Simultaneous Localization and Mapping

1st Yifan Xu A53253258
The Electrical and Computer Engineering Department UCSD
La Jolla, US
yix247@eng.ucsd.edu

***Abstract*—This paper is a report for the second project in ECE276 Sensing and Estimation in Robotics, UCSD.**

***Index Terms*—SLAM, particle filter, localization, mapping**

I. INTRODUCTION

As the world continues to move deliberately toward a transportation system driven by autonomous vehicles in the purpose of reducing accidents, traffic congestion and freeing people, the technologies to achieve this goal became really important. To make vehicles drive autonomous, an accurate simultaneous localization and mapping (SLAM) method is needed.

In this project, I implemented simultaneous localization and mapping using odometry, inertial, 2-D laser range, and RGBD measurements from a differential-drive robot. But the sensor data are not good. In order to get a better result, I used the particle filter algorithm to find an accurate world position for the robot.

II. PROBLEM FORMULATION

The main problem to this SLAM project was to find the most accurate location of the moving robot with encoders, IMU, 2D Lidar data and the particle filter algorithm. Therefore, it is more likely to make a reasonable and accurate map.

A. Mapping

Mapping process needs to transfor the lidar data information from the lidar coordinate to the world coordinate.The lidar data can be transformed via ${}_wT_{bb}T_l$ where ${}_wT_b$ is the homogeneous rotation and transformation matrix from robot's body to world and ${}_bT_l$ is the homogeneous rotation and transformation matrix from lidar to robot's body.

We initial the robot first position at the world (0,0) and map the first lidar scan in to the map, as our first step ground truth map.

B. Localization Prediction

By using the differential-drive motion model with input from the encoder measurments and the IMU yaw rate with additive noise, we can predict the next pose of our robot. In this project, I used the discrete-time model with time discretization τ:

$$x_{t+1} = x_t + \tau(v_t sinc(w_t\tau/2)cos(\theta_t + w_t\tau/2)$$

$$y_{t+1} = y_t + \tau(v_t sinc(w_t\tau/2)sin(\theta_t + w_t\tau/2)$$

$$\theta_{t+1} = \theta_t + \tau w_t$$

C. Localization Update

Implementing the particle filter algorithm is the most important part of localization update, also in the whole project.

In the localization update, based on the predicted particle positions and weights $(\mu^{(i)}_{t+1|t}, \alpha^{(i)}_{t+1|t}), i = 1, ...N$ and the next laser scan data, we can get the global laser data coordinates via ${}_wT_{bb}T_l$. Compared with the current ground truth map, we can find the best match. Then we can get the updated particles positions and weights $(\mu^{(i+1)}_{t+1|t}, \alpha^{(i+1)}_{t+1|t}), i = 1, ...N)$.

D. Texture Mapping

For texture mapping, we need to combine the RGB image data and disparity image data together and transform from image plane to world plane.

Given the values d at location (i,j) of the disparity image, we can obtain the depth and associated color by:

$$dd = -0.00304 * d + 3.31$$

$$depth = 1.03/dd$$

$$rgbi = (i*526.37 + dd*(-4.5*1750.46) + 19276.0)/585.051$$

$$rgbj = (j * 526.37 + 16662.0)/585 : 051$$

We use the Pinhole Camera Model. The projection and intrinsics:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fs_u & fs_\theta & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \frac{1}{z_o} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix}$$

The extrinsics:

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} = \begin{bmatrix} R_{oc}R^T_{wc} & -R_o c R^T_{wc} p_{wc} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

We call Eular angles the roll(ϕ), pitch(θ),yaw(ψ) angles. The ZYX intrinsic rotaion:

$$R = R_z(\psi)R_y(\theta)R_x(\phi) =$$

$$\begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}$$

III. TECHNICAL APPROACH

For this project, we had two tasks. First, SLAM. Second, texture mapping.

A. SLAM

- **Particle Filter** As I mentioned in the last section, the particle filter is the most important technical approach for SLAM. The main idea of particle filter is let the particles distributed in different location and check which location has the highest possibility to be the real position where the robot is. In the project, I initiated 100 particles at the origin of the world frame and added noise on its position (x, y) and the yaw angle orientation θ. Initiated the ground truth map with the first laser scan at the beginning. Then, during the prediction and update process which mentioned above, we calculated the correlation of the next lidar scan in world frame (based on different particles) and the ground truth map. Choose the particle with the biggest correlation value as the robot real position. In order to increase the accuracy, we calculated the correlation with a grid of values around all particle positions. Also, we could adjust particles positions according to the correlation matrices. Repeat the process until it finished, we will get the map.

B. Texture mapping

- **RGBD** To implement the texture mapping, we need the RGB and depth information which can be obtained by RGB and depth cameras. Since two cameras are not located in the same position. We need to map them together first. After combine the RGB data and depth data, we need to transfer them from the image frame to the world frame (based on the moving robot position) as equations showed in the last section. After got the data in the world frame, since we wanted to map the texture of floor, I set a threshold on the height in the world coordinate to get the data which is belong to the floor. Then, we could map the RGB color data in the corresponding pixels in the map. Therefore, we will get a map with floor texture.

IV. Results

A. Training Result

1) Trainning Set 20: Both SLAM and Texture results are shown below.

2) Trainning Set 21: Both SLAM and Texture results are shown below.

B. Testing Result

1) Trainning Set 23: SLAM results are shown below.

C. Discussion

From the figures we can see, the SLAM method successfully find the routes that the robot drove in the hallway in different circumstance(all three data sets). The texture can be mapped on the floor at the right position too(both training data sets). However, the accuracy of the SLAM method is still needed to be improved. When there is a big turn, then it's highly possible that the sensor returned some data with big noise. Therefore, it will be more easily to decrease the map accuracy, like shifting the map slightly. Especially, in the test set, there are four harsh 90 degree turning. Also, in the data set 21, we can see the texture mapping slightly tiled to the left on the left hallway. From the RGB images we could know, there are ramps. So, the height of the robot is actually change. Since in this project we only used yaw angle, there will be some deviations.

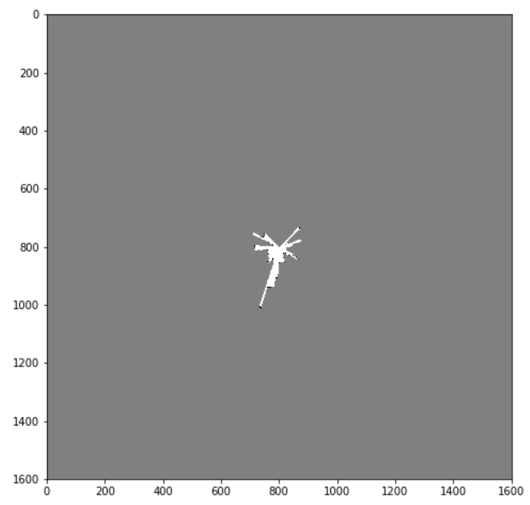

Fig. 1. DataSet 20 SLAM

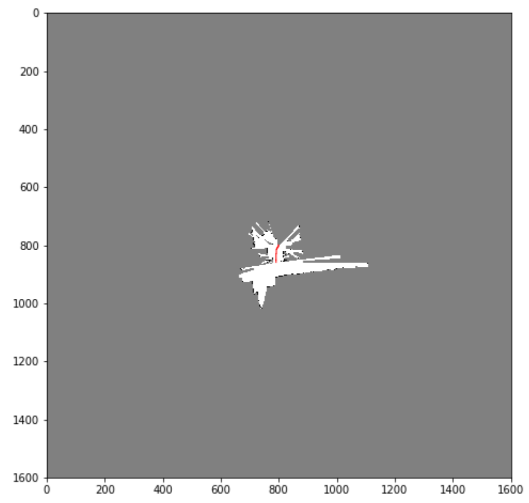

Fig. 3. DataSet 20 SLAM

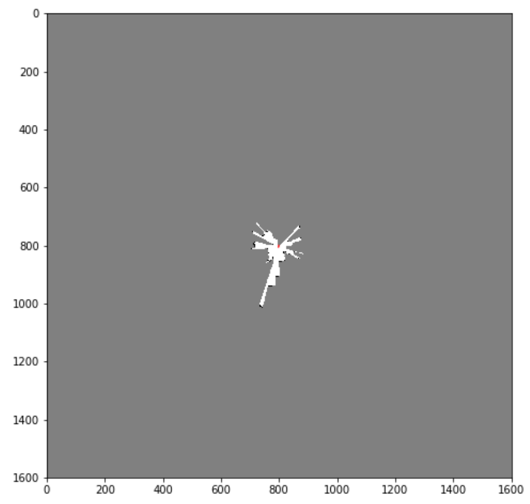

Fig. 2. DataSet 20 SLAM

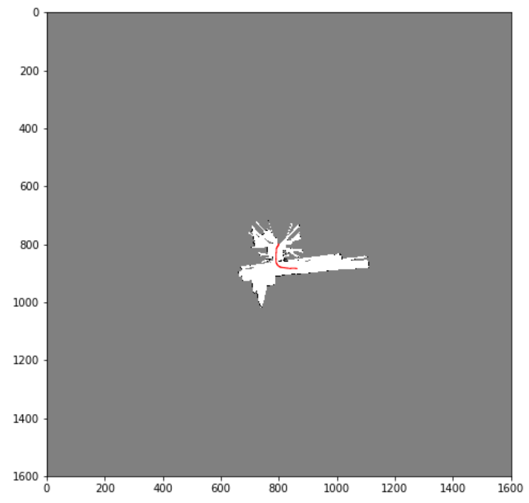

Fig. 4. DataSet 20 SLAM

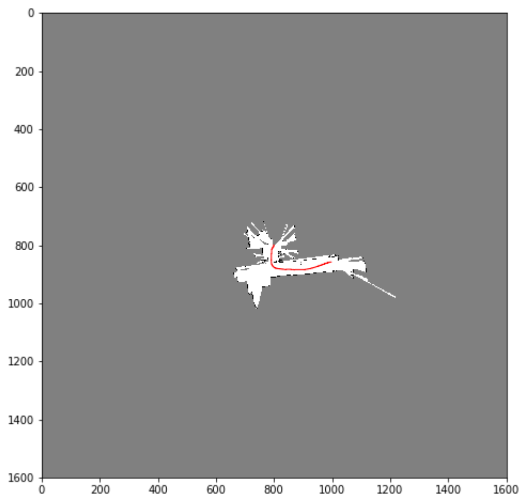

Fig. 5. DataSet 20 SLAM

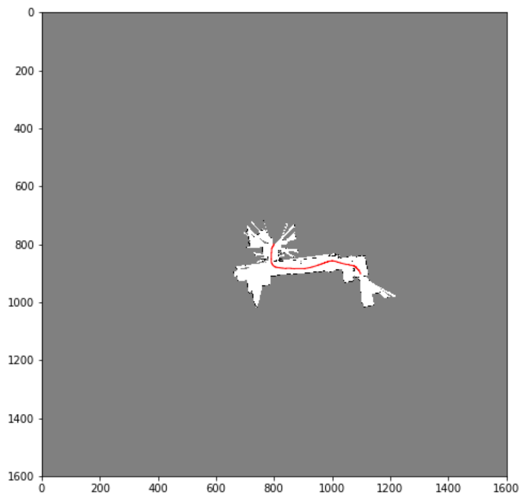

Fig. 6. DataSet 20 SLAM

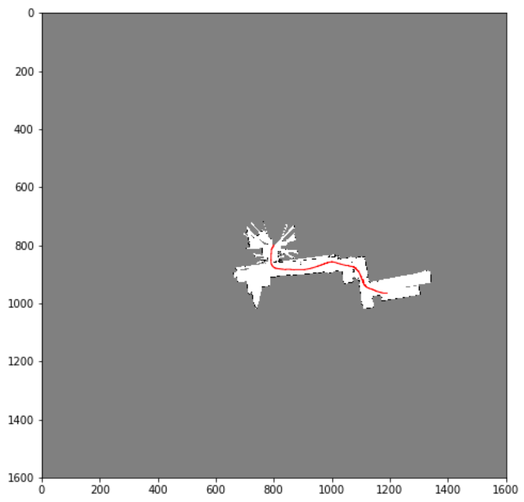

Fig. 7. DataSet 20 SLAM

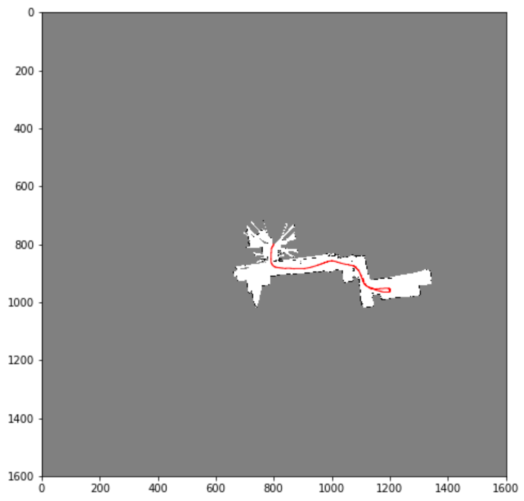

Fig. 8. DataSet 20 SLAM

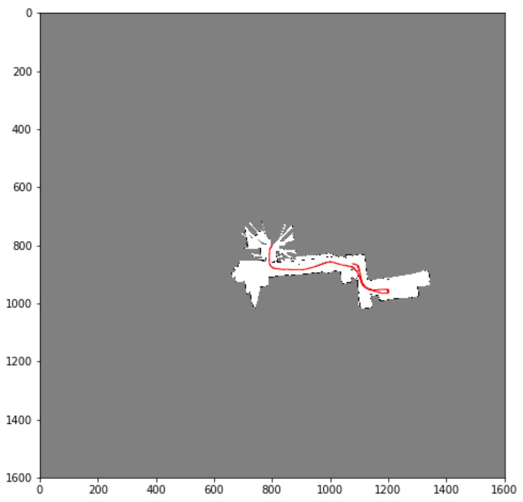
Fig. 9. DataSet 20 SLAM

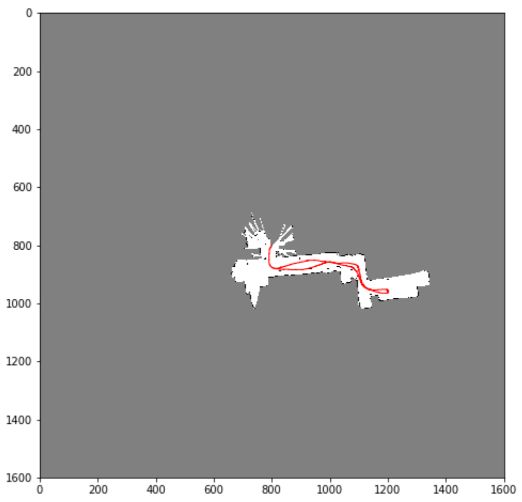
Fig. 11. DataSet 20 SLAM

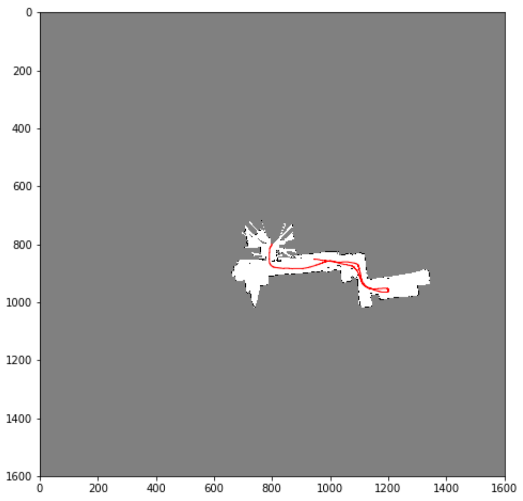
Fig. 10. DataSet 20 SLAM

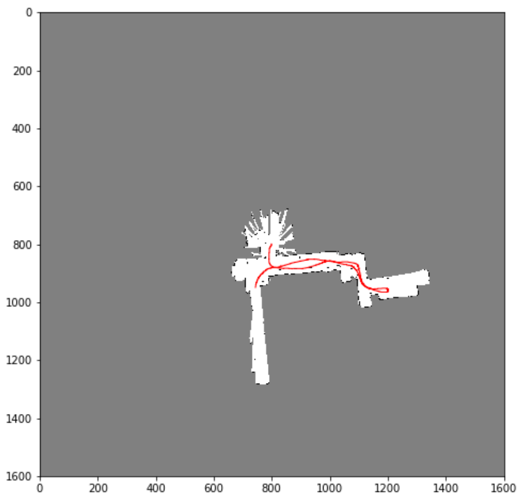
Fig. 12. DataSet 20 SLAM

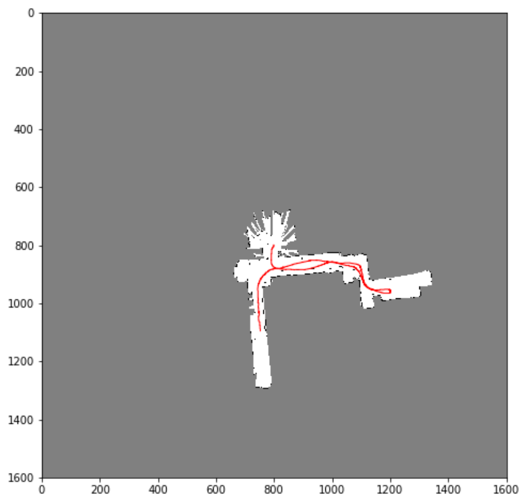

Fig. 13. DataSet 20 SLAM

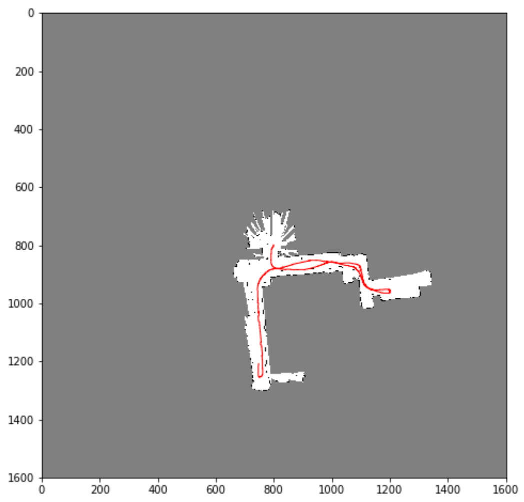

Fig. 15. DataSet 20 SLAM

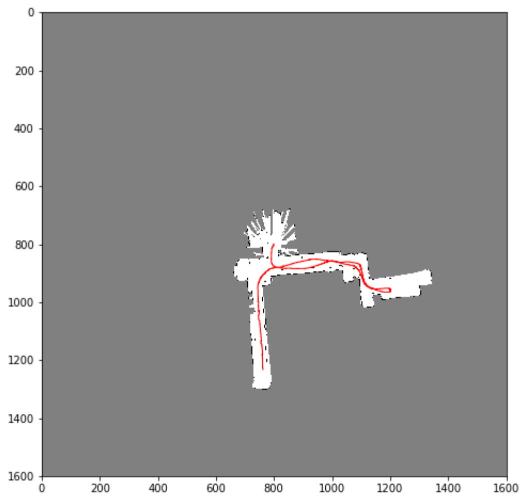

Fig. 14. DataSet 20 SLAM

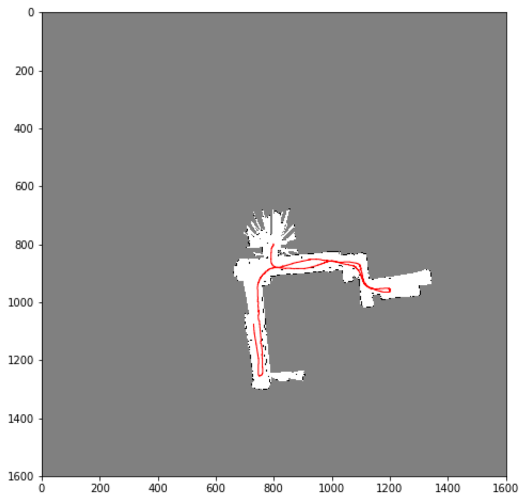

Fig. 16. DataSet 20 SLAM

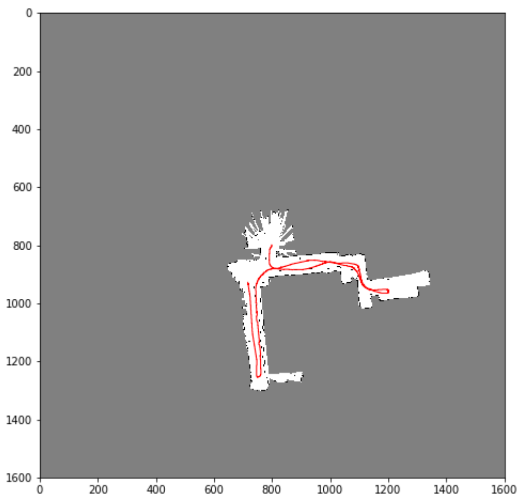
Fig. 17. DataSet 20 SLAM

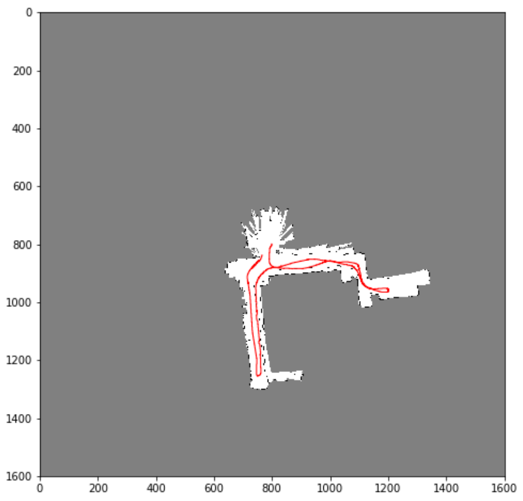
Fig. 18. DataSet 20 SLAM

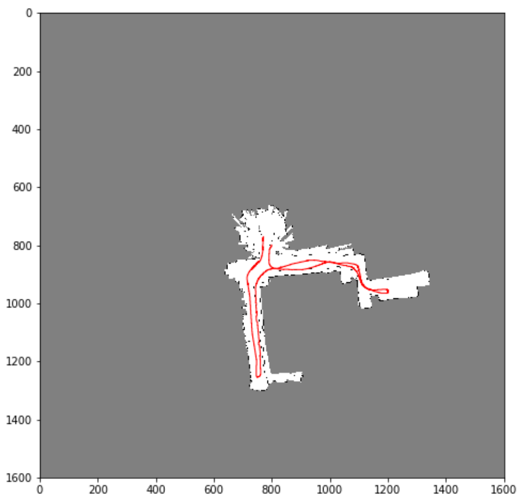
Fig. 19. DataSet 20 SLAM

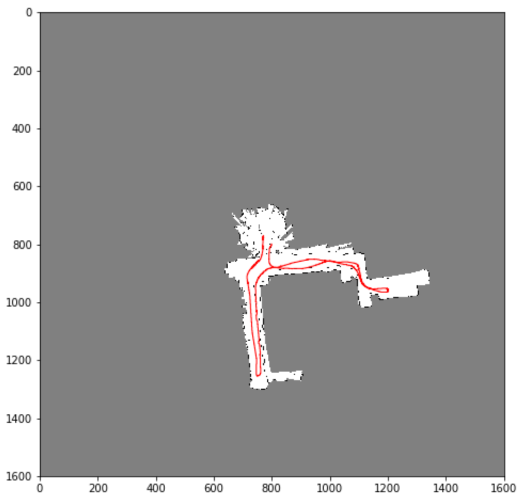
Fig. 20. DataSet 20 SLAM

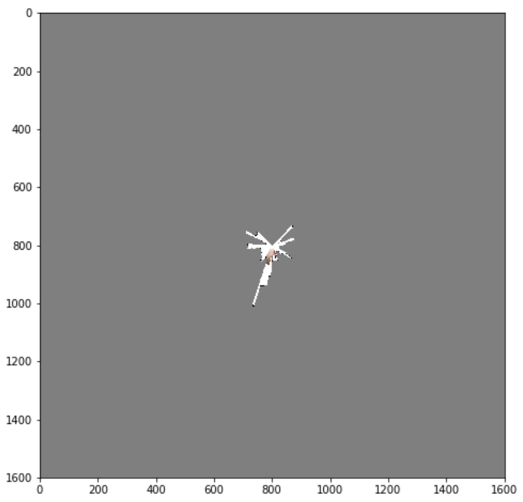

Fig. 21. DataSet 20 Texture

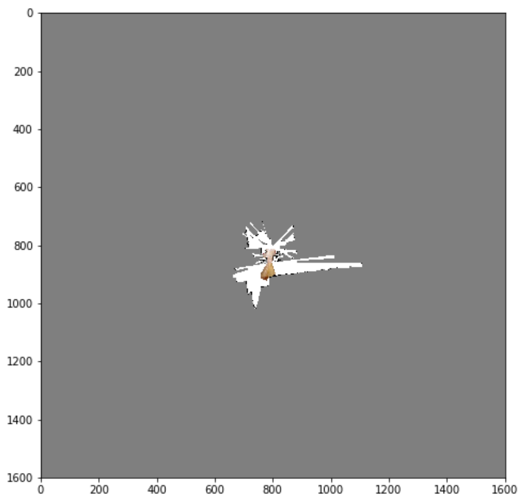

Fig. 23. DataSet 20 Texture

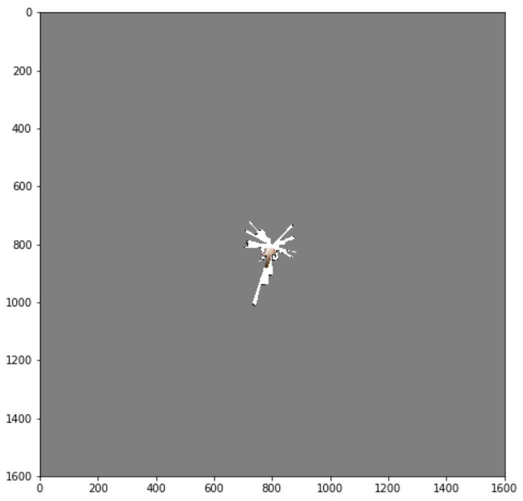

Fig. 22. DataSet 20 Texture

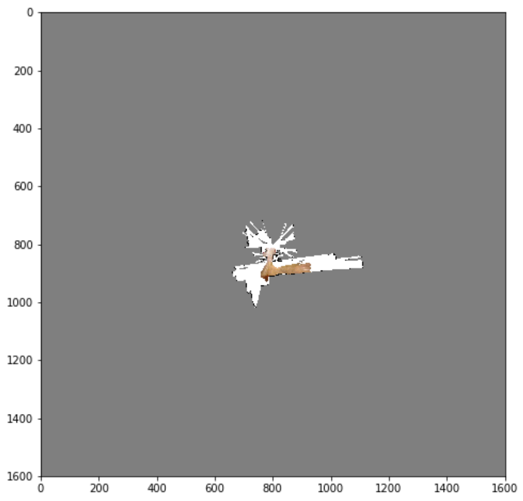

Fig. 24. DataSet 20 Texture

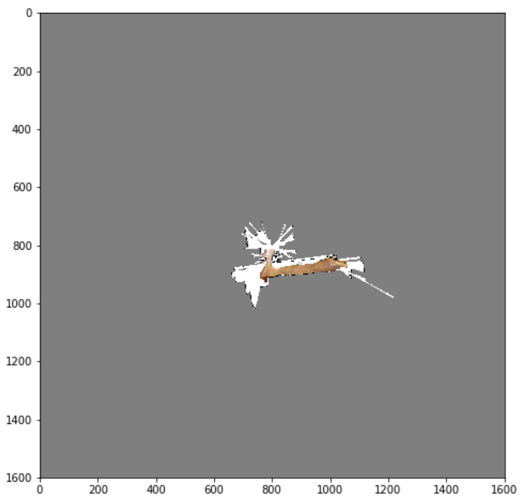
Fig. 25. DataSet 20 Texture

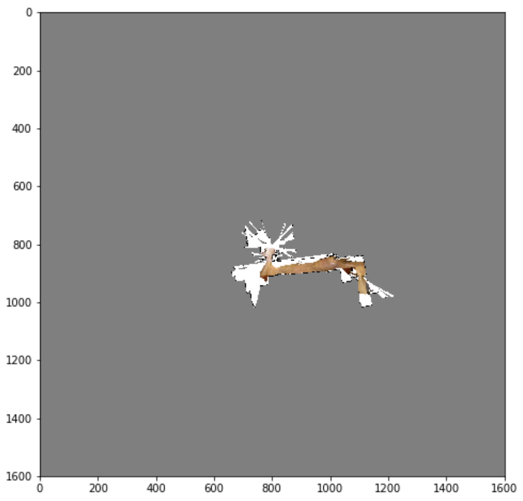
Fig. 26. DataSet 20 Texture

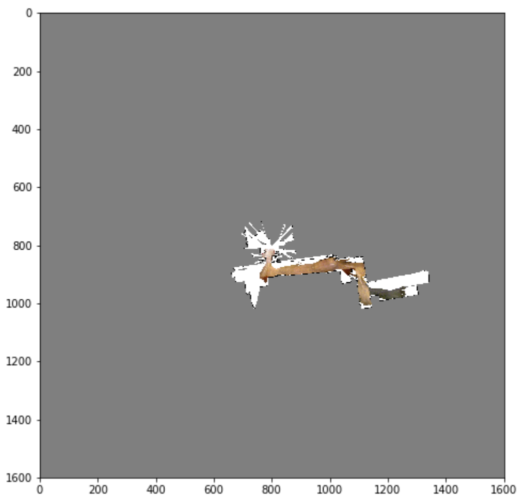
Fig. 27. DataSet 20 Texture

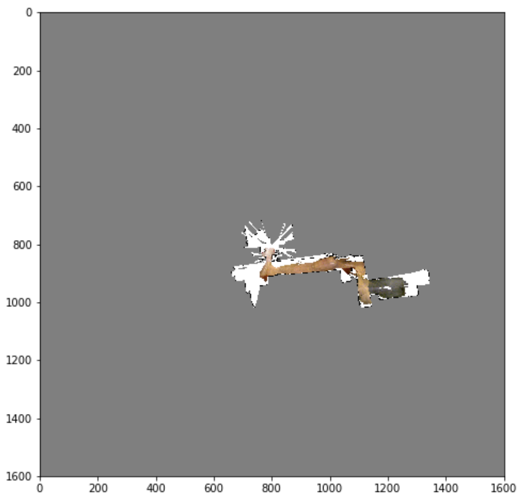
Fig. 28. DataSet 20 Texture

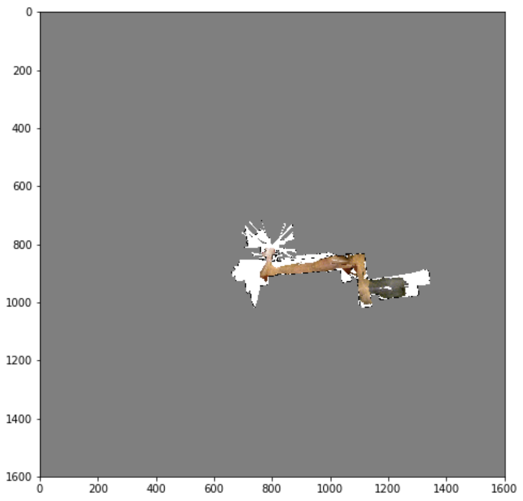

Fig. 29. DataSet 20 Texture

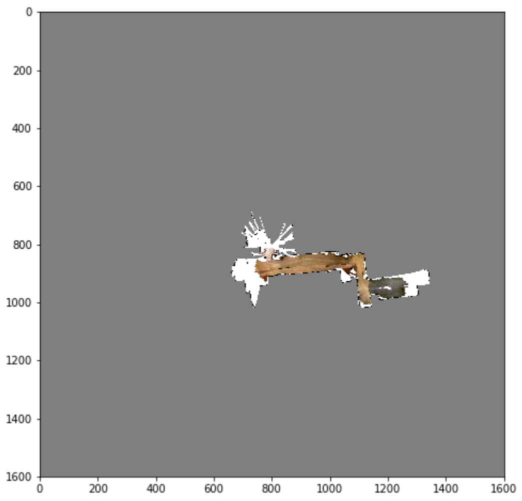

Fig. 31. DataSet 20 Texture

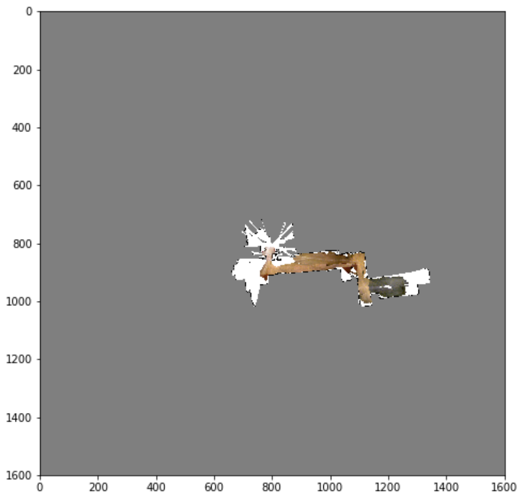

Fig. 30. DataSet 20 Texture

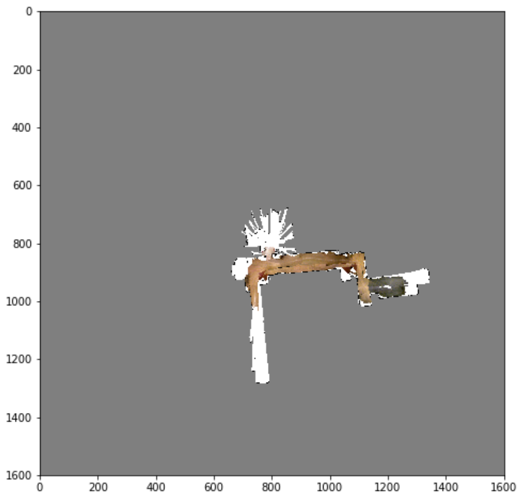

Fig. 32. DataSet 20 Texture

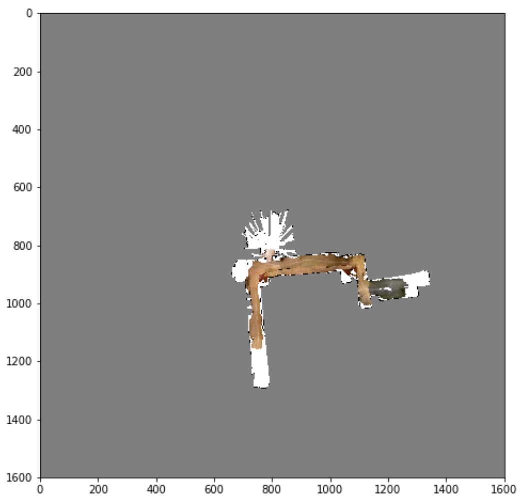
Fig. 33. DataSet 20 Texture

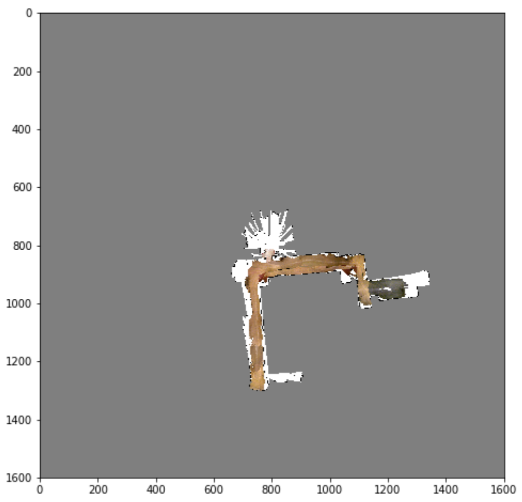
Fig. 35. DataSet 20 Texture

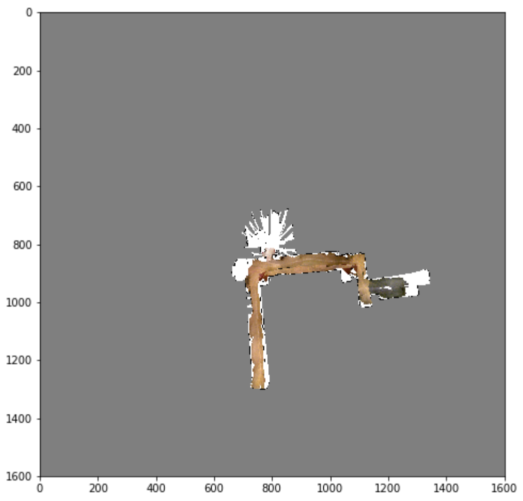
Fig. 34. DataSet 20 Texture

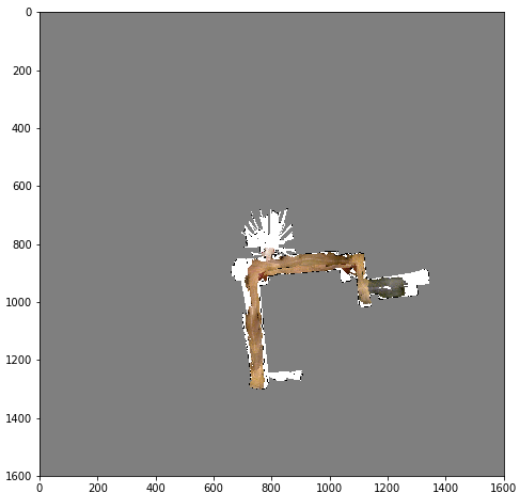
Fig. 36. DataSet 20 Texture

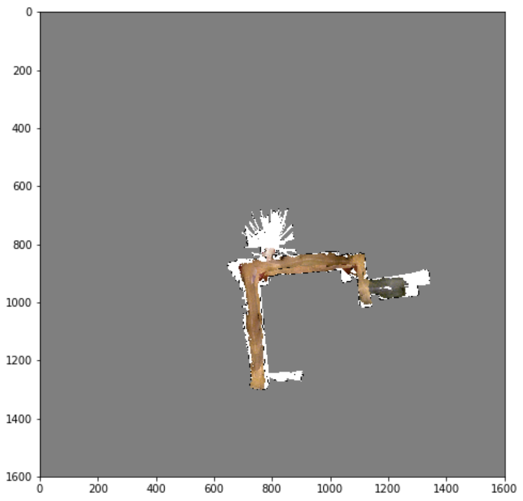

Fig. 37. DataSet 20 Texture

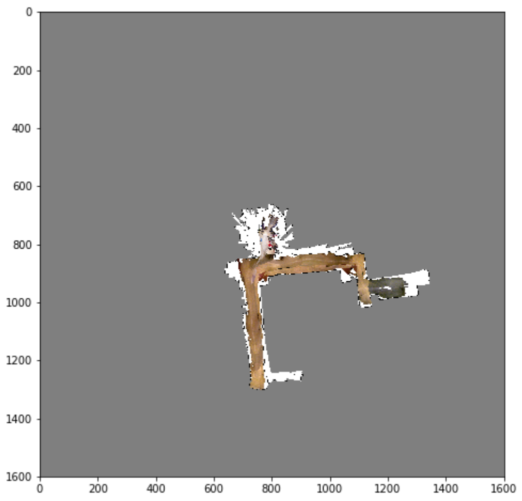

Fig. 39. DataSet 20 Texture

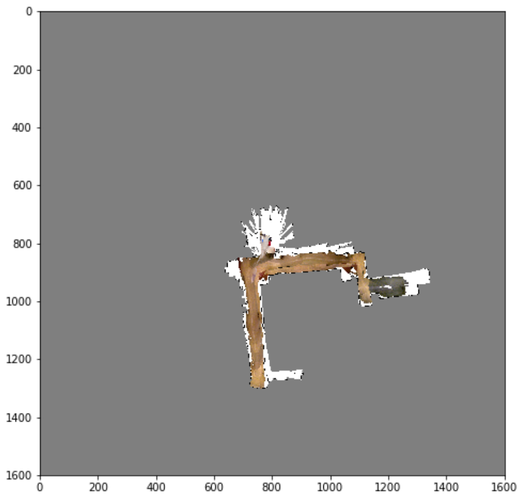

Fig. 38. DataSet 20 Texture

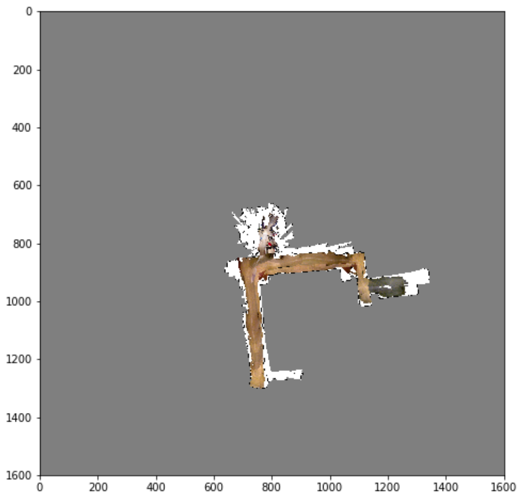

Fig. 40. DataSet 20 Texture

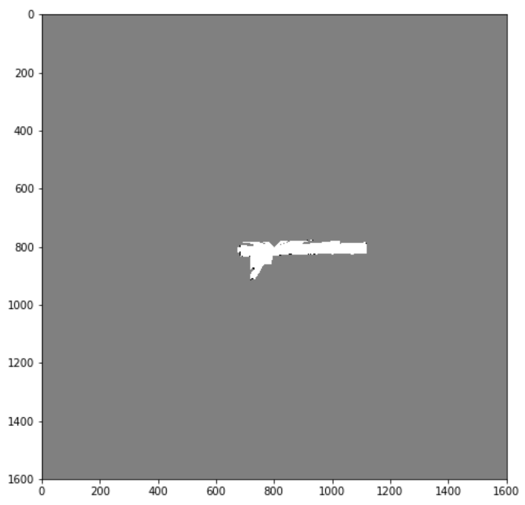
Fig. 41. DataSet 21 SLAM

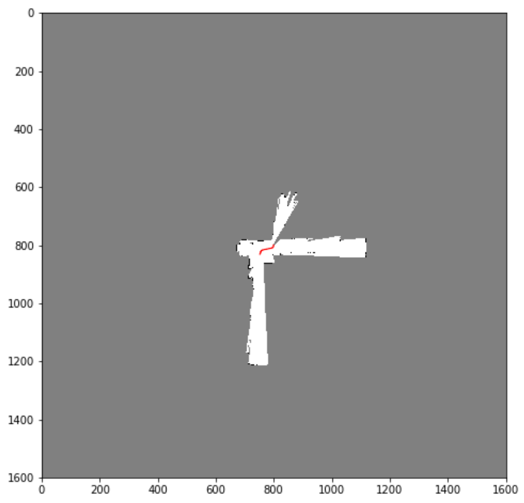
Fig. 43. DataSet 21 SLAM

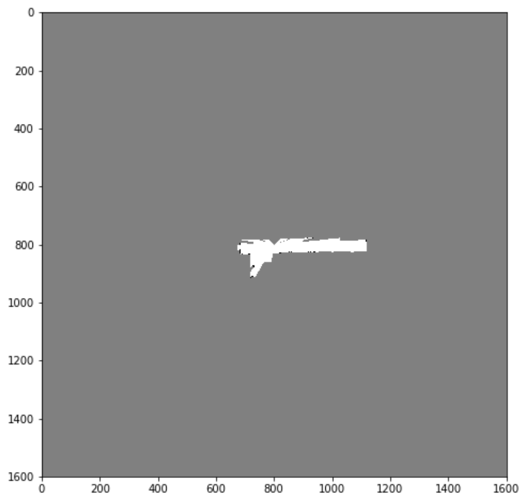
Fig. 42. DataSet 21 SLAM

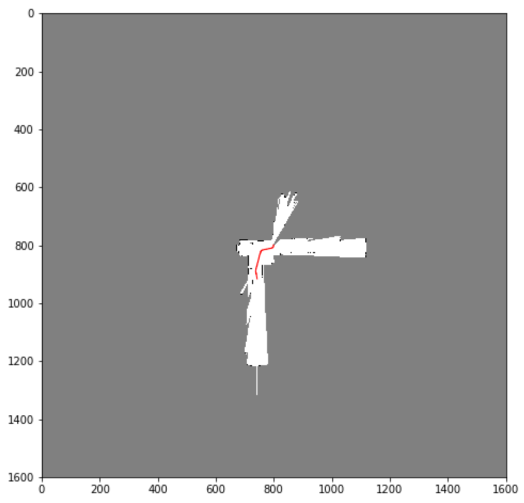
Fig. 44. DataSet 21 SLAM

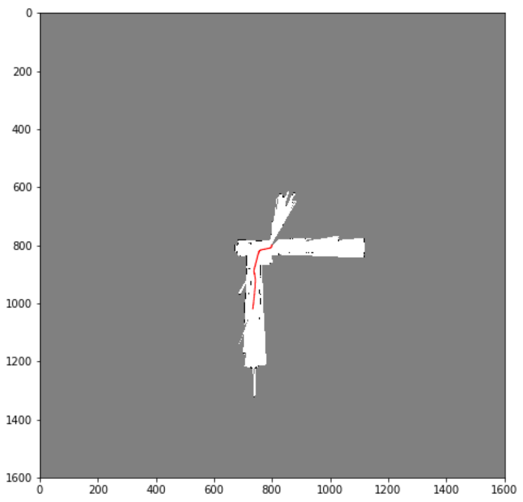

Fig. 45. DataSet 21 SLAM

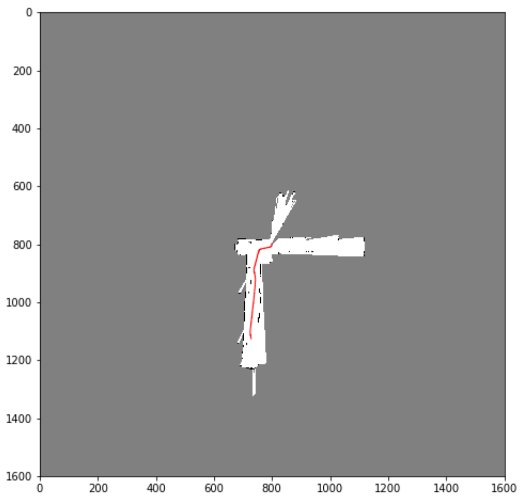

Fig. 46. DataSet 21 SLAM

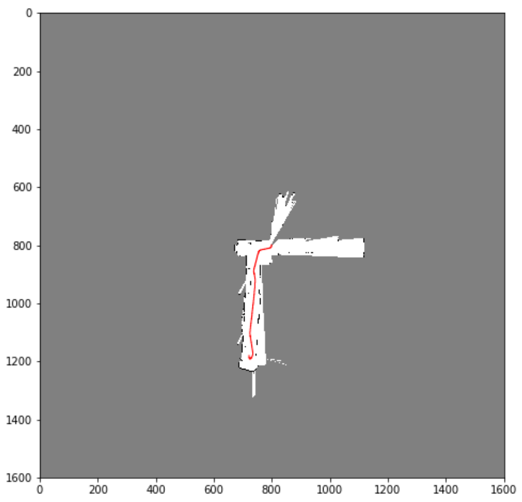

Fig. 47. DataSet 21 SLAM

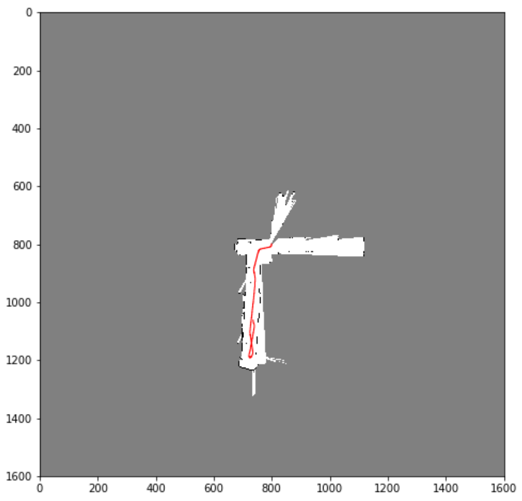

Fig. 48. DataSet 21 SLAM

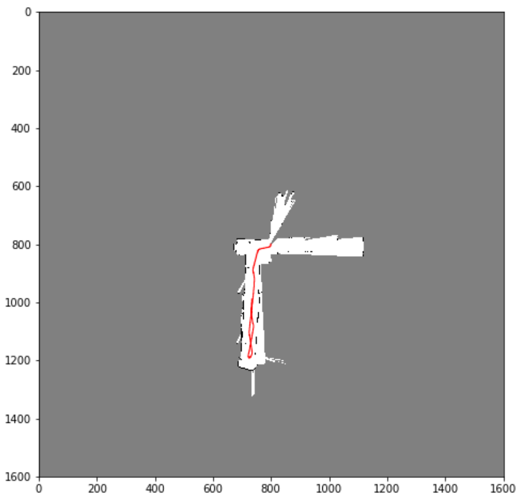

Fig. 49. DataSet 21 SLAM

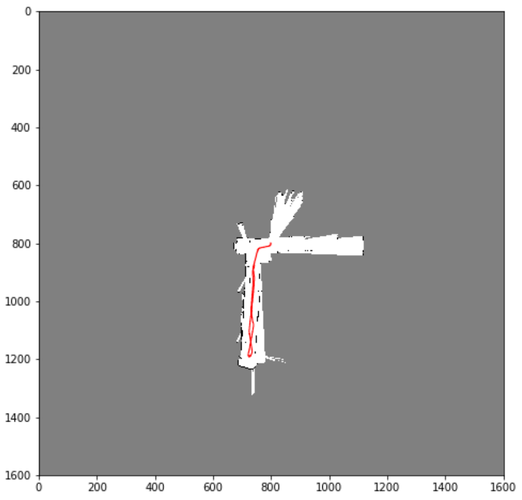

Fig. 50. DataSet 21 SLAM

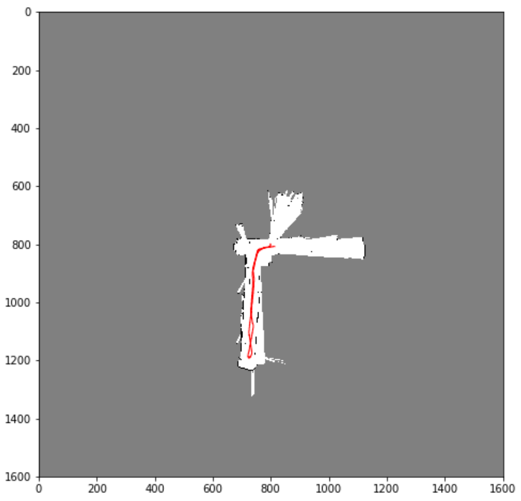

Fig. 51. DataSet 21 SLAM

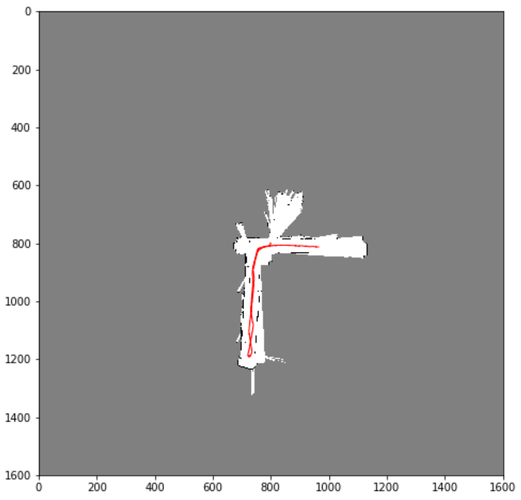

Fig. 52. DataSet 21 SLAM

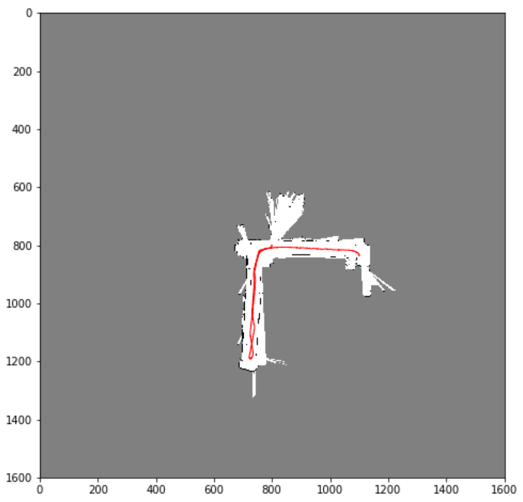

Fig. 53. DataSet 21 SLAM

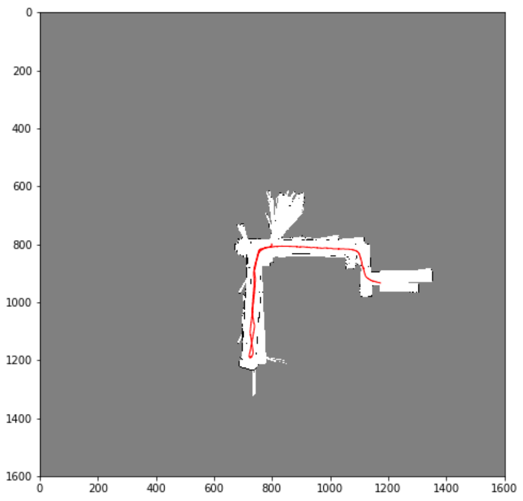

Fig. 54. DataSet 21 SLAM

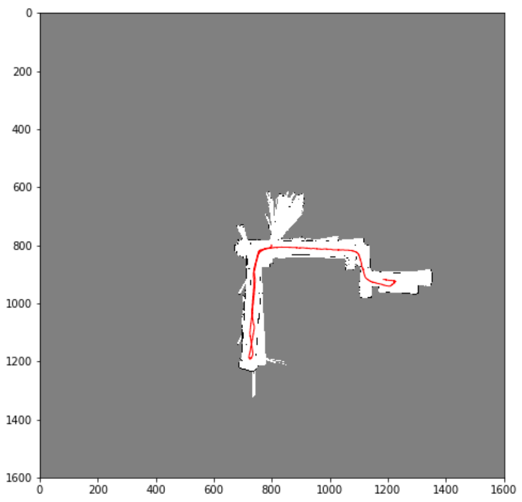

Fig. 55. DataSet 21 SLAM

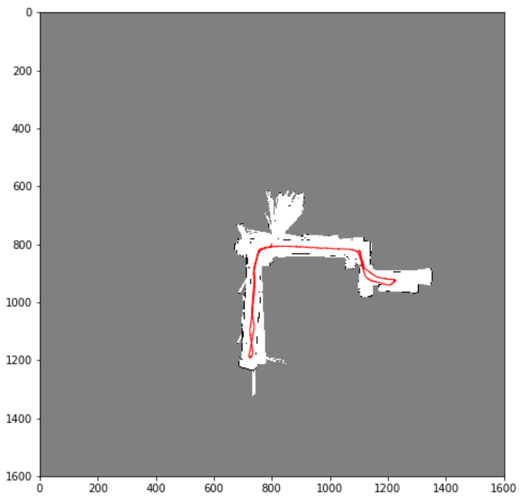

Fig. 56. DataSet 21 SLAM

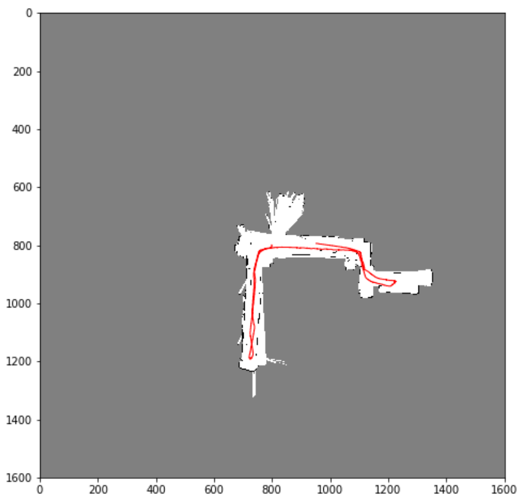

Fig. 57. DataSet 21 SLAM

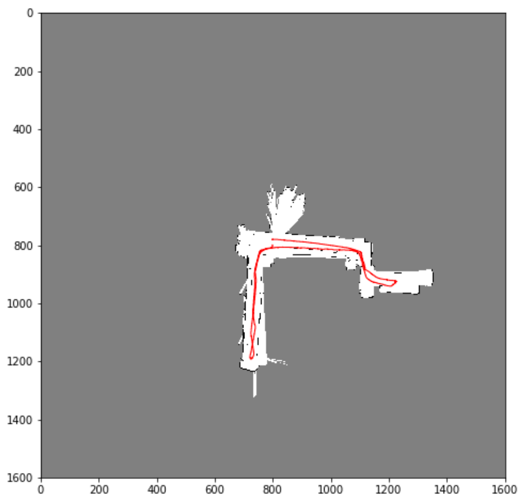

Fig. 59. DataSet 21 SLAM

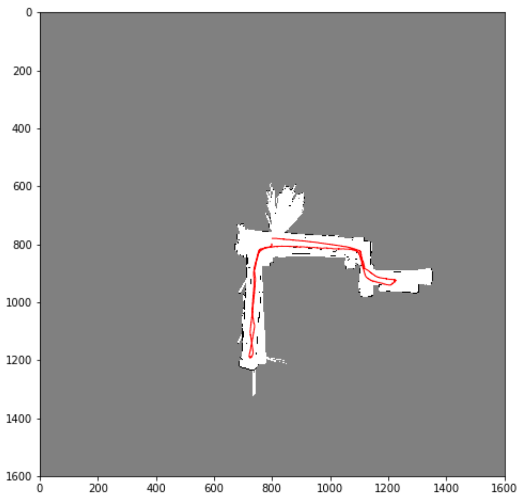

Fig. 58. DataSet 21 SLAM

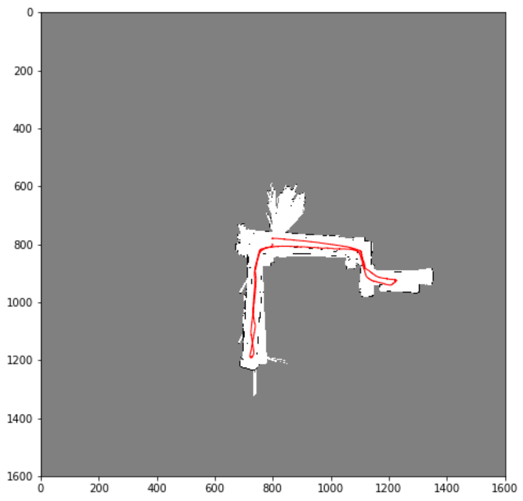

Fig. 60. DataSet 21 SLAM

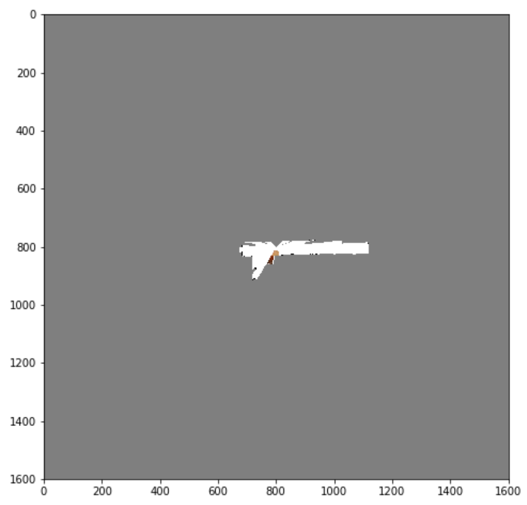

Fig. 61. DataSet 21 Texture

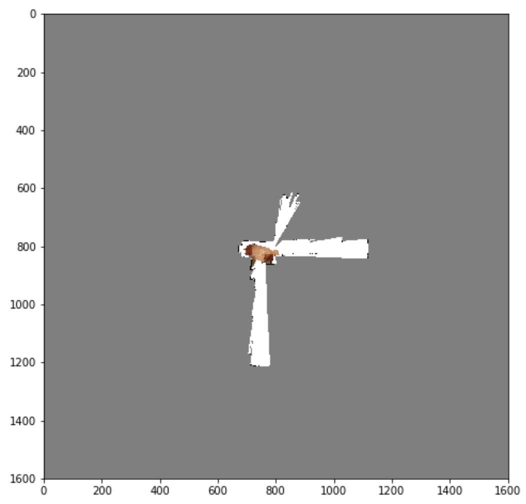

Fig. 63. DataSet 21 Texture

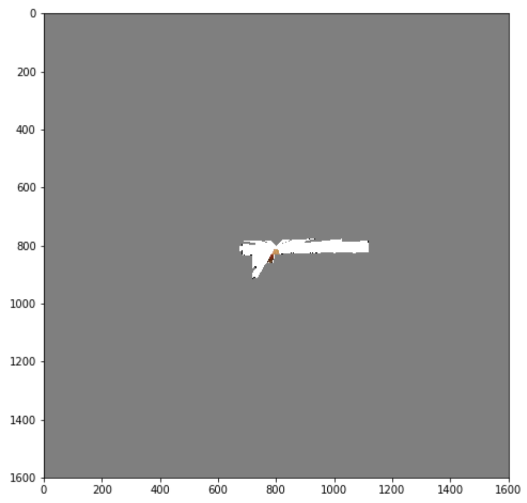

Fig. 62. DataSet 21 Texture

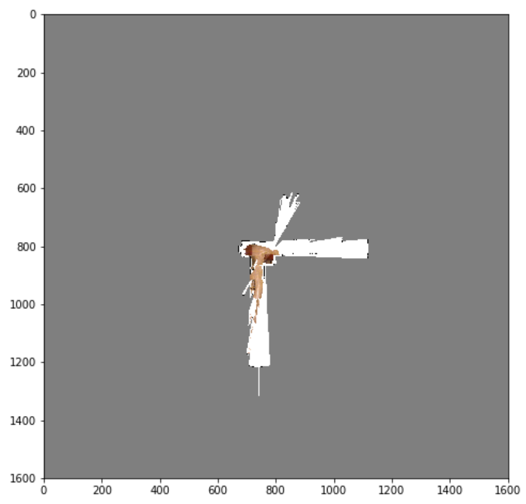

Fig. 64. DataSet 21 Texture

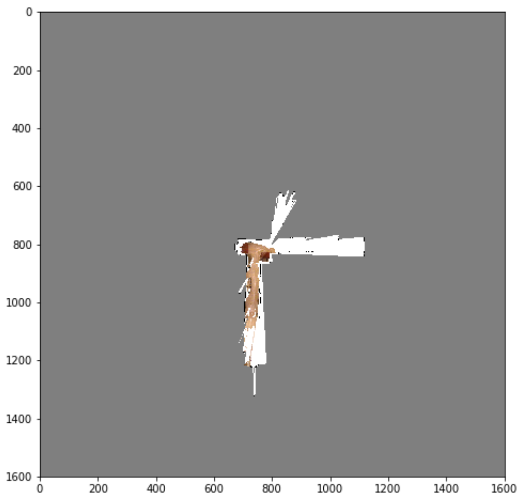

Fig. 65. DataSet 21 Texture

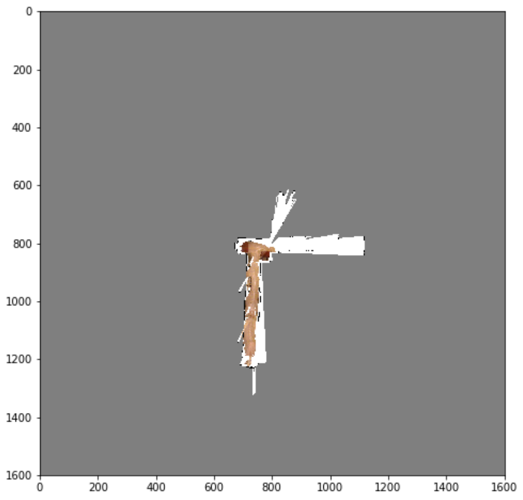

Fig. 66. DataSet 21 Texture

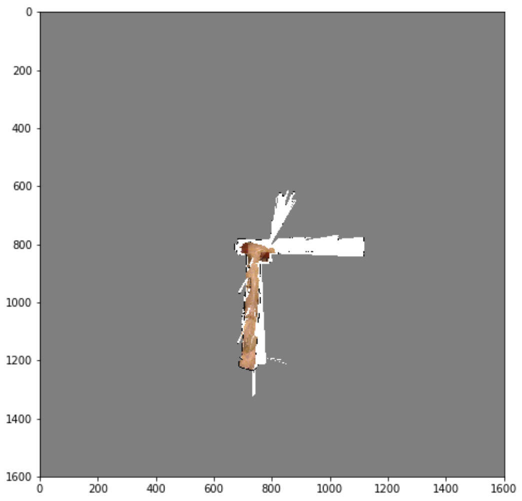

Fig. 67. DataSet 21 Texture

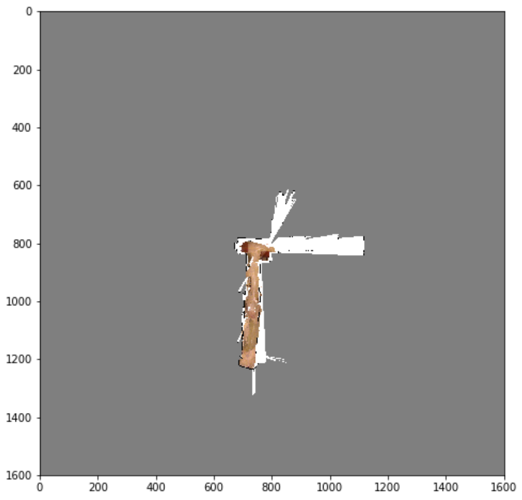

Fig. 68. DataSet 21 Texture

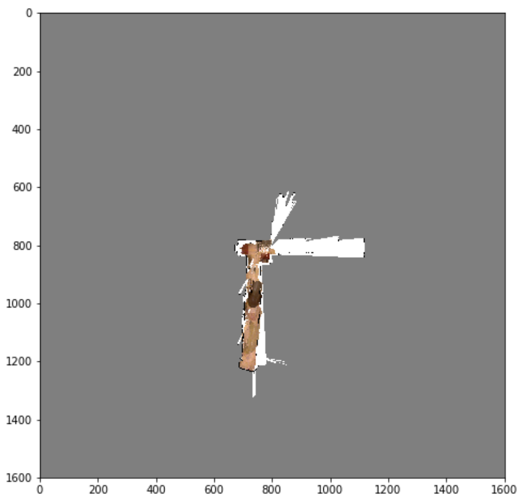

Fig. 69. DataSet 21 Texture

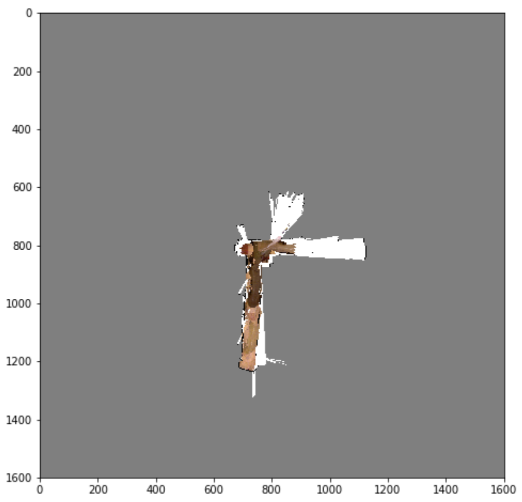

Fig. 71. DataSet 21 Texture

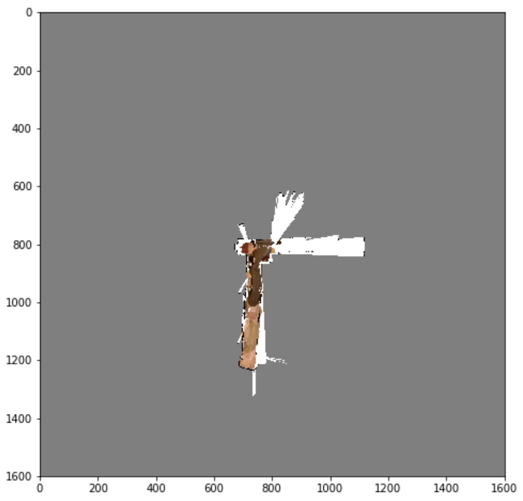

Fig. 70. DataSet 21 Texture

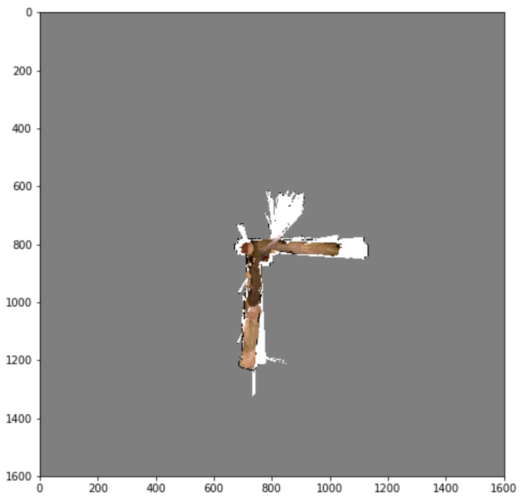

Fig. 72. DataSet 21 Texture

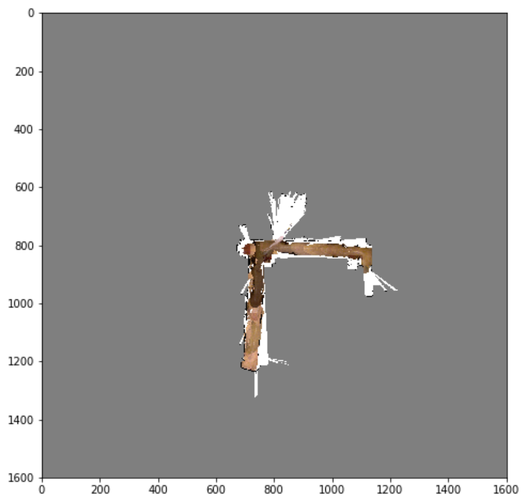
Fig. 73. DataSet 21 Texture

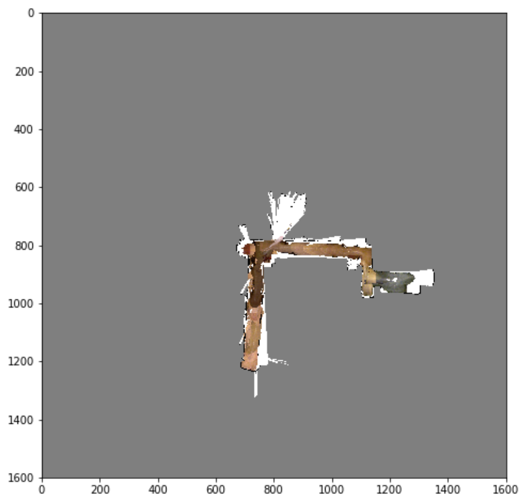
Fig. 75. DataSet 21 Texture

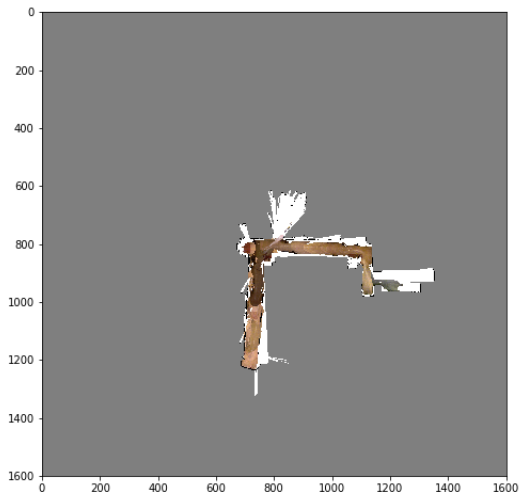
Fig. 74. DataSet 21 Texture

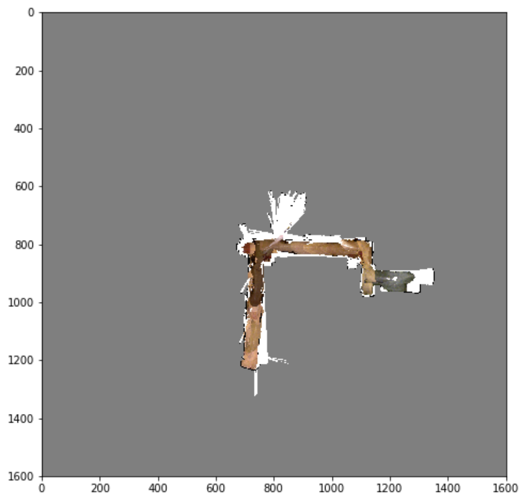
Fig. 76. DataSet 21 Texture

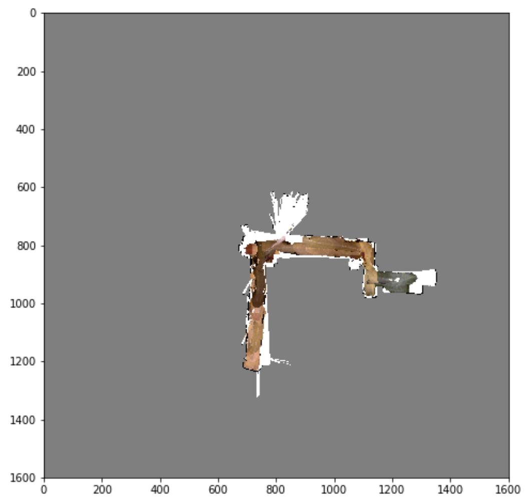

Fig. 77. DataSet 21 Texture

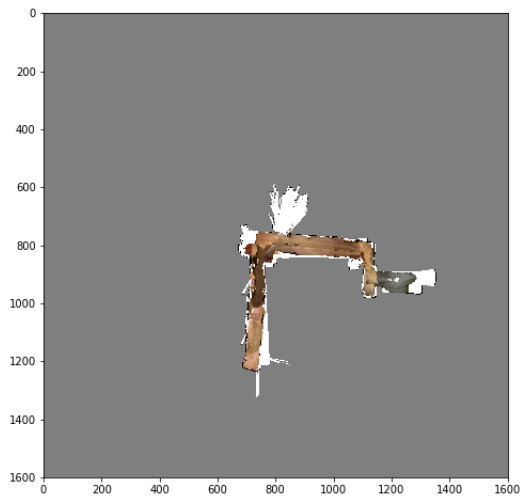

Fig. 79. DataSet 21 Texture

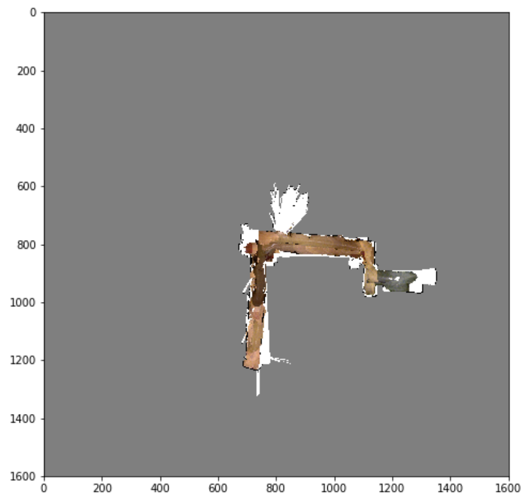

Fig. 78. DataSet 21 Texture

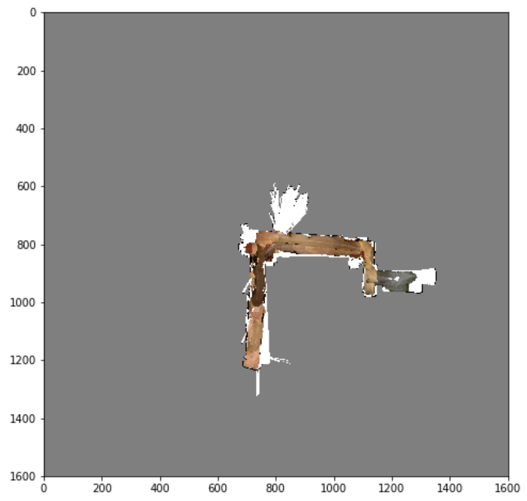

Fig. 80. DataSet 21 Texture

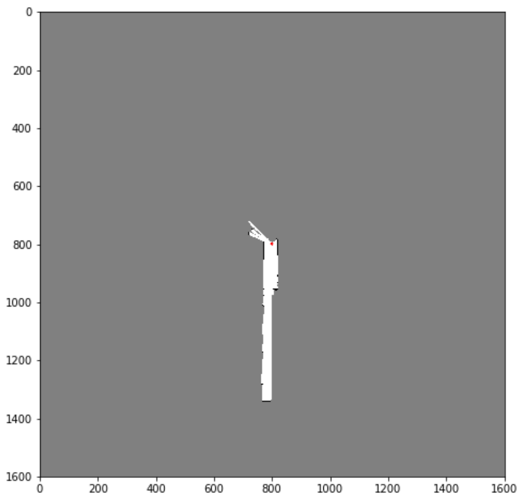
Fig. 81. DataSet 23 SLAM

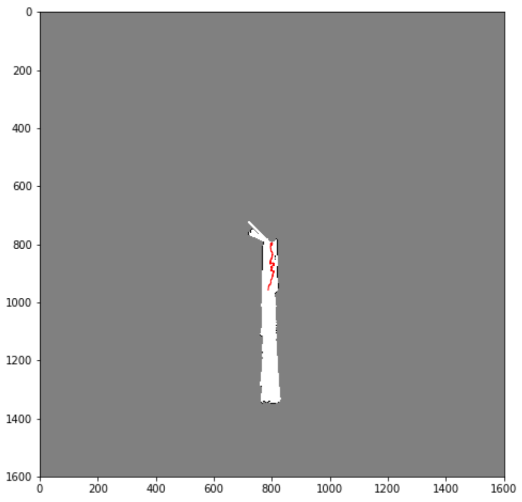
Fig. 83. DataSet 23 SLAM

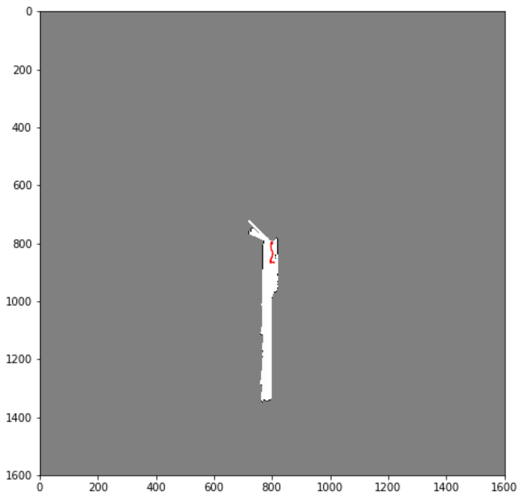
Fig. 82. DataSet 23 SLAM

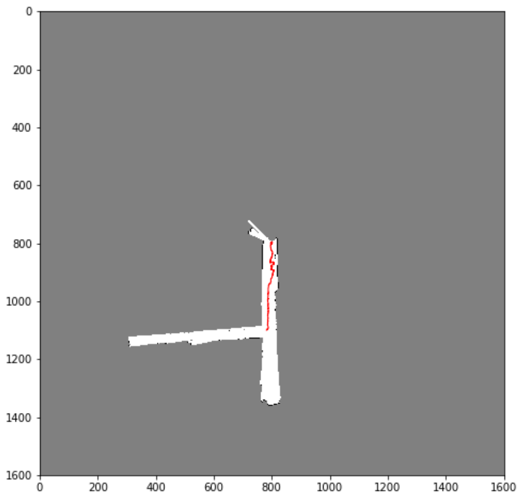
Fig. 84. DataSet 23 SLAM

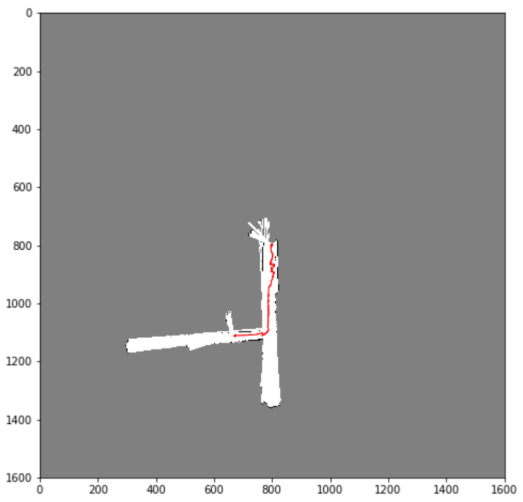

Fig. 85. DataSet 23 SLAM

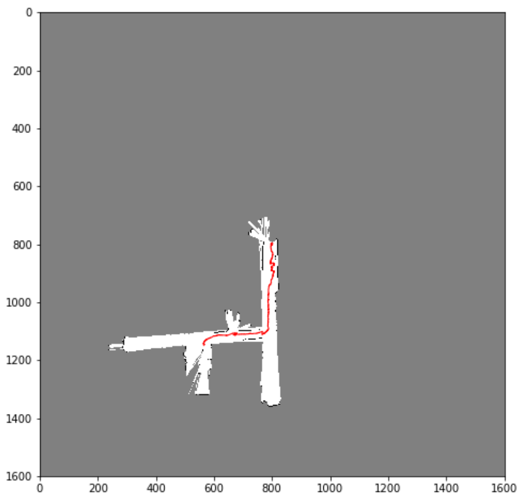

Fig. 86. DataSet 23 SLAM

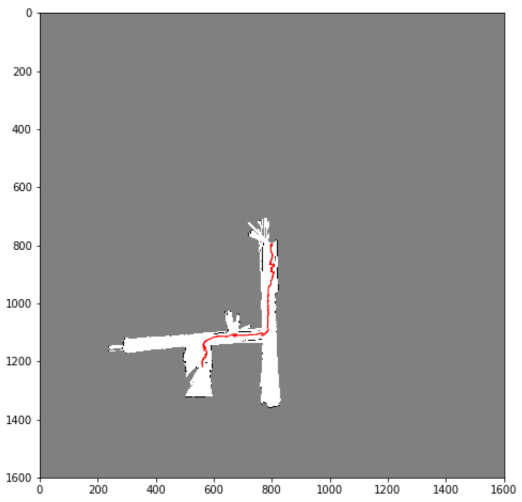

Fig. 87. DataSet 23 SLAM

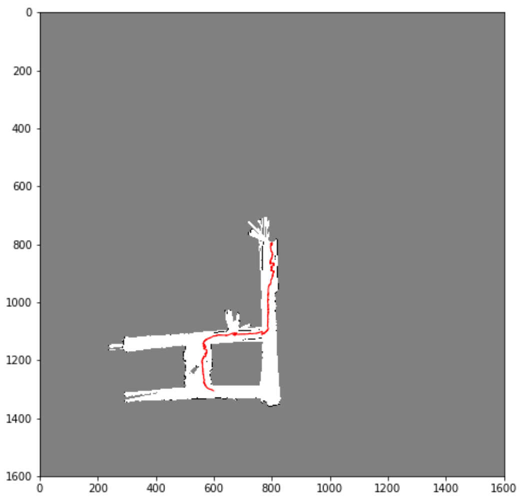

Fig. 88. DataSet 23 SLAM

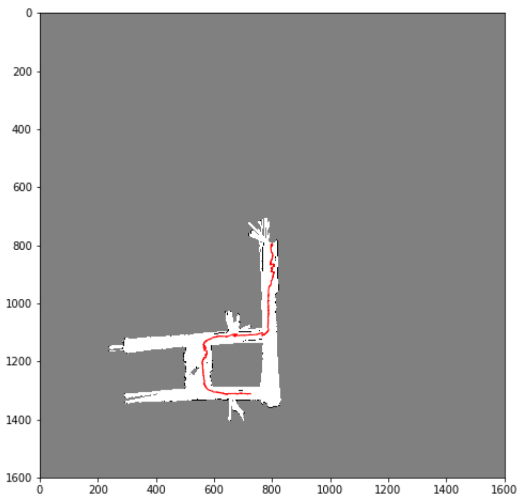

Fig. 89. DataSet 23 SLAM

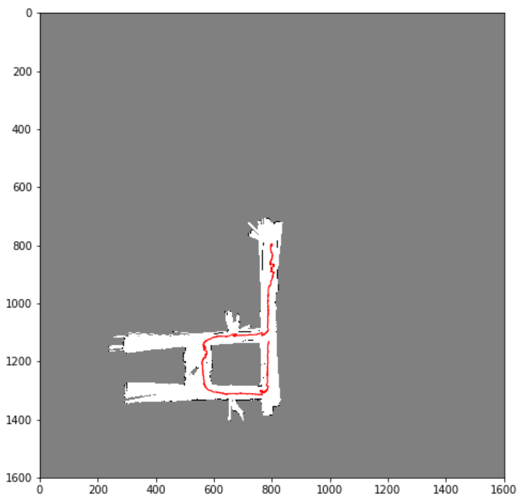

Fig. 91. DataSet 23 SLAM

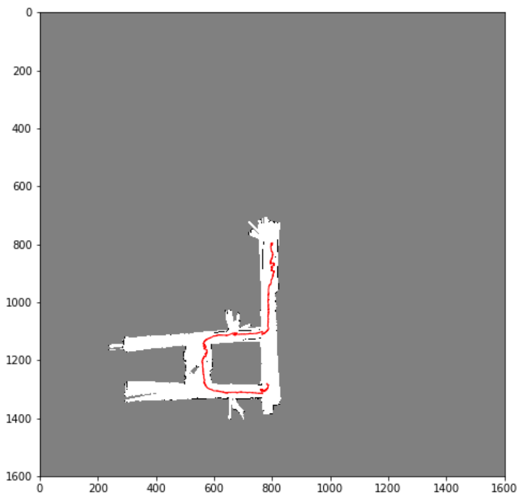

Fig. 90. DataSet 23 SLAM

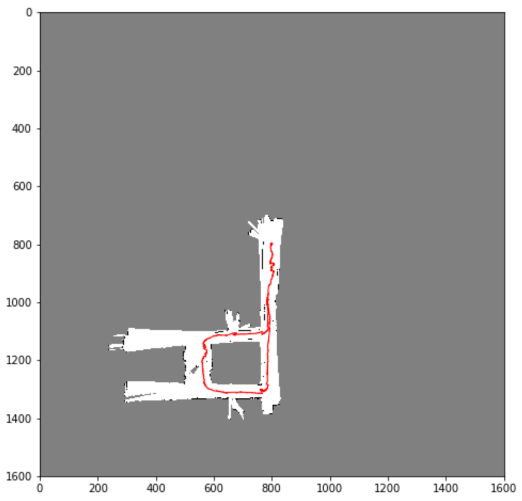

Fig. 92. DataSet 23 SLAM

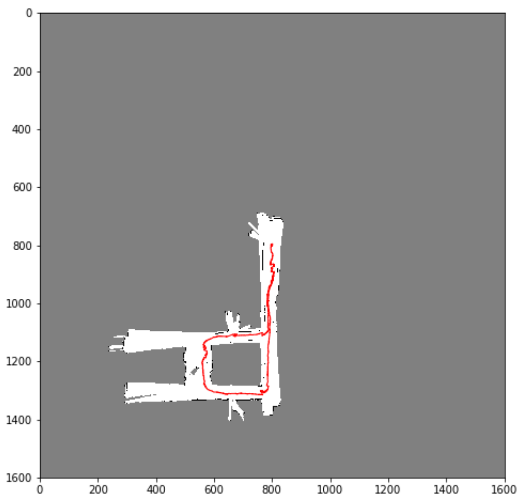

Fig. 93. DataSet 23 SLAM

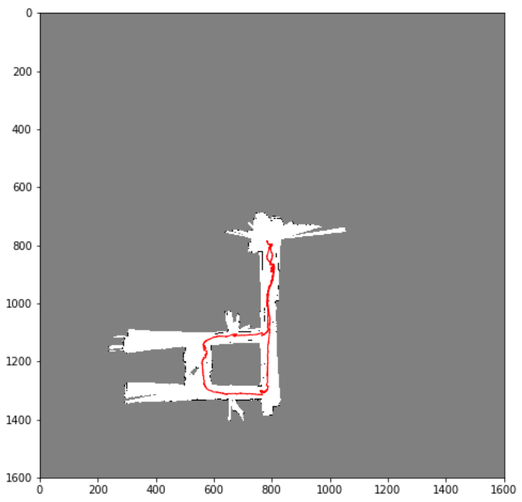

Fig. 95. DataSet 23 SLAM

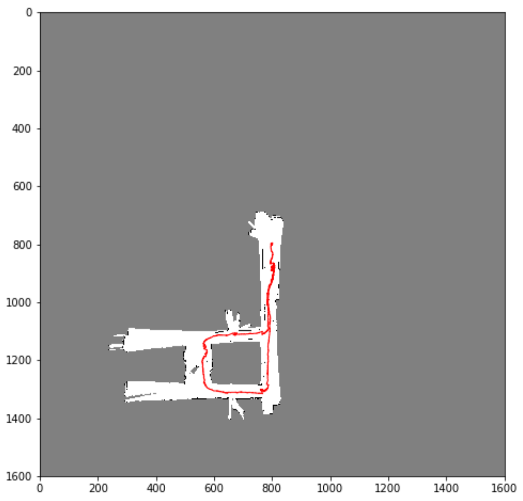

Fig. 94. DataSet 23 SLAM

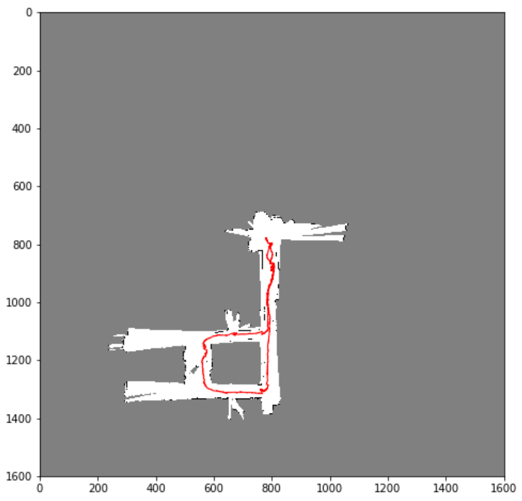

Fig. 96. DataSet 23 SLAM