6.
Conclusions

This work explored the main SLAM solutions, given a detailed example for implementing the EKF-SLAM and FastSLAM; although the adopted solution, for the proposed application, was the DP-SLAM.

Thus, to perform SLAM without odometry information, the following algorithms were implemented:

- A simulator to acquire 3D data from structured environments; implemented on MatLab® platform.
- A scan matching algorithm, to get robot displacement without odometry information, based in genetic algorithm optimization for the Normal Distribution Transform; also implemented on MatLab® platform, using the Differential Evolution library acquired from reference [22].
- A modified DP-SLAM, which included a new motion model: the scan matching motion model; implemented on Visual C++ platform.

This method can deal as well with 3D environments, as long as the LRF is mounted on a rotating platform on the mobile robot, to allow both horizontal and vertical scans. As the robot moves, the LRF acquires 2D scans. Where and when 3D data is required, the robot stops and the platform rotates synchronized with the LRF to perform further 2D scans on different plane orientations. After traveling through a desired path, the acquired data is processed to obtain 2D or 3D maps.

In this way, the main contribution of this thesis is the application of DP-SLAM without the need for odometry information, just by using a single LRF. Furthermore, this comes with some considerations and advices, along with specific conclusions, detailed next.

6.1. DP-SLAM Conclusions

DP-SLAM is a powerful tool in 2D mapping so that, it doesn't need a separated loop detection algorithm.

Such a high performance comes with a high computational cost. Thus, DP-SLAM is an off-line mapping algorithm, due to the huge data it manages. The smallest experiment ("Mine") took about 1 hour to be evaluated and the longest ("Diiga") took about 5 hours, using a processor AMD Turion 64 with 2.1GHz. The time complexity of DP-SLAM is analyzed in detail in [6].

The Motion Model in DP-SLAM has a considerable influence in the mapping performance. Poor models will need a huge number of particles to obtain moderate results. Thus, the Scan Matching Motion Model proposed in this work could be improved to include holonomic robots, so that they can manage both Δx displacement errors as well as in Δy errors.

The "low map" size in LSLAM ("piece of data" in Figure 4.19) as proposed by [6] is a function of the number of iterations. Thus, [6] proposes between 75 to 150 iterations. But in the analysis in this thesis, these values didn't give good results. Basically, 150 iterations don't guarantee that the robot travels enough distance to be considered a low map. The robot could get hundreds of scans without leaving the same room. In all experiments made by DP-SLAM in this work, a low map is only created when robot travels about 3m in an approximately straight path. It was observed that, when a low map is created (the LSLAM algorithm finishes) within a fast robot rotation, the next low map (that uses a portion of the previous map) suffers major ambiguities and tends to produces poor particles.

In LSLAM, DP-SLAM uses the current map for particle weighing. The best particle is used to grow the map, adding the new observation. If the robot displacement is higher than the LRF's maximum range, then DP-SLAM does not have enough pieces of the map for particle weighing. Thus, it is recommended to have a LRF with maximum range large enough to avoid this undesired situation.

As a way of demonstrating the proposed method performance, the simulated map (Figure 5.8) is compared with the acquired map using DP-SLAM (Figure 5.24). Thus, in order to compare them, the contours (walls) of both maps are selected and then converted to bitmaps. In this case the pixel to pixel comparison gives 93.34% of similarity, this mean 6.66% of wrong pixels.

6.2. Scan Matching Conclusions

The time of convergence in DE optimization depends on population size and generation number. Population size of 100 with 50 generations takes approximately 18s using a processor AMD Turion 64 with 2.1GHz. This relatively long time, coupled with time consumed by DP-SLAM algorithm, make the proposed method an offline but robust solution.

Errors in Scan Matching, the first or any other, are outweighed by the motion model proposed in Section 5.2. Note also that the first scan or any other have the same level of reliability; thus, if the first scan, and consequently the first grid map, is not good, it can be improved by the subsequent iterations (subsequent scans), in the sense of a probabilistic fusion.

Misalignments found in scan matching (for Δy displacement) are due to the cell size defined in Section 2.3.3. I.e. small details in the environment are blurred in NDT representation, which leads to unclear limits in alignment. One solution would be to reduce the cell size, but this creates an NDT representation with sharp shape; this case, DE optimization would need a quite larger population to achieve convergence. However, with some considerations, the cell size could be dynamically defined, depending on the measured LRF values. Thus, if LRF returns small distances, for instance less than 3m, then the cell size would be reduced to 500 mm. For higher distances, 1m of cell size would be a good choice. This idea was not implemented in this thesis, but it suggests future works to determine the distance threshold for each cell size choice.

The LRF's maximum range, as in DP-SLAM, is a very important parameter in scan matching. Larger maxima mean more data (about the environment) in one scan, which helps to improve the alignment.

The scan matching proposed was implemented in Matlab®, while DP-SLAM was implemented in C++ (using the code from [8]). It is possible, of course, to use the same computer language for Scan Matching and DP-SLAM. To increase the algorithm speed, it is recommended to implement the scan matching in C++ as well. A powerful library for evolutionary computing, GCOM [37], can help in such task. This is a suggestion for future works.

There are solutions for 3D scan matching in non-planar terrain [38], [39] and [40]. Thus, they could be used as a basis to extend the proposed method to a 3D mapping in uneven terrain, using as well a single LRF without odometry information.

6.3. 3D Mapping Conclusions

The 3D maps presented in Section 5.4 use point cloud representation. Thus, the quality of these maps depends on LRF error, because they are simply plotted using the 2D trajectory given by DP-SLAM. However, DP-SLAM fuses probabilistic observations made for each cell, so, as the cell is more often observed as occupied, the darker it becomes. This idea could be applied in 3D representations as well. Thus, 3D observation may also be fused, similarly to what is done in 2D. Note that this does not mean using a 3D cell for localization or perception model: this means only using the 3D cells to print a 3D map output by fusing the available observations made at each 3D cell. This idea, however, was not implemented in this thesis but it is another suggestion for future works.

Another representation for 3D maps could be in the form of geometric planes. The Delaunay triangulation is a popular algorithm to create a simplified representation of the environment in the form of triangles acquired from a point cloud. There are commercial and open source software [41] specialized in processing this kind of data.