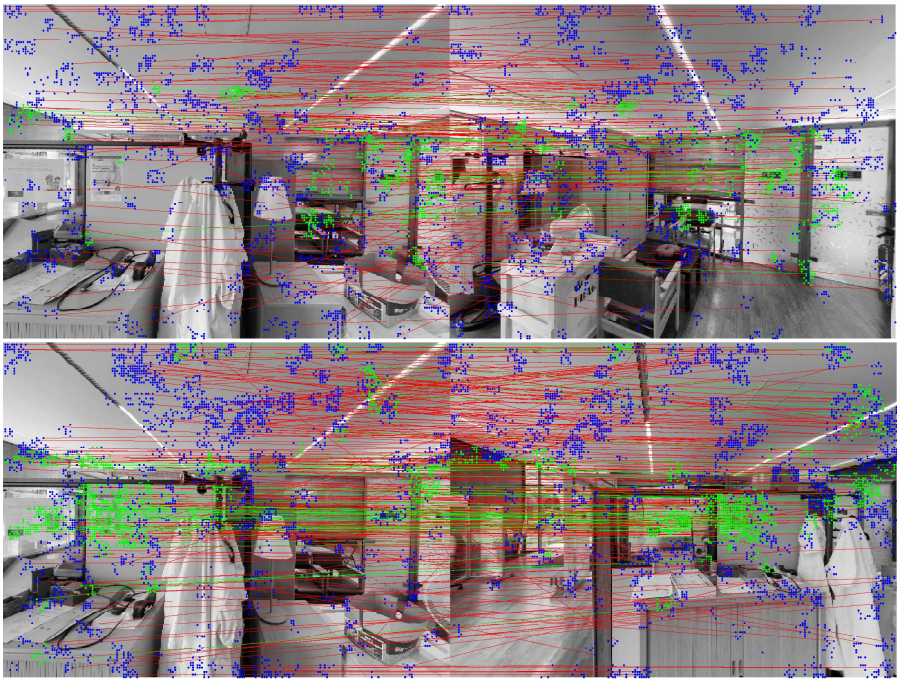

Figure 2: Dense feature verification with query on left and database image on right. Dense feature points (**blue**) and their tentative matching (**red**) are verified using up to two homographies. Supporting feature matches are displayed in **green**. Upper pairs shows matching of database image with highest ranking after image retrieval for query id 275. Below is the highest ranked database image after the **geometrical verification** step re-ranking. The feature matching is performed on RGB images, the gray-scale is used for visualisation only. We also display only 5% of matches (lines) for better clarity.

2.4 Pose Estimation

In this step we use known depth information of the database images to estimate camera pose. The geometrically verified feature points form image-to-image (2D-2D) correspondence pairs. We use the 2D-2D information to link the 3D depth data of the database image with the query 2D feature points forming a set of 2D-3D correspondences. Having access to reliable correspondences between image points and points in space, we establish a relative position of the image capture with respect to the points in space and consequently to the database image with known pose w.r.t. world frame. The relative pose is estimated via standard P3P solver using locally optimized RANSAC[7]. The pose with highest support among correspondences is selected. The quality of pose can be measured in terms of reprojection errors as visualized in Figure 3.

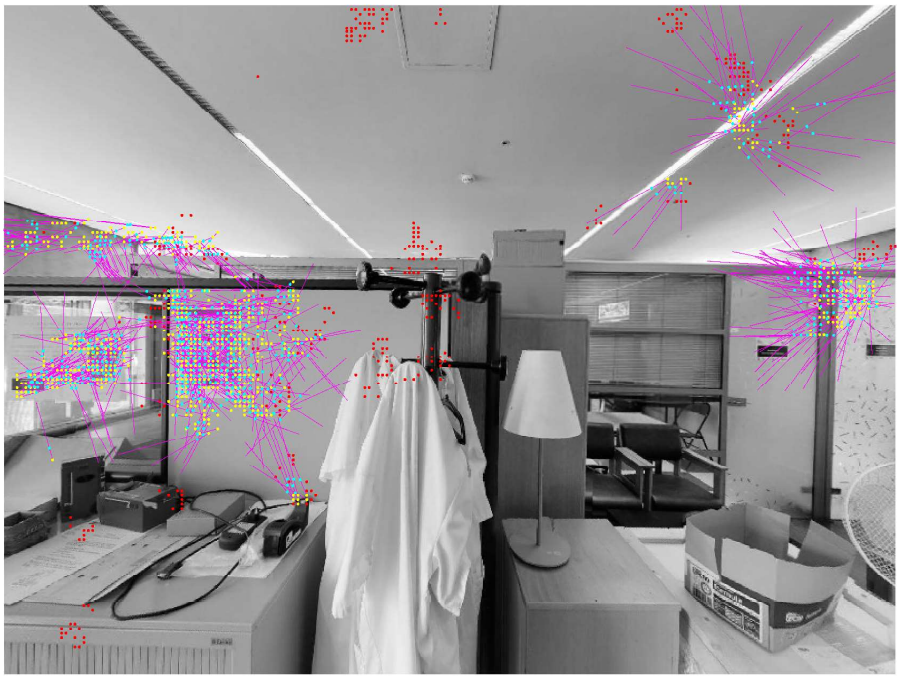

Figure 3: The query feature points (**yellow**) are reprojected back into the query image (**cyan**). The magnified ($s = 5$) reprojection error (**magenta**) shows the direction of error. If more reprojection errors display a clear pattern it is a sign that pose is not well estimated. The feature points with larger reprojection errors than set threshold ($t = 2.5°$ ray angular error) are considered as outliers (**red**).

2.5 Camera Pose Verification

In order to re-rank the candidate camera poses and ultimately select one as a result we perform a verification step. We use the mesh model of the environment and we render a synthetic image from the candidate pose. We aim to select a pose which produces the most similar synthetic image to the query. The similarity is estimated over the whole image. The blank areas of synthetic image are filled by interpolation or extrapolation of nearby pixels. The similarity is estimated as the inverse of median square error over Root SIFT descriptor extracted by DenseSIFT extractor [8, 9]. We use implementation of VLFeat [10] package. The pose with the highest similarity is then selected as a response to the query. The Figure 4 shows the final query to synthetic image comparison for the result pose of query id 275 and Figure 5 shows the top five considered reference poses. The query id 275 has been localised with $0.07m$ translation and $2°$ angular error w.r.t. to the reference pose.

Figure 4: The comparison of the query image (left) and synthetic view (second to left). The candidate pose was estimated from a reference database image (right). The image second to right provides a **blend** of query and synthetic image to better inspect residual misalignment.

Figure 5: The comparison of query image (left) and top five pose candidates, presented as blends (ordered by quality from second-to-left to right image).

3 Modeling Broca Hospital

To validate that our approach is suitable for the hospital environment, we created a precise virtual 3D model of Broca hospital. The model is based on visual data collected as a set of panoramic sweeps by a very precise Matterport3D measurement unit. We have models of two distinct wings within the hospital: the Day Care and Living Lab facilities (see Figure 6) in two different view conditions. As **D2.1** aims to provide an initial solution to the visual self-localisation it is expected that a map of the environment in suitable format is provided. The models as well as the sets of panoramas captured with known poses are used as the foundation of Broca Hospital environment map.

The panorama set is split into two distinct subsets: the **database** panoramas (90%) and the validation **query** panoramas (10%). The localisation map is built from the former.

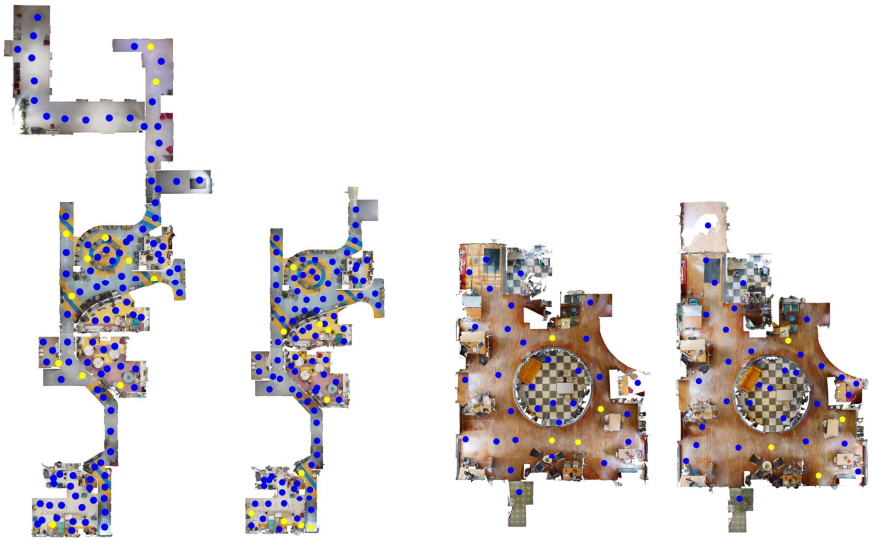

Figure 6: Broca Hospital dataset: Floor plans of DayCare (two on the left) and Living Lab (two on the right) models. Models are named hospital 1, hospital 2, living lab 1, living lab 2 from left to right respectively. **Blue** dots display poses of database panoramas and **yellow** display panoramas used as queries.

3.1 Localisation Map

We construct our image database by cutting panoramic pictures into a set of photographs without panoramic distortion and 106°, 90° horizontal and vertical field of view respectively. 12 cutouts with 30° yaw steps are created in three pitch levels totalling in 36 cutouts to cover the whole panoramic view. To add depth information we project the corresponding mesh model into the panorama reference pose. Figure 7 shows an example on one of the panoramas. The database consist of 9144 cutout images extracted from 254 panoramas.

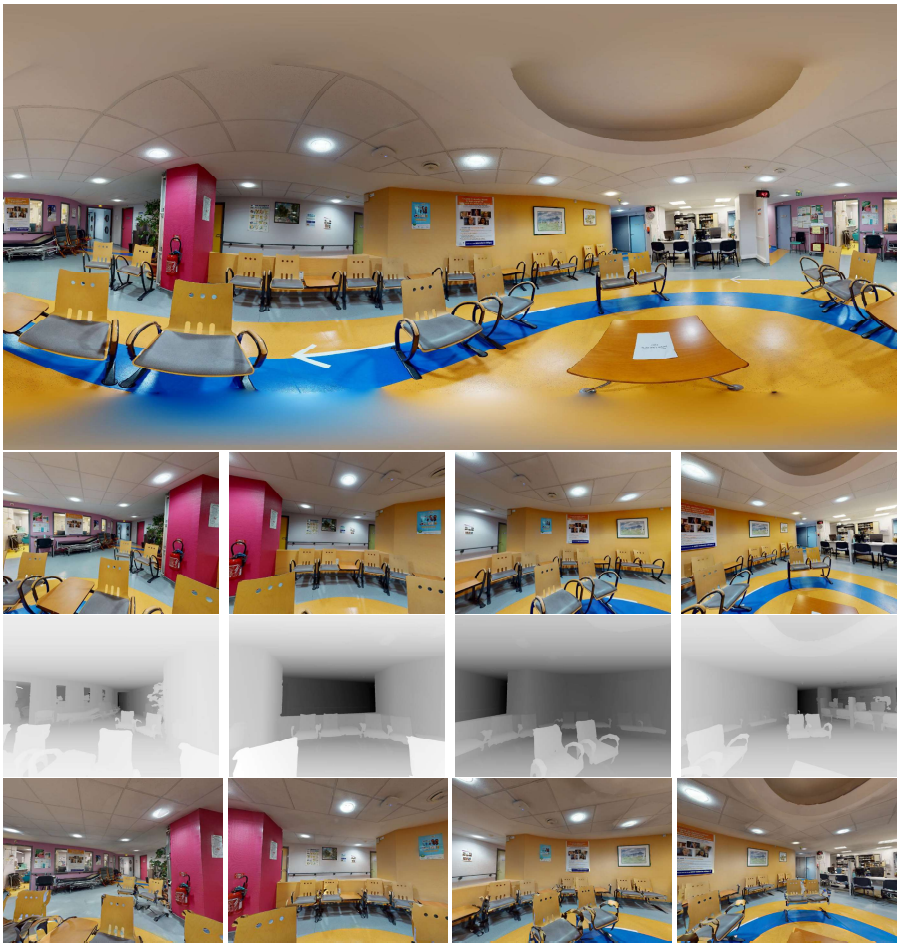

Figure 7: The panorama (top) is cut into a set of 36 undistorted images – cutouts (second row). The depth information(third row) is extracted from 3D model (bottom row) by projecting the model into the estimated cutout camera pose.

We observed a varying inconsistency between reference rotations of panoramas provided by Matterport and the 3D model. We replaced the reference rotation information by our estimate. See Figure 8 for comparison. We estimate the true rotation by finding the best alignment of a single cutout to synthetic views generated by sampling mesh projections under varying rotations.

Figure 8: Blended mesh projection with a cutout of a panorama. The reference rotation (left) compared to our estimate (right).

4 Validation

To assess suitability of proposed the method in the hospital environment, we extract 10% of panoramas prior map creation and evaluate the localisation on this query subset. The query panoramas are preprocessed in similar fashion as database panoramas with the only exception that only horizontally aligned cutouts are used, resulting in only 12 cutouts per panorama compared to 36 in case of database panoramas. The reason to use horizontal cutouts only is that cutouts facing floor or ceiling are very difficult to assess even for a human observer. The query set consist of 360 query images extracted from 30 panoramas selected randomly over all four models.

As can be seen from Figure 6, the models taken at the same part of hospital cover almost the same area. There are some appearance changes as we removed blinds, curtains etc. for the second version of models. An example of appearance changes can be seen in Figure 9. The corresponding models were aligned for purposes of evaluation. I.e. if a query was taken in one model of a certain location but localised in a second model of the same location, we mark it as correct up to distance and angular thresholds.

Figure 9: Visual difference between hospital 1 and 2 (left pair) or living lab 1 and 2 (right pair).

5 Results

The main evaluation metric is the ratio of correctly localised queries under a certain distance and certain angular error. We fix the angular threshold to 10°and we evaluate for different distance thresholds ranging from zero up to two meters. The results displayed in the floor plan can be seen in Figure 12 and the cumulative results over varying threshold are displayed in Figure 13.

As can be seen, we achieve 94% correct poses (see examples in Figure 10) within a very strict threshold of only $0.25m$. The achieved precision is very high and is higher than best performing methods in most standard benchmarks [11]. The error is directly derived from imperfections of 3D model and depth map estimates. From our experience in similar facilities, the imperfections are often in range of units of cm and it is extremely difficult to localise with better precision than $10cm$ as even evaluation results are uncertain in this domain because of the possible reference "ground truth" pose errors.

Even though the achieved precision is very high as it can compete with or directly outperform state-of-the-art localisation results, it is difficult to predict without real world experiments if this precision will be sufficient to achieve all goals set in the project. We have plans for multiple improvements to increase the precision if needed.

From the inspection of the failure cases we see that it is (Figure 11) most often due to lack of visual information, visual similarity, or repetitive patterns in the field of view. These issues are notoriously difficult and failures are hard to avoid.

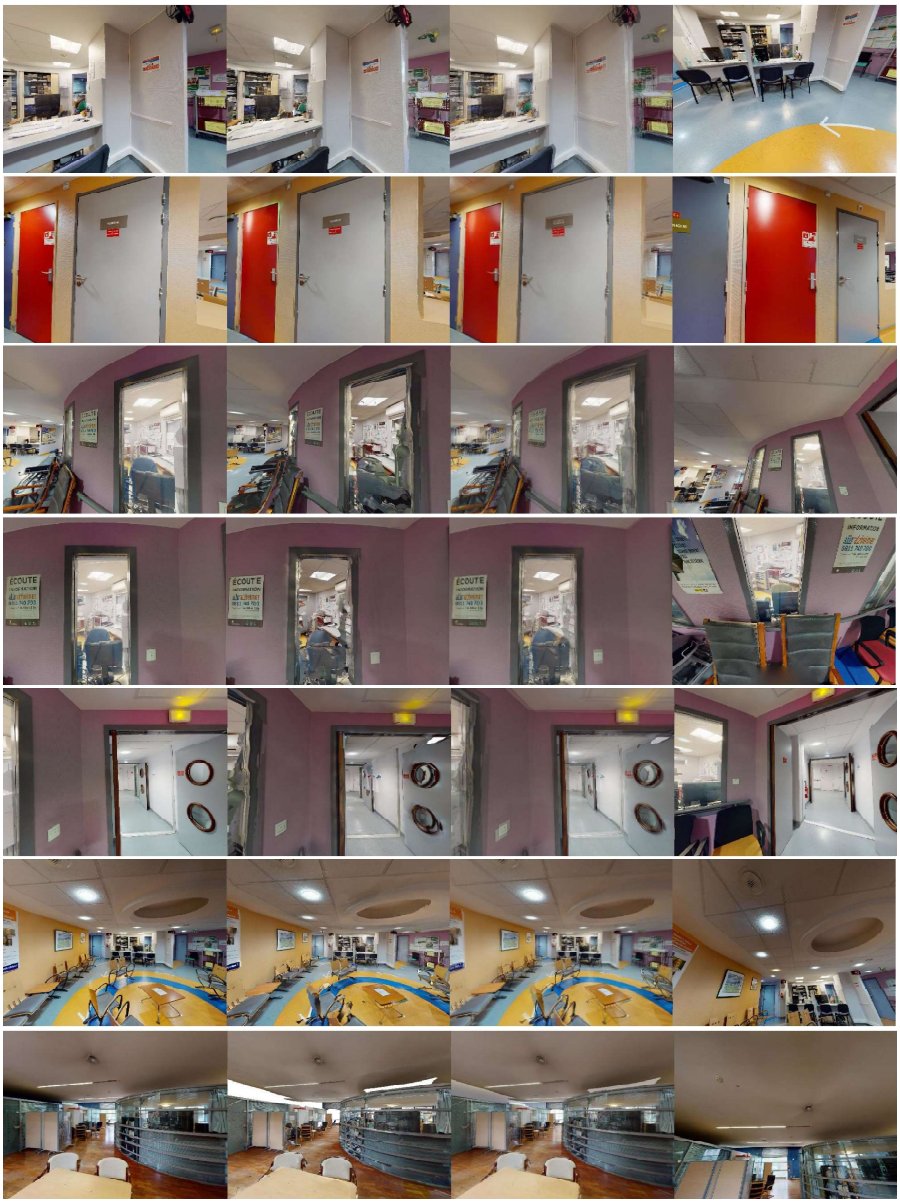

Figure 10: Columns from left to right: Well localised query images, projected mesh from estimated pose, blend of query and mesh projection, database image used for relative pose estimation.

Figure 11: Columns from left to right: The failure queries, projected mesh from estimated pose, blend of query and mesh projection, database image used for relative pose estimation.

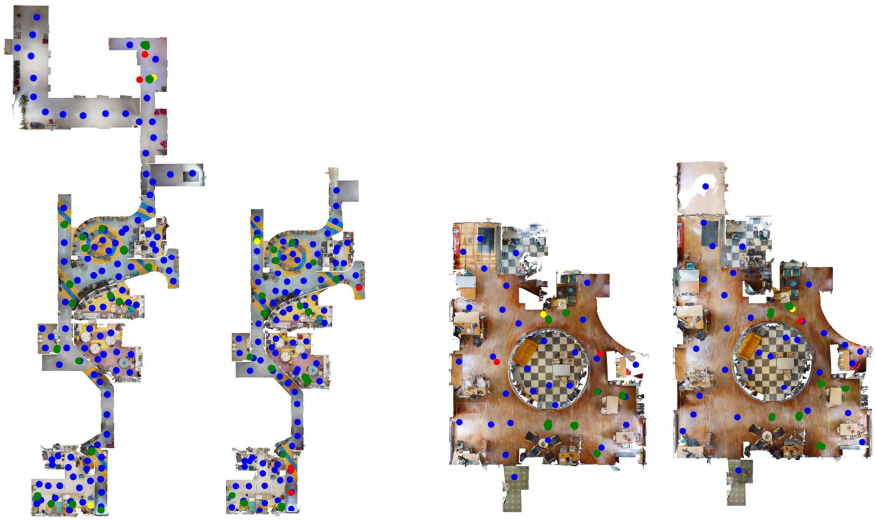

Figure 12: Results of localisation. **Green** are displayed queries with translation error lower than $0.25m$ and angular error below $10°$. **Yellow** are displayed queries with translation error between $0.25m$ and $1m$ and **red** are shown ones with error $> 1m$. The database panoramas are displayed in **blue**.

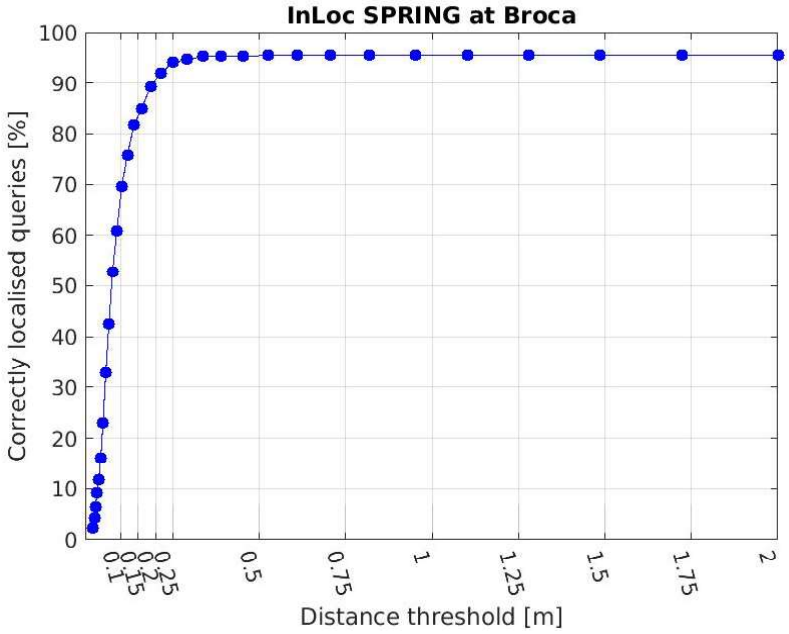

Figure 13: Final results of Broca dataset evaluation. Results are displayed as percentage of correctly localised queries within certain distance threshold in meters and fixed 10°angular threshold.

6 Odometry Integration

The achieved success rate on its own is reasonably high and we would like to achieve similar results in the final results of the project. It is important to realize that even though the environment is realistic, this model and query imagery are somewhat artificial. We aim to develop methods suitable for populated spaces and that brings many challenging problems. We can expect various forms of occlusion into query images, caused by people and their belongings naturally occurring in the environment. We can also expect temporal changes of the environment itself as furniture may be moved around for various reasons and therefore the map will not mirror query images as reliably as it is in our initial experiment.

To compensate for these expected issues, we plan to integrate odometry information available on-board of ARI. We will mirror human behaviour in case the field of view is confusing. Natural thing is to turn around or slightly move to get additional information. We will collect a series of query images with known relative position and localise based on this batch query. We believe that the additional reliable constraint in the form of known relative position between

query images based on ARI's odometry will help us compensate for the new challenges introduced in less artificial setup.

Even though we have plans to implement and explore these modifications, we were mostly not able to do so, as we don't have access to the robot yet. We believe that the simulation test we performed reflect reality well enough, and we are confident that we will be able to achieve comparable results doing real world experiments. These will be conducted as soon as we have access to the robot to support this claim. Results with possible corrections will be published as part of the followup deliverable D2.4 latest.

7 Future Work

We have many ideas how to further enhance quality of localisation. We list here a few main lines which we want to follow.

7.1 Image Undistortion

Evaluation in real world will bring a new challenge, which was not present in our experiment. The image acquisition method – most commonly rolling shutter and actual lenses of the camera may cause various distortions of the image. This can be neglected for most standard cameras as the main and often sufficient parameter – focal length is generally available in EXIF image metadata. For cameras with severe distortion it might not be enough. That can be compensated by fist camera model selection [12] followed by accurately modeling the camera parameters and image undistortion which is then passed to the localisation module.

7.2 Image Retrieval

Even though we did not observe any failure cases due to image retrieval yet, we expect this to occur in experiments where we introduce various sources of occlusion (people, temporary object in the scene etc.). Therefore we consider to examine possibilities of applying self-attention transformer networks which were recently introduced in the image domain[13].

7.3 Feature Matching, Geometrical Verification and Pose Estimation

We also look in the possibility to further enhance quality by finding more robust tentative feature matches [14] and its verification by more complex constrains than just two homographies. This step is crucial as reliable matching with well estimated depth is paramount to successful localisation. We also explore possibility of direct pose estimation from images. The neural rendering techniques

NeRF[15] use a completely different methodology and its preliminary results look very promising.

8 Software

We present four software packages:

1. **Localisation Module** – described localisation software derived from work of H. Taira et al. **InLoc**[1]
2. **Map Construction Module** – a package of tools to convert 3D model with panoramas into an actual localisation map
3. **Communication Module** – a communication tool to send a query image to the localisation server and receive a response from the ROS[16] environment.
4. **Visitor Module** – a dataset exploration tool using AIHabitat [17] which allows simple tour in any of four datasets.

We provide also directly the localisation map which is quite large (500GB), it is publicly available at:
https://data.ciirc.cvut.cz/public/projects/2020SPRING/SPRING_Broca_dataset

8.1 Localisation Module

The localisation pipeline is a combination of MATLAB and Python3 implementation. While the main script is written in MATLAB, specific functions such as mesh model handling is done in Python. We provide a simple script to reproduce our results. It is located in

```
./SPRING/SPRING_Demo/demo_SPRING_dataset.m
```

folder of the localisation module. The module is publicly available at:
https://data.ciirc.cvut.cz/public/projects/2020SPRING/
SPRING_localisation_module.tar.gz
This is still a research version of our code as the development continues.

8.2 Map Construction Module

The map building tools can be found as part of localisation module on path:

```
./SPRING/Build_SPRING/*
```

The tools allow to construct a localisation database based on panoramas accompanied by mesh model and panorama poses w.r.t. the model.

The tools are again, a combination of MATLAB and Python3. The Python part is responsible for image and model handling, generating cutouts and depth maps. The MATLAB part precomputes netVLAD embeddings for database images and sets up links and environmental variables for the localisation module.

8.3 Communication Module

As the Localisation module is quite large and requires a lot of computation, we assume its usage off-board the robot, preferably on a nearby workstation. We provide a simple communication tool to send queries from ROS to a workstation. It follows simple client-server architecture and is available at:
https://data.ciirc.cvut.cz/public/projects/2020SPRING/
SPRING_communication_module.tar.gz
The package contains both client and server side. A ROS client sends a HTTPS post with image data. The server handles the incoming HTTPS request, reads out the image and forwards the query to MATLAB. After query is evaluated, the server side returns response to the request and the ROS client receives the 6DOF pose as .json

8.4 Visitor Module

The visitor module builds on top of AI Habitat [17] and it offers a simple way to virtually walk through the individual models. The module is available here:
https://data.ciirc.cvut.cz/public/projects/2020SPRING/
SPRING_visitor_module.tar.gz

9 Conclusion

We have presented a fully operational localisation module. The module estimates pose of an image capture by assessing visual similarity by image retrieval followed by geometric verification and camera pose estimation and finally ranked by visual similarity of estimated camera pose with the query image. We presented a newly created Broca dataset – a realistic indoor environment and have shown that our module works and is able to localise over 94% of query images within $0.25m$ of the reference camera pose. We could not include robot's odometry but we provided plan to integrate it as we believe it will be an important tool to further enhance the results when deployed to cluttered and populated spaces.

We attach additional visual results in from of Appendix to provide deeper insight of the results achieved.

References

[1] 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.

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[4] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[5] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 883–890, 2013.

[6] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[7] Karel Lebeda, Jirı Matas, and Ondrej Chum. Fixing the locally optimized ransac–full experimental evaluation. In *British machine vision conference*, volume 2. Citeseer, 2012.

[8] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918. IEEE, 2012.

[9] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T Freeman. Sift flow: Dense correspondence across different scenes. In *European conference on computer vision*, pages 28–42. Springer, 2008.

[10] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1469–1472, 2010.

[11] Carl Toft, Will Maddern, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, et al. Long-term visual localization revisited. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[12] Michal Polic, Stanislav Steidl, Cenek Albl, Zuzana Kukelova, and Tomas Pajdla. Uncertainty based camera model selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5991–6000, 2020.

[13] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[14] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[16] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[17] 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.

A Appendix – Additional Results

We provide additional visual results over 10 randomly selected queries.

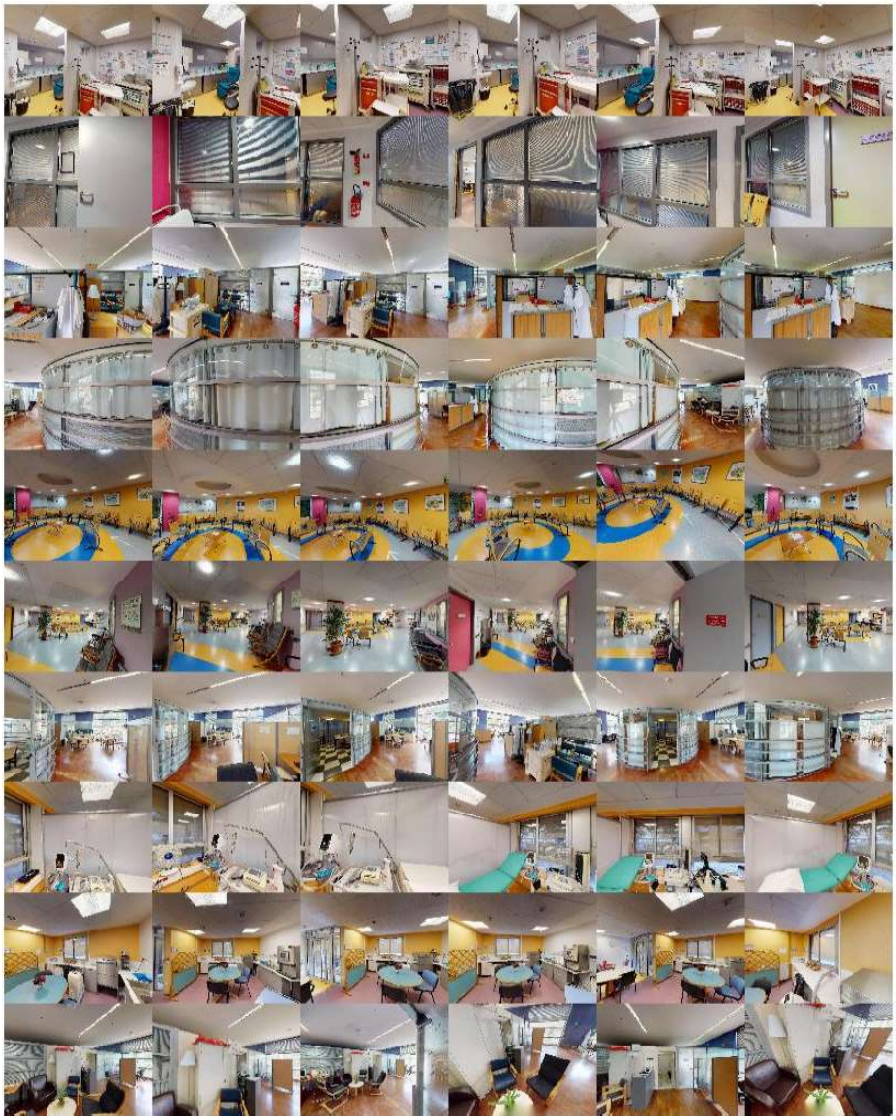

Figure 14: **Image retrieval** step results for 10 randomly sampled queries. Leftmost image is the query image followed by the top five retrieved images, sorted from left to right. $Ids = [158, 137, 275, 284, 67, 174, 360, 229, 250, 265]$

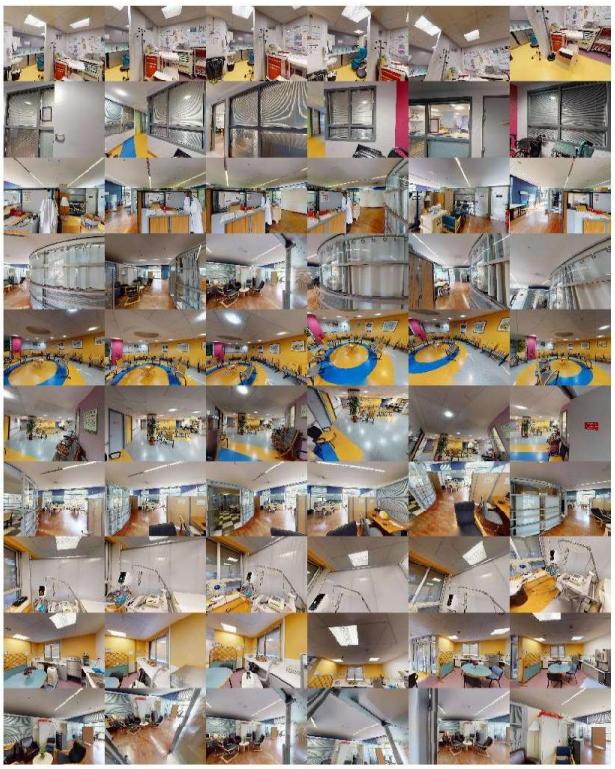

Figure 15: Re-ranking **after geometrical verification** step results for 10 randomly sampled queries. Leftmost image is the query image followed by the top five re-ranked images, sorted from left to right. $Ids = [158, 137, 275, 284, 67, 174, 360, 229, 250, 265]$

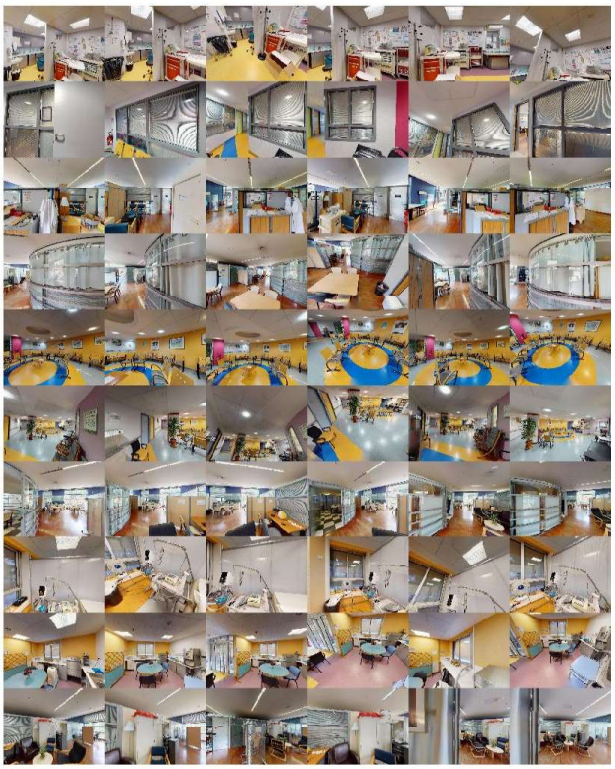

Figure 16: Final re-ranking **after pose verification** step for 10 randomly sampled queries. Leftmost image is the query image followed by the top five re-ranked images, sorted from left to right. $Ids = [158, 137, 275, 284, 67, 174, 360, 229, 250, 265]$

A.1 Query Id 67

Figure 17: **Query Id: 67** Image retrieval ranking

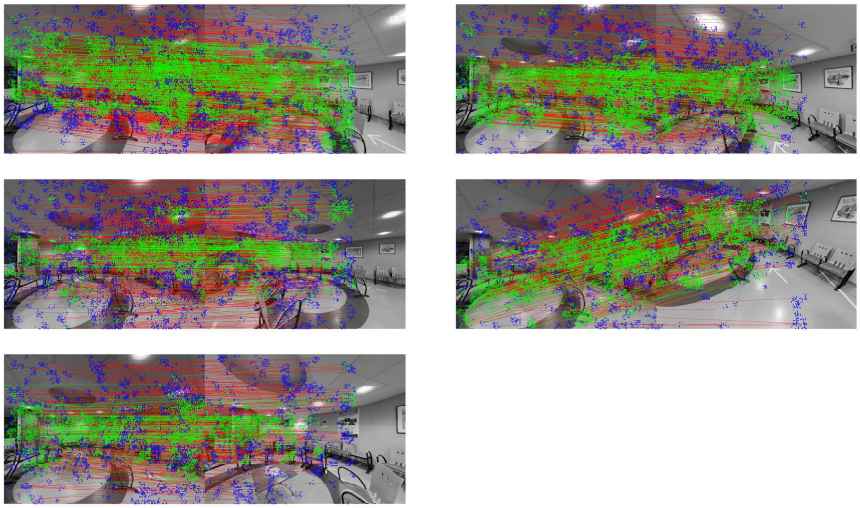

Figure 18: **Query Id: 67** Dense matching

Figure 19: **Query Id: 67** Geometrical Verification re-ranking

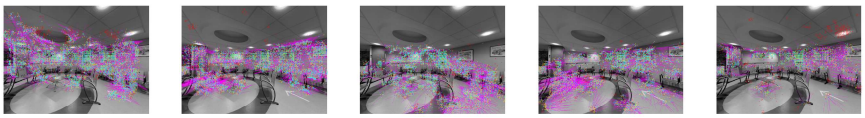

Figure 20: **Query Id: 67** Reprojection errors after pose estimation

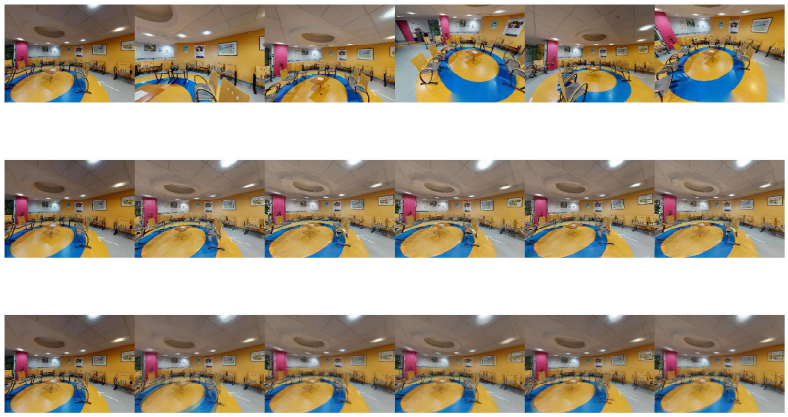

Figure 21: **Query Id: 67** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.2 Query Id 137

Figure 22: **Query Id: 137** Image retrieval ranking

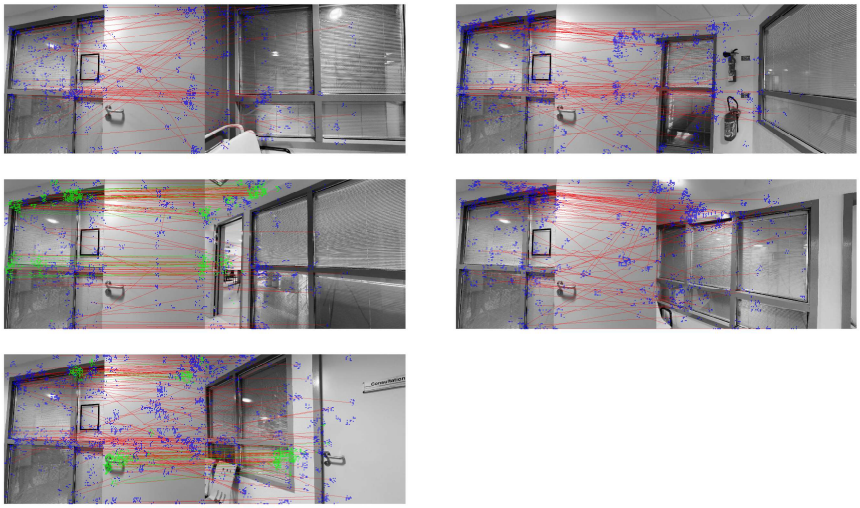

Figure 23: **Query Id: 137** Dense matching

Figure 24: **Query Id: 137** Geometrical Verification re-ranking

Figure 25: **Query Id: 137** Reprojection errors after pose estimation

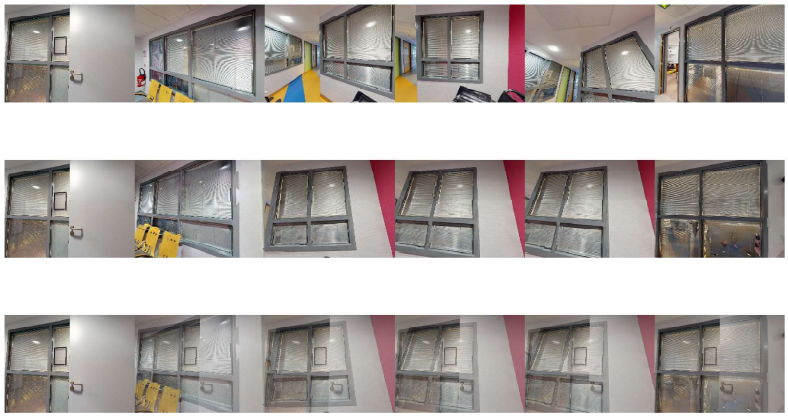

Figure 26: **Query Id: 137** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.3 Query Id 158

Figure 27: **Query Id: 158** Image retrieval ranking

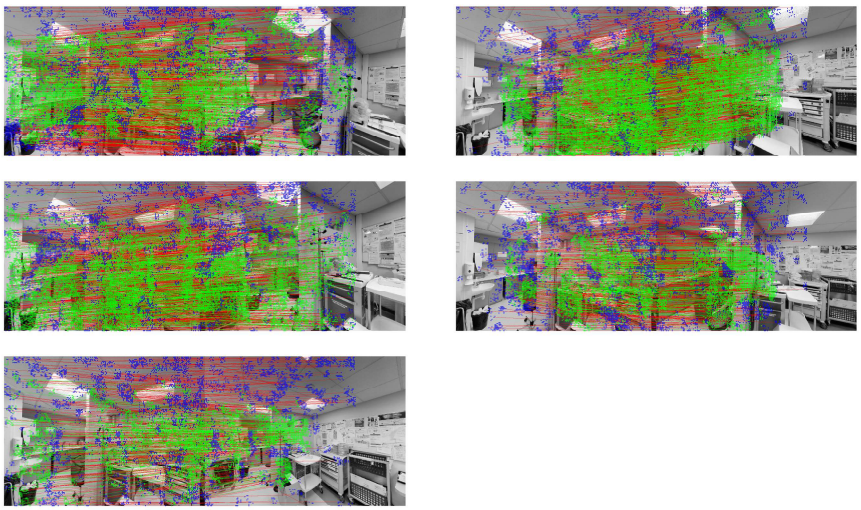

Figure 28: **Query Id: 158** Dense matching

Figure 29: **Query Id: 158** Geometrical Verification re-ranking

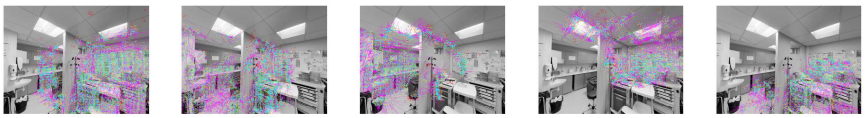

Figure 30: **Query Id: 158** Reprojection errors after pose estimation

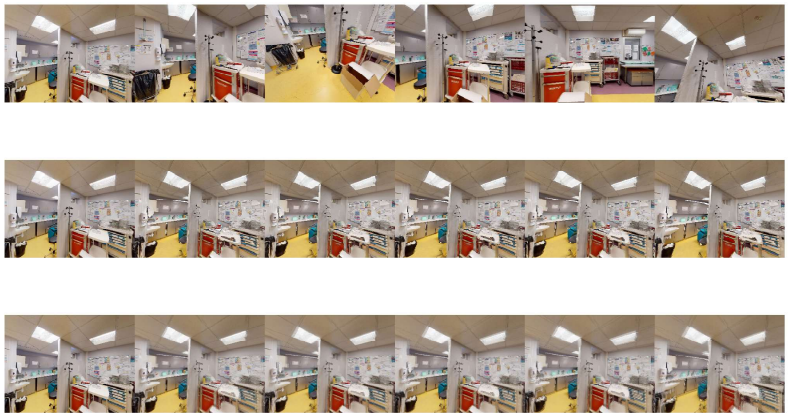

Figure 31: **Query Id: 158** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.4 Query Id 229

Figure 32: **Query Id: 229** Image retrieval ranking

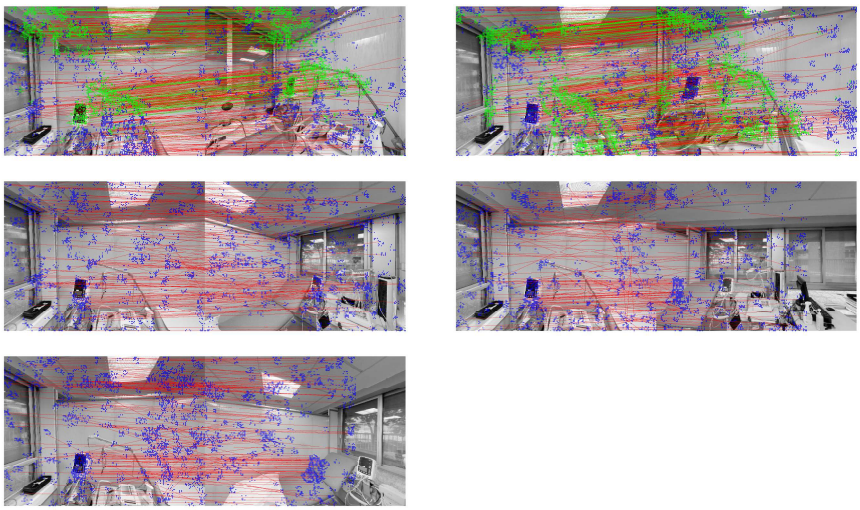

Figure 33: **Query Id: 229** Dense matching

Figure 34: **Query Id: 229** Geometrical Verification re-ranking

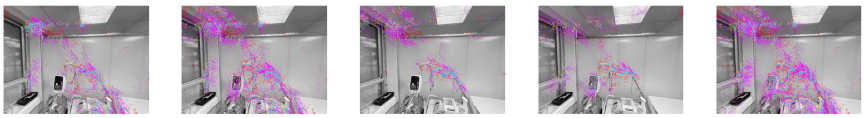

Figure 35: **Query Id: 229** Reprojection errors after pose estimation

Figure 36: **Query Id: 229** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.5 Query Id 250

Figure 37: **Query Id: 250** Image retrieval ranking

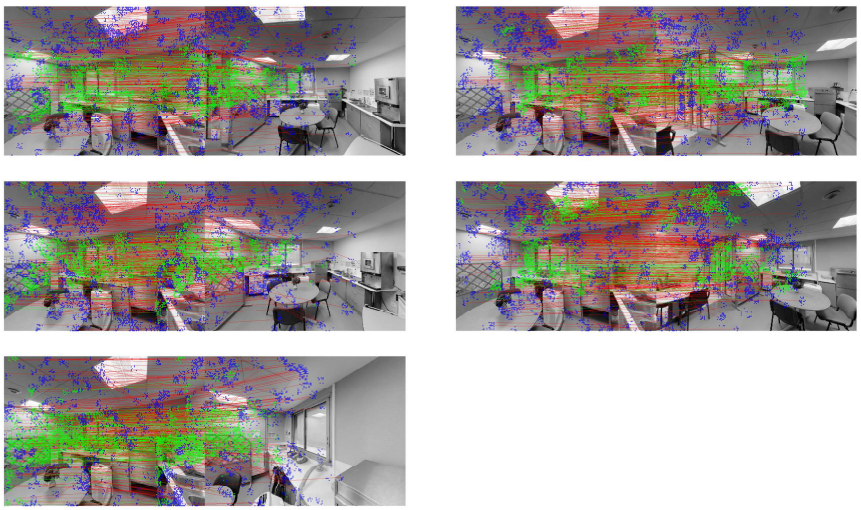

Figure 38: **Query Id: 250** Dense matching

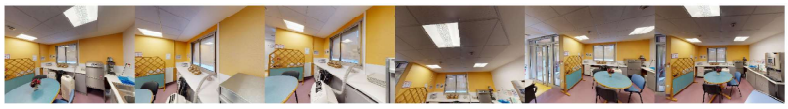

Figure 39: **Query Id: 250** Geometrical Verification re-ranking

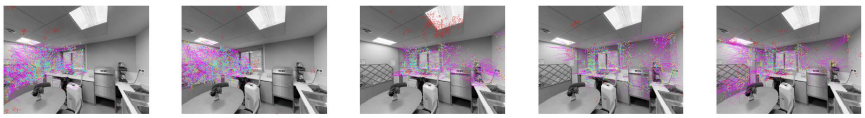

Figure 40: **Query Id: 250** Reprojection errors after pose estimation

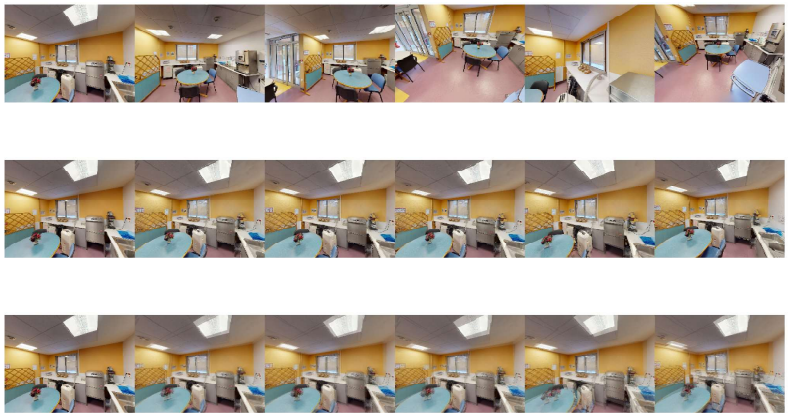

Figure 41: **Query Id: 250** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.6 Query Id 265

Figure 42: **Query Id: 265** Image retrieval ranking

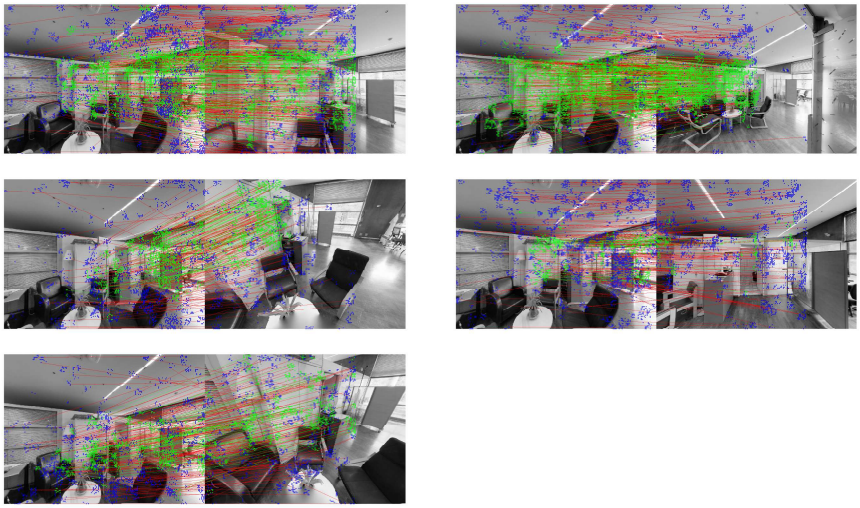

Figure 43: **Query Id: 265** Dense matching

Figure 44: **Query Id: 265** Geometrical Verification re-ranking

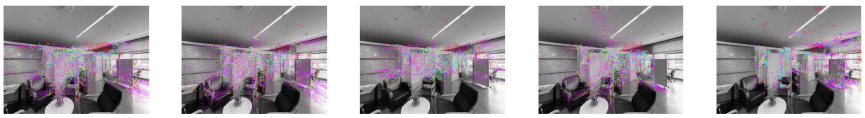

Figure 45: **Query Id: 265** Reprojection errors after pose estimation

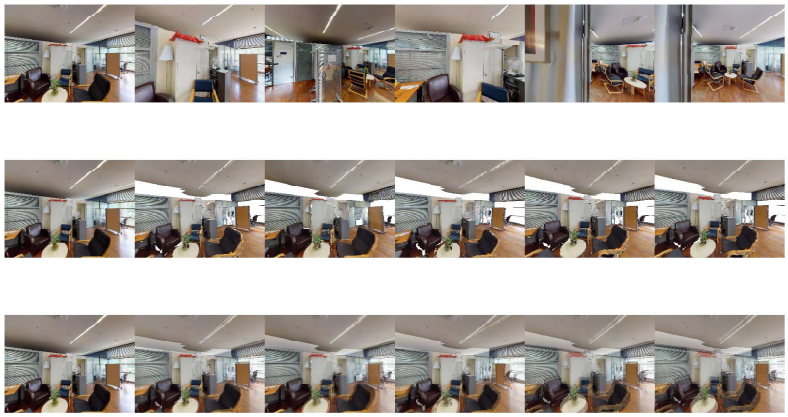

Figure 46: **Query Id: 265** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.7 Query Id 275

Figure 47: **Query Id: 275** Image retrieval ranking

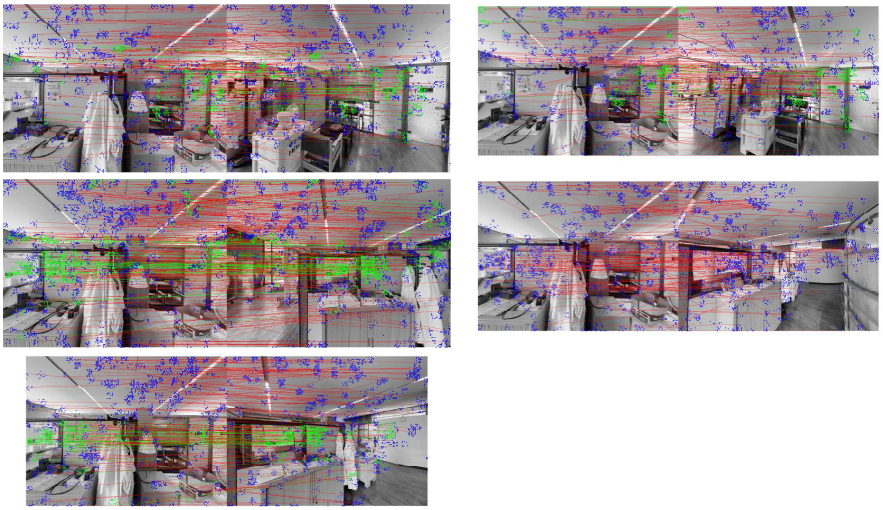

Figure 48: **Query Id: 275** Dense matching

Figure 49: **Query Id: 275** Geometrical Verification re-ranking

Figure 50: **Query Id: 275** Reprojection errors after pose estimation

Figure 51: **Query Id: 275** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.8 Query Id 284

Figure 52: **Query Id: 284** Image retrieval ranking

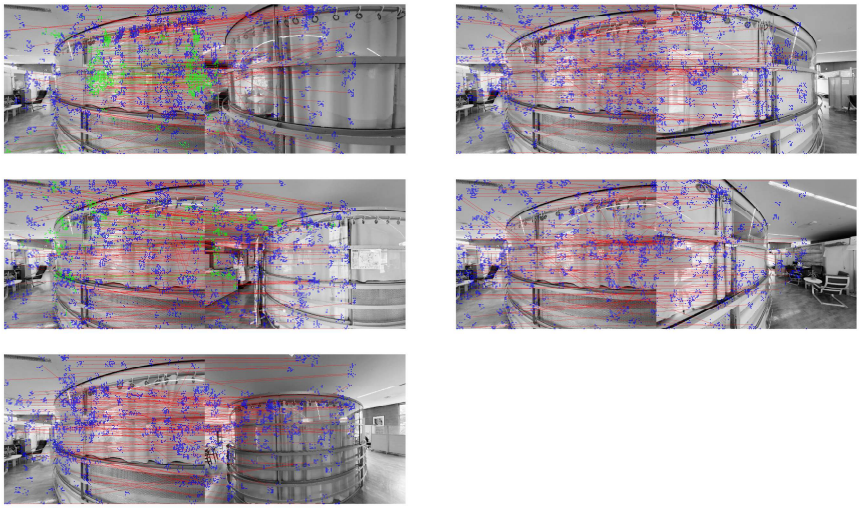

Figure 53: **Query Id: 284** Dense matching

Figure 54: **Query Id: 284** Geometrical Verification re-ranking

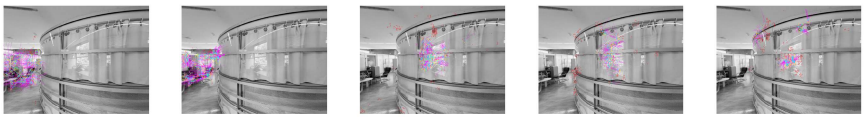

Figure 55: **Query Id: 284** Reprojection errors after pose estimation

Figure 56: **Query Id: 284** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)

A.9 Query Id 360

Figure 57: **Query Id: 360** Image retrieval ranking

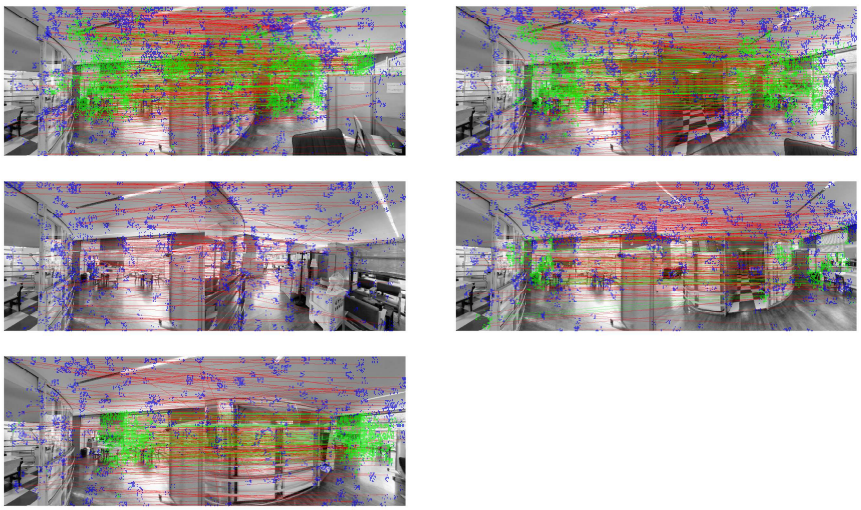

Figure 58: **Query Id: 360** Dense matching

Figure 59: **Query Id: 360** Geometrical Verification re-ranking

Figure 60: **Query Id: 360** Reprojection errors after pose estimation

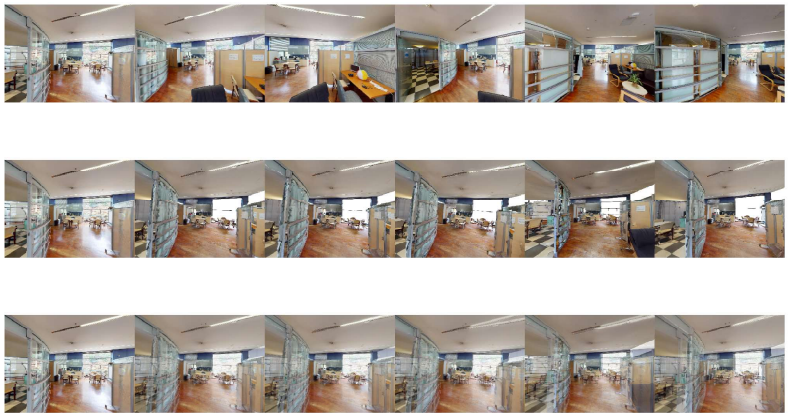

Figure 61: **Query Id: 360** Pose verification re-ranking (top), synthetic view (center), blend with query(bottom)