Mobile Robots with Novel Environmental Sensors for Inspection of Disaster Sites with Low Visibility

Grant agreement no: 645101

Project start: January 1, 2015

Duration: 3.5 years

Deliverable 1.8

Software Toolkit - SLAM Toolkit for the navigation sensors

Due date: month 40 (April 2018)

Lead beneficiary: ORU

Dissemination Level: Public

Main Authors:

Malcolm Mielle (ORU)
Martin Magnusson (ORU)

Contents

1. Introduction

This document is a technical report within the scope of the Horizon 2020 project SmokeBot. In search and rescue scenarios, a robot needs to perform navigation in unknown environments. The SmokeBot project focuses on civil robots supporting fire brigades and aims to enhance the performance of robots under harsh conditions, for example: smoke, dust or fog.

This report provides an overview of the software toolkit developed in Task 1.5, "SLAM in Complex and Dynamic Environments". Task 1.5 is part of Work Package 1, "3D Structure Perception with Limited Vision", and aims at provoding structural maps, particularily in environments with low visibility. The SLAM system.

The software developed is based on the developments on using sketch maps and other rough priors as input from T6.1, the radar/lidar sensor fusion developed in T1.4 and the MPR radar hardware from T1.1, as well as ORU's previous work on robust mapping [3].

2. Radar and lidar SLAM

The mapping and localisation toolkit can use range information from radar, lidar, or both, using the sensor fusion algorithm detailed in D1.7.

As a baseline implementation, we initially used an off-the-shelf SLAM implementation: Gmapping.[1] Using the MPR, we found that Gmapping, which uses a Rao-Blackwellised particle filter together with occupancy grid maps for SLAM, produced results that were already usable for navigation in small but smoke-filled environments.

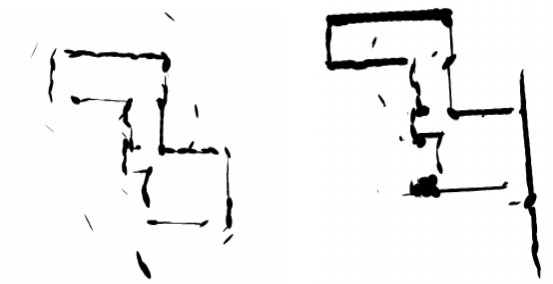
In later work within Task 1.5, we have performed a comparative analysis of radar and lidar sensing for localisation and mapping, in a paper submitted to the IROS 2018 conference [1]. In this paper, we present a quantitative comparison of the accuracy of radar (MPR) vs lidar (Velodyne VLP-16) sensing for indoor localization and mapping. We show that while producing slightly less accurate maps than a lidar, the radar can accurately perform SLAM and build a map of the environment, even including details such as corners and small walls.

We also evaluated the quality of mapping and localisation using the radar using two SLAM algorithms: Gmapping and NDT Fuser [3]. NDT Fuser is a pose tracking and mapping algorithm that, instead of an occupancy grid, uses the NDT map representation. With NDT, the map is represented which is a coarse grid where each cell stores a Gaussian fitted to the local surface shape, rather than having a fine grid with a single occupancy value per cell. In previous work, mapping and navigation algorithms using NDT maps have been shown to provide fast and accurate localisation with low memory requirements, compared to e.g. occupancy grids [4].

NDT fuser led to smoother trajectories and lower displacement in position overall. On the other hand, a feature-sparse corridor was shortened somewhat in the NDT maps, due to a loss of detail in the coarser map discretisation. Still, both SLAM algorithms obtained similar localization accuracy, with mean displacements of less than 0.04 m compared to the ground truth.

1 *http://wiki.ros.org/gmapping*

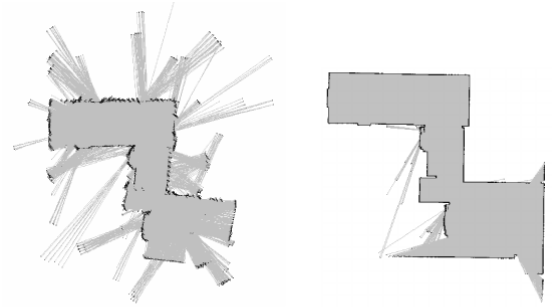
Our evaluation shows that the radar is a valid alternative to lidar sensing, if the situation calls for it. This is especially important to know when considering to use radars in high risk operations, such as disaster scenarios and emergency. Figures 1 and 2 show maps created by both a laser scanner and the radar, using two different SLAM algorithms. To conclude, with the RGT-V sensor suite it is possible to produce geometric maps also in low-visibility environments.



(a) NDT fuser map built by with the radar on the first run.

(b) NDT fuser map built by with the lidar on the first run.

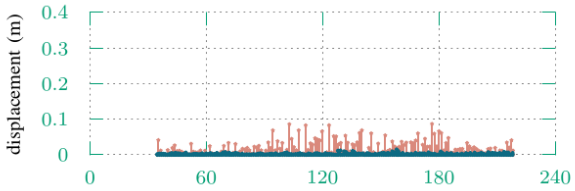
Figure 1: Comparison of radar and lidar maps built with NDT Fuser.



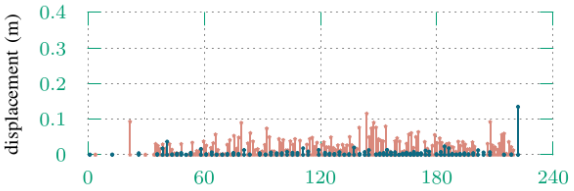
(a) Gmapping map built by with the radar on the first run.

(b) Gmapping map built by with the lidar on the first run.

Figure 2: Comparison of radar and lidar maps built with Gmapping.



(a) NDT fuser first run. At every step, the displacement is under 0.1 m for both sensors.



(b) Gmapping first run. Apart from one radar and one lidar update, the displacement stayed under 0.1 m for both sensors.

Figure 3: Absolute displacement in position (in meters) during the the mapping run shown in Figures Figure and Figure, for NDT Fuser (a) and Gmapping (b). The red lines show errors when using the radar, and the blue lines show errors when using the lidar. While having higher displacement than the lidar, the radar's displacement is generally under 0.1 m for both algorithms.

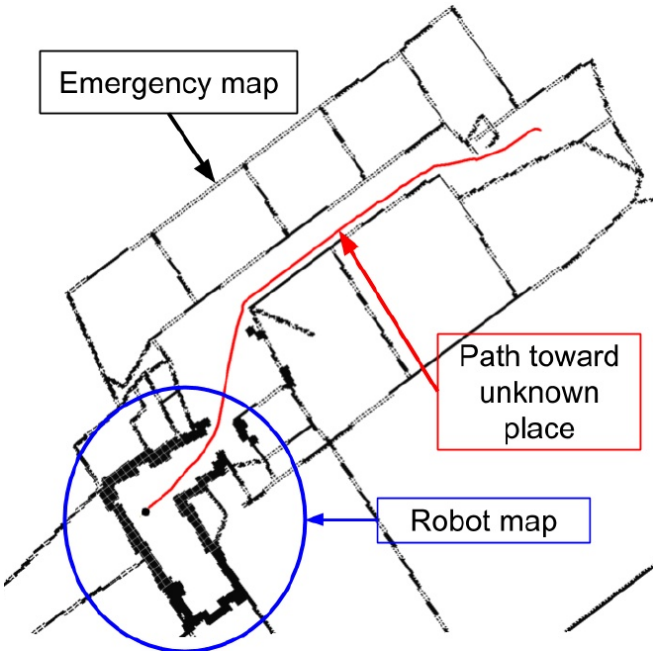


Figure 4: Illustrating the result of a prior emergency map that has been matched to a partial robot map, completing the robot map with unobserved information.

3. Graph-based SLAM with prior information

In addition to performing standard SLAM using the lidar and radar components of the RGT-V sensor suite, Smokebot's Task 1.5 also includes SLAM using prior information, where available. In Task 6.1 we have developed a SLAM system that integrates prior map information in the form of emergency maps for "auto-completing" the robot's map with yet unseen information [2], as shown in Figure 4..

For the final SLAM system for complex and dynamic environments, we also include a step that localises the robot by matching the range readings to the prior emergency map. This allows us to correct both errors in the prior map (by non-rigidly aligning the walls of the prior map to the map constructed by the robot) and the robot-built map (by using the structure of the prior map to potentially compensate for SLAM errors due to lack of observable features in a low-visibility scenario).

Prior information is often available before a robot starts exploring an environment. We have focused on using emergency maps since they are a support often used by firemen. However, emergency maps can be outdated and/or the scale can be locally wrong; sometimes on purpose, to facilitate map understanding. The system we developed takes in account those errors in the emergency map and corrects them using the robot measurements. In the meantime, the emergency map's topology is used to correct errors in the robot map, e.g. bent corridors due to drift.

In our previous implementation [2], we were integrating an emergency map in SLAM, using it to complete the robot mapping with unexplored parts of the map, we now use localisation in the emergency map to correct both errors in the prior and the robot-built map.

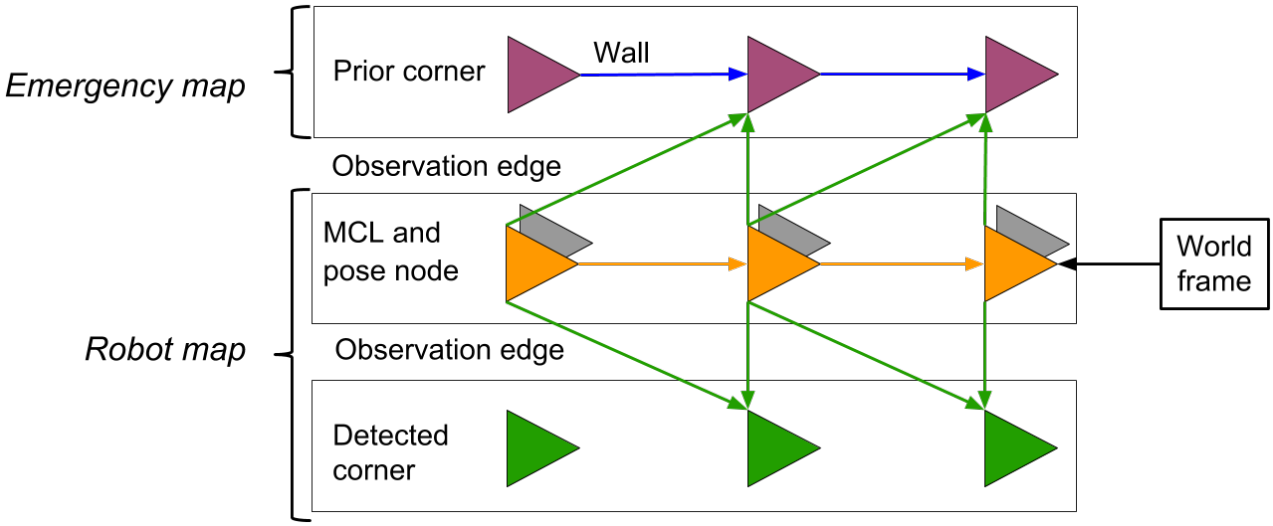


Figure 5: Graph-SLAM formulation

The final toolkit as per D1.8 uses a graph-SLAM formulation to optimize information from the robot map and the emergency map into one consistent representation. The robot builds sub-maps of the environment using its sensor data, while, at the same time, the same sensor data is used to localize in the emergency map using Monte-Carlo Localization (MCL).

The localization and its covariance are used to associate corners in the emergency map and the robot maps. For each robot pose associated with a sub-map, the equivalent MCL pose is saved. MCL poses can be considered as "where the robot thinks it is in the emergency map when considering the lidar input". Hence, the strategy to find associations between the emergency map and the robot map is based on the covariance, or "certainty" of the localization in the emergency map.

The following Section 3.a outlines the data association step of findng mathing corners in the two map representations, and Section 3.b contains information on how to run the software toolkit, and the parameters used. We are currently preparing a journal article submission presenting the approach used in the D1.8 toolkit in further detail, as well as the experimental results obtained.

3.a How to find associations between both map modalities

Figure 5 shows the graph structure that underlies the SLAM system. The MCL poses, the pose nodes, and the detected corners are created during the mapping process. To perform localization in the emergency map, the emergency map is first converted to a NDT map. Then, Monte-Carlo localization is used, using the laser scans as input and the NDT emergency map as a model map. This enables the robot to know where it is in both its environment using the mapping result, and in the emergency map using the localization result. Observation edges between the pose nodes and the emergency map corners are found using the following steps:

1. During the mapping process, every robot pose is associated with an equivalent MCL pose and a sub-map built using the sensors information. The MCL poses correspond to the pose of the robot in the emergency map, while the robot pose are the poses of the robot in its environment.

2. Every sub-map is moved from the robot pose to the equivalent MCL pose. Hence, every corner found in the sub-map will now be where they would be if seen by the robot at the MCL pose, i.e. the sub-maps are positioned where they would be if fitted on the emergency map according to the localization.

3. We score the likelihood that a detected corner corresponds to an emergency map corner by using the MCL pose covariance and the mahalanobis distance as follows:

   - We calculate the mahalanobis distance using the MCL covariance:

     $$mahalanobis\,distance = (pose\,prior - pose\,landmark) \circ (MCLcov^{-1}(pose\,prior - pose\,landmark))$$

   - We then calculate the probability that the corners are "colliding", i.e. the corners' similarity in position:

     $$probability = e^{mahalanobis\,distance}$$

   - This probability will be only slightly larger than zero if both corner poses "fit in the covariance". Hence we only need to look for a score slightly above zero to get a matching ; we use 5%.

4. If the prior and robot map corners match, we create an observation from the robot pose to emergency map corner equal to the observation from the robot pose to the matching robot map corner. The observation's covariance is the equivalent MCL pose covariance.

The information matrix used in the "wall" edges is designed such that the walls of the emergency map are free to extend or shrink during the back-end optimisation step, but it is costly to rotate them. The covariance of elements from the robot map is determined by the mapping process. Hence, the only parameters to manually set are the one controlling how much can the emergency map's walls be extended or shrinked in the optimization.

3.b How to run the algorithm

Two launch files need to be launch to run the code: one for the mapping and localization, and another for the graph-SLAM optimization. Here are the parameters of the launch files.

Mapping and localization parameters:

- use_mcl: if true, use MCL in the emergency map.
- use_graph_map_registration: if true, use NDT mapping.
- scale_gaussian_mcl: scaling factor for the Gaussian in MCL localization
- cell_neighborhood_size_mcl_in_meters: if MCL is allowed to use cells further away from the actual cell scan, the parameter is the size of the neighbor that MCL is allowed to search.
- use_euclidean_mcl: use the Euclidean distance as a measure for the cell fitness in MCL.
- use_mean_score_mcl: use the mean value of all cells within the neighborhood

- use_euclidean_for_long_distances: only use the Euclidean distance if the cell is further than the distance_euclid parameter.
- use_hybrid_strategy_mcl: use an hybrid strategy in MCL where either we find a cell corresponding exactly to the scan and we use it as is, or we consider the neighbor and use the mean of all cell fitness. This parameter forces the use of the euclidean distance.
- cov_x_mcl: starting cov along the x axis for the MCL.
- cov_y_mcl: starting cov along the y axis for the MCL.
- cov_yaw_mcl: starting cov along the yaw axis for the MCL.

Auto-complete Graph (ACG) parameters

- pause_for_testing: if true, the program will stop at every step.
- use_prior: if true, integrate the emergency map.
- optimize_prior: if true, the emergency map is optimized to fit the robot map.
- use_robot_maps: if true, the mapping given by the robot is used.
- use_corner: extract corners from the robot map.
- use_corner_orientation: use the corner orientation as a parameter for corner matching.
- corner_covariance: use an approximation of the corner covariance given the NDT used to find it.
- own_registration: register the sub-maps.
- world_frame: the world frame
- sensor_frame: the sensor frame
- gaussian_scaling_factor: scaling factor of the MCL covariance. Set to 1.
- threshold_score_link_creation: probability above which the corner are considered the same and matched. This should be only slightly above zero.
- prior_file: file localization for the emergency map image.
- max_deviation_corner_in_prior: minimum angle to extract a corner from the emergency map.
- scale: scale of the emergency map.

4. References

[1] Malcolm Mielle, Martin Magnusson, and Achim J. Lilienthal, "A comparative analysis of radar and lidar sensing for localization and mapping", in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018. *Under review.*

[2] Malcolm Mielle, Martin Magnusson, Henrik Andreasson, and Achim J. Lilienthal, "SLAM auto-complete: Completing a robot map using an emergency map", in IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), 2017.

[3] Todor Stoyanov, Jari Saarinen, Henrik Andreasson, Achim J Lilienthal, "Normal distributions transform occupancy map fusion: Simultaneous mapping and tracking in large scale dynamic environments", in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

[4] Jari Saarinen, Todor Stoyanov, Henrik Andreasson and Achim J. Lilienthal, "Fast 3D Mapping in Highly Dynamic Environments using Normal Distributions Transform Occupancy Maps", in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

SLAM auto-complete: completing a robot map using an emergency map

Malcolm Mielle, Martin Magnusson, Henrik Andreasson, Achim J. Lilienthal

***Abstract*— In search and rescue missions, time is an important factor; fast navigation and quickly acquiring situation awareness might be matters of life and death. Hence, the use of robots in such scenarios has been restricted by the time needed to explore and build a map. One way to speed up exploration and mapping is to reason about unknown parts of the environment using prior information. While previous research on using external priors for robot mapping mainly focused on accurate maps or aerial images, such data are not always possible to get, especially indoor. We focus on emergency maps as priors for robot mapping since they are easy to get and already extensively used by firemen in rescue missions. However, those maps can be outdated, information might be missing, and the scales of rooms are typically not consistent.**

We have developed a formulation of graph-based SLAM that incorporates information from an emergency map. The graph-SLAM is optimized using a combination of robust kernels, fusing the emergency map and the robot map into one map, even when faced with scale inaccuracies and inexact start poses.

We typically have more than 50% of wrong correspondences in the settings studied in this paper, and the method we propose correctly handles them. Experiments in an office environment show that we can handle up to 70% of wrong correspondences and still get the expected result. The robot can navigate and explore while taking into account places it has not yet seen. We demonstrate this in a test scenario and also show that the emergency map is enhanced by adding information not represented such as closed doors or new walls.

I. Introduction

In a search and rescue scenario, speed and efficiency are key. First responders need to locate victims, and get them out (alive) of the disaster site as quickly as they can. Each of those tasks usually takes up several hours, in harsh and dangerous conditions, putting the lives of first responders at risk. Autonomous robots could reduce the time spent by first responders on the disaster site, and speed-up operations. On the other hand, one does not want to spend much time in exploration and map building, two critical tasks needed by robots. One way to tackle this problem is to integrate prior information into Simultaneous Localization And Mapping (SLAM), to reason on still unknown parts of the environment

For indoor environments, emergency maps are probably the easiest prior maps to get. A use case is for firemen, who easily have access to emergency maps during their missions. However, the maps can be outdated, new changes to the building will not be represented, and the scale might not be uniform, to make the map easier to interpret.

Center of Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden. firstname.lastname@oru.se

This work was funded in part by the EU H2020 project SmokeBot (ICT-23-2014 645101) and by the Swedish Knowledge Foundation under contract number 20140220 (AIR)

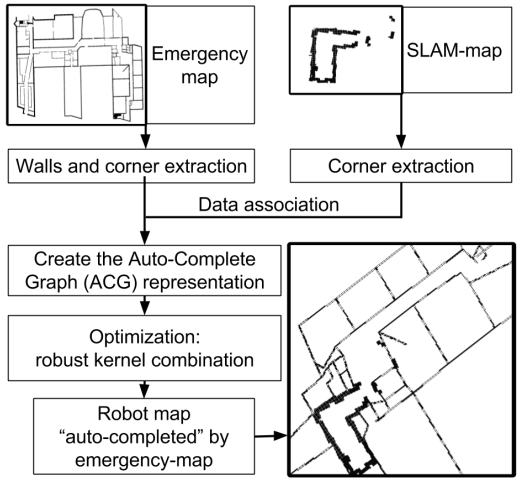


Fig. 1. Auto-complete process.

Previous works on SLAM with prior information focused on using either a topological map depicting objects [1] or a map representing an environment with no distortions or errors, e.g. aerial maps [2]–[4] or highly accurate models of the environment [5], [6]. While the methods mentioned above can deal with outdated data, they assume that the prior is a metrically accurate map. It is not straight forward to integrate emergency maps in SLAM using approaches from the literature [1]–[6] due to the non-uniform scale.

We aim at enhancing the map built during SLAM (i.e. SLAM map) by using information from an emergency map: the SLAM map is completed with information from the emergency map, and the emergency map inaccuracies are corrected using the sensors' measurements.

II. Method outline and contributions

We create a graph representation, that we call the auto-complete graph (ACG), fusing information from the SLAM map and the emergency map into one. Nodes in the ACG are the corners from the emergency map, the robot poses, and the corners in the SLAM map. Graph edges are added between connected corners in the emergency map (along the walls), between consecutive robot poses (from scan registration), between corners in the SLAM map and robot poses (observations edges), and between corresponding corners in both maps. The ACG is optimized, fusing information from the emergency and SLAM maps. The whole process is illustrated in Fig 1.

The contributions of the paper are :

- A formulation of graph-based SLAM that incorporates information from a rough prior map with uncertainties in scale and detail level.

- An optimization strategy adapted to the new graph formulation, based on a combination of robust kernels.

We also demonstrate a method for extracting corners in maps using the NDT (normal-distributions transform) representation.

Related works are covered in more detail in Section III. We show how to build the ACG in Section IV and how to optimize it in Section V. The method is tested and validated in Section VI and Section VII.

III. Related work

Vysotska and Stachniss [6] match maps obtained from OpenStreetMap onto the robot scans. They use the robot position and ICP [7] to introduce a correcting factor in the error function used in the graph-SLAM. The method allows up to $\pm 40^\circ$ of error in orientation between the map and the robot's heading for outdoor maps but can not correct an emergency map since its local scale may not correspond to the robot scans and ICP can not deform the map.

Parsley and Julier [5] integrate planes extracted from Ordnance Survey MasterMap[1] as constraints in a graph representation, use gating to remove planes that do not correspond to any planes in the scans, and match corresponding prior and SLAM planes using RANSAC. While the framework produces a more accurate map than SLAM without prior information, the gating is done using a distance metric, assuming uniform scale. Hence, correspondences between planes from the prior and the SLAM map are assumed to be correct. We assume that correspondences between the emergency and SLAM maps will introduce errors in the graph. Thus, we constrain the SLAM map using the prior while correcting the prior to complete the SLAM map.

Another idea is to consider the prior map as a topological representation of the environment. Shah and Campbell [1] use a human-provided map representing buildings to create waypoints using the Voronoi–Delaunay graph. The prior map is matched onto the robot map to estimate an affine transformation. Indoor, one could use objects, but annotated maps are harder to get than emergency maps.

Oßwald, Bennewitz, et al. [8] developed an exploration strategy that uses hand-made topo-metrical maps and the traveling salesman problem to find the most efficient global exploration path in the prior. Since they focus on the exploration method, there is no clear explanation on how to match the prior onto the SLAM map.

Skubic, et al. [9]–[11] use a sketch map interface for navigation. Objects are closed polygons drawn by the user and are described and matched to objects seen by the robot using histogram of forces. Their method can not be used on emergency maps with no objects represented.

In our previous work [12], we developed a method to find correspondences between a sketch map and a ground truth map. To interpret the sketch map, we use a Voronoi diagram, with a thinning parameter, that we extract as a graph and we use an efficient error-tolerant graph matching algorithm to find correspondences. However, the method cannot be used directly on a SLAM map due to their high level of noise.

Freksa, Moratz, et al. [13] use a schematic map to navigate a robot. They find correspondences between the schematic map and the environment by matching corners and tested their method on a simulated environment with three rooms. By definition, their prior map and data association are perfect, making the method too simple for emergency maps.

Kümmerle, Steder, et al. [4] used aerial maps as prior in a graph-based SLAM. They use edges in both modalities and Monte Carlo localization to find correspondences between stereo and three-dimensional range data, and the aerial images. However, they assume a constant scale in the aerial image, which we can not do with emergency maps.

Boniardi, Behzadian, et al. [14] present an approach for robot localization and navigation based on a hand-drawn sketch of the environment. They use an extension of the Monte Carlo localization algorithm to track the robot pose and approximate the deformation between the sketch map and the real-world using two scale factors. However, two scale factors will not correct local deformations of the emergency map.

IV. Auto-complete-graph construction

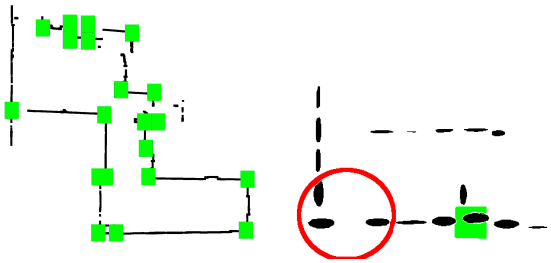
We developed a graph-based SLAM representation that incorporates information from an emergency map and SLAM map into one map, by using corners as common landmarks. Corners are easy to extract in emergency maps since walls are drawn clearly, making corners salient. We describe the method to extract elements from SLAM maps in Section IV-A, and from emergency maps in Section IV-B, before presenting the ACG formulation in Section IV-C.

A. Processing the SLAM map

We use NDT [15], [16] as the representation of the robot's map. NDT makes it easy to extract salient corners, and allows for efficient scan registration [17], [18], planning [19] and localization in both 2D and 3D. NDT is a grid-based representation where each cell stores a Gaussian distribution representing the shape of the local surface. While mapping an environment with a range scanner, an NDT map can be built incrementally by representing each scan as an NDT grid, then registering and fusing it with the previous map [16]. From this, to build the pose-graph representation used later to build the ACG, a partial NDT map is built iteratively until the robot goes further than a certain distance, at which point a new NDT map is started. Each pose where we started to build a partial NDT map is a pose-node in the graph, and possesses its corresponding partial NDT map as an attribute. More on this process is detailed in Section IV-C.1. Note that the approach presented in this paper does not hinge on using NDT as the representation of the robot map. Other representations could be used, as long as it is possible to extract salient corners.

To find corners in each NDT map, we analyze every cell that is occupied, i.e. each cell that has a Gaussian. A cell

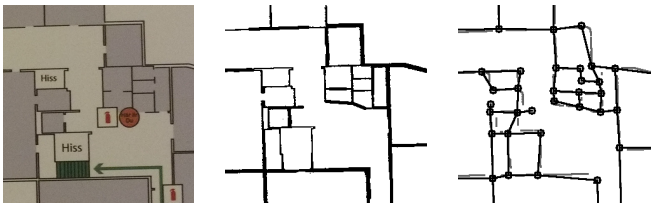
[1] https://www.ordnancesurvey.co.uk/business-and-government/products/mastermap-products.html



(a) Full NDT map with corners (in green) detected using a neighborhood size of two.

(b) When using a neighborhood search size of 1, a NDT-corner was undetected since it is next to an empty cell.

Fig. 2. A NDT map, with the Gaussians in black, and extracted corners, represented by green squares.



(a) Partial view of an emergency map from a larger picture, taken by a 20MP phone camera.

(b) Emergency map with only the walls represented. There are some deformations due to icons removal.

(c) Graph extracted from the prior.

Fig. 3. Pre (Fig 3a) and post processed emergency map (Fig 3b and Fig 3c). Because of the scale uncertainties and artifacts due to symbols, the graph extracted is not an accurate representation of the environment.

is a corner if: 1) it has more than one neighbor with a Gaussian, 2) the main eigenvector of two of those neighbors' Gaussian's form an angle between 80 and 100 degrees. By calculating the rays' collision point, the estimated position of the corner is determined. However, this method depends on the size of the neighborhood observed at every point. Some corners can be overlooked if one considers a small neighborhood that does not have enough measurements, as seen in Fig 2b. In our work, the neighborhood size considered is 2. A resulting NDT map with detected corners can be seen in Fig 2a.

B. Processing the emergency map

Emergency maps possess few features, as in Fig 3a. The maps usually represent walls and some elements, such as the position of extinguishers, stairs, or toilets. We removed manually those extra elements as they might not be consistently represented between different emergency maps [20].

A corner in the emergency map is, either a place where the line orientation abruptly deviates with an angle of 45° or more, or a place where a line splits into multiple lines, i.e a crossing. Using a line follower algorithm, all corners and the lines between them are extracted from the emergency map, as in Fig 3c. The line follower is able to take into account uneven line thickness of the walls in the thresholded image of the emergency map.

C. Graph formulation

The ACG is built using the information extracted previously from the SLAM and emergency maps. The structure

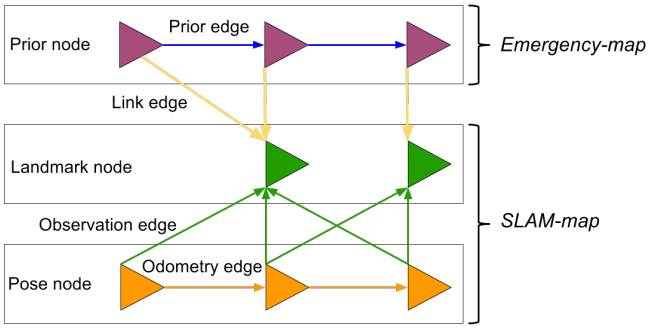


Fig. 4. The ACG structure fusing information from the SLAM and emergency maps. Prior-nodes are in purple, prior-edges in blue, pose-nodes and odometry-edge in orange, landmarks and their observations in green, and link-edges in yellow. We will use this color code in all pictures.

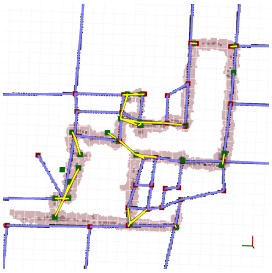
of the ACG can be seen in Fig 4.

1) Elements from the NDT maps: robot poses are added as pose-nodes in the ACG, while odometry-edges are the registrations between partial NDT maps associated to each consecutive pose-node. Corners extracted from the NDT maps are landmark-nodes in the ACG. The observation measurements between landmarks and the pose-nodes from where they were detected are observation-edges. Since the partial NDT maps associated with each pose node are rigid, observation-edges must reflect this. Hence, their covariances' standard deviation is $\sqrt{0.05}$ m for the $\vec{x}$ and $\vec{y}$ axis, to only allow for small movements of landmark-nodes around their respective pose-node.

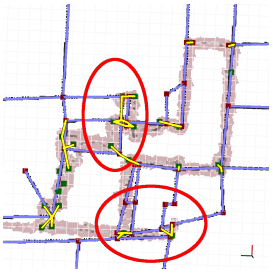
2) Elements from the emergency map: the corners from the emergency map are prior-nodes in the graph and walls between them are prior-edges. We initialize the prior-nodes' positions using a transformation given by two known equivalent points in both maps. Prior-edges' lengths are computed from the distance between prior-nodes.

While emergency maps have uncertainties in scale and proportion, they are structurally consistent and we aim at preserving that consistency where needed: prior-edges (i.e walls in the emergency map) should be hard to rotate but easy to stretch or shrink. Thus, each prior-edge has a covariance with a high value along its main axis but a small one on the perpendicular axis. The covariance matrix can be defined with $\Sigma V = VL$ where Σ is the covariance matrix, V is the matrix whose columns are the eigenvectors of Σ and L is the diagonal matrix whose non-zero elements are the corresponding eigenvalues. By computing $\Sigma = VLV^{-1}$ we obtain the desired translation-covariance. To align the covariance with the edge axis, we define the first eigenvector as the direction of the edge, while the second eigenvector is a perpendicular vector. We associate a high eigenvalue to the first eigenvector and a small eigenvalue to the second one. The calculation of the first eigenvalue depends on the original, non-optimized, length of the prior-edge; we experimented with different values in Section VI-B. The second eigenvalue is manually set to a small value: 0.005.

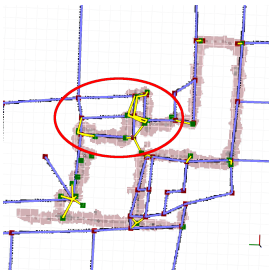
3) Matching the corners: to associate each corner extracted from the NDT maps with potential correspondences in the prior map, an edge between every landmark-node



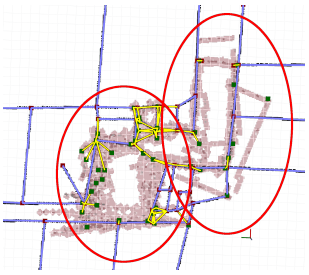
(a) Optimization result when using a Huber kernel first followed by DCS. The emergency map is fitted onto the SLAM map.



(b) Result of the optimization when no robust kernel is used. Large deformations due to incorrect corner association are visible in the red ellipses.



(c) Optimization result when only a Huber kernel is used. Deformations due to incorrect corner association are still visible in the red ellipses.



(d) Optimization result when only DCS is used. The partial NDT maps have been moved and don't fit together.

Fig. 5. Different results of the optimization using different optimization strategies. Link-edges are in yellow, prior-edges in blue and the SLAM map is in light brown.

and every prior-node is added to the ACG, if the distance between them is less than a certain threshold. Those link-edges represent a zero transformation and their covariance's standard deviation is set to $\sqrt{0.5}$ m on the $\vec{x}$ and $\vec{y}$ axis to account for possible noise. Effectively, the linked corners are allowed to move uniformly around each other, but they can't go far from each other without increasing cost.

V. Optimization and SLAM back-end

Now that we have a graph representation of the environment, the next step is to optimize it to complete the SLAM map with not yet seen parts of the environment. With each new pose-node added to the ACG, we find all possible correspondences between prior and landmark-nodes, before running 10 optimization iterations with a Huber kernel, followed by 20 iterations with Dynamic Covariance Scaling [21], [22] (DCS) to increase robustness to the very high number of outlier edges. The Huber kernel is a parabola in the vicinity of zero and increases linearly at a given level $j * j > k$. Its cost-function is as follow:

$$\rho(x) = \begin{cases} \frac{x^2}{2}, & \text{if } |x| \leq k \\ k(|x| - \frac{k}{2}), & \text{otherwise} \end{cases} \quad (1)$$

While this kernel guarantees unicity of the solution since it is a convergent ρ-function, the result is still influenced by incorrect link-edges between non-corresponding corners (Fig 5c). To remove the effect of remaining incorrect link-edges, we use DCS to dynamically scale down the information matrices in edges that introduce a large error in the graph.

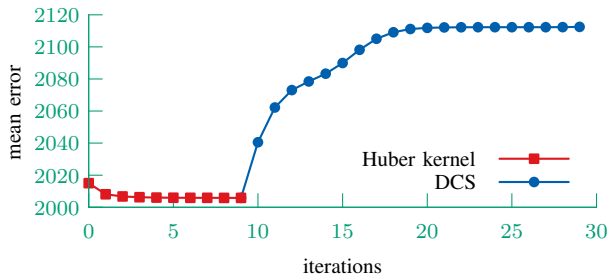


Fig. 6. Convergence plot for our optimization strategy: the mean error is calculated over 25 runs. For each run, optimization with the Huber kernel reaches stability at around 5 iterations, while DCS does after 13 additional iterations. DCS allows some link edges to move further from their mean value so the total error plotted increases after optimization

VI. Parameter evaluation

We evaluated our method on Örebro University's dataset [2], using a picture of the emergency map, taken by a phone, as the prior (see partial image in Fig 3a). A public implementation is available online [3].

A. Influence of the robust kernel on the optimization

We evaluate the influence of the robust kernels and confirm that using a combination of a Huber Kernel and DCS is a good optimization strategy.

In Fig 5a, one can see the result of the optimization when using a Huber kernel for 10 iterations first, followed by 20 iterations with DCS: the emergency map is fitted onto the SLAM map and we obtain the expected result. Fig 6 shows the mean error value at each iteration step, over 28 runs. One can see that the Huber kernel reaches stability in around 5 iterations and DCS reaches stability in around 13 iterations.

With no robust kernel, the emergency map is not correctly fitted on the SLAM map: rooms from the prior are deformed in a non-realistic way, as can be seen in Fig 5b in the red ellipses. The emergency map is better fitted when using a Huber kernel, as in Fig 5c, but still does not represent reality. When using only DCS, the SLAM map is corrupted since the error of some edges is scaled down without first converging toward an approximate solution (Fig 5d). The rough scale of the emergency map introduces high errors in some link-edges and, with more than 50% of wrong link-edges, DCS alone can not converge toward the optimal solution.

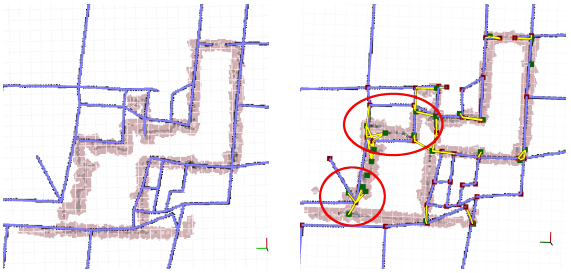
B. First eigenvalue of prior-edges' covariance.

We evaluate the influence of the eigenvalue corresponding to the eigenvector along each prior-edge (discussed in Section IV-C.2). This parameter controls how likely prior-edges are to extend or shrink.

As seen in Fig 7b, using 1% of the edge's length leads to the prior map not being changed but translated and rotated to fit the SLAM map, without correcting its inaccuracies.

[2] http://wiki.ros.org/perception_oru/Tutorials/Using%20NDT%20Fuser%20to%20create%20an%20NDT%20map

[3] https://github.com/MalcolmMielle/Auto-Complete-Graph



(a) No optimization. The SLAM map is simply drawn over the prior.

(b) Using 1% of a prior-edge length to calculate its covariance. Red circles: the prior is too rigid to fit the SLAM map.

(c) Using 50% of a prior-edge length to calculate its covariance. The prior and SLAM map are correctly fitted together.

(d) Using 100% of a prior-edge length to calculate its covariance. The prior was not rigid enough and is uncorrectly fitted on the SLAM map due to wrong link-edges.

Fig. 7. Results of the optimization with different percentage of the egde's length used for the prior's covariance.

Using 100% of the edge's length gives too much flexibility to the prior-edges and leads the ACG to a suboptimal solution, where walls from the emergency map don't match equivalent walls in the prior map, as can be seen in the red circles in Fig 7d. Using 50% gives the best result, since the prior is fitted onto the SLAM map, and incorrect corner correspondences do not influence the result, as seen in Fig 7c.

C. Maximum number of outlier link-edges

We estimated the maximum amount of wrong link-edges we can have in the graph before the optimization fails. We ran 26 optimizations with added noise on the initial poses, totaling 13 successes and 13 failures, where a failure means that at least one prior landmark is not at the correct position in the SLAM map. The percentages of outliers for each optimization are presented in Fig 8.

To confirm that the number of outliers is correlated with the success of the optimization, we first test if the mean of the success sample is significantly less than the mean of the failures sample. The z-score of the max and min of the percentage of outliers for success and failure samples are within a 3σ event and we can assume normality. We ran Welsch's t-test, which does not assume equality of variances. The sign of the t-statistic is important since we are testing a "less than" hypothesis of a one-tailed t-test. If $t < 0$ and $p_{value}/2 < 0.05$, we can reject the hypothesis that both distributions are similar. With our sample, $t = -4.756$ and the $p_{value} = 8.377 \cdot 10^{-5}$, showing a significant

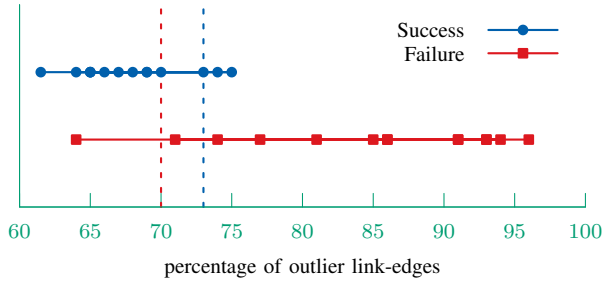
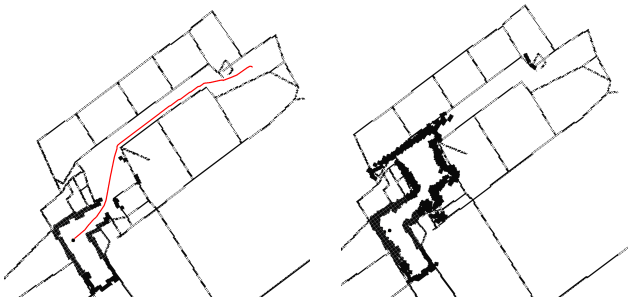


Fig. 8. Success cases are in blue, and failure ones in red. The blue vertical bar is 1.5 standard deviation above the success mean, and the red one is 1.5 standard deviation under the failure mean.



(a) The robot found a path to a room in the emergency map that was not yet visited by the robot.

(b) The robot added a door that wasn't closed in the emergency map and the path can not be found anymore.

Fig. 9. Searching for a path at different exploration times.

difference between the two distributions. For the failure case, 1.5 standard deviations under the mean represents 70% of outliers, meaning 86% of the failure cases have more than 70% of outliers. Thus, our method can usually handle up to 70% outliers. Anecdotically, the optimization can fail under 70%, like one case in Fig 8 but it should be noted that the final result only had 2 nodes over 13 wrongly matched and was very close to a success case. For the success case, 1.5 standard deviations above the mean is equivalent to 73%, which means that 86% of the success cases have under 73% of outliers. The optimization is likely to fail if there is more than 73% of outliers, however, it can occasionally succeed.

VII. Real Navigation Scenario

We tested the ACG in a real navigation scenario. The test was conducted in the basement of Örebro University on the Taurob platform[4], using a Velodyne and the same prior as before. We used 2 m as the minimum distance between two corners before creating a link-edge, to get less than 71% of outliers as seen in Section VI-C, and 50% as the eigenvalue for the prior-edges' covariances. The process runs in real-time on an Intel core i7 CPU at 2.30GHz, in about 1.3 seconds. Hence, the total time is slower than the time needed to create a partial NDT map.

To create an occupancy grid from the auto-complete graph, we fused all partial NDT maps positioned on their relative pose-node in one occupancy grid. Then, we draw the prior map by adding occupied cells where prior-edges are. Using this map, it was possible to find paths to places the robot had

[4]http://taurob.com

not yet explored. In Fig 9a, the robot has only been exploring for a short time but can find a path toward its destination, even though it has not explored that part of the environment yet. In Fig 9b, the robot has collected more information and now knows that the first path it found toward its goal is not practicable. A door is closed and the robot can not pass through it.

VIII. Future work

In the future, we plan on investigating improved correspondences between the emergency map and the SLAM map by introducing corner orientation. Also, we will evaluate our method using other SLAM datasets for which emergency maps, or other rough prior maps, are available.

IX. Conclusion

We developed a formulation of graph-based SLAM, incorporating information from a rough prior, which has uncertainties in scale and detail level. We also presented an optimization strategy adapted to this new graph formulation.

We use corners, in the prior and SLAM maps, as a common element to find correspondences, and create a graph fusing information from both modalities. To obtain robust optimization results, we first use a Huber kernel followed by DCS, allowing up to 70% of wrong correspondences.

Contrary to other works [1]–[6], we do not use the prior map to bind the SLAM map. We match the prior map onto the SLAM map to complete missing information and unexplored areas, while accounting for the uncertainties of the prior. Experiments showed that the SLAM map completed with information from the emergency map enables the robot to navigate and plan the exploration while taking in account non-explored places.

References

[1] D. C. Shah and M. E. Campbell, "A qualitative path planner for robot navigation using human-provided maps," *The International Journal of Robotics Research*, vol. 32, no. 13, pp. 1517–1535, Nov. 1, 2013, ISSN: 0278-3649, 1741-3176.

[2] M. Persson, T. Duckett, and A. J. Lilienthal, "Fusion of aerial images and sensor data from a ground vehicle for improved semantic mapping," *Robotics and Autonomous Systems*, From Sensors to Human Spatial Concepts, vol. 56, no. 6, pp. 483–492, Jun. 30, 2008, ISSN: 0921-8890.

[3] M. P. Parsley and S. J. Julier, "Towards the exploitation of prior information in SLAM," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 2991–2996.

[4] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based SLAM using aerial images as prior information," *Autonomous Robots*, vol. 30, no. 1, pp. 25–39, Jan. 2011, ISSN: 0929-5593, 1573-7527.

[5] M. P. Parsley and S. J. Julier, "Exploiting prior information in GraphSLAM," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2638–2643.

[6] O. Vysotska and C. Stachniss, "Exploiting building information from publicly available maps in graph-based SLAM," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 4511–4516.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992, ISSN: 0162-8828.

[8] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss, "Speeding-up robot exploration by exploiting background information," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 716–723, Jul. 2016, ISSN: 2377-3766.

[9] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, and A. Schultz, "Using a hand-drawn sketch to control a team of robots," *Autonomous Robots*, vol. 22, no. 4, pp. 399–410, Mar. 20, 2007, ISSN: 0929-5593, 1573-7527.

[10] M. Skubic, C. Bailey, and G. Chronis, "A sketch interface for mobile robots," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 1, IEEE, 2003, pp. 919–924.

[11] G. Parekh, M. Skubic, O. Sjahputera, and J. M. Keller, "Scene matching between a map and a hand drawn sketch using spatial relations," in *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, 2007, pp. 4007–4012.

[12] M. Mielle, M. Magnusson, and A. J. Lilienthal, "Using sketch-maps for robot navigation: Interpretation and matching," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct. 2016, pp. 252–257.

[13] C. Freksa, R. Moratz, and T. Barkowsky, "Schematic maps for robot navigation," in *Spatial Cognition II*, DOI: 10.1007/3-540-45460-8_8, Springer, Berlin, Heidelberg, 2000, pp. 100–114, ISBN: 10.1007/3-540-45460-8_8.

[14] F. Boniardi, B. Behzadian, W. Burgard, and G. D. Tipaldi, "Robot navigation in hand-drawn sketched maps," in *Proceedings of the European Conference on Mobile Robotics (ECMR)*, 2015.

[15] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, IEEE, 2003, pp. 2743–2748.

[16] T. Stoyanov, J. Saarinen, H. Andreasson, and A. J. Lilienthal, "Normal distributions transform occupancy map fusion: Simultaneous mapping and tracking in large scale dynamic environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 4702–4708.

[17] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-NDT," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[18] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, and A. Birk, "Beyond points: Evaluating recent 3d scan-matching algorithms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 3631–3637.

[19] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Path planning in 3d environments using the normal distributions transform," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 3263–3268.

[20] U. J. Dymon, "An analysis of emergency map symbology," *International Journal of Emergency Management*, vol. 1, no. 3, pp. 227–237, 2003.

[21] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 62–69.

[22] P. Agarwal, "Robust graph-based localization and mapping," PhD thesis, PhD thesis, University of Freiburg, Germany, 2015.