CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS

Tinatin Verdzeuli

InLoc Visual Localization for ARI robot

Department of Cybernetics

Thesis supervisor: **doc. Ing. Tomáš Pajdla, Ph.D.**

Applied Algebra and Geometry Group, CIIRC CTU, Prague, August 2021

http://aag.ciirc.cvut.cz/

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Verdzeuli Tinatin**	Personal ID number:	**472711**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Electrical Engineering and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

InLoc Visual Localization for ARI robot

Bachelor’s thesis title in Czech:

InLoc vizuální lokalizace pro robota ARI

Guidelines:

1) Study InLoc visual localization and its modifications[1, 2] as well as ARI software environment.
2) Implement access to existing InLoc functionality from ARI environment.
4) Demonstrate and evaluate InLoc localization on data provided by ARI robot.

Bibliography / sources:

[1] Arandjelović, R.; Gronat, P.; et al. NetVLAD: CNN architecture for weakly supervised place recognition. In IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[2] Taira, H.; Okutomi, M.; et al. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2018, ISSN 1063-6919, pp. 7199–7209, doi:10.1109/CVPR.2018.00752.
[3] 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 CVPR 2019.
[4] P. Lucivnak. Visual Localization with HoloLens. MSc Thesis, CIIRC CTU in Prague 2020.

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Tomáš Pajdla, Ph.D., Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **25.02.2021** Deadline for bachelor thesis submission: ___________

Assignment valid until: **30.09.2022**

doc. Ing. Tomáš Pajdla, Ph.D.	prof. 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

.

Date of assignment receipt   Student’s signature

Acknowledgements

Thank you to my supervisor, doc.Ing. Tomáš Pajdla, for providing guidance and feedback throughout this project. Special thanks to Ing. Stanislav Steidl, for his advise and guidance. Also, I want to express my gratitude to Anna Zderadickova and Michal Polic who took the time to clarify my questions. Finally, I Thank my parents who supported and encouraged me all the way.

Declaration

Abstract

This paper is about implementing access to existing InLoc functionality from ARI environment. For this we studied ROS, InLoc visual localization and its modifications as well as ARI software environment, and create a ROS python package for InLoc, integrating it with robot operating system and making InLoc available to be used on robots such as ARI. Lastly, we demonstrate access to InLoc from ARI ROS environment.

Contents

Chapter 1

Introduction

Contents

1.1 Social Robots And Indoor Localization Problem

Social robots have already been introduced into public spaces, such as museums, airports, commercial malls and hospitals. Developing robust socially pertinent robots in healthcare [1] as well as other industries can have a tremendous social impact and economic value.

In order to properly perform their roles in public, robots must move, see, hear and communicate with several actors, in complex, unstructured and populated spaces. However, because of its limitations, today's Human-Robot Interaction technology is not well-suited for complicated tasks. This leads to not being well accepted by a large percentage of users. Thus, there are limitations that must be overcome, such as creating a sensor and knowledge based robust robot perception in complex, unstructured and populated environments. Which brings us to the main focus of this work - indoor localization based on camera data and existing maps for ARI robot.

This work is an integration of already existing large-scale indoor visual localization approach [2] and a humanoid social robot - ARI. This robot combines Service Robotics and Artificial Intelligence in one single platform, it is able to perform a wide range of multimodal expressive gestures and behaviors, and is suitable for research in Human-Robot-Interaction, perception, cognition, navigation, and localization. InLoc is capable of determining location, with high accuracy based on camera images. InLoc approach to solving the localization problem is to build a 3D map of the building and then use a camera to estimate the current position and orientation of the robot. However, InLoc is not yet compatible with ARI thus we created a wrapper software that integrates InLoc with ROS and runs on ARI or any other robot.

1.2 Objectives

1. InLoc visual localization must be reviewed [2, 3, 4, 5, 6]. As well as ARI manual [7] and ARI ROS Software [8, 9].
2. A new ROS package must be created, with a node that communicates to a server (sends images and receives pose estimation).
3. InLoc should run on the server, with a new dataset, by receiving the query images send by ARI ROS node.
4. The performance of the newly implemented node shall be tested.

1.3 This Work Is Organized The Following Way

Chapter 2 contains background on the topic of indoor visual localization, as well as theory about software and robot which this paper relies on.

Chapter 3 describes an implementation of the new ROS package, which gives ARI access to InLoc functionality.

Chapter 4 demonstrates testing of the implemented code, using data provided by ARI robot.

Chapter 5 is a summary of this work, whether or not it has fulfilled the assignment and possible future work.

Chapter 2

Theory

Contents

This chapter provides a theoretical and technical background on topics we are dealing within this thesis.

2.1 ARI Robot

ARI is is PAL Robotics' humanoid social robot [7] [Figure 2.1], which can perform a wide range of multimodal expressive gestures and behaviors, and is suitable for research in Human-Robot-Interaction, Speech recognition, perception, cognition, navigation, localization and SLAM.

The following is a list ARI´s main dimensions:

- Height - 165 cm

- Width - 53 cm
- Depth - 75 cm
- DoF Head - 2
- DoF Arms - 4 (x2) Optional
- DoF Hands - 1 (x2) Optional
- DoF Mobile Base - 2

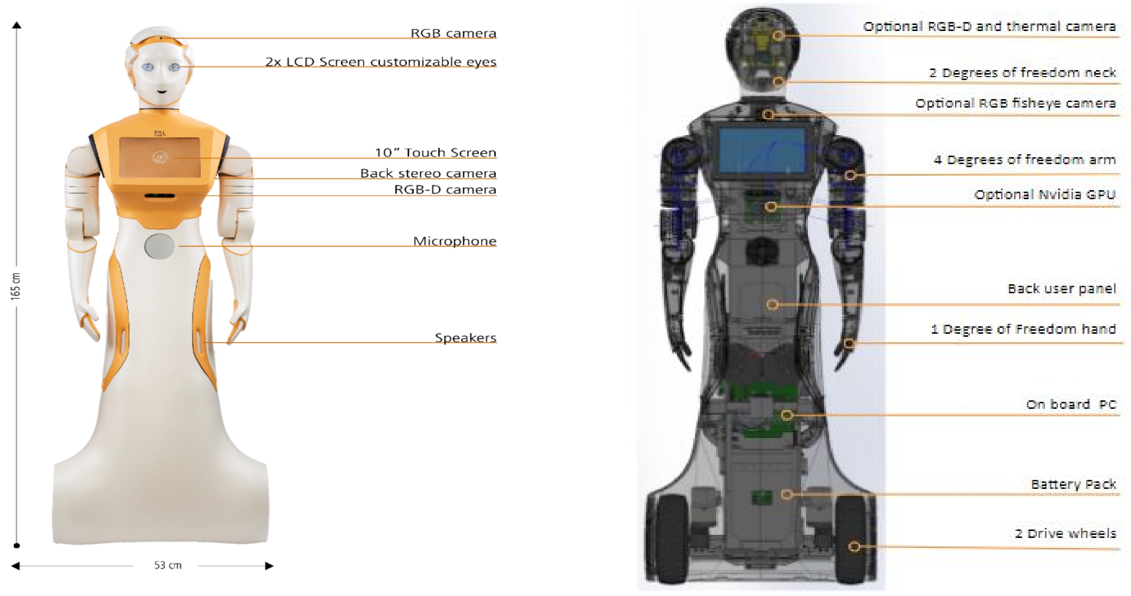


Figure 2.1: ARI's components: Humanoid Torso, 2 DoF head, 16 RGB LEDs per ear, Eyes LCD screens with custom animations, 40 RGB LED ring on the back, Touchscreen 10.1" 1200x800 Projected Capacitive, 802.11 a/b/g/n/ac/ad 5 GHz and 2.4 GHz, 802.11 a/b/g/n/ac/ad 5 GHz and 2.4 GHz, Ethernet 1000 Base, 4 x High Performance Digital Microphones array, Optional head camera: Head Camera Sony 8MegaPixel (RGB), Head Intel Realsense D435i (RGB-D), Torso Camera Intel Realsense D435i (RGB-D), Torso Back Intel Realsense D265 (stereo-fisheye), 2x HiFi Full-range Speakers, Thermal camera (optional).

2.2 Sensors

As shown in [Figure 2.1 and Figure 2.3], there are many sensors mounted on the robot. However for this localization task we will need RGB-D head camera and Front Stereo-fisheye camera.

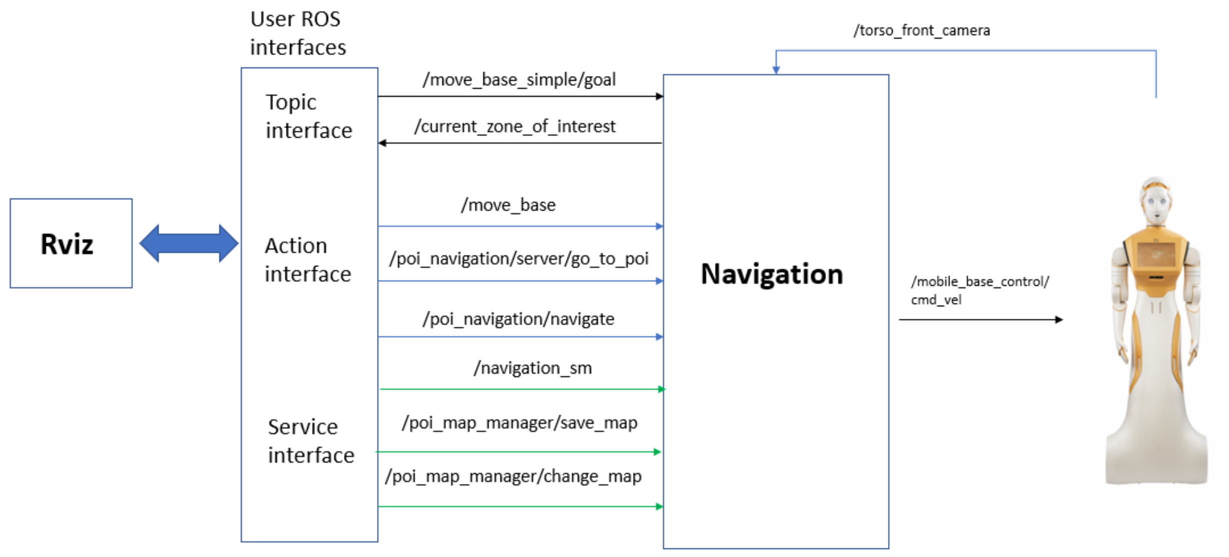


Figure 2.2: ARI ROS Navigation System

Stereo RGB-D camera (Intel Realsense D435i), as shown in [Figure 2.3], is mounted on the frontal side of the torso below the touch-screen, as well as inside ARI's head, and provides RGB images along with a depth image obtained by using an IR projector and an IR camera. The depth image is used to obtain a point cloud of the scene. It has an integrated IMU sensor unit mounted at the base to monitor inertial forces and provide the altitude.

Frontal and back stereo-fisheye cameras [Figure 2.3]: The frontal camera is positioned just above the touch-screen and the back camera, above the emergency button. They publish stereo images at 30 frames per second, provides stereo, fisheye, black and white images, and also publishes IMU data.

2.3 Existing Software

ARI Already has a basic navigation system [Figure 2.2], which uses Visual SLAM to perform mapping and localization using the RGB-D camera of the torso. This system works by detecting keypoints or features from the camera input [Figure3] and recognising previously seen locations in order to create a map and localize. The map obtained is represented as an Occupancy Grid Map (OGM) that can later be used to make the robot localize and navigate autonomously in the environment. the user can communicate with the navigation software using ROS topics, actions and services.

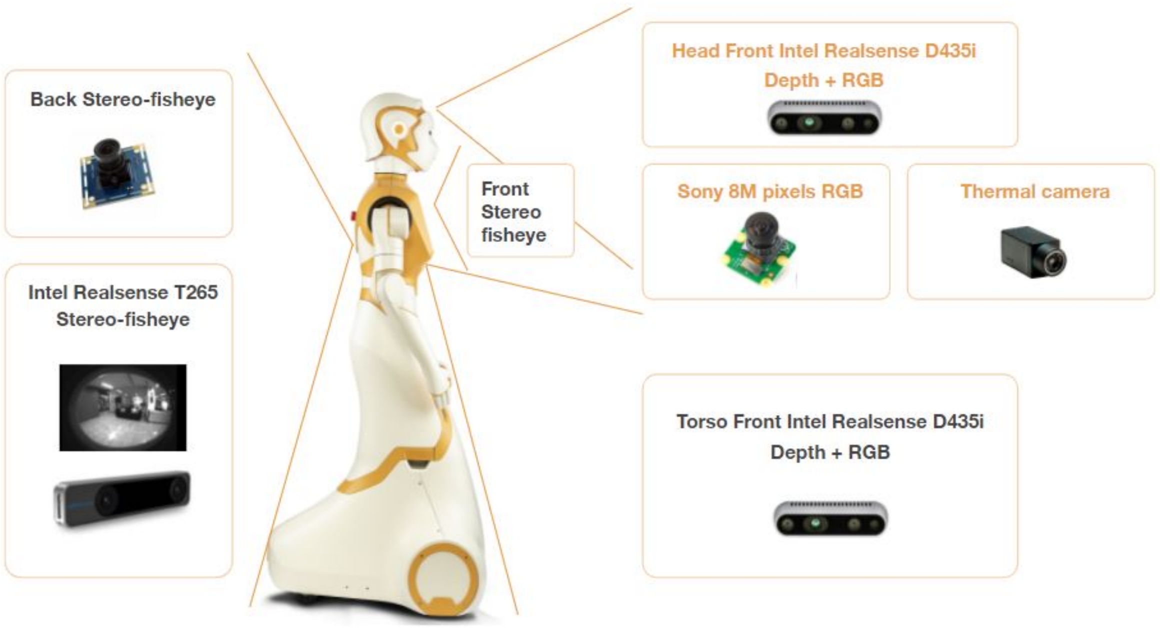

Back Stereo-fisheye
Intel Realsense T265 Stereo-fisheye
Front Stereo fisheye
Head Front Intel Realsense D435i Depth + RGB
Sony 8M pixels RGB
Thermal camera
Torso Front Intel Realsense D435i Depth + RGB

Figure 2.3: Cameras mounted on the robot

2.4 ROS

ROS - Robot Operating System is an open source collection of software frameworks for robot software development. It is a flexible framework for writing robot software, and is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

ROS is the standard robotics middleware used in ARI. The comprehensive list of ROS packages used in the robot are installed in different locations of the SSD and are classified into three categories:

- Packages belonging to the official ROS distribution melodic.
- Packages specifically developed by PAL Robotics, which are included in the company's own distribution, called ferrum.
- Packages developed by the customer.

2.5 RVIZ

RVIZ is a 3D visualization tool for ROS. Using RVIZ we can create a map of environment and localize the robot by just moving it around, while reading its sensor data [Figure 2.4]. Thus, enabling us to create a very basic localization and autonomous navigation system.

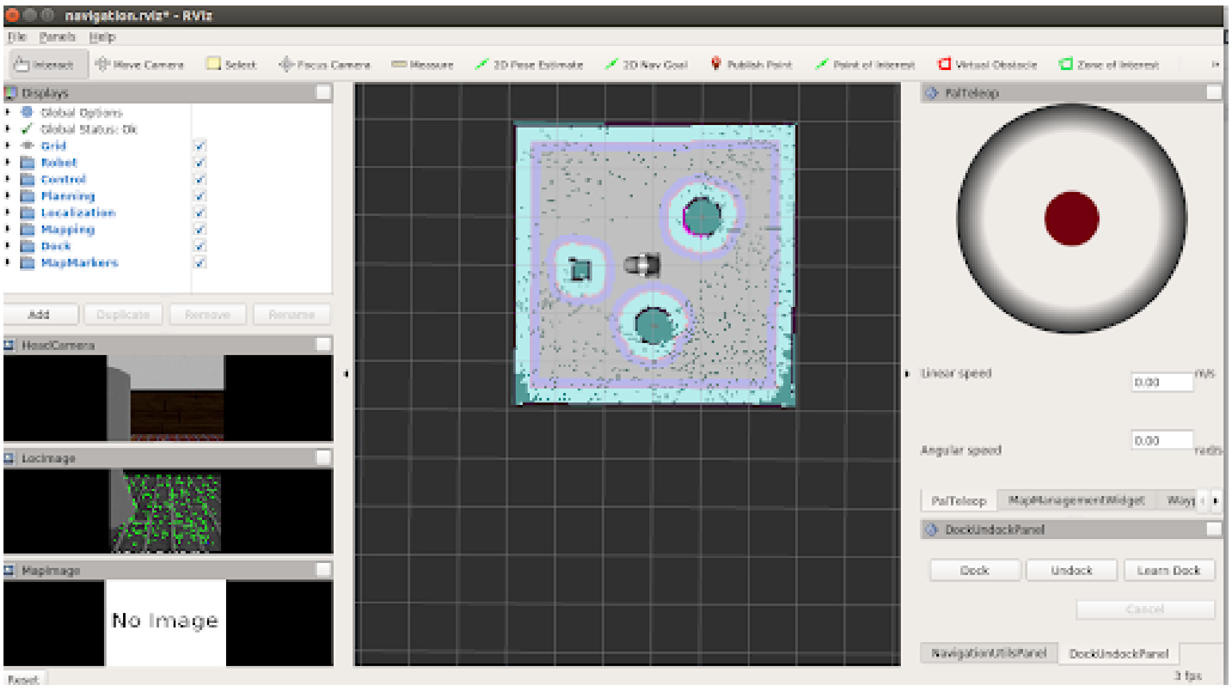


Figure 2.4: Robot navigation system in RVIZ

2.6 6DoF

Six degrees of freedom (6DoF) refers to the freedom of movement of a rigid body in three-dimensional space. Specifically, the body is free to change position as forward/backward, up/down, left/right translation in three perpendicular axes, combined with rotation about three perpendicular axes: yaw, pitch, and roll.

2.7 Indoor Localization Method

Autonomous navigation and Localization in any kind of environment is a necessity for robotic intelligent systems. Successful navigation requires both to localization and path planning. Indoor localization has received less attention compared to Large-scale localization in urban areas. Furthermore, it is a harder problem, due to that, in small distances, even small changes in viewpoint lead to large changes in image appearance. For the same reason, occluders also have a stronger impact. Buildings often have many repetitive elements (corridors, rooms, doors, chairs, etc. . .). Also, the appearance of indoor scenes highly dynamic and can change during a day (lighting, moving furniture, etc. . .).

InLoc - Indoor Visual Localization with Dense Matching and View Synthesis [10, 2], shows a improvement of correctly localized queries by 17–20 percent over other existing methods [2]. However, InLoc pose verification is robust up to a certain level of scene changes, e.g., illumination changes and some amount of misalignment, but cannot deal with extreme changes in the scene such as very large occlusions or when the view is dominated by moving objects.

InLoc is a Visual localization method targeted for indoor environments, that predicts the 6DoF (freedom of movement: forward/backward, up/down, left/right, yaw, pitch, and roll) pose of a query photograph with respect to a large indoor 3D map [Figure 2.5]. This method carefully introduces dense feature extraction and matching in a sequence of progressively stricter verification steps.

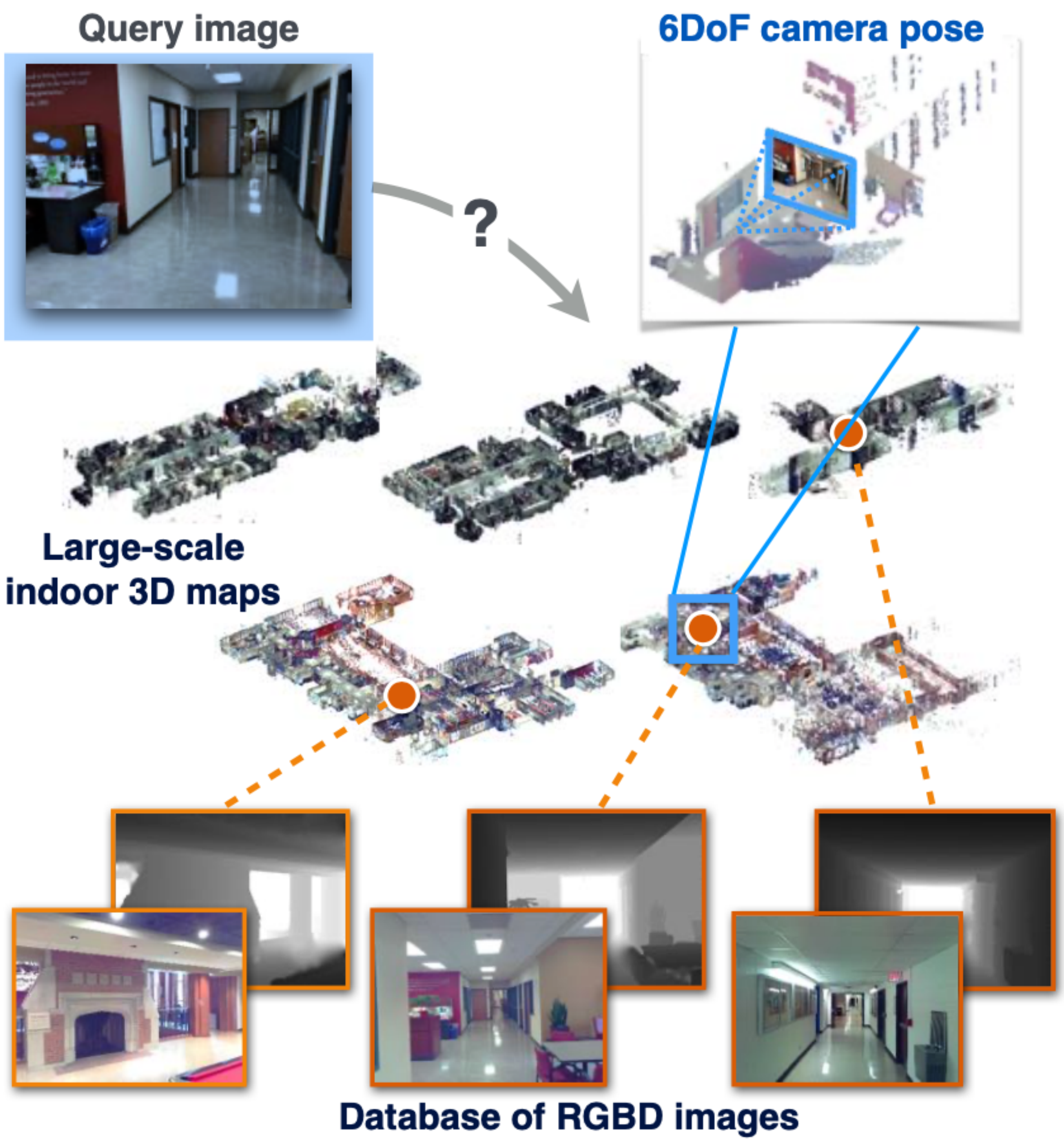


Figure 2.5: Large-scale indoor visual localization [2]

The method proceeds along three steps:

1. efficient retrieval of candidate poses that ensures scalability to large-scale environments.
2. pose estimation using dense matching rather than local features to deal with textureless indoor scenes.
3. pose verification by virtual view synthesis to cope with significant changes in viewpoint, scene layout, and occluders.

InLoc approach starts with an image retrieval step, using a compact image representation [3] that scales to large scenes. After retrieving potentially relevant database images, pose estimation is done in two steps: First, dense matching of CNN descriptors is used, it captures spatial configurations of higher-level structures. Second, step is based on virtual view synthesis that can accurately verify whether the query image depicts the same place by dense pixel-level matching, again not relying on sparse local features.

InLoc dataset is composed of a database of RGBD images geometrically registered to the floor maps augmented with a separate set of RGB query images taken by handheld devices. The provided query images are annotated with manually verified ground-truth 6DoF camera poses in the global coordinate system of the 3D map.

For each query reference pose generation is computed as follows:

1. Selection of the visually most similar database images.
2. Automatic matching of query images to selected database images.
3. Computing the query camera pose and visually verifying the reprojection.
4. Manual matching of difficult queries to selected database images.
5. Quantitative and visual inspection.

Indoor visual localisation with dense matching and view synthesis is a new method, which addresses three main challenges:

1. Lack of sparse local features: Indoor environments are full of large textureless areas such as walls, ceilings and floors. Thus, it's hard to detect features. This problem is overcome by using multi-scale dense CNN features for both image description and feature matching.

2. Large image changes: Indoor environments include movable objects, such as furniture and people. This can cause severe occlusions when viewed from a close distance. This problem is overcome by dense feature matching. Image descriptors are extracted from a convolutional neural network and match higher-level structures of the scene rather than relying on matching individual local features.

3. Self-similarity: Due to many symmetric and repetitive elements, Indoor environments are often very self-similar(corridors, tiles, chairs, doors, etc...). To overcome this problem, This approach incorporates both the positive and negative evidence by counting matching and non-matching pixels across the entire query image. It counts the negative evidence, i.e., what portion of the image does not match, as well as the positive evidence, i.e., what portion of the image does match, to decide whether two views are taken from the same location. This is achieved by explicit pose estimate verification based on view synthesis.

2.8 Simulation Testing

Training and testing embodied AI agents in the real world is slow, dangerous, expansive and difficult to replicate, thus we use simulations such as AI Habitat and Gazebo [Figure 2.6], which enables us to first develop a promising approach, and afterwords transfer it to a physical platform.

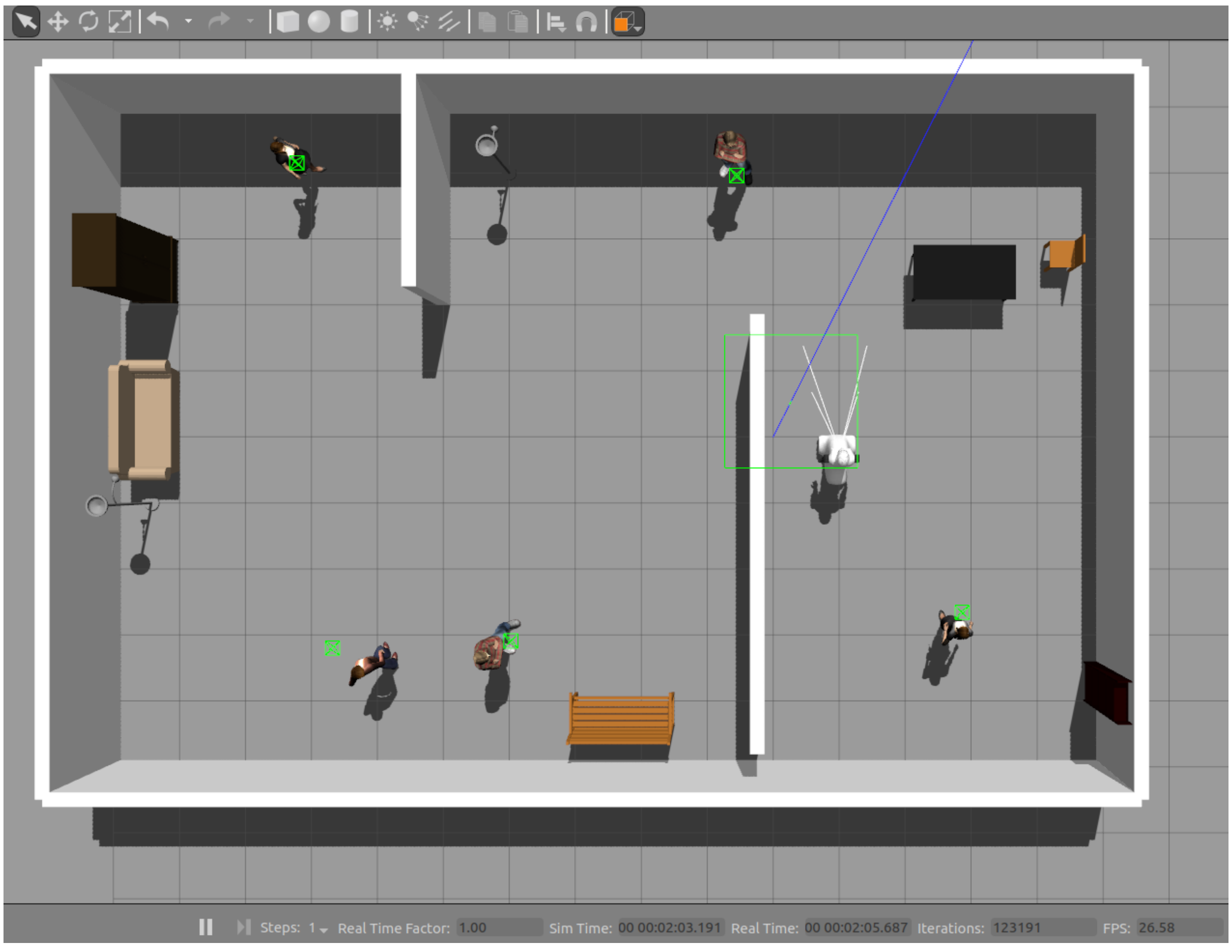
2.9 Gazebo

Gazebo is an open-source 3D robotics simulator, used in robot operating system (ROS), to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. ARI already has a default environment created in Gazebo to test robots basic navigation, we used it as a basic testing simulation, before switching to the main testing data using AI Habitat.

2.10 AI Habitat

AI Habitat - is a simulation platform for research in Embodied AI. AI Habitat enables training of robots in a highly photo realistic and efficient 3D simulator, before transferring the learned skills to reality. This gives AI agents more active perception, long-term planning, learning from interaction, and holding a dialog grounded in an environment.

Overall, Habitat consists of Habitat-Sim and Habitat-Lab. In addition we are using Habitat ROS for out robot simulation.



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:02:03.191 Real Time: 00 00:02:05.687 Iterations: 123191 FPS: 26.58

Figure 2.6: ARI Gazebo Simulated Environment

- Habitat-Sim is a high-performance physics-enabled 3D simulator with support for: 3D scans of indoor/outdoor spaces, CAD models of spaces and piecewise-rigid objects, Configurable sensors, Robots described via URDF, Rigid-body mechanics.
- Habitat-Lab is a modular high-level library for end-to-end development in embodied AI — defining embodied AI tasks, configuring embodied agents, training these agents, and benchmarking their performance on the defined tasks using standard metrics.
- Habitat-ROS connects AI Habitat simulator with ROS to take advantage of photo-realistic 3D environments and easily simulate tasks such as visual SLAM and navigation. This enables us to: navigate Habitat environment with ROS navigation packages, navigating Gazebo environment with Habitat trained agent, navigating Habitat environment with Habitat trained agent with ROS in the loop.

Chapter 3

Implementation

Contents

For implementation of InLoc approach on ARI robot, we decided to use ROS, and will be coding in Python, visualisation of a robot and testing it in a virtual environment was done using Gazebo and RVIZ. In this section we discuss the Python Package, that was implemented to give ARI access to the InLoc functionality on the server.

CD at the end of the paper contains the code described below.

3.1 ARI InLoc Wrapper Node

In order to integrate InLoc with ARI robot, we creates a new ROS package for ARI existing software. In the code this package is called "ros-server-communication". This package "communication.py"node, that robot can call when it needs to localise using InLoc.

"communication.py" node has a wrapper class, when initiated it subscribes to sensor data published by ARI's other nodes: RGB-D head camera or Stereo-fisheye camera topics. After obtaining the image, if needed, "camera-callback" function can modify it. Lastly, "send-image" function sends image to a server, waiting for response.

3.2 ARI Server Communication

In order for server to connect and communicate with the ARI's "communication.py" node, we implemented "matlab-server-img.py". It uses Socket to connect two nodes on a network to communicate with each other. One node listens on a particular port at an IP, while the other node reaches out to the other to form a connection. The server "matlab-server-img.py" forms the listener socket while the client "communication.py" reaches out to the server using "image-send" function.

3.3 Running the code

1. Install Ros and ARI software [7, 8].
2. Create and build a catkin workspace:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

3. create a package in catkin workspace and build it:

```
catkin_create_pkg ros_server_communication rospy roscpp
cd ~/catkin_ws/
catkin_make
```

4. Copy the "communication.py" into "ros-server-communication/src" directory.
5. Start ARI simulation [8].

```
#example:
roslaunch ari_gazebo ari_gazebo.launch public_sim:=true
rosrun key_teleop key_teleop.py
```

or start ROS:

```
roscore
```

6. Start the Server node matlab-server-img.py:

```
 cd /"path-to-server-node"
 python matlab_server-img.py  --matlab_path "Path to the matlab" --
matlab_script_folder "Path to the folder containing the .m file"  --
matlab_function "Name of the Matlab function to run"
```

```
# example :
python matlab_server-img.py  --matlab_path "/usr/local/MATLAB/R2020b/
bin/matlab" --matlab_script_folder /home/user/InLoc  --matlab_function
InLoc_run
```

7. Start the ARI communication node:

```
rosrun ros-server-communication communication.py
```

or

```
cd /"path-to-python-file"
python communication.py
```

3.4 The Dataset For ARI Visual Localization

To run InLoc using new dataset, we must prepare features, image lists, and retrieval scores in ”inputs” directory:

- Image list

  query-imgnames-all.mat contains string cell array named query-imgnames-all, that consists of image names of queries.

  cutout-imgnames-all.mat contains string cell array named cutout-imgnames-all, that consists of paths of cutout images.

- Image retireval scores

  scores.mat contains array named score. Contains the similarity score between query in each row and database in each column.

- Features

  Dense features for queries and databases are in

  ```
  inputs/features/query/.../img-name.features.dense.mat
  # and
  inputs/features/database/.../img-name.features.dense.mat
  ```

Chapter 4

Implementation Testing

Contents

In order to show that communication node is correctly working - it is able to send image data and receive the localization results from the server, we conducted following tests.

4.1 Simulated

In existing simulated Gazebo environment, ARI would navigate while publishing sensor data, wrapper node was subscribed to fisheye camera, modified the image and sending it to the server node. On the server some other modifications where done and this image was returned back.

We tested our client-server algorithm in a ARI Gazebo simulation until sensor data was correctly retrieved, send and answer was received [Figure 4.1]. Thus, making sure nodes were properly communicating before we started any real environment experiments.

4.2 Real Environment

For creating an image database suitable for indoor InLoc evaluation, an experiment to obtain a set of perspective images from university laboratory was conducted. Robot navigated in the laboratory, while recording of the data was done using rosbag command-line tool. From rosbag file images were exported into a separate file. However, fisheye and RGB

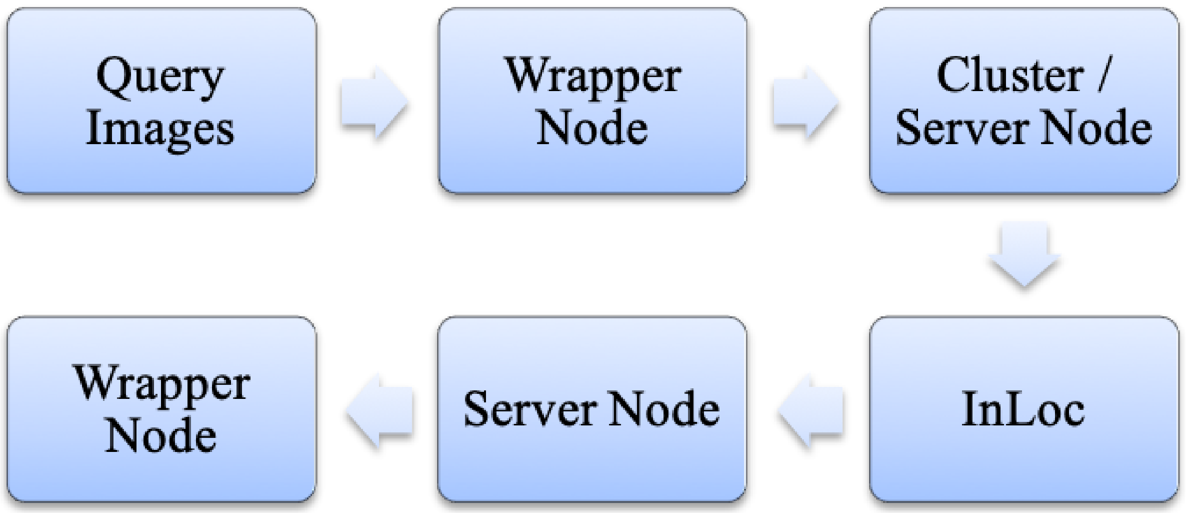


Figure 4.1: Node Communication Diagram

cameras on the robot malfunctioned before mentioned experiment were done, and working camera data was not suitable for testing InLoc. As of now, the mentioned cameras are not functional. Thus, we could not obtain new satisfactory dataset from experiments conducted in university laboratory. Instead, we tested InLoc on already existing dataset. But, this made it impossible to conduct real time experiments on ARI, which could have better demonstrated and evaluated the implementation results.

Chapter 5

Conclusion

Contents

5.1 Results

We learned about indoor localization methods, mainly InLoc software; ROS system and how it works; how to connect two nodes on a network to communicate with each other, such a Socket programming; Also, learned about ARI Robot software and hardware.

Server node can successfully communicate with ARI ROS node, recieve image data and run matlab scripts on server (such as InLoc-demo). Thus, As a result of this project, implementation of InLoc functionality to ARI robot was done successfully [Figure 5.1]. The implementation can also work on other robots with ROS system, by modifying names of sensor topics inside a wrapper node and creating a new InLoc dataset.

We also planned to test InLoc localization on ARI in real time, however, due to time constraints and some technical problems we were unable to demonstrate InLoc on ARI in real time. The main sensors for this localization method, that could provide query data were front Stereo fisheye and RGB head cameras, and both were not functional at the time of experiments.

5.2 Summary

The InLoc Visual Localisation for ARI robot has the following steps. (1) Given a query image taken by ARI Stereo-fisheye camera, the ROS "communication" node transforms the image and sends it to the server, waiting for response. (2) On the Server, InLoc obtains a set of candidate images by finding the N best matching images from the reference image database registered to the map. (3) For these N retrieved candidate images, it computes the query poses using the associated 3D information that is stored together with the database images. (4) It re-ranks the computed camera poses based on verification by view synthesis. (5) "wrapper" node receives highest ranked camera pose and compares it to the Map of current environment.

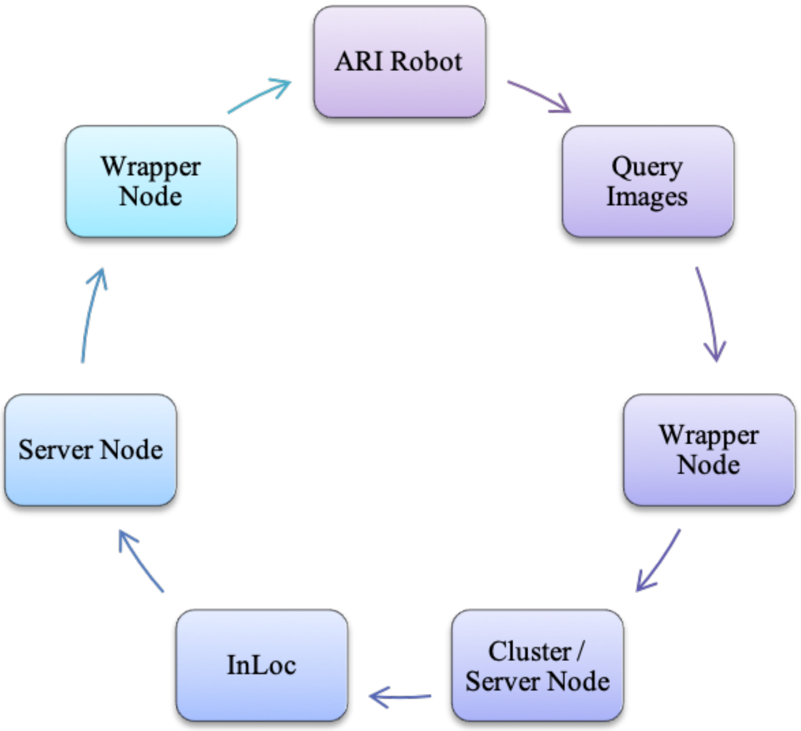


Figure 5.1: InLoc Visual Localization for ARI robot Diagram

5.3 Future Work

We would like to do real time testing on ARI InLoc functionality in university environment. We also, could integrate InLoc with existing ARI SLAM software, thus creating a more effective method. In addition, experiments can be done, testing both methods, and comparing InLoc to ARI localization and navigation system. Thus finding some shortcomings that could be improved on by integrating two different approaches. Finally, Simulated environment of university laboratory could be created for Gazebo and AI Habitat, for better simulation testing.

Bibliography

[1] E. H.-I. research and innovation action (RIA), “Spring: Socially pertinent robots in gerontological healthcare.” [Online]. Available: https://spring-h2020.eu/

[2] A. T. 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).

[3] R. P. A. T. J.Sivic., “Netvlad: Cnn architecture for weakly supervised place recognition,” (2016).

[4] H. I. J. M. J. T. T. A.Torii., “Is this the right place? geometric-semantic pose verification for indoor visual localization,” (2019).

[5] P. Lucivnak., “Visual localization with hololens,” (2020).

[6] A. R. J. M. T.Pajdla., “24/7 place recognition by view synthesis,” (2018).

[7] P. Robotics, “Ari official website.” [Online]. Available: https://pal-robotics.com/robots/ari/

[8] PalRobotics, “Ari software simulation.” [Online]. Available: http://wiki.ros.org/Robots/ARI

[9] Facebook, “Ai habitat official website.” [Online]. Available: https://aihabitat.org/

[10] H. M. T. M. M. J. T. A. InLoc, “Indoor visual localization with dense matching and view synthesis.” [Online]. Available: http://www.ok.sc.e.titech.ac.jp/INLOC/

Appendices

CD Content

In Table 1 are listed names of all root directories on CD.

Directory name	Description
Thesis	The thesis in pdf format
Ros-server-communication	python codes for client-Server nodes: communication.py and matlab-server-img.py
CMakeLists	File for building software packages.
package.xml	File that defines properties about the package.
communication.py	ROS client node, that communicates with InLoc.
matlab-server-img.py	Server node
readme.md	Instructions

Table 1: CD Content

List of abbreviations

In Table 2 are listed abbreviations used in this thesis.

Abbreviation	**Meaning**
ROS	Robot Operating System
RVIZ	aROS visualization tool
API	application programming interface
HRI	Human Robot Interaction
6DoF	Six degree of freedom
InLoc	Indoor Visual Localization
IMU	Inertial Measurement Unit
URDF	Unified Robot Description Format
CAD	Computer-aided design
IR	Infrared

Table 2: Lists of abbreviations