Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2016

SLAMIT: A Sub-map based SLAM system

On-line creation of multi-leveled map

Karl Holmquist

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

SLAMIT: A Sub-map based SLAM system On-line creation of multi-leveled map

Karl Holmquist

LiTH-ISY-EX--16/5021--SE

Supervisor: **Tommaso Piccini, Gustav Häger**
ISY, Linköpings universitet

Examiner: **Michael Felsberg**
ISY, Linköpings universitet

Division of Computer Vision
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Sammanfattning

I många situationer efter en stor katastrof, såsom den i Fukushima, är området ytterst farligt för människor att vistas. Det är i sådana miljöer som semi-autonoma robotar kan begränsa risken för människor genom att utforska och kartlägga området på egen hand. Det här exjobbet fokuserar på att designa och implementera ett mjukvarubaserat SLAM system med real-tidspotential användandes en Kinect2 sensor.

Exjobbet har fokuserat på att skapa ett system som tillåter effektiv lagring och representering av kartan för att tillåta utforskning utav stora områden. Det görs genom att separera kartan i olika abstraktionsnivåer, vilka korresponderar mot lokala kartor sammankopplade med en global karta.

Strukturen av system har tagit hänsyn till under utvecklingen för att tillåta modularitet. Vilket gör det möjligt att byta ut komponenter i systemet.

Det här exjobbet är brett i det avseende att det använder tekniker från flera olika områden för att lösa de sub-problem som finns. Några exempel är objektdetektion och klassificering, punkt-molnsregistrering och effektiva 3D-baserade okupationsträd.

Abstract

In many situations after a big catastrophe such as the one in Fukushima, the disaster area is highly dangerous for humans to enter. It is in such environments that a semi-autonomous robot could limit the risks to humans by exploring and mapping the area on its own. This thesis intends to design and implement a software based SLAM system which has potential to run in real-time using a Kinect 2 sensor as input.

The focus of the thesis has been to create a system which allows for efficient storage and representation of the map, in order to be able to explore large environments. This is done by separating the map in different abstraction levels corresponding to local maps connected by a global map.

During the implementation, this structure has been kept in mind in order to allow modularity. This makes it possible for each sub-component in the system to be exchanged if needed.

The thesis is broad in the sense that it uses techniques from distinct areas to solve the sub-problems that exist. Some examples being, object detection and classification, point cloud registration and efficient 3D-based occupancy trees.

Resumen

Después de grandes catástrofes, cómo la reciente en Fukushima, está demasiado peligroso para permitir humanes a entrar. En estás situaciones estaría más preferible entrar con un robot semi-automático que puede explorar, crear un mapa de la ambiente y encontrar los riesgos que hay. Está obra intente de diseñar e implementar un sistema SLAM, con la potencial de crear está mapa en tiempo real, utilizando una camera Kinect 2.

En el centro de la tesis está el diseño de una mapa que será eficiente alojar y manejar, para ser utilizado explorando áreas grandes. Se logra esto por la manera de la separación del mapa en distintas niveles de abstracción qué corresponde a mapas métricos locales y una mapa topológica que conecta estas.

La estructura del sistema ha sido considerado para permitir utilizar varios tipos de sensores, además que permitir cambiar ciertas partes de la sistema.

Esté tesis cobra distintas áreas cómo lo de detección de objetos, estimación de la posición del sistema, registrar nubes de puntos y alojamiento de 3D-mapas.

I want to thank everyone who has helped me during my master thesis, especially my two supervisors who have been greatly helpful during the course of the thesis. I'd also like to thank my examiner and everyone at CVL for the help and interest shown. As well as for my family and friends for their support.

Contents

Notation

Common terms

Term	Description
Key-frame	Selected frame from camera feed which are used in calculations
Registration	Process of matching two point sets

Abbreviations

Abbreviation	Representation
SLAM	Simultaneous Localisation And Mapping
R-CNN	Region based - Convolutional Neural Network
GICP	Generalized Iterative Closest Point
V-SLAM	Visual - Simultaneous Localisation And Mapping
IMU	Inertial Measurement Unit
CNN	Convolutional Neural Network
YOLO	You Only Look Once (An object detector)
BA	Bundle Adjustment
RANSAC	RANdom SAmple Consensus
T-SDF	Truncated - Signed Distance Function
LIDAR	Light Imaging, Detection And Ranging
FOV	Field Of View
KLT-TRACKER	Kanade-Lucas-Tomasi Tracker
SVM	Support Vector Machine

1

Introduction

Robotics is becoming more common in everything from everyday tasks like vacuum cleaning to more specialized tasks such as industrial assembly. These robotic systems are mainly used for monotonous, repeating tasks and in situations that pose a risk to humans, such as surveying a disaster area.

In disaster areas, there is the risk of being cut-off from the operator or delays in communications. To cope with such situations, the system must be semi-autonomous, to accomplish some tasks without direct control from a human operator. To perform these actions, it is required that these robotic systems accurately track their own movements and create a map of the environment as it is explored. This problem is known as Simultaneous Localization And Mapping, SLAM.

This thesis is related to the CENTAURO EU project; a project which intends to study and construct a semi-autonomous robot for exploring and mapping a disaster area. The project description on the official project page says:

"Disaster scenarios, like the Fukushima nuclear accident, clearly show that the capabilities of today's disaster response robots are not sufficient for providing the needed support to rescue workers. The CENTAURO project aims at development of a human-robot symbiotic system where a human operator is telepresent with its whole body in a Centaur-like robot, which is capable of robust locomotion and dexterous manipulation in the rough terrain and austere conditions characteristic of disasters. [...]"

"For routine manipulation and navigation tasks, autonomous robot skills will be developed. This will allow for taking the operator partially out of the control loop, which will be necessary to cope with communication latencies and bandwidth limitations and to reduce the operator workload. [...]" [1]

[1] For more information on the project, please refer to the official CENTAURO project page: `https://www.centauro-project.eu/`.

1.1 Goal

The thesis aims to design and implement a software system which robustly and accurately maps the environment while exploring it, known as SLAM. To achieve high accuracy, it also needs to counteract the inevitable drift caused by sensor noise. Detecting when the system has returned to a previously visited area, known as loop-closure detection, limits the effects of this drift.

The system is also capable of automatically separating the map into sub-maps as more of the environment is explored. This limits the memory requirements of the system for storing and processing the map. In addition, it will speed up operations of the system by allowing processing sub-maps individually. The system is also designed to be able to reach real-time performance. The description of the sensors which have been available during the design is seen in section 1.2. These sensors intend to reflect the future capabilities of the CENTAURO platform.

1.2 Tools

Some of the tools and utilities which were available for this project:

Hardware

The lab cart created for the iQMatic[2] project has been used to record some test data. The following sensors were available to mount on the lab cart:

- Kinect 2 camera - provides RGB-D data
- Occam omni-stereo camera
- A set of research grade cameras equipped with wide angle lenses
- IMU- for estimating movements

Software libraries

The following libraries, including their dependencies, have been used:

- ROS - Robot Operating System
- PCL - point-cloud Library
- Octomap - Octree based map storage
- OpenCV - Computer vision library
- Software for monocular SLAM developed at the CVL-department of ISY, LiU
- g^2o - Graph optimization framework

The ROS library handles inter- and intra-process communication. It permits separating the system in distinct parts and define how each part will communicate with the other parts. Each of these parts is a separate ROS node, where each node corresponds to a single operating system process.

[2] A desciption of the iQMatic project is available at `http://www.vinnova.se/sv/Resultat/Projekt/Effekta/2009-02186/iQMatic---finansiering-av-sista-aret/`

1.3 Problem formulation

One of the problems faced by a rescue-robot is mapping the terrain and position itself. This is an absolute necessity for path planning in an unexplored area since no information about the environment is known beforehand.

To solve these problems, there are three sub-problems which will be the primary focus of this thesis.

1.3.1 Localization and map building

An informative map should contain sufficient information to give a good view of the situation while avoiding clutter. For example, in order to traverse from one side to the other of a big indoor environment, the relevant information is which rooms connects to which and if there are any obstacles close-by. While detailed information about areas in each room are of less interest. However, when the robot desires to pick up objects or manipulate the environment such fine-grained detail is essential.

In order to handle both of these situations the complete system needs to represent the environment in multiple abstraction levels, so that the appropriate one can be used.[3]

- Topological map - contains large scale information about how the different areas connect to each other.
- Navigational map - contains information necessary for navigating in each area.
- Manipulational map - contains detailed information of a sub-part of the area, used for interacting with objects.

Since both the map and the starting position are unknown, the system must localize itself while creating the map around it.

1.3.2 Loop-closure

The localization or rather the pose estimation in all SLAM systems will have some inaccuracies. These inaccuracies will accumulate while exploring more of the environment, causing the estimated position to drift from the true position. This is particularly apparent when returning to an area that has been previously mapped. This is because the position estimate of the system will not correspond to the actual position because of inaccuracies. If the drift is too large, the system might miss-interpret a revisited area as a new area.

The loop-closure problem is to detect when the system revisits the same area and to correct the positions so that the map is consistent with the information.

[3] See deliverable WP5 in the CENTUARO project: `https://www.centauro-project.eu/intranet/dissemination/deliverables-1/d51_centauro_navigation_concept.pdf`

1.3.3 Separation of the map into sub-maps

The map represents the visited parts of the environment and will grow as the system explores more of the environment. If the map is too large to fit into memory, it will be impossible to update efficiently. To allow efficient manipulation of the map, it will be separated into sub-maps.

The separation of the map into sub-maps requires a consistent separation method; assuring that the same area starts and ends close to the same position when visited multiple times. Inconsistent separation into sub-maps risks complicating the loop-closure detection, as the sub-maps only have a minor overlap between themselves.

1.4 Layout of thesis

Five chapters span the content of this thesis, including chapter 1, the introductory chapter. Chapter 2 contains related work and discussions coupled to these. Chapter 3 describes the proposed method for the system and the sub-systems; it also describes the implementation of these in the complete system. The results obtained from the different sub-systems are presented in chapter 4. Finally, chapter 5 discusses the results and mentions possible improvements and changes of approach in the system to further improve the obtained results.

2

Background

This chapter covers related work on SLAM systems, on object detection, and recognition. It also contains work regarding point-cloud registration and representation. Object detection and recognition methods will later be used for loop-closure detection.

2.1 SLAM systems

SLAM systems has a relatively long history within the robot community. Leonard and Durrant-Whyte [17] proposed early a system on the topic. They describe the problem of Simultaneous Localization And Mapping with the help of a motion model and a system model.

Much of the early work and implementations have focused on 2D mapping with robots for path-planning. More modern work has increasingly focused on 3D-mapping; made possible by the increased processing power and the more accurate sensors available. Such work allows for both manipulation tasks, picking-up objects, as well as for path-planning for avionic systems, such as quadcopters.

Many of the SLAM-systems differ in the approaches taken to this problem regarding using different kinds of sensors. Some of the principal sensors that are in use are, IMU, LIDAR, and RGB-cameras.

The IMU approach is common and is relatively accurate when used together with a motion model. It has the drawback of only being able to estimate the pose of the system itself and require a separate sensor to map the environment. This leads to the problem of time-synchronization between the two sensors as they collect data at different rates. This is complicated by each sensor having a separate non-synchronized clock.

The usage of LIDAR is common as well, especially in combination with IMU measurements. One reason for that combination being used is because most LI-

DAR systems scan the environment at a high speed. Registering these scans from a stationary sensor is straight-forward. However, if the sensor is in motion it is essential to have a good estimate of the pose corresponding to each time scan. The high sample rate of the IMU allow reducing the need for interpolation between pose estimates and it because of this that the two are commonly used together.

Multiple of the state-of-the-art systems today use RGB-cameras, they achieve impressive results, as seen on the KITTI odometry benchmark [8]. This makes a visual based SLAM system feasible, both in terms of performance as for the easy access and attractive price-point of RGB- and RGB-D cameras.

Different branches exist within the V-SLAM community. Defining these based on the number of cameras used, we see three major branches. The monocular systems use one camera and tracks features from one frame to the next to estimate the pose [19]. These do not depend on a fixed known configuration between cameras and are thus easily set up and moved from system to system. However, the monocular systems cannot estimate the scale of the motion. One alternative is to replace the RGB camera with a RGB-D camera such as the Kinect. Another alternative is to use an IMU as an extra sensor [18].

The other main branch is the stereo camera systems. These require a fixed and known relative pose between the cameras. This makes the setup of the system more complex as a calibration for the cameras has to be performed. In exchange, they can estimate the scale and are thus not in need of any other sensor.

Some systems use more than two cameras [12]. Increasing the number of cameras increases the complexity of the system. In exchange, it permits creating a wider field-of-view which can give more robust and accurate pose estimations. For example, the Occam Omni-stereo camera which has a number of time-synchronized stereo cameras, resulting in a 360 degree field of view with synchronized images. This also has a proprietary driver which allows for receiving a depth image per camera pair.

The choice of sensor for this thesis fell on the Microsoft Kinect 2 camera, mostly because of ease of use and the availability of open-source drivers. As for the structure of the system, since real-time operation is required for the CENTAURO system, the number of possible options was somewhat limited. The choice fell on a smaller two-step system; resulting in accurate pose estimation without the use of post-processing optimization.

2.2 Loop-closure detection

There are essentially three approaches to loop-closure detection, map-to-map-, image-to-map- and image-to-image based detectors. Williams et al. [24] compares one method per approach and evaluates them in regard to robustness and precision.

Williams et al. [24] mentions the problem of finding features that are sufficiently unique for the map-to-map approach by Burgard et al. [1], which affect the robustness of the system. The image-to-map approach by Williams et al. [23] and the image-to-image approach by Cummins and Newman [4] are found to be

more robust, both reaching a higher recall and precision than the map-to-map approach [1]. However, Williams et al. [24] notices that the image-to-map based approach tested is more memory intensive than the image-to-image based approach, which makes it less suitable to use in an on-line SLAM system.

The tested image-to-image based approach[4] was the predecessor to the FAB-MAP 2.0 [5], which uses a global bag-of-word based approach. This bag-of-word requires offline training to create a good vocabulary based on the type of environment. The bag-of-word contains extracted features for some key-frames. The work done by Persson [19] mentions that it is possible to replace the SURF features used in the original paper with faster feature descriptors, such as FAST or STAR with minimal impact on recall.

One note of importance regarding the image-to-image approaches, such as FAB-MAP, is that they are sensitive for view-point changes when revisiting the same scene. The system requires that the scene is seen from the same view-point to limit the number of false-positives. To detect view-based loop-closures are often wanted in SLAM systems but in this thesis, which uses loop-closures in the topological map, a map-to-map based solution is more suitable. Additionally, the map-to-map based approach can also be done extremely sparse, saving time, and permitting estimating the relative transform between the two maps. Another possible approach to achieve map-to-map matching is by using a registration between the two local-maps, e.g. by doing a point-cloud registration.

The available point-cloud registration algorithms are generally either too slow or too inaccurate to achieve a reliable result when matching two clouds which might be missing a good initial relative pose and might also not be completely overlapping. The color supported generalized iterative closest point, GICP, [15] method is a relatively fast algorithm for point-cloud registration. The drawback being the accuracy when a good initialization is missing and the point-clouds are differing to a high degree. While the approach suggested by Danelljan et al. [6] is more accurate and does not need an initialization, it comes at the cost of longer processing time. However, neither author provided results for point-clouds with smaller overlap.

2.3 Object detection

Object detection aims at detecting one or several classes of objects in an image. This is typically done using a supervised learning method such as support vector machine, SVM, or a convolutional neural network, CNN. Some systems use only RGB, while others require RGB-D sensors or another way of measuring depth [11]. There are also some which segments point-clouds to detect the object type [9].

CNN based detectors has recently made a great deal of progress. One of which is the faster R-CNN [21], which manages to achieve good accuracy at a reasonable speed by combining the training of a region proposal network and the object classification network, there is also the slightly less accurate but significantly faster You Only Look Once, YOLO [20]. Both methods uses a combined net for

detecting and classifying. Earlier works have instead been using two separate nets for predicting bounding boxes and classifying these boxes. Combining these steps results in increased processing speed which is highly desired in a real-time system. However, the YOLO method struggles with small objects particularly if there are multiple clustered together.

The most commonly used benchmark for object detection is the PASCAL VOC data-set[7]. This provides a benchmark with a few categories depending on task to perform and allowed training data. The typical measurement of performance is the area under the precision-recall curve; precision being the percentage of detections being correct. While the recall being the percentage of correct detections found. This is often written as mAP calculated as the mean of the area underneath the precision-recall curve for all object classes.

The YOLO detector uses three different models; the medium sized model manages to reach 45 fps while maintaining good performance. While the biggest model manages to reach a similar accuracy and a significantly higher speed than that of the faster R-CNN [21].

2.4 Efficient map storage and access

Map storage research has resulted in several methods to save and visualize point-clouds. One of the more expensive one is to transform the point-cloud by the relative transform before merging all the points into a single point-cloud. This method gives the most accurate depiction of the environment since it saves all available information in the point-clouds. However, it makes the map grow extremely fast as well, while not necessarily helping with scene understanding.

Instead, a more appealing method is to use a voxel grid and storing it efficiently in the form of an octree structure where each voxel contain information of occupancy. This can be highly efficient since it is possible to decide maximum resolution when adding point-clouds and it requires much less memory to save than a complete point-cloud.

The problem with this approach is that measurement errors and passing objects can cause trouble if not handled in some way, since noise effects the occupancy information of the voxels making false-detection a big problem. Using a probability based octree, such as Octomap [14] solves this problem. It maintains the highly efficient storage form of octrees, while also allowing for uncertainty in the measurements, caused by sensor noise.

2.5 Sub-map separation

Over time, multiple methods have been proposed within the field of sub-map based SLAM. One recent system, proposed by Kähler et al. [16] is similar to the proposed system by sharing the goal of achieving real-time performance and creating a dense 3D map. It is also similar in using 3D data as input, and explicitly mentions the Kinect as a possible source. This system also uses sub-maps to

increase the accuracy and limit computational requirements. The sub-map splitting is also based on what is in view. The main difference in how to represent the map during creation is that it uses a truncated signed distance function, TSDF to represent the current view. The size of the maps is also smaller than the size of the proposed system in this thesis. This difference in size is caused by the difference in the approach taken. The proposed system intends to maintain an intuitive representation of the region, by creating a map per region, while the approach suggested in [16] mainly uses the sub-maps to increase efficiency by processing a set of nearby poses together.

3

Method

This chapter describes the proposed system. First, an overview of the system is provided. Second, each sub-system is described in detail.

3.1 Overview

Figure 3.1 shows the general structure of the system. It shows the RGB-D frame used to estimate the pose and register the point-clouds. The current sub-map saves the detected objects which are deemed static and their detected positions to detect loop closure. This is done by comparing the set of detected static objects in the current sub-map with the earlier created sub-maps.

Upon detecting a loop-closure, the system re-optimizes the topological map to make it consistent; since consistency between the estimated pose and the actual pose is not guaranteed because of sensor drift. This drift cannot be detected unless a loop-closure has been detected. If no loop-closure is detected a check is made whether to start a new sub-map and, if that is the case, it will add it as a new node in the topological map. Otherwise nothing will be done to the topological map and the system starts over with a new frame.

Figure 3.2 shows more detail about the pose estimation and registration step. It shows the extraction of points from the image using FAST features and the tracking of these points using a standard KLT-tracker to estimate the pose by the Perspective-n-Point, PnP, algorithm. Filtering is applied to the incoming frames, only allowing frames where sufficient translation between these has occurred to be used to estimate the position. These frames are known as key-frames. When a new initial pose has been found, it will be used as an initiation of the color supported GICP. The first time the point-clouds are registered, the scale will be estimated since the essential matrix estimation only gives the translation up to scale.

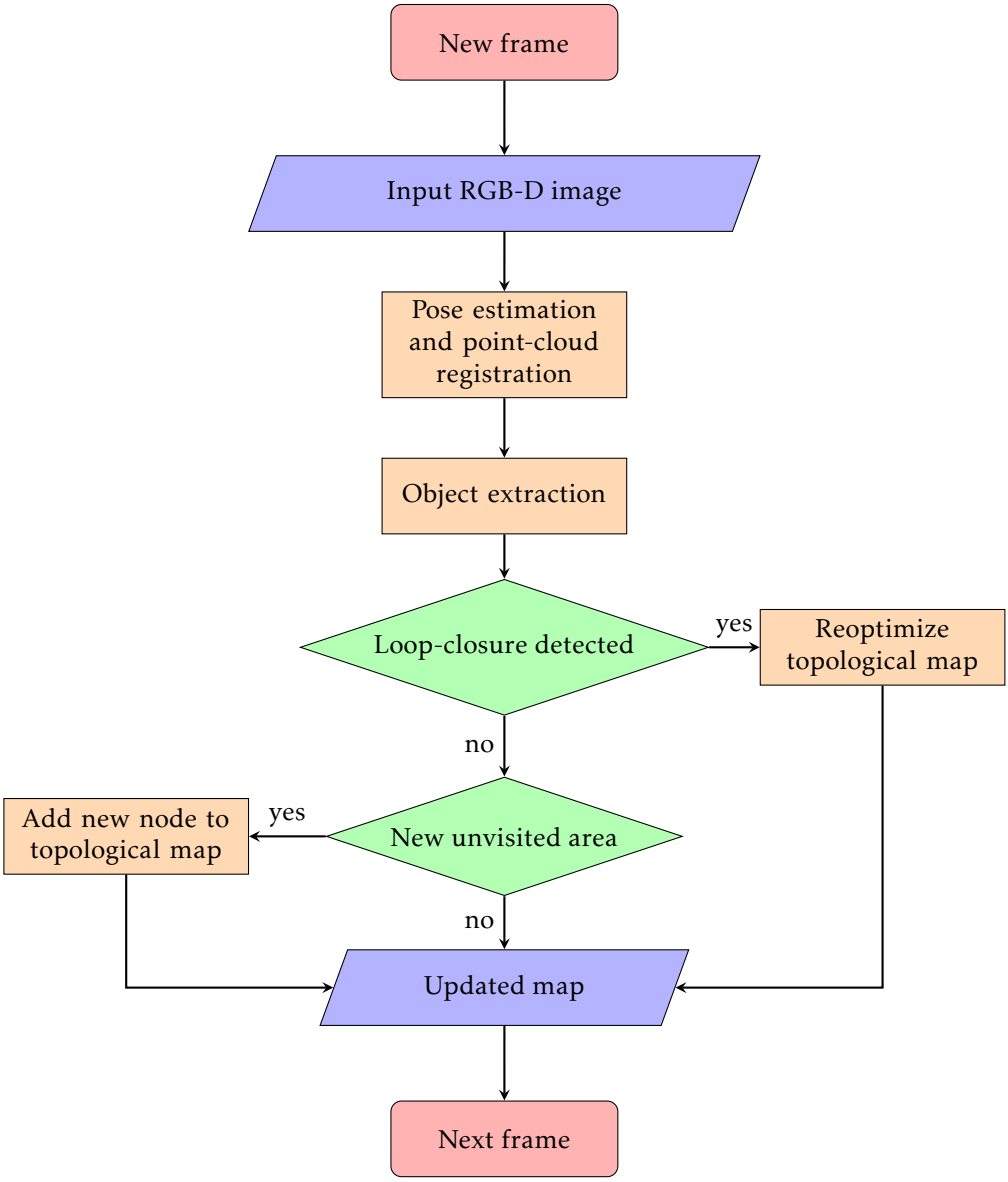


Figure 3.1: *Overview of the system: Key-frames are extracted from the input RGB-D images. These key-frames are used to estimate a pose and create a point-cloud if not available from another source. For each key-frame, the object detector detects and extracts the static objects in-view. Using these objects, the system decides if an earlier area is currently revisited. Based on this result the topological map is updated, by re-optimizing the positions of the sub-maps, adding a new area or, doing nothing.*

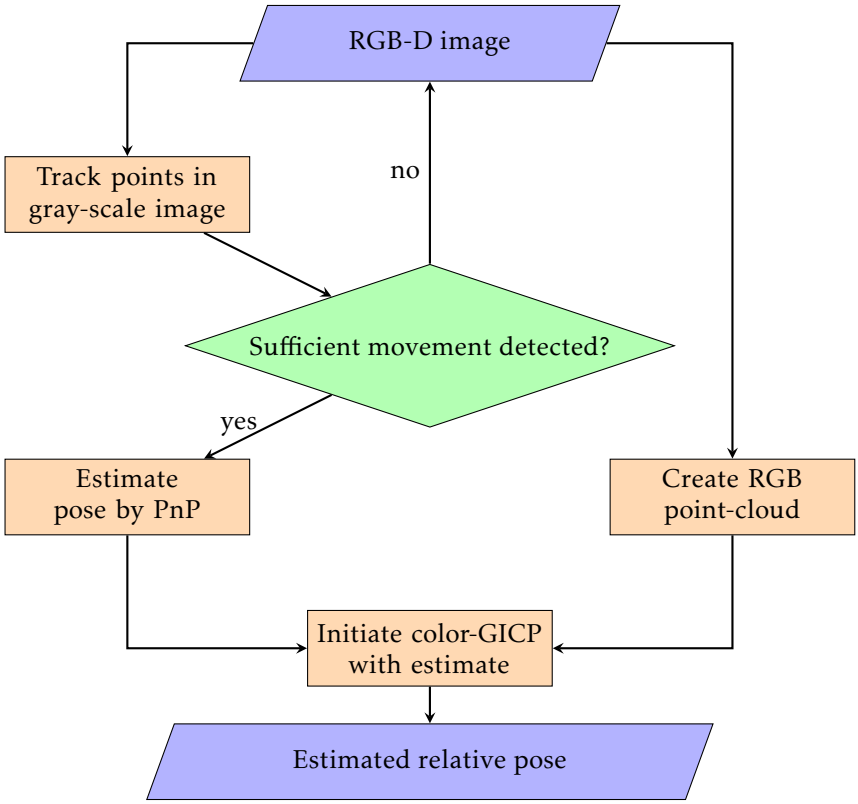
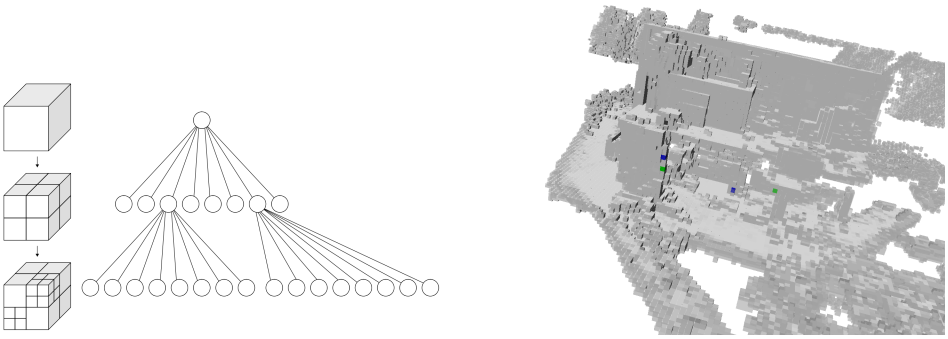


Figure 3.2: *Detailed overview of pose estimation and point-cloud registration: Key-frames are chosen based on the detected movement from tracking key-points from frame-to-frame. For each key-frame an initial pose estimation is done, which is refined by using the estimate as input to a point-cloud registration algorithm.*



(a) *(b)*

Figure 3.3: *a) Visualization of the underlying octree structure of Octomap b) Visualization of occupied voxels of an Octomap*

3.2 Multi-resolution map building

The reason for creating a multi-resolution map or rather a map with multiple abstraction levels, is to allow for visualization based on what is of interest at the moment as well as being able to expand the map when more of the environment is explored without a need to process the entire map.

A map using multiple abstraction levels also allows separately optimizing the different levels, both the topological map as well as all sub-maps. The optimization can be done cheaper as the number of parameters to optimize is smaller than it would be in a single level map.

Proposed solution

Unlike most SLAM systems that use a single frame-to-frame graph, this system has a higher level region-to-region graph as well as multiple sub-maps. The proposed method is representing the map in three distinct levels of abstraction:

- Topological map - is represented as a region-to-region graph containing various nodes and the translation and rotation which relates neighbouring nodes.
- Navigational map - uses an Octomap [14] to represent one of the nodes in the topological-map.
- Manipulational map - uses an even finer Octomap to represent a part of the sub-map and is either anchored to the navigational map or a separate entity which is removed then the current area is exited.

The graph-map makes it possible to only process or visualize a subset of the total map. Additionally, it allows the loop-closure optimization to be simpler, limiting the number of variables required in the optimization.

An example how the topological map, the graph map, would be after follow-

ing the route depicted in figure 3.4 can be seen in figure 3.5. The relative transform from node i to node j is shown as T_{ij}.

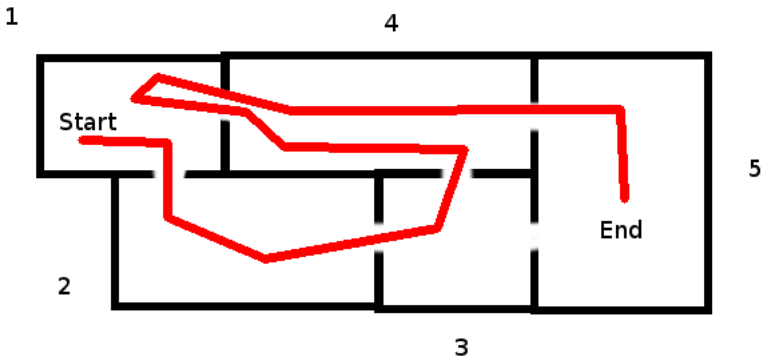


Figure 3.4: *Illustration of a path through an imaginary indoor environment*

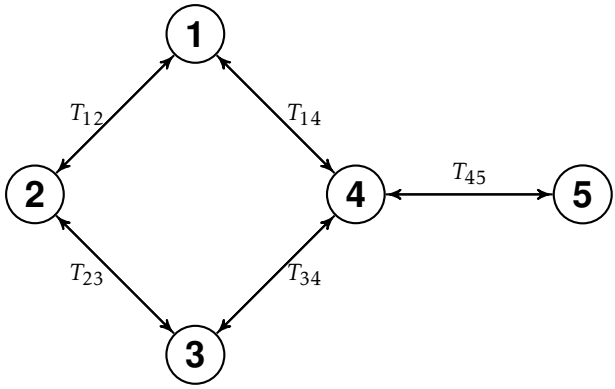


Figure 3.5: *Resulting graph from figure 3.4 after loop-closures*

3.3 Pose estimation

The pose estimation is one of the fundamental components, having a major effect on the performance of the other system components.

Proposed solution

The proposed SLAM system uses a two-step method. The first step is to estimate an initial pose. This is done by using parts of a monocular V-SLAM system developed at CVL. Once an initial pose estimate is obtained, a refinement step using GICP [15] is performed.

The initial pose is found by estimating the essential matrix between two key-frames, which results in the relative rotation and translation. However, the estimate is missing a correct scale since this initial estimation step uses a monocular system without scale estimation. To determine the scale, the refinement step is necessary.

From the third frame and onward, the scale factor is propagated to following frames by using the relation:

$$\alpha_{1,3} T_{1,3} = \alpha_{1,2} T_{1,2} \oplus \alpha_{2,3} T_{2,3} \tag{3.1}$$

The three different scale factors are denoted α and the relative transforms between key-frames as T. $\alpha_{1,2}$ is either known or set to one for the first set, after which all scale factors are related to that of the first.

The pose estimation will be unreliable if the essential matrix is close to singular. In order to avoid this degenerate case, key-frames are used for the actual estimation. These key-frames are selected such that sufficient translation between them exists. Since the true translation is unknown at the moment of receiving a frame, the translation is instead estimated using the pixel distance that the tracked key-points from the last key-frame have moved.

The second step is used for refining the estimate and estimating the scale factor. The scale factor is estimated by doing a point-cloud registration without an initial guess of the transform. Once the registration is performed the length of the translation relative to the initially estimated translation vector is the scale factor. When the scale factor has been set, it is used to scale the following initial estimates. These estimates are used to initialize the registration method by transforming one of the point-clouds. This helps the registration converge faster, and it also allows for doing a sub-sampling of the point-clouds to increase the speed of the algorithm. Down-sampling the point-clouds results in an increase in speed since the complexity of the registration method is, at least, $\mathcal{O}(n \log n)$ depending on the implementation. One of the more costly steps is the nearest neighbour search, which is done by a kd-tree and has a complexity of $\mathcal{O}(n \log n)$.

If the first step fails and loses track, the system will reset the scale factor and start over. The transform between the first pose before the failure and after is found by performing a non-initialized registration step, similar to the scale estimation step.

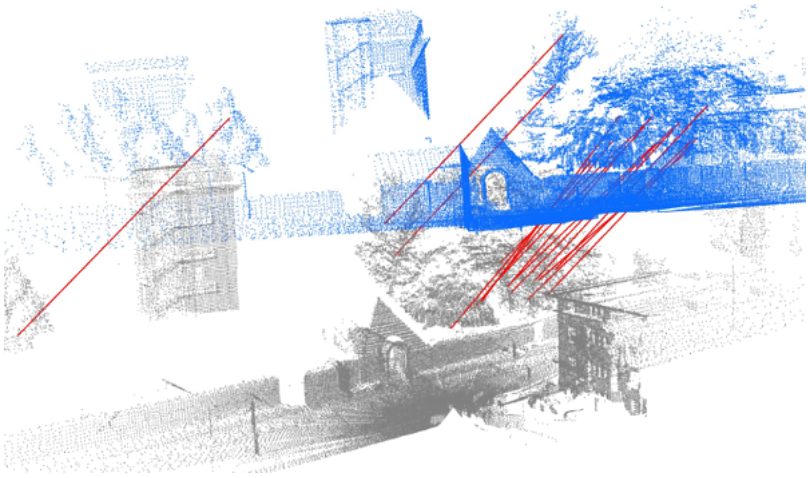


Figure 3.6: *Illustration of point cloud registration*

3.4 Object detection and localization

The object detection problem is localizing objects in an image. The detection should be rotation- and scale- invariant in order to detect the object regardless of their 6D pose. In a system for previously described use cases, it is also important that the detection can be run at a reasonable speed and acceptable accuracy.

Proposed solution

The proposed method uses the YOLO detector [20], since this method is able to achieve real-time performance, e.g. 30+ fps, while maintaining a good accuracy and also is available as open source with pre-trained weights. A more thorough comparison to other algorithms is given in section 2.3. r The method uses only the RGB-channels from the key-frames chosen by the pose estimation step. It starts by using the YOLO detector to detect the objects currently in view, and extracting the feature vector for the grid point in the center of the bounding box from one of the layers in the CNN.

These are both passed to the map builder, which uses ray-tracing to detect the first occupied node in the direction of the center of the detected object. This node is seen as the 3D-position of the object. Each detected object is passed to the list of objects of the current region. If it finds another object which matches both in class, appearance and position, the position and covariance of the matching object is updated. Otherwise a new object is added.

3.5 Loop-closure detection

If a loop exists in the graph, the loop-closing can be done to counteract the accumulated drift caused by inaccuracies in the pose estimation. This drift is seen

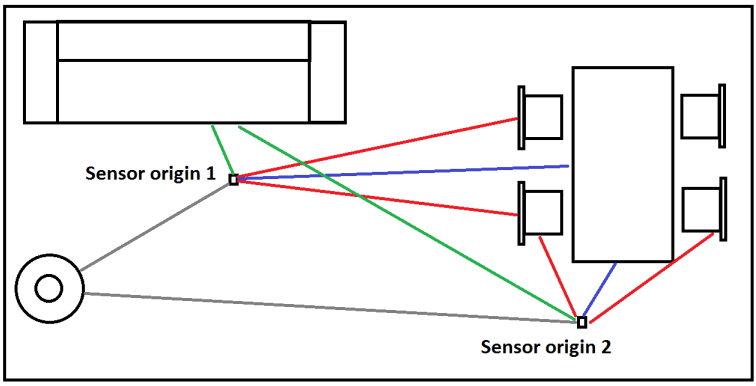


Figure 3.7: *Illustration of ray-tracing from two distinct positions, creating two distinct static object sets. The two sets are similar enough to allow detecting that the sensors both are in the same room.*

clearly when a previously seen area is revisited.

In this work a method for loop-closure using a bag-of-objects is suggested. This approach extracts and classifies a set of static objects in view and projects them into 3D-space. Each object will have a position, an object class and a feature vector that should be discriminative enough to differentiate it from objects of the same class. These objects can later be used when an area is revisited. Matching the objects in the two areas is a registration problem.

This bag-of-object approach has the strength of not requiring that the views match, instead it is sufficient that a subset of the same objects has been recognized. As the registration is done on the level of objects, it is not required to have the same view of the object for the registration to work. The registration is done using the position of the objects.

Proposed solution

Each node in the graph map contains a list of objects, containing a 3D-position, y, a class label, c, a covariance matrix describing the uncertainty in the measurement, C, as well as a feature vector extracted from the same CNN which was used to detect them, f.

In order to limit the number of configurations of matching objects needed to be tested. The first step is to find possible matches between the two sets of objects, Ω_1 and Ω_2. First each of the elements of Ω_1 is matched to all objects with the same class from Ω_2. For each of these matches, the Euclidean distance between the two feature vectors, f, is calculated, and if they are sufficiently similar

the match is added to a list of matches. From the list of matches the most probable match is selected as the matching object. The following equation shows the matching between landmark $l_{1,i} \subset \Omega_1$ and $l_{2,j} \subset \Omega_2$.

$$j = \arg\min_j \left(F\left\{ l_{1,i} == l_{2,j} | c_{1,i} == c_{2,j} \right\} \right) \tag{3.2a}$$

$$F\left\{ l_{1,i} == l_{2,j} \right\} = \frac{1}{N} \sum_{k}^{N} \sqrt{f_{1,i,k}^2 - f_{2,j,k}^2} \tag{3.2b}$$

The matching based on class and appearance will limit the possible number of matches to a size which makes it possible to register the two sets of objects to each other. All the matching object-positions in homogeneous coordinates are concatenated to two matrices, $\bar{y}_1$ and $\bar{y}_2$, one for each set.

$$\bar{y}_i = \begin{pmatrix} l'_{i,0} & \cdots & l'_{i,m} \end{pmatrix} \tag{3.3}$$

From these two matrices, the relative transform is calculated using the pseudo-inverse of the second set, $\bar{y}_2$.

$$\bar{y}_1 = H\bar{y}_2 \tag{3.4a}$$

$$\epsilon_{reproj} = \sum_{k=0}^{m} \sqrt{y_{1,k}^2 - Hy_{2,k}^2} \tag{3.4b}$$

In the ideal noise-free case, (3.4a) will hold but generally this is not the case. The resulting transform, H, is instead the least-squares solution that minimizes the re-projection error of transforming the landmarks in the first set to the second, as seen in (3.4b).

The re-projection error, ϵ_{reproj}, is used to measure if the solution is accurate enough to be considered an actual match between the two sets.

3.6 New area detection

The detection of a new area is deeply related to the structure of the proposed method for mapping. As more of the environment is mapped, the map has to be split into regions. This splitting can be done in various ways as long as it is consistent over visits. The consistency is required for the loop-closure detection to be more accurate. If the area is mistakenly split into two separate areas the loop-closure detection will have fewer objects to work with.

Proposed solution

The proposed system intends to find an estimate of the entropy of the newly mapped area, the entropy being a measurement of the information that the newly

added point-cloud adds to the map. Rocha et al. [22] defines the entropy of the map, H at time t_k as the sum over all voxels of the differential entropy, h, for each coverage pdf, p, of a certain voxel, c_l.

$$H(t_k) = \sum_{l \in y} h\left(p^k(c_l)\right) \tag{3.5}$$

The differential entropy is the continuous version of entropy and is defined for a continuous random variable with the pdf $f(x)$ and the domain S as.

$$h\left(f(x)\right) = -\int_S f(x) log\left(f(x)\right) dx \tag{3.6}$$

For further detail about the coverage pdf of each voxel $p^k(C_l)$ see [22]. This calculation of the entropy is computationally heavy and complex. Instead the entropy is estimated by making the assumption that the detected objects are relatively evenly dispersed inside of the area. This will in many cases not hold true, but it is generally sufficient to get an accurate estimate for a much smaller computational cost.

The proposed solution is fast as well as relatively consistent between visits. This method uses a line-of-sight approach, which for each object in the current area checks if that objects is in sight. If the check shows that too many of the objects are out of sight, it is regarded as a new area has been entered. To detect if a certain object is in sight or not, a ray-cast is performed in the region map. The check for new area is only performed after a certain number of objects have been detected in the current region. This is done to limit erroneous exists that, for example, flickering objects can cause.

3.7 Loop-closure optimization

When a loop-closure has been detected the topological map, the region-to-region graph, needs to be optimized. This optimization is done over all nodes and not only the ones part of the loop, in order to minimize the total error:

$$\epsilon = \sum_{k=1}^{m} \left| \bar{y}_k - T_{k-1,k} \bar{y}_{k-1} \right| \tag{3.7}$$

Proposed solution

The proposed solution uses the g2o framework and the sparse Cholesky factorization from CHOLMOD [2]. More exactly it uses the sparse on-line graph optimizer available in the g2o framework with standard solver.

The optimization is done only over the nodes and edges in the graph, no landmarks or other features are added since the nodes are independent areas with little overlap.

In the case of a loop-closure the edge is added to the graph while the node itself isn't. This is done because the node is not a new node but rather a revisited one, so the graph is set to go from the last node to the revisited node. An example of a resulting graph is seen in figure 3.5.

4

Results

This chapter contains the experimental results as well as the strengths and weaknesses of the system and subsystems. The system was tested qualitatively in a few scenarios. The data used was internally recorded data of a corridor in the basement of the B-building as well as the freely available freiberg3_long_office_household data-set[1]. [2]

4.1 Memory footprint comparison

This section will compare the memory footprint of an Octomap in comparison to a PCL point cloud. The octomap maintains a small memory requirement in comparison to saving all point-clouds individually. This comes at the cost that the added poses cannot be changed after their corresponding point-clouds have been added to the octomap. However, the octomap can be converted to a point-cloud again by using the center of each node as the coordinates of the point. This allows for registration between two different nodes after a detected loop-closure. The most expensive operation of the map building is inserting new point-clouds into the octomap.

It is also worth the time to compare the saved space by inserting the point-clouds into a discrete representation. The octomap structure stores both occupied and unoccupied nodes while point-clouds represent the occupied points only. The .pcd format, which is used internally in the proposed system, can contain multiple fields of varying precision; this system uses four channels of float precision; X, Y, Z and the color RGB merged together. This gives a size per point of 4 fields * 4 Bytes = 16B per point. The Kinect 2 depth sensor has the

[1] The Freiberg RGB-D data set can be found here

[2] All tests has been done using an Intel® Xeon(R) CPU W3565 processor@3.2GHz and a Nvidia® GeForce GTX 750 Ti graphics card, the data itself has been played from recorded ROS bags

size 640*480 = 307200 points which gives a total file size per point-cloud from a Kinect 2 sensor as 16B * 307200 ≈ 5MB per point-cloud. In comparison, the octomap format .ot uses 8 bytes of data per node if color is included. This gives for an area of 15m*15m*4m at 4cm resolution a file size of 112MB, which is approximately equal to twenty-two point-clouds. However, when running the system on a data-set of an office, the octomap size only grew to 16.5MB after inserting 164 point-clouds. This shows that for normal areas during mapping, the octomap size will be less than 200MB and most likely much less.

The octomaps data structure is more efficient than the point-cloud structure; which combined with the graph-map structure results in that the memory usage for the map storage is kept low at all times. The precision lost in the octomap compared to in a point cloud can be made relatively small as long as not the system fails in accurately locating itself.

4.2 Failure cases and refinement analysis

The initial pose estimator has two possible degenerate cases causing the algorithm to fail or reach a local minimum instead of the global minimum. The first case of near-singular essential matrix has been discussed earlier in section 3.3 and is mitigated by extracting key-frames.

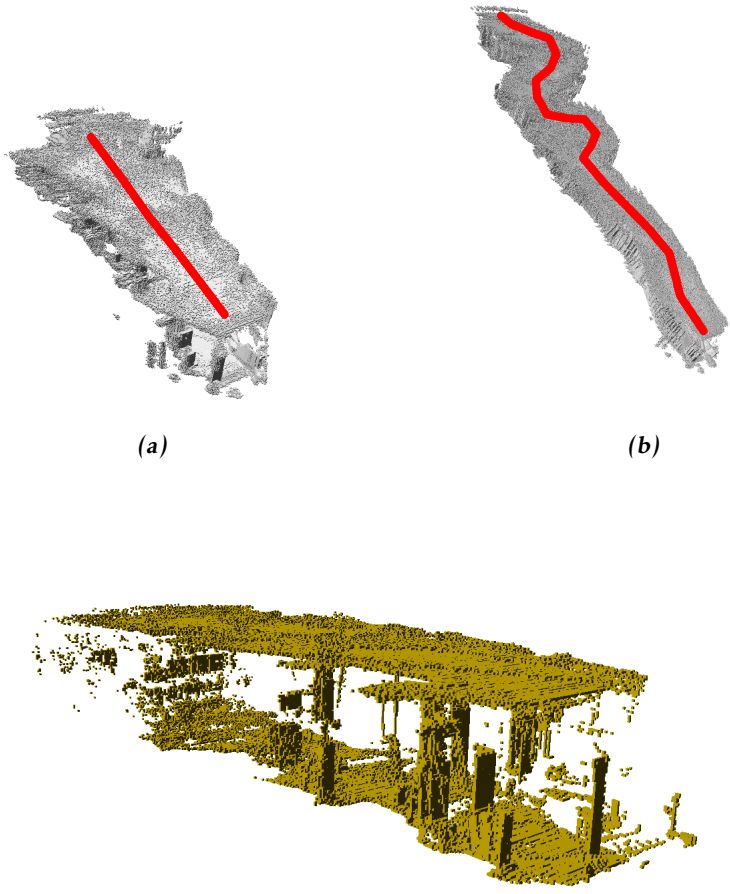
The second degenerate case appears when all points are approximately at a plane parallel to the image plane. In this case, it is impossible to calculate a good estimate of the rotational component. This is caused by the fact that in a pure rotation all points move a distance which is directly proportional to the distance from the center of rotation. If the whole scene is at an equal distance to the camera center, a translation and rotation will appear equal.

Another problem occurs when the first estimated pose, either from the initial pose estimation step or from the point-cloud registration step, is bad. This causes the scale estimate to be bad which can cause the refinement step to fail in the following frames, giving either a wrong solution or causing it to diverge.

Studying the resulting map from the system run on a data-set of a corridor, the importance of the refinement step can be seen. In figure 4.1b the point-cloud registration step has only been used to initialize the scale but not to refine the pose. This causes a lot of artifacts as seen along the walls where a repeating pattern can be discerned; this pattern comes from the use of incorrect poses then inserting the point-clouds. In comparison to figure 4.1a, a much clearer segmentation of objects is seen, without repeating patterns or similar artifacts.

Figure 4.1c shows a segmented version of the map obtained by using the refinement step and shows mainly the bigger connected surfaces, floor, ceiling as well as pillars, and tables among other things. One of the RGB images from the sequence can be seen in figure 4.2. It shows why the limited field of view, FOV, of the forward mounted camera together with all the objects along the walls that obstruct the view to the sides causes most of the walls to be missing in the 3D-map.

The corridor data-set is a relatively nice one, contain much structure and mostly translations. Another data-set which was recorded in the old TekNat li-



(a) *(b)*

(c)

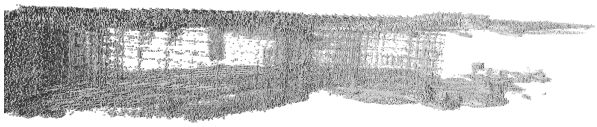
Figure 4.1: *Illustration of the need of refining the initial pose estimation. (a) Complete algorithm using point-cloud registration for refining the estimate. (b) The point-cloud registration has only been used to estimate the scale and not for refinement. (c) Clearer image of (a) showing parts of the inside of the corridor.*

Figure 4.2: *Image from the corridor used in one of the data sets*

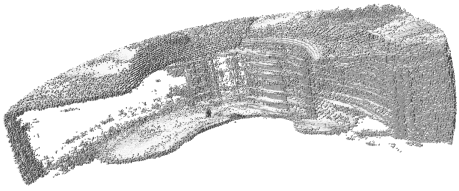
brary at the LiU university is a much harder problem. Mostly because of the big open area in the middle with little structure, especially in the point-cloud data.

Figure 4.3 shows the same test case as above, with and without the refinement step, for the TekNat data-set. Both gives a similar result where the structure of the room cannot easily be discerned. The trajectory on the other hand is mostly wrong in the scale estimation and have a relatively large drift, especially in the rotational component. This error is most likely due to the limited structures the pillars and the walls gives since when studying the first part of the sequence, it manages quiet well to create the map as long as the sofa and tables are in focus.

An image of the environment is seen in figure 4.4. Both of these data-sets are too small to show significant drift in the Octomaps above. In larger environments, however, it will be apparent unless loop-closure is applied.



(a)



(b)

Figure 4.3: *Illustration of one of the situations when the refinement step cannot converge to an accurate solution. (a) Complete algorithm using point-cloud registration for refining the estimate. (b) The point-cloud registration has only been used to estimate the scale and not for refinement.*

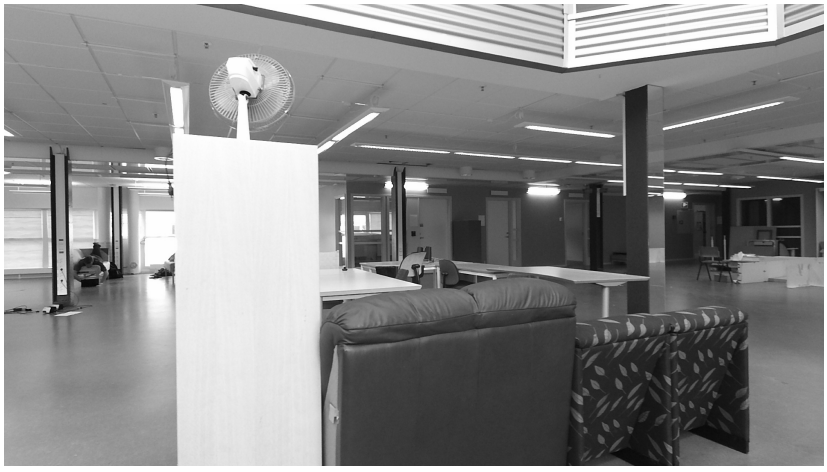


Figure 4.4: *Image from TekNat library at LiU*

4.3 Loop-closure detection

In order to evaluate the loop-closure detection, the results will be compared with one of the state-of-the-art systems available, the FAB-MAP 2.0[5]. The freiberg 3_long_office_household data-set is used to compare the two systems, the FAB-MAP systems is given 3 different image views of one side of the office, all of which are recognizable as being of the same scene for a human observer. Figure 4.5 shows the different views that are used in the test. Figure 4.5b is the most different compared to the other views while figure 4.5a and figure 4.5c are pretty similar.

The FAB-MAP system uses pre-trained vocabularies to be able to extract and recognize features that are suitable to discern different scenes. In this test two different pre-trained vocabularies has been used, the OxfordVocab_SURF_11k trained mostly outside in Oxford as well as the IndoorVocab_10k_KMeansRadius Based which is trained indoors.

(a) View 1

(b) View 2

(c) View 3

(d) View 1

Figure 4.5: *Shows four views from the freiberg data set which are used to test FAB-MAP 2.0. The first and fourth is the exact same image used to illustrate the result of a perfect match*

(a) *Indoor vocab*

Image:	a)	b)	c)	d)
a)	1	-	-	-
b)	2.59e-08	9.99e-01	-	-
c)	1.18e-03	1.95e-04	9.99e-01	-
d)	9.16e-01	1.97e-02	2.31e-03	6.21e-02

(b) *Oxford vocab*

Image:	a)	b)	c)	d)
a)	1	-	-	-
b)	3.26e-08	9.99e-01	-	-
c)	1.46e-03	2.11e-04	9.98e-01	-
d)	9.16e-01	1.97e-02	2.31e-3	6.21e-02

Table 4.1: *Each row contains the probability that the current frame matches to an earlier visited view followed by the probability of the view being from a new scene. If all views are unrelated to the others, the matrix will be identity. All values have been rounded to have the precision of 3 digits plus exponent.*

Both vocabularies fail on relatively small variations in view-point and only detect a match between the two exact same images; resulting in a 91 % probability of them being the same. The complete results of the algorithms can be seen in table 4.1.

The loop closure detector implemented in the proposed system, can be improved in various ways. One is taking a more robust approach to the registration problem. The system matches each landmark in the respective nodes to the landmark with the highest correspondence value, which is not necessarily the correct and may in that case introduce outliers. But in the current implementation, these matches are deemed as correct and from the set of correspondent 3D-points the affine transform between the two are found by solving the least-squares problem in (3.4a). The formulation of the least-squares problem is linear, making it relatively sensitive to outliers. This is in practice not a problem thanks to the robustness of using objects.

4.4 Object detection

The object detection is sufficiently robust and gives accurate bounding boxes for the detected objects despite using the smallest of the pre-trained models. This model causes some error detections, but most of these are just temporal and occur on objects that have not been present in the training data.

These wrong detections are irrelevant as they can be filtered out, e.g. by counting number of occurrences. If there is a consistent false object detection, it can be used just as well as any other object; as the most important property is that they

are consistent, due to the matching working mainly on the feature descriptor. An example output image from the YOLO detector can be seen in figure 4.6.

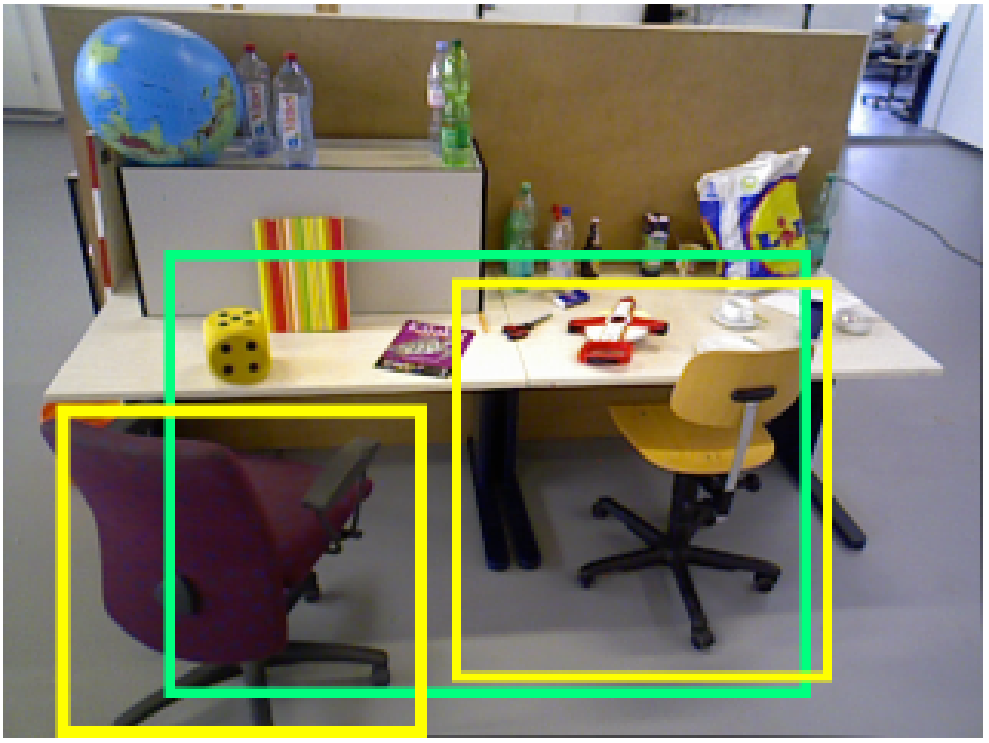


Figure 4.6: *YOLO detection of an office scene: Yellow boxes mark chairs and green boxes mark the desk*

The positioning of these objects are done in the octomap and are reasonably accurate. One problem is the positioning of concave objects, such as chairs and tables. Figure 4.7 shows the positions of the objects in the octomap. Both of the chairs are positioned accurately, since the back of the chair is in the center of the bounding box. The table, on the other hand, is more complex since the center of the bounding box is underneath the table, close to the legs of the table. These kind of misses in the object positioning is a common characteristic of objects such as tables that are big and concave. The inaccuracies of the positioning of distinct object types has to be taken in account when clustering object detections through time. Figure 4.7 shows the clustered detections of a scene from a few different view-ports. The image illustrate the problem caused by tables by detecting it as two separate objects. In this case it would be beneficial to adjust the covariance matrix that are used to estimate the probability of the objects matching.

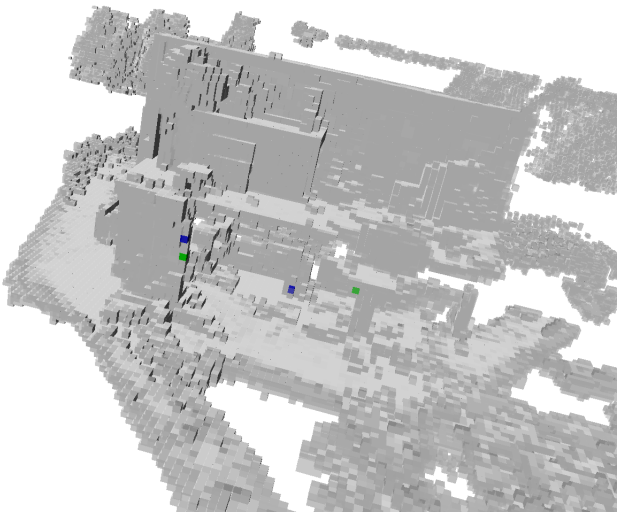


Figure 4.7: *Octomap with projected object detections. Blue = table, Green = chair*

4.5 Topological map

The method described in section 3.6 work well in most cases. It has a clear drawback of not being able to detect a new area if no object has been detected in the current one. There is also the problem of boundary checking between two areas, should the system exit an area and then return directly, it will not detect that a new area has been entered. In practice this is not a problem, as a loop closure will be detected if it reenters the same area again.

Another possible problem could be seen by running the system on the freiberg 3_long_office_household data set. This data set contains a room with relatively few objects along the walls and an office in the middle of the room, with two desks separated by a movable wall, as partly seen in figure 4.8.

Since most of the objects are seen from one of the sides of the wall and not the other, the system detects that a new area has been entered as it travels around to the other side of the wall. This is to be expected and could in certain cases also be desired in order to separate larger areas into well-defined sub-maps. Thus, it is important to note that the geometry of the room and the placement of the objects inside of it affect the new area detector to a large degree.

Even though the new area detector is not one of most time-crucial parts in the system it still has a relatively low computational complexity which depends on the number of objects and their distance from the current position

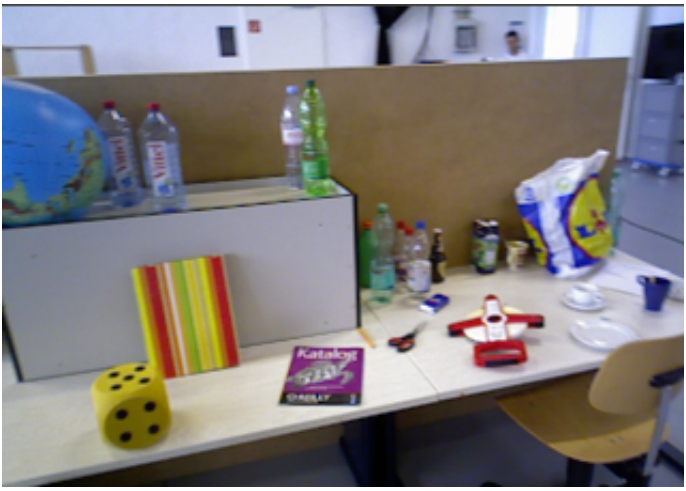


Figure 4.8: *Image from the freiberg3_long_office_household data set: Showing one side of the separated office*

4.6 Computational complexity

Unfortunately, the system is still too heavy to be reliably run in 30fps, which is the frame-rate of the Kinect 2 sensor. This can be corrected by further optimizing some of the components. At the moment, the most time-critical part to still lag-behind is the pose estimation component. The initial pose registration part has been run earlier at a 60fps on another setup while in the current setup it only reaches a maximum of about 10fps. The point-cloud registration has also been tested and can be run at a lower frequency of about every fourth key-frame without affecting the pose accuracy. However, this largely depends on the structure of the environment and how the sensor movements are. Note that it is frames per second not key-frames per second.

About 6fps on average is reachable in the current implementation on the test system. Using a more high-performance system and further optimization of the components would allow for a higher fps.

5

Conclusions

This final chapter will describe some of the potential areas of further research connected to this system, followed by a summary of the thesis.

5.1 Future work

In this section, some of the possible improvements that were discovered during the implementation and testing of the system are presented. These improvements are left as a future work for those interested.

Relaxing the real-time requirement

If the ability to view the current map in real-time is relaxed, it is possible to receive a more accurate map by creating the map after leaving the region. This is possible by expanding the local SLAM system with a bundle adjustment, BA, step which further refines the poses. As well as adding a view-based loop-closure detector inside each region, e.g. FABMAP 2[5]. The drawback is that the scale-estimation and the point-cloud registration must wait until the area is exited to be able to optimize the poses, limiting the ability to get a correct map of the environment during exploration.

Building the map after leaving an area will also affect the other components. The method for detecting new areas requires a different approach, since the map is not created before the area is exited. Additionally, the object positioning method is required to be changed if the positions are needed during exploration. One alternative method to position the objects is by triangulation[13]. In this case the detected objects can be added to the optimization as well, augmenting the SLAM system with landmarks as well. This permits direct usage of the optimization described by Graham et al. [10].

Increasing accuracy of object positioning

The positioning of 3D objects is currently implemented in a straight-forward way, a more robust and precise method is to use a distributed ray-trace[3]. A distributed ray-trace can be done by sending multiple rays randomly distributed within the bounding box of the object. This method of calculating a ray-trace is more computationally expensive. But will result in more robust and accurate detected positions, especially if the object contains holes, such as a pin-chair.

Training the object detector

The YOLO detector[20] used in the system, uses the pre-trained weights available on the official website. These weights are trained on the VOC2007[7] data set which contains 20 classes. In this system, it would be beneficial to retrain the system on other objects while removing some of the ones which exist in VOC2007, such as cats, dogs, and people which all tends to be highly mobile and cannot reliable be used to detect loop-closures. It is also possible to tailor the detector to better match objects likely to be found in the kind of areas which it will be used in.

Registration after loop-closure

As mentioned in section 4.3, the least-square algorithm uses a linear cost function which is sensitive to outliers. These outliers can be filtered by imposing a threshold on the number of observations of it before deeming it accurate. While in environments with lots of objects, both actual objects and outliers, there is also the possibility to filter out outliers by using RANSAC. Another possible improvement is to calculate the weighted least squares solution instead, weighting each of the measurements by an estimate of the covariance matrix of the object position. This lessens the impact of objects that likely are outliers.

New area detection

New areas can be detected in various ways and even though the current approach is relatively exact it relies heavily on the fact that all areas has several detectable objects relatively evenly distributed. Randomly selecting a few voxels in the map and performing ray-tracing in their direction can be a more robust alternative.

Another approach which might be useful is training a detector for detecting doors, passages and similar boundaries. This might have limited uses in outdoor environments and would eventually struggle in a catastrophe area where much of the building has collapsed.

5.2 Summary

The focus of the thesis has been to design and implement a SLAM system, combining state-of-the-art sub-components to create a system which can handle large

environments in close to real-time. Even though it does not reach real-time performance in the current implementation, it is possible to reach that speed with further optimizations of the system. The structure and design of the system are possible to use in the scenarios the CENTAURO project intends to function; giving a relief for rescue-workers by making it possible to survey the environment for dangers before letting personnel enter the site.

Bibliography

[1] W. Burgard, O. Brock, and C. Stachniss. *Mapping Large Loops with a Single Hand-Held Camera*, pages 352–. MIT Press, 2008. ISBN 9780262255868. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6280107. Cited on pages 6 and 7.

[2] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):22:1–22:14, October 2008. ISSN 0098-3500. doi: 10.1145/1391989.1391995. URL http://doi.acm.org/10.1145/1391989.1391995. Cited on page 20.

[3] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18(3):137–145, January 1984. ISSN 0097-8930. doi: 10.1145/964965.808590. URL http://doi.acm.org/10.1145/964965.808590. Cited on page 34.

[4] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. doi: 10.1177/0278364908090961. URL http://ijr.sagepub.com/cgi/content/abstract/27/6/647. Cited on pages 6 and 7.

[5] M. Cummins and P. Newman. Highly scalable appearance-only slam - fab-map 2.0. *Robotics Science and Systems (RSS)*, 2009. URL http://www.roboticsproceedings.org/rss05/p39.pdf. Cited on pages 7, 28, and 33.

[6] M. Danelljan, G. Meneghetti, F. Khan, and M. Felsberg. A probabilistic framework for color-based point set registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL http://www.cvl.isy.liu.se/research/cogvis/colored-point-set-registration/index.html. Cited on page 7.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Re-

sults, 2007. URL http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. Cited on pages 8 and 34.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. URL http://www.cvlibs.net/datasets/kitti/eval_odometry.php. Cited on page 6.

[9] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2154–2161, Sept 2009. doi: 10.1109/ICCV.2009.5459471. URL http://ieeexplore.ieee.org/document/5459471/?arnumber=5459471. Cited on page 7.

[10] M. C. Graham, J. P. How, and D. E. Gustafson. Robust incremental slam with consistency-checking. *International Conference on Intelligent Robots and Systems (IROS)*, 2015. URL http://ieeexplore.ieee.org/document/7353363/. Cited on page 33.

[11] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. *Learning Rich Features from RGB-D Images for Object Detection and Segmentation*, pages 345–360. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10584-0. doi: 10.1007/978-3-319-10584-0_23. URL http://dx.doi.org/10.1007/978-3-319-10584-0_23. Cited on page 7.

[12] A. Harmat, M. Trentini, and I. Sharf. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments. *Journal of Intelligent and Robotic Systems*, 78(2):291–317, 2015. URL http://link.springer.com/article/10.1007/s10846-014-0085-y. Cited on page 6.

[13] J. Hedborg, A. Robinson, and M. Felsberg. Robust three-view triangulation done fast. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 152–157, June 2014. doi: 10.1109/CVPRW.2014.28. Cited on page 33.

[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi: 10.1007/s10514-012-9321-0. URL http://octomap.github.com. Software available at http://octomap.github.com. Cited on pages 8 and 14.

[15] M. Korn, M. Holzkothen, and J. Pauli. Color supported generalized-icp. *International Conference on Computer Cision Theory and Application (VISAPP)*, pages 592–599, 2014. URL http://www.is.uni-due.de/fileadmin/literatur/publikation/korn14visapp.pdf. Cited on pages 7 and 16.

[16] O. Kähler, V. A. Prisacariu, and D. W. Murray. Real-time large-scale dense 3d reconstruction with loop closure. *European Conference on Computer Vision (ECCV)*, 2016. Cited on pages 8 and 9.

[17] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442–1447 vol.3, Nov 1991. doi: 10.1109/IROS.1991.174711. Cited on page 5.

[18] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *UAV*, 2010. URL `http://rpg.ifi.uzh.ch/docs/UAV10_nuetzi.pdf`. Cited on page 6.

[19] M. Persson. Online monocular slam (rittums). Master's thesis, Linköping University, Department of Electrical engineering, CVL, 2013. URL `http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A771912&dswid=-7286`. Cited on pages 6 and 7.

[20] J. Redmon, S. Kumar Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL `http://arxiv.org/abs/1506.02640`. Cited on pages 7, 17, and 34.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. URL `https://arxiv.org/abs/1506.01497`. Cited on pages 7 and 8.

[22] R. Rocha, J. Dias, and A. Carvalho. Entropy-based 3-d mapping with teams of cooperative mobile robots: a simulation study. Technical report, Technical report, ISR, University of Coimbra, Portugal, 2004. URL `http://ap.isr.uc.pt/archive/33.pdf`. Cited on page 20.

[23] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardos. An image-to-map loop closing method for monocular SLAM. In *Proc. International Conference on Intelligent Robots and and Systems*, 2008. Cited on page 6.

[24] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular {SLAM}. *Robotics and Autonomous Systems*, 57(12):1188 – 1197, 2009. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2009.06.010. URL `http://www.sciencedirect.com/science/article/pii/S0921889009000876`. Inside Data Association. Cited on pages 6 and 7.

Index