

**Theory and Technology of Robotics**

**Course Project:**

***Implement Simultaneous Localization and Mapping (SLAM) with MATLAB***

**Group members:**

**Run He, Rui Qi, Yuchen Song, Ruoyao Tian, Jialong Zeng**

**Spring 2022**

**28/04/2022**

Contents

[1. Introduction 1](#_Toc102390441)

[1.1 Background and results 1](#_Toc102390442)

[1.2 Problem restatement 1](#_Toc102390443)

[2. Problem Formulation 1](#_Toc102390444)

[2.1 Problem assumptions. 1](#_Toc102390445)

[2.2 Dataset Introduction 2](#_Toc102390446)

[3. Methodology 2](#_Toc102390447)

[3.1 Work Flow Overview 2](#_Toc102390448)

[3.2 Implementation Details 2](#_Toc102390449)

[a) Configuration of LIDAR Parameters 2](#_Toc102390450)

[b) Data Loading and Preprocessing 3](#_Toc102390451)

[c) Relationship between Local coordinate and global coordinate 3](#_Toc102390452)

[d) Data initialization 4](#_Toc102390453)

[e) Transformation from Local Coordinate to Global Coordinate 4](#_Toc102390454)

[f) Creation of Grid Map 4](#_Toc102390455)

[g) Rough Estimation for pose 5](#_Toc102390456)

[h) Scan to Scan Match 5](#_Toc102390457)

[i) Add a Key Scan 6](#_Toc102390458)

[j) Loop Closure Optimization 7](#_Toc102390459)

[k) Plotting 8](#_Toc102390460)

[4. Results and Analysis 8](#_Toc102390461)

[4.1 Impact of close loop detection 8](#_Toc102390462)

[4.2 Impact of occupancy grid map resolution 10](#_Toc102390463)

[5. Conclusion 11](#_Toc102390464)

[Reference 11](#_Toc102390465)

1. **Introduction**
   1. Background and results

In order to complete tasks safely and independently, a robot must realize its position in the environment. This problem can be viewed as simultaneous localization and mapping (SLAM), which is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. SLAM has become hot issues of mobile robot research in recent years, and it has also become one of the core processes to realize fully autonomous robot.

This paper introduces and implements a point nets map construction algorithm based on 2D Lidar scan data, we introduce Loop Closure Optimization, which optimizes the pose while building the map and predicting the pose sequence of the mobile robot to reduce the cumulative error in the modeling process[1]. We adjusted the values of some key parameters and analyze their overall impact on the map constructed by the algorithm.

* 1. Problem restatement

We have a set of 2-D lidar scans collected in a real environment (Deutsches Museum, each batch of scan is considered as data frames), and we will solve the following problems based on this dataset and prescribed key parameters of the lidar from which the data was collected:

1. Estimate and optimize the pose of the mobile robot when acquiring the data frame.
2. Calculate the occupancy grid coordinates of the global occupancy grid map based on the collected data frame and the pose estimation obtained from 1).
3. Derive a global map based on the pose estimation sequence and the global occupancy grid map coordinates.
4. **Problem Formulation**
   1. Problem assumptions.
5. The size of the mobile robot is negligible and is treated as a point in the model
6. The pose changes (Δx, Δy, Δθ) of the mobile robot in adjacent data frames are generally regarded as approximately consistent.
7. Insufficient consideration of the pose transformation capability of mobile robot between adjacent data frames.
8. The probe rotation time of the data frame obtained by the lidar scanning is negligible, i.e., each set of scanning data is collected at the same time.
9. There were no moving obstacles in the environment when data is collected.
10. The lidar did not malfunction when collecting data, and did not collect extremely anomalous data points.
    1. Dataset Introduction

The map data we use is “2D Cartographer Backpack – Deutsches Museum”[2]. It was collected using a 2D LIDAR backpack. Each bag contains data from an IMU which is on a horizontal LIDAR.

We download the .bag file and transform it as a .mat file, which is more convenient for us to use MATLAB for simulation. The dataset consisting of multiple batches of LIDAR scanning data, including the ranges and angles measured for each laser beam, as shown in equation x.

1. **Methodology**
   1. Work Flow Overview

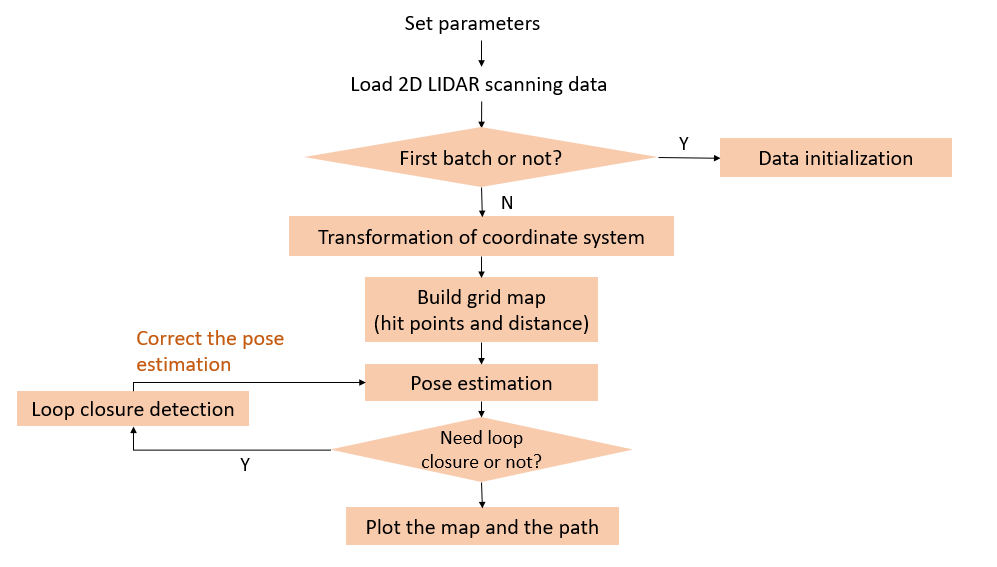


Figure 1. Work flow of our program

* 1. Implementation Details

1. Configuration of LIDAR Parameters

Before performing SLAM, we need to configure the sensor parameters of 2D LIDAR at first. They are shown in details in table 1. These parameters must match the LIDAR which is used in our dataset, otherwise, the dataset will be meaningless.

Table 1. Sensor parameters of 2D LIDAR

|  |  |
| --- | --- |
|  |  |
|  | Total number of laser beams emitted by the scanning |
|  | Time for one scan |
|  | Time increment |
|  | Minimum scanning range |
|  | Maximum scanning range |
|  | Minimum scanning angle |
|  | Maximum scanning angle |
|  | Angle increment between two consecutive scanning laser beams |

1. Data Loading and Preprocessing

Mobile robot processes Lidar scan data in batches. The form of the original scan data for batch is as follows:

In this simulation project, not all original scan data is reliable. We set the upper limit of the available scan data’s range and filter out available scan data based on the threshold. Equation below shows the data preprocessing method.

Then we convert the available data from polar coordinates to Cartesian coordinates.

1. Relationship between Local coordinate and global coordinate

We take the initial position of the mobile robot as the origin, the horizontal direction as axis, and vertical direction as axis. This forms the **global coordinate system**. The robot’s initial pose is considered as the pose in the global coordinate system.

When the robot is doing the scan, assume that the rotation angle of the mobile robot relative to the global coordinate system is , and the direction vector between the robot and the origin of the global coordinate system is . Take the current position of the robot as the origin, the direction of as axis. This forms the **local coordinate system**. Relationship between these two coordinate systems is shown in figure 2.

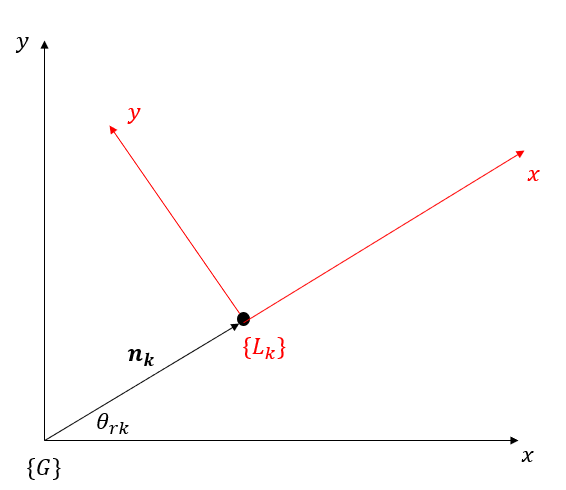


Figure 2. Relationship between the two coordinate systems

1. Data initialization

In this project, the initial position of the robot is set at the origin of the global coordinate system, and its initial pose is considered as the pose in the global coordinate system. Therefore, we directly add the first batch available scan data to the global map as a result of the first data processing.

1. Transformation from Local Coordinate to Global Coordinate

After the first batch, the robot should transform the available data from local coordinates to global coordinates according to its current pose, and then does the pose estimation.

We assume that when the robot begins its scan, the pose of it in global coordinate system is , the local coordinate of the scan point of is , the global coordinate of the scan point is .

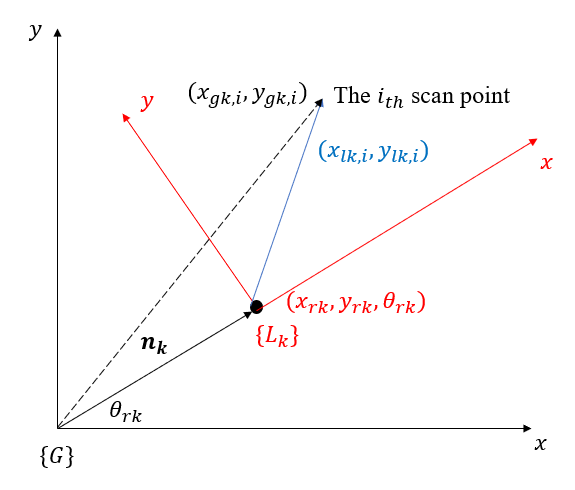


Figure 3. Demonstration of relative pose

According to the concept of relative pose in robotics theory, the translation matrix is written as follows:

By using , we can calculate the global coordinates of available data .

1. Creation of Grid Map

We also need to construct grid map for pose optimization, which will be used to adjust the current pose.

Firstly, we get the global coordinate of scan data surrounding the current pose. Secondly, we find the left-up corner and the right-down corner of the point set. Then we denote them as vectors. Based on these two vectors and a specific resolution, the grid map size can be calculated.

Next, the coordinates of the scan data will be transformed from global coordinates to grid coordinates. They are the **hit points** defined by us of the scan.

In grid map of point occurrence, we assign the cells with grid coordinates of the hit points with. For each cell in grid map, calculate the distance to its nearest hit point. Finally, we get the distance map of the pose. Figure 4 shows the grid map.

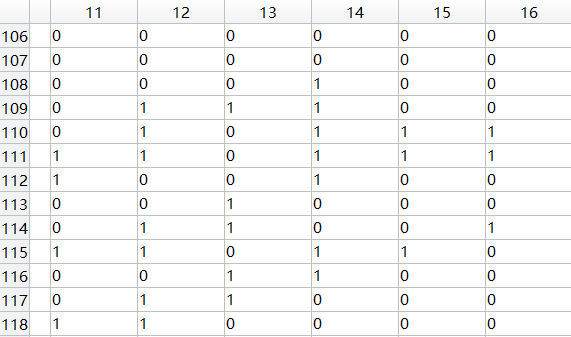
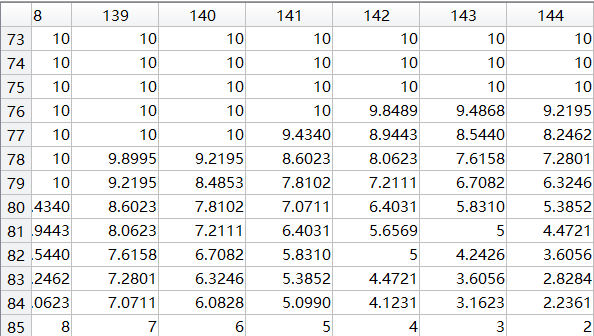
 

Figure 4. Grid maps of point occurrence and corresponding distance

1. Rough Estimation for pose

In our project, we assume that the robot does not move fast and less disturbance occurs when the robot moves, which we can just focus on the kinematics of the robot. Then we can assume that the velocity of the robot between two scans may not change severely. With such assumption, next pose can be roughly estimated by the previous poses.

Noted that the pose as , , at the k-th scan, we can have:

Finally, we can have the pose estimation of k+1-th scan as:

1. Scan to Scan Match

Although we can get a rough estimation of the pose through the previous method, we still need a more accurate pose for the map building. To get a better pose estimation, we used the scan to scan match method to the optimize the previous guessed pose.

Since we have assumed that the robot does not move fast and the time span between two scans is small, the grid map generated by previous and next pose should be similar. Then we can use the similarity of two grip maps to optimize the estimated pose.

In the previous grid map generation, we can get a distance grid map and a hits grid map. We can use the hit grid map generated by the next scan to extract the distances in the previous map then sum as the score. If two maps are similar, the coordinates of hits in the hits grip map will extract small distances in the distances grid map, which means that the smaller score gets better estimation.

Denote the and as the vectors used for estimate score, where u and v are distance and hits grid map respectively. Then the optimization problem can be illustrated as follows:

To get a solution in the optimization problem, we use down hill searching in this project. Based on the guessed pose, we search the surrounding poses with certain resolution and get a pose with most similar grid map with lowest score.

**Scan to Scan Match**

*best\_score = , times = 0 , pose\_new = pose\_pre*

***while****(pose\_new == pose\_pre* ***and*** *0 < times < 3)*

***if*** *times != 0* ***then*** */= 2*

*times += 1*

***for******do***

***for******do***

***for******do***

*v =* **grid\_hits**(*)*

*score =*

***if*** *score < best\_score* ***then***

*pose\_new =*

*best\_score = score*

To reduce the calculation time, we only search for the surrounding of the estimated pose once with the resolution. If the score will not get smaller during one process, the resolution will be set smaller and rerun the process. To avoid the situation that the score always larger than the best score, the resolution will get smaller with only a certain times. The process of this algorithm can be described as above.

After optimizing the pose by the scan data, the transformation of the local map to the global frame can be done via the transformation of scan points to the global.

1. Add a Key Scan

In this function, we want to add a new key scan (pose) to the map on condition that the robot moves any distance or angle that exceeds the threshold. We will continually use this key scan data to update the map. For the details of this function, we firstly check whether the difference between the last pose and current pose is too large, if yes, we would say the pose estimation is not correct and use the last pose instead. After predicting and transforming the next pose, we use the “hits” to select new points and add these new points to our map data.

1. Loop Closure Optimization

From the previous step, we can obtain a better pose estimation of the robot. However, there may be large accumulated errors during the process when keeping using the previous pose. To reduce the accumulated error, we can use the loop closure optimization method.

The basic idea of loop closure optimization is to optimize the new pose via a far previous pose which do not have much accumulated error. The selected pose used to optimize the new pose should be far enough in time, but close to the new pose in space, which means that the pose has much less accumulated error and local map is similar. The path of the robot can be seen as nearly a loop, see in the figure, that’s so-called loop closure.

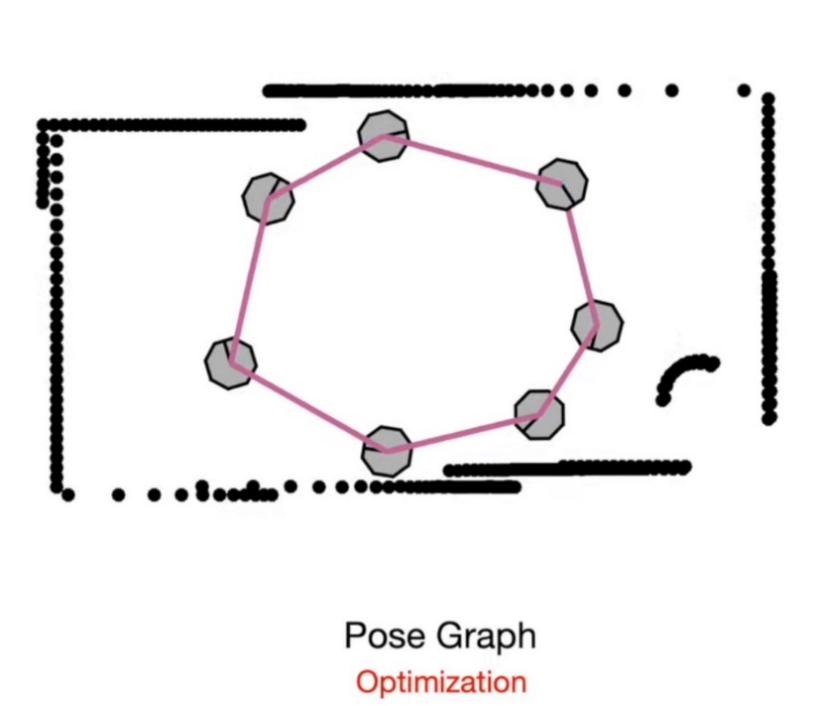


Figure 5. Demonstration of close loop optimization

To select a proper pose as the previous pose, which also call detecting a loop, we first back skip some pose to assure that the selected pose is far enough. Then we keep searching back until the distance between the new pose is smaller than a certain threshold, we can think that comes to a loop. Then we can get to the next step to revise the new pose.

The optimization algorithm is similar to the scan to scan match process with much smaller resolution. After running this optimization algorithm, we can get a better optimization of the pose with less accumulated error.

Since the loop closure optimization can be quite slow with small resolution for optimization, we may not use this method each scan. We optimize at some pose with a minimum distance between.

The process of the loop closure optimization can be shown as below:

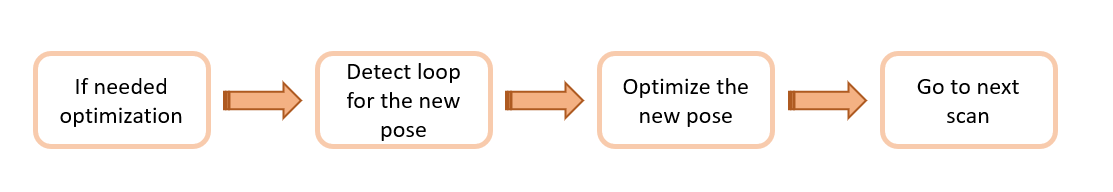


Figure 6. process of the loop closure optimization

1. Plotting

Finally, for plotting part, we first transform our scan to global coordinates again. Then, we plot world map (black points map), our scan data as well as the path of the robot (blue dots trajectory). Also, for each scan, we plot the several lines to analogue the laser emitted by the robot itself to get the scan data (red lines).

1. **Results and Analysis**
   1. Impact of close loop detection

Using the original search parameters and without close loop detection, the mobile robot trajectory and the constructed map (Deutsches Museum) is shown in the following figure.

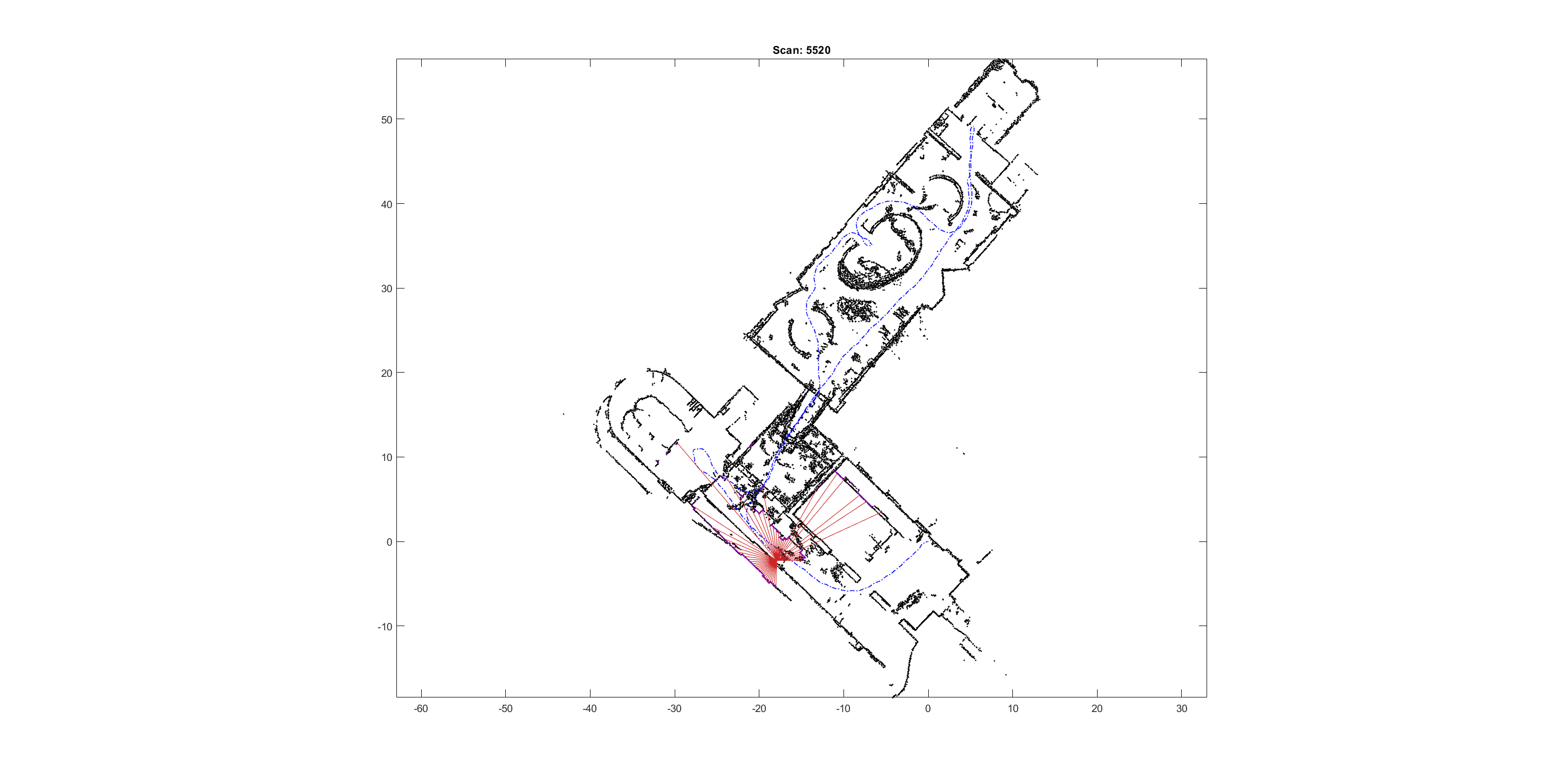


Figure 7. Map generated without close loop detection

It can be seen from the figure that since the mobile robot has no error correction mechanism, the cumulative error gradually increases, and an obvious offset phenomenon occurs at the end of the robot path.

To this end, we introduce close-loop detection. According to the prescribed matching degree threshold, we find the data frame with the highest degree of similarity in the paths that have been created before, and consider that there is a close loop at this position. At the same time, we will correct the current data frame according to the matched frame, so as to eliminate the accumulated error caused by the close loop path between the matched frame and the current frame in the subsequent matching. The result with the addition of close loop detection algorithm is shown as follows.

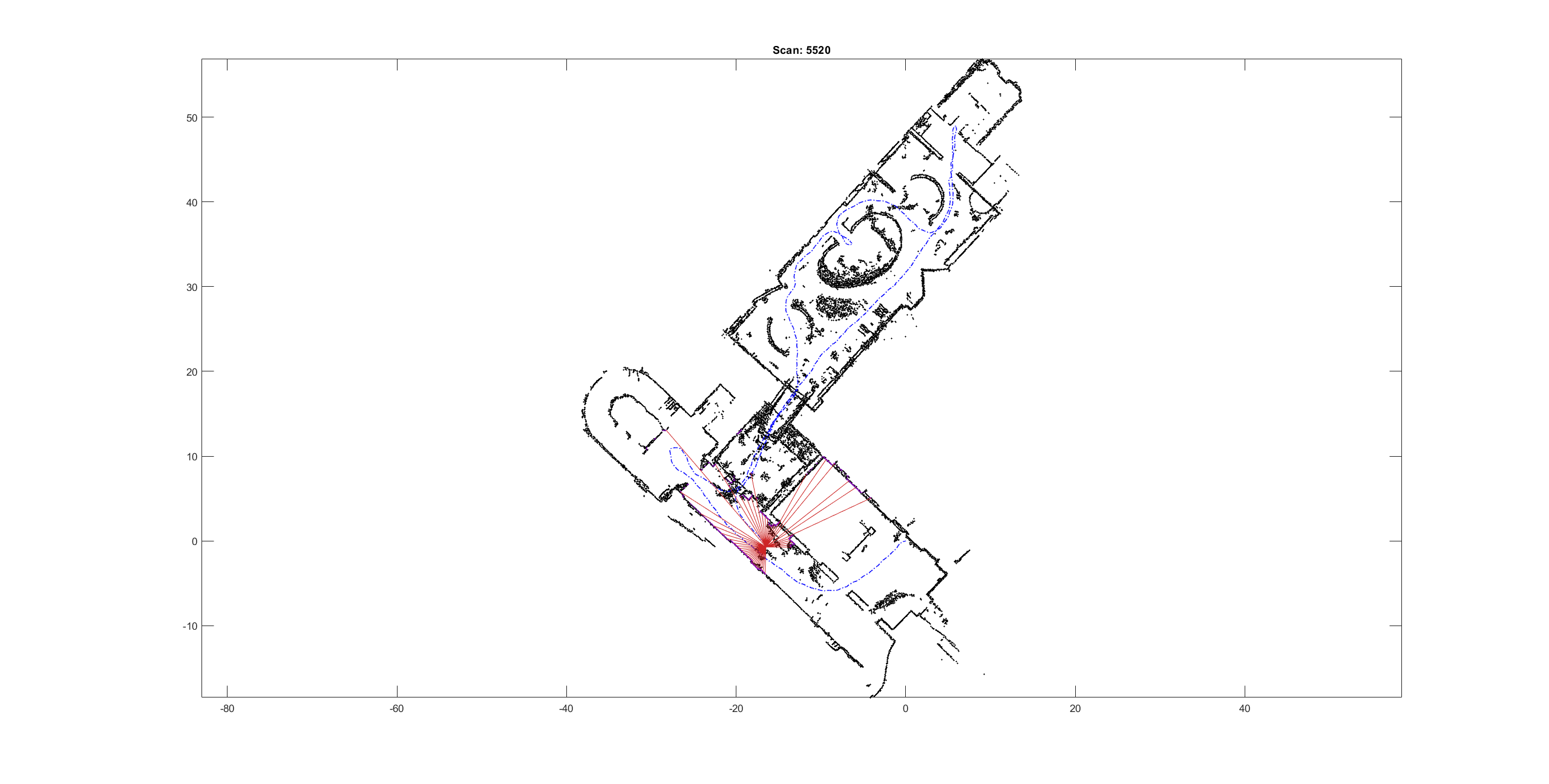


Figure 8. Map generated with close loop detection

Comparing the output results with the results without close-loop optimization, it is obvious that the offset is significantly weakened in the map construction process. We believe that the close-loop optimization algorithm successfully reduces the accumulated error caused by the path that forms the close loop. This defect is caused by the algorithm itself. The algorithm of this project only performs local optimal estimation for the pose of a single point where the close-loop is detected, which results in a very insignificant optimization effect. A more general close loop optimization algorithm should optimize the path key points enclosed by the previous close loop pose and the latest pose, so as to improve the scope of optimization effect.

We also tried other Lidar scan data from Deutsches Museum.

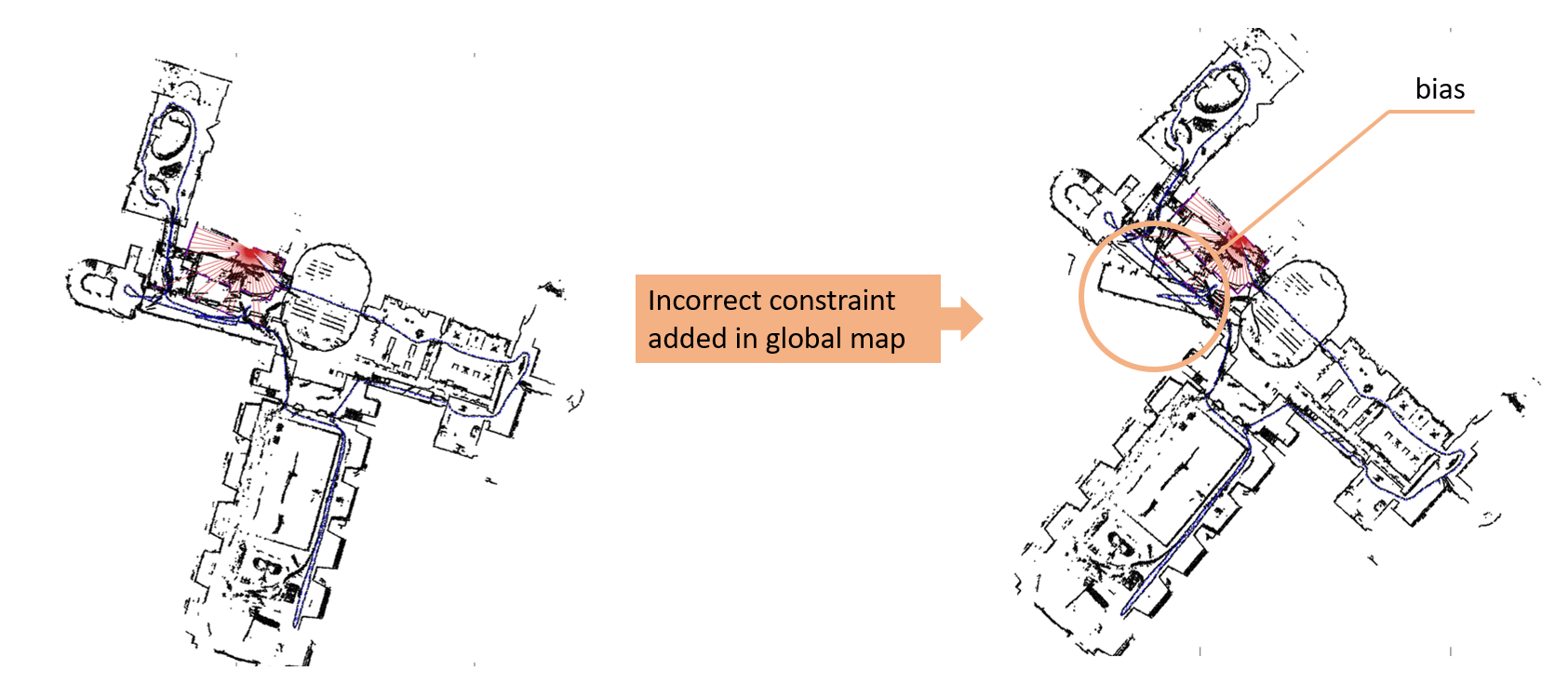


Figure 9. Map generated with another dataset

The slam algorithm is not suitable for map construction for an environment that contains relatively large vacancy, and path with frequent cross-links should be avoided. Due to the lack of an anti-disturbance mechanism, the algorithm is sensitive to lidar noise, and it can be improved via performing a preliminary selection of lidar scan based on the intensity column in the dataset.

We can find that since the pose estimation is completely determined by the degree of similarity with the current map, there is a lack of error correction mechanism. At the same time, the scan point in the close-loop of the close-loop optimization algorithm will be considered as the correct point and considered as a global constraint. Hence, when an incorrect loop occurs, the error will be further enlarged and cause obvious distortion in the constructed map. Solutions to such problems can be considered in future research.

* 1. Impact of occupancy grid map resolution

In order to improve the efficiency of the algorithm, the matching in the data frame is based on the local occupancy grid map, and the conversion of the coordinates in the Cartesian coordinate system into the occupancy points in the grid map requires certain resolution configuration.

In theory, increasing the grid resolution to twice the original will improve the accuracy of fast matching and achieve better pose estimation. For this reason, the resolution of the grid map is doubled. The result is as follows:

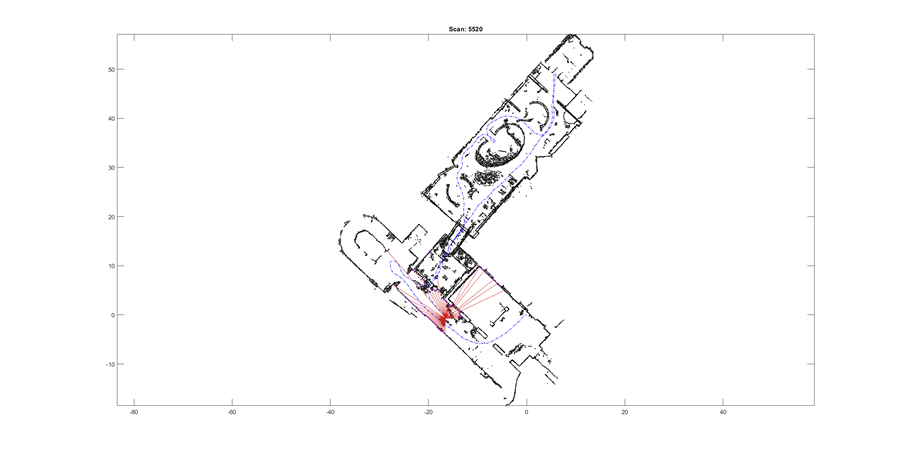


Figure 10. Map generated with 2 times of default resolution

Compared with the previous figure, the offset is significantly reduced. It can be roughly considered that the increase of the resolution (double) improves the accuracy of the pose estimation, and thus achieves a better SLAM outcome.

To further increase the resolution, the following results are obtained:

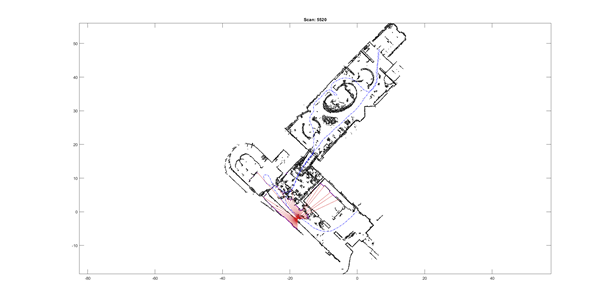


Figure 11 Map generated with 4 times of default resolution

It can be found that the map produces obvious offset, and this phenomenon may be caused by the method for evaluating the similarity between data frames. Since the similarity score is defined as the numerical sum of the overlapping areas of the distance matrix, high resolution may cause the score occupancy coordinates to be sensitive to small perturbations, and since fast matching is based on the hill-climbing algorithm, the rapidly oscillating score may make it difficult to search for a reliable local optimal solution.

1. **Conclusion**

In this project, we mainly focus on a Lidar SLAM with the dataset obtained in a museum. We got to know how the Lidar data works and run a simulation in MATLAB. During the research, we got to know the knowledge of the model of 2D mobile robot, frame transformation, pose estimation via both scan to scan match and loop closure optimization. Finally, we can get a point map after using all the scan data.

There can be still some problems in our project: some assumption in our project may not be valid and the optimization algorithm cannot be precise enough to find a global optimum. These problems may be fixed via more advanced and completed methods in the future.

**Reference**

1. Hess, Wolfgang, et al. "Real-time loop closure in 2D LIDAR SLAM." *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016.
2. The Cartographer Authors. "Public Data -- Cartographer ROS documentation". *Google Cartographer ROS*, 24 June 2020, https://google-cartographer-ros.readthedocs.io/en/latest/data.html.
3. 任乾. "激光SLAM | cartographer论文(2) ". *Zhihu*, 4 Dec. 2019, https://zhuanlan.zhihu.com/p/93713565.
4. MATLAB. "Understanding SLAM Using Pose Graph Optimization | | Autonomous Navigation, Part 3". *Youtube*, 8 July 2020, https://youtu.be/saVZtgPyyJQ.
5. 卫浩. "激光SLAM简单入门（1）-matlab读取bag文件并保存为mat文件". *Zhihu*, 4 Dec. 2018, https://zhuanlan.zhihu.com/p/51577331.
6. 卫浩. "激光SLAM简单入门（2）-代码解读和分析". *Zhihu*, 5 Dec. 2018, https://zhuanlan.zhihu.com/p/51617565.
7. meyiao. "LaserSLAM". *Github*, 13 June 2017, https://github.com/meyiao/LaserSLAM.