Article

SLAM on the Hexagonal Grid

Piotr Duszak

Institute of Automatic Control and Robotics, Warsaw University of Technology, 02-525 Warsaw, Poland; piotr.duszak@pw.edu.pl

Abstract: Hexagonal grids have many advantages over square grids and could be successfully used in mobile robotics as a map representation. However, there is a lack of an essential algorithm, namely, SLAM (simultaneous localization and mapping), that would generate a map directly on the hexagonal grid. In this paper, this issue is addressed. The solution is based on scan matching and solving the least-square problem with the Gauss–Newton formula, but it is modified with the Lagrange multiplier theorem. This is necessary to fulfill the constraints given by the manifold. The algorithm was tested in the synthetic environment and on a real robot and is entirely fully suitable for the presented task. It generates a very accurate map and generally has even better precision than the similar approach implemented on the square lattice.

Keywords: simultaneous localization and mapping; hexagonal grid; Lidar; SLAM

1. Introduction

There are three ways to fill a plane with congruent regular polygons: with equilateral triangles, squares, or equilateral hexagons [1]. The second method has many applications, starting with building chain fences through image registration, display, and processing and ending with map representation, path planning, and localization.

However, this way of filling the plane has disadvantages, such as two types of the neighborhood (by vertices or edges; hence, the distance between neighbors is not always equal to one). With the hexagonal lattice, there is no such problem. Furthermore, a regular hexagon has more axes of symmetry than a square. Moreover, hexagons have the highest area to perimeter ratio of all shapes, with which the plane can be filled.

For these reasons, hexagonal grids commonly occur in the natural environment and have many applications. They can be seen in beehives, rock formations, turtle shells, and even on Saturn’s north pole [2] (see Figure 1). This grid has great endurance properties; hence, it is used in materials such as graphene [3]. The human retina (see Figure 1) is also formed in such a way [4]. These inspirations have led to the developments in hexagonal image processing [5]. It is a very promising field of research, but has one big drawback. There are no easily accessible devices to capture or display such images.

Another field in which the hexagonal grid can be useful is mobile robotics. In this case, all advantages can be applied, and there is no impediment (unlike in the previous paragraph) because the map is created based on sensors, such as lidar, which are not based on CCD matrices (unlike most cameras). It was shown that the hexagonal grid could be used as a map representation and has better properties than the square grid [6,7]. However, regardless of accuracy, it is worth investigating the possibility of building such a...
map to exploit its desirable properties, such as better path-finding. To meet this expectation, an algorithm for simultaneous localization and mapping on a hexagonal grid is presented.

Figure 1. Hexagonal structures occurring in nature. (a) Saturn's north pole [4]. (b) Cross-section of human retina [2].

2. Related Work
2.1. Hexagonal Grid

A large amount of work has been conducted on hexagonal grids. For example, they were tested in terms of geometric properties such as transformations [9,10] or distance between two points [11–13]. Furthermore, algorithms for drawing straight lines (analogous to Bresenham’s algorithm) and circles were invented [14,15].

Several articles have focused on hexagonal image processing, as a substantial part of robotics. They describe basic algorithms used in computer vision [5,16], e.g., gradient operators [17], edge detection [18–20], morphology operations [20], fast Fourier transform [21], filtering [19,22], and features extractions [23–25]. Recently, hexagonal grids have been considered in machine learning frameworks as well [26–28]. Although all these papers show the benefits of using hexagonal grids in image processing, there are currently no effortlessly available devices for capturing such images. Therefore their use in this area is limited.

Using hexagonal grids is a relatively new approach in mobile robotics. Most research refers to path planning [29–31], which turns out to be shorter and smoother than designated on an ordinary square grid. It was also shown that lines, circles, and splines are better represented on a hexagonal lattice [7]. This is important because obstacles often have such shapes. In [6], algorithms for 2.5D map creation was proposed. This generates a map based on a point cloud of the whole area. Therefore, a critical algorithm is lacking to build a map in real-time, based on subsequent data portions.

2.2. SLAM

The problem of simultaneous localization and mapping is essential for mobile robotics. It was first introduced in [32], where they solve the problem using a robot equipped with sonar and so-called beacons to match subsequent sensor readings. They also apply an extended Kalman filter to improve the quality of computed estimations. Since then, scientists and engineers have been intensely interested in this problem, and there are now many solutions worth mentioning.

In [33], the authors presented an algorithm using Rao–Blackwellized particle filters and scan-matching procedures to obtain the objective. In [34], the main idea was to use the Gauss–Newton algorithm to match scans. The approach shown in [8] combines scan-to-submap matching with graph optimization based on the branch and bound algorithm. All solutions mentioned in this paragraph are based on the square grid.

There are also many papers comparing various SLAM algorithms [35–37] and their applications, for example, in rescue operations [38].
Recently, there were attempts to use machine learning techniques in simultaneous localization and mapping, e.g., neural networks [39–41] and reinforcement learning [42]. Several attempts at SLAM based on hexagonal grids have been described [43,44]. In the former, the authors used a robot equipped with an event-based embedded dynamic vision sensor aimed at the ceiling and particle filter to estimate localization. They used collision sensors located on a bumper ring to notice obstacles. Therefore, it requires some characteristic elements on the ceiling and detects objects only from a very short distance. In the latter, there is a lidar-based algorithm described that relies on matching maps at different moments in time.

The algorithm presented here does not need particular markers. Due to being based on lidar, it can recognize walls and obstacles from a range. Moreover, unlike solutions described in [8,33,34], it uses the hexagonal grid instead of the square one. Similarly, like in [44], it is based on scan matching, but here the entire map is taken into account, not just the edge pixels.

3. Localization and Mapping

3.1. Problem Formulation

SLAM, as the name suggests, is the problem of creating a map of the environment and estimating the pose of the robot at the same time, based on consecutive sensor readings over time. In this paper, the native map on the hexagonal grid is built, not using the square lattice case. Each cell represents the probability of being occupied. It was assumed that the robot is equipped with lidar because it is one of the most popular sensors used in mobile robotics, and it gives sufficient data for this problem.

Although odometry is also very popular, it was decided not to use such information for the following reasons:

- Odometry is not very accurate;
- The system that does not require odometry is more flexible;
- One of the aims of this article is to compare properties between hexagonal and square lattices.

3.2. Solution Overview

The algorithm presented here was inspired by [34], because the solutions described there take into account the fact that a map is built on a grid. It consists of several steps as shown in Figure 2.

![Figure 2. Overview of the algorithm.](image)

First, the robot obtains data from lidar. Readings are in the form \((\phi_i, r_i)\), where \(r_i\) is a successive range in the direction of \(\phi_i\). Therefore, it is necessary to convert them to the proper coordinate system.

The next part is matching the new point cloud to the existing map. It is based on the Gauss–Newton algorithm for solving a non-linear least-square problem. It requires access to every point on the map and calculating derivatives; therefore, a new method must be constructed. This step uses transformation found from the previous reading as the starting point of the Gauss–Newton iteration method.

The last step is the actualization of the map combined with the discretization of the transformed new lidar scan. In the following subsections, all phases are described. However, to make the article more accessible, the unique coordinate system used for the hexagonal grid is first presented.

3.3. Coordinate System

In this research, the so-called cube coordinate system is used [9,45]. It was chosen because of the high symmetry level. It has three axes, rotated every 120°. It can be
interpreted as a coordinate system on a plane with one redundant axis, but also as a Cartesian coordinate system in 3D, but only on a plane, satisfying the equation:

$$x + y + \zeta = 0$$  \hspace{1cm} (1)

The cube coordinate system is presented in Figure 3. The third axis is denoted by \(\zeta\) to be consistent with [6] and to distinguish it from the normal third dimension.

\[\begin{align*}
  x_i &= r_i \cdot \cos(\phi_i) \\
  y_i &= r_i \cdot \cos\left(\frac{2}{3}\pi - \phi_i\right) \\
  \zeta_i &= r_i \cdot \cos\left(\frac{4}{3}\pi - \phi_i\right)
\end{align*}\]  \hspace{1cm} (2)

where:
- \(r_i, \phi_i\) are the original polar coordinates of the \(i\)-th point from the lidar scan;
- \(x_c, y_c, \zeta_c\) are continuous cube-hexagonal coordinates.

3.4. Polar to Cube Conversion

Conversion from polar to cube coordinates is analogous to the conversion from the polar to Cartesian system. It can be described by the following formulas:

3.5. Map Access

The map is stored as a discrete hexagonal grid, but the Gauss–Newton algorithm (which is used in the next step) requires values and gradients everywhere, so it is necessary to interpolate values between cells. It was achieved for both occupancy probability and derivatives by linear interpolation.

For the given point \(P_{m}\), with continuous coordinates, the occupancy value \(M(P_{m})\) and gradient \(\nabla M(P_{m}) = (\frac{\partial M}{\partial x}(P_{m}), \frac{\partial M}{\partial y}(P_{m}), \frac{\partial M}{\partial \zeta}(P_{m}))\) can be computed based on three nearest points with integer coordinates (see Figure 4) as the affine combination according to the formula:

$$M(P_{m}) = \alpha M(P_1) + \beta M(P_2) + \gamma M(P_3)$$  \hspace{1cm} (3)

where:
- \(M(\cdot)\) is the occupancy probability;
- \(P_1, P_2, P_3\) are the nearest points, shown in Figure 4;
- \(\alpha, \beta, \gamma\) denote the weights of the corresponding points.
Figure 4. Point $P_m$ with continuous coordinates and its nearest integer neighbors $P_1$, $P_2$, $P_3$.

In order to calculate weights, it is necessary to solve the following equation:

$$
\begin{bmatrix}
    x_1 & x_2 & x_3 \\
    y_1 & y_2 & y_3 \\
    \zeta_1 & \zeta_2 & \zeta_3 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
    \alpha \\
    \beta \\
    \gamma
\end{bmatrix}
= 
\begin{bmatrix}
    x \\
    y \\
    \zeta
\end{bmatrix}
$$

(4)

It always has a solution, because points $P_1$, $P_2$, $P_3$ are not collinear and for every point $x_i + y_i + \zeta_i = 0$. Taking into account the distribution of a point in the grid, Equation (4) can be transformed into the following:

$$
\begin{bmatrix}
    x_1 & x_1 & x_1 + 1 \\
    y_1 & y_1 + 1 & y_1 \\
    \zeta_1 + 1 & \zeta_1 & \zeta_1 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
    \alpha \\
    \beta \\
    \gamma
\end{bmatrix}
= 
\begin{bmatrix}
    x \\
    y \\
    \zeta
\end{bmatrix}
$$

(5)

which has a very simple solution:

$$
\begin{align*}
\alpha &= 1 + (x_1 - x) + (y_1 - y) \\
\beta &= 1 + (\zeta_1 - \zeta) + (x_1 - x) \\
\gamma &= 1 + (y_1 - y) + (\zeta_1 - \zeta)
\end{align*}
$$

(6)

During calculating derivatives, it is important to remember that variables $x$, $y$, and $\zeta$ are dependent on each other. Hence, partial derivatives for $\alpha$ are equal:

$$
\begin{align*}
\frac{\partial\alpha}{\partial x} &= -0.5 \\
\frac{\partial\alpha}{\partial y} &= -0.5 \\
\frac{\partial\alpha}{\partial \zeta} &= 1
\end{align*}
$$

(7)

Similarly, partial derivatives for $\beta$ and $\gamma$ can be computed. Hence, derivatives of the occupancy probability are equal:

$$
\begin{align*}
\frac{\partial M}{\partial x} (P_m) &= -0.5 \cdot M(P_1) - 0.5 \cdot M(P_2) + M(P_3) \\
\frac{\partial M}{\partial y} (P_m) &= -0.5 \cdot M(P_1) + M(P_2) - 0.5 \cdot M(P_3) \\
\frac{\partial M}{\partial \zeta} (P_m) &= M(P_1) - 0.5 \cdot M(P_2) - 0.5 \cdot M(P_3)
\end{align*}
$$

(8)
3.6. Scan Matching

In this step of the algorithm, new lidar data are matched to the already existing map. It was inspired by [34], but adapted to the hexagonal grid and cube-coordinate system. This approach is based on fitting beam endpoints to cells on the map, that are already recognized as occupied. This solution was chosen, because it is grid-based, so it can fully exploit the good properties of the hexagonal lattice.

Scan matching process is expected to be more accurate when operating on the hexagonal grid because there is only one type of neighborhood (as mentioned in Section 1) and the average distance from one cell to all neighbors is less for the hexagonal lattice than for the square lattice (if one assumes the surface areas of both polygons is one than the average distance is equal to 1.07 for the former case and it is 1.27 for the latter; see Figure 5).

\[
\begin{align*}
\mathbf{l} &= \sqrt{\frac{2}{3}} \\
\mathbf{l} &= \sqrt{\frac{2}{3}}
\end{align*}
\]

Figure 5. Comparison of the neighborhood for square and hexagonal lattice. Distances given when area of polygons is one. (a) The distance to all neighbors is the same at approximately 1.07. (b) There are two types of neighborhood. Average distance to the neighbor is approximately 1.21.

During this step of algorithm, there is needed the rigid transformation \( \xi = (p_x, p_y, p_z, \psi)^T \), which minimizes:

\[
\xi^* = \arg\min_{\xi} \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2
\]

where \( S_i \) are the coordinates of the \( i \)-th scan endpoint after applying transformation \( \xi \). Denote the \( i \)-th scan endpoint coordinates as \( s_i = (x_i, y_i, \zeta_i)^T \). Then, \( S_i(\xi) \) can be computed according to the formula:

\[
S_i(\xi) = \frac{1}{2} \begin{bmatrix}
2 \cos \psi + 1 & 1 - \cos \psi - \sqrt{3} \sin \psi & 1 - \cos \psi + \sqrt{3} \sin \psi \\
1 - \cos \psi + \sqrt{3} \sin \psi & 2 \cos \psi + 1 & 1 - \cos \psi - \sqrt{3} \sin \psi \\
1 - \cos \psi - \sqrt{3} \sin \psi & 1 - \cos \psi + \sqrt{3} \sin \psi & 2 \cos \psi + 1
\end{bmatrix} \begin{bmatrix}
x_i \\
y_i \\
\zeta_i
\end{bmatrix} + \begin{bmatrix}
p_x \\
p_y \\
p_z
\end{bmatrix}
\]

(10)

The first part of Formula (10) is a rotation by angle \( \psi \) around the axis, which goes through zero and is perpendicular to the plane, given by the formula \( x + y + \zeta = 0 \); the second part is simply the translation.

Finding \( \xi \) is based on the Gauss–Newton method, but it is necessary to find the solution that meets the condition \( p_x + p_y + p_z = 0 \). Therefore, in this paper, the Gauss–Newton method is combined with the method of Lagrange multipliers, which can be used for finding local extremes on manifolds.

Denote \( f(\xi) = \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \) and \( G(\xi) = p_x + p_y + p_z \). Therefore, according to the Lagrange multiplier theorem finding the minimum of function \( f \) on manifold \( G(\xi) = 0 \) requires solving the following equation system:

\[
\nabla f(\xi) = \lambda \nabla G(\xi) \\
G(\xi) = 0
\]

(11)
To calculate partial derivatives of function \( f \), the part under the sum is expanded in the Taylor series in the following way:

\[
  f(\xi + \Delta \xi) \approx \sum_{i=1}^{n} \left[ 1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta \xi \right] \]

(12)

Now, it is easier to calculate \( \nabla f \) and substitute it into Equation (11):

\[
  2 \sum_{i=1}^{n} \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}^T \left[ 1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta \xi \right] = \begin{bmatrix} \Lambda \\ \Lambda \\ \Lambda \\ 0 \end{bmatrix}
\]

(13)

\[
p_x + p_y + p_z = 0
\]

(14)

In Section 3.5, it was described how to compute \( M(\cdot) \) and \( \nabla M(\cdot) \). To compute \( \frac{\partial S_i(\xi)}{\partial \xi} \), it is necessary to use Equation (10) to obtain:

\[
  \frac{\partial S_i(\xi)}{\partial \xi} = \begin{bmatrix} 1 & 0 & 0 & -x_i \sin \psi + \frac{\xi - y_i}{\sqrt{3}} \cos \psi \\ 0 & 1 & 0 & -y_i \sin \psi + \frac{\xi - x_i}{\sqrt{3}} \cos \psi \\ 0 & 0 & 1 & -\zeta_i \sin \psi + \frac{\xi - z_i}{\sqrt{3}} \cos \psi \end{bmatrix}
\]

(17)

When \( \Delta \xi \) is calculated, it is now possible to step toward the minimum of function \( f \). The starting point of this modified Gauss–Newton iteration formula is transformation calculated based on the previous reading from the lidar.

### 3.7. Map Actualization

After obtaining new data from lidar and determining the new transformation \( \xi \), the map is updated straightforwardly, i.e., beam endpoint coordinates after applying \( \xi \) are marked as occupied, and cells between the robot and endpoints are marked as free.

However, due to the discrete character of the hexagonal grid, there is a need to convert continuous cube coordinates to its discrete version. Such an algorithm was described in [9]. For completeness, it is also presented here. The aim is to determine the coordinates
of the nearest cell, when continuous coordinates \(x_c, y_c, \zeta_c\) are given. First, the auxiliary
variables are calculated:

\[
\begin{align*}
\hat{x}_d &= \lfloor x_c / w \rfloor \\
\hat{y}_d &= \lfloor y_c / w \rfloor \\
\hat{\zeta}_d &= \lfloor \zeta_c / w \rfloor
\end{align*}
\]

(18)

where \(w\) is the width of one cell.

Next, the following Algorithm 1 is applied:

**Algorithm 1 Discretization of Hexagonal Coordinates.**

```plaintext
if \(\hat{x}_d + \hat{y}_d + \hat{\zeta}_d = 0\) then
    \(x_d \leftarrow \hat{x}_d\)
    \(y_d \leftarrow \hat{y}_d\)
    \(\zeta_d \leftarrow \hat{\zeta}_d\)
else
    if \(\{x_c / w\} \geq \{y_c / w\}\) and \(\{x_c / w\} \geq \{\zeta_c / w\}\) then
        \(x_d \leftarrow -\hat{y}_d - \hat{\zeta}_d\)
        \(y_d \leftarrow \hat{y}_d\)
        \(\zeta_d \leftarrow \hat{\zeta}_d\)
    else if \(\{y_c / w\} \geq \{x_c / w\}\) and \(\{y_c / w\} \geq \{\zeta_c / w\}\) then
        \(x_d \leftarrow \hat{x}_d\)
        \(y_d \leftarrow -\hat{x}_d - \hat{\zeta}_d\)
        \(\zeta_d \leftarrow \hat{\zeta}_d\)
    else
        \(x_d \leftarrow \hat{x}_d\)
        \(y_d \leftarrow \hat{y}_d\)
        \(\zeta_d \leftarrow -\hat{x}_d - \hat{y}_d\)
    end if
end if
```

where \(\{ \bullet \}\) denotes the fractional part.

Analogous to [34], to prevent from falling in a local minimum, three maps are kept
with different resolutions. Each map has cells with a surface area four times larger than the
previous one. During the matching process, first the transformation is found on a coarse
map; next, this transformation is used as input for a more accurate map, and so on.

4. Results

The algorithm presented here was tested in an artificial environment, simulated by
ROS (Robot Operating System) [46] and compared to the Hector SLAM [34], available as a
package on ROS, and also on the real mobile robot in the laboratory. Additional tests were
conducted on actual data provided by the MIT Stata Center (http://projects.csail.mit.edu/
/stata/, accessed on 16 August 2022) [47].

4.1. Simulation

The simulation was prepared with ROS stage (http://wiki.ros.org/stage, accessed on
16 August 2022). During simulation tests, ground truth of the robot is known, so there is a
possibility of comparing the localization precision of the presented algorithm with another
one. Several scenarios were prepared, differing in the following properties:

- Location of obstacles—two types of map (Figure 6);
- Resolution of built map—two resolutions: area of one cell on the most accurate map
  equals 0.01 m\(^2\) or 0.04 m\(^2\);
- Angle between initial direction of robot and walls—three angles: 0°, 45°, and 60°;
- Resolution of lidar—two resolution: 1000 points in one reading or 360 points.
Figure 6. Two types of synthetic maps. (a) Maze from robotcraft competition (https://robotcraft.ingeniarius.pt/, https://github.com/ingeniarius-ltd/robotcraft_maze, accessed on 16 August 2022). (b) Synthetically generated rooms.

During each test, two parameters were measured: mean square error of position and mean square error of angle. Results for the algorithm presented here and for comparison for hector_mapping (http://wiki.ros.org/hector_mapping, accessed on 16 August 2022) can be seen in Table 1. Test parameters are presented in the following order: map type (maze or rooms), area of cell on the most accurate map (0.01 m² or 0.04 m²), angle (0°, 45°, or 60°), and number of points from lidar (1000 or 360).

Table 1. Localization error for Hexagonal SLAM and hector_slam for various situations.

| Test Number | Test Parameters | MSE of Position for Hexagonal SLAM in Meters | MSE of Position for Hector_slam in Meters | MSE of Angle for Hexagonal SLAM in Radians | MSE of Angle for Hector_slam in Radians |
|-------------|-----------------|---------------------------------------------|------------------------------------------|-------------------------------------------|------------------------------------------|
| 1  | maze 0.01 m² 0° 1000 | 2.4 × 10⁻² | 6.1 × 10⁻² | 1.8 × 10⁻⁴ | 1.3 × 10⁻⁴ |
| 2  | rooms 0.01 m² 0° 1000 | 5.8 × 10⁻² | 6.3 × 10⁻² | 8.0 × 10⁻⁵ | 6.9 × 10⁻⁵ |
| 3  | maze 0.04 m² 0° 1000 | 5.01 × 10⁻² | 1.4 × 10⁻¹ | 1.7 × 10⁻⁴ | 3.4 × 10⁻⁴ |
| 4  | maze maze 0.01 m² 45° 1000 | 2.3 × 10⁻² | 6.0 × 10⁻² | 2.4 × 10⁻⁴ | 2.6 × 10⁻⁴ |
Table 1. Cont.

| Test Number | Test Parameters | MSE of Position For Hexagonal SLAM in Meters | MSE of Position For Hector_slam in Meters | MSE of Angle For Hexagonal SLAM in Radians | MSE of Angle For Hector_slam in Radians |
|--------------|-----------------|---------------------------------------------|-------------------------------------------|--------------------------------------------|------------------------------------------|
| 5            | maze 0.01 m² 60° 1000 | 1.5 × 10⁻²                                  | 4.1 × 10⁻²                                | 1.7 × 10⁻⁴                                | 4.2 × 10⁻⁴                              |
| 6            | maze 0.01 m² 0° 360 | 2.4 × 10⁻²                                  | 5.3 × 10⁻²                                | 2.0 × 10⁻⁴                                | 1.2 × 10⁻⁴                              |

As can be seen, the precision of the estimated position was always better for the SLAM on the hexagonal grid (usually two times better). Some tests precisions of the estimated angle were slightly better for the hexagonal grid and some for the square grid, but the difference here was minimal. Increasing the cell width twice also doubled the error, which is expected. The angle between the starting direction of the robot and the walls did not affect the results. The algorithm also coped with the reduced resolution of the lidar. Figure 7 shows the generated map for Test 1, and Figure 8 shows the ground truth position of the robot during Test 1 and estimation obtained from hector and hexagonal SLAM.

Figure 7. Map of the maze generated by the algorithm.

The algorithm was tested on Ubuntu 20.04.4 LTS, AMD Ryzen 7 4800H CPU 1.4 GHz, RAM 32 GB (manufactured for Lenovo in China). On this hardware, one loop of the algorithm took an average of 1.5 × 10⁻² s for a 360-point lidar and 2.9 × 10⁻² s for a 1000-point lidar.
4.2. Laboratory

The algorithm was also tested on a real mobile robot, i.e., the Husarion ROSbot 2.0 (https://husarion.com/manuals/rosbot/, accessed on 16 August 2022). It is equipped with RpLidar with a 360° field of view and a range of up to 8 m. During experiments in the laboratory, the robot was driven through the maze (see Figure 9) remotely controlled by an operator at an average linear speed of approximately 0.07 m/s and at an average angular speed of approximately 0.02 rad/s. However, this time the accurate position of the robot was not available during the whole run.

To overcome this, the starting and ending positions of the robot were the same (a special marker was attached to the floor for this purpose), and the estimation error was measured after the execution of one full lap. The center of the lidar was established as the reference point of the robot, and the starting and ending position was measured with a
caliper relatively to markers attached to the floor. Angle error was not measured due to the inability to determine its ground-truth value, even after one full lap. Two tests were carried out with different arrangements of walls. Final error values were equal to $3.1 \times 10^{-2}$ m for both tests.

4.3. MIT Stata Center Data Set

MIT Stata Center Data Set was collected in buildings belonging to MIT in 2011 and 2012. It contains information about odometry, cameras recordings, and—most importantly for this research data—from lidar. The authors used Willow Garage PR2 equipped with Hokuyo UTM-30LX Laser Scanner. Moreover, they share ground truth positions for some recordings with declared accuracy of 2–3 cm.

Two tests were provided, differing in their starting position and a robot’s motion trajectory. During the first one (from 27 January 2012) the robot was driving at an average linear speed of 0.57 m/s (a maximum linear speed was 1.28 m/s) and at an average angular speed of 0.11 rad/s (a maximum angular speed was 0.70 rad/s). Speeds during the second one (from 3 April 2012) were similar, namely: an average linear speed of 0.54 m/s, a maximum linear speed of 1.10 m/s, an average angular speed of 0.15 rad/s, and a maximum angular speed of 0.78 rad/s.

Figure 10 shows the ground truth position of the robot during the first test, using MIT Data and estimations obtained from hector and hexagonal SLAM. Analogous results for the second one are shown in Figure 11. Those experiments confirm the results obtained from the simulations, i.e., the trajectory determined by the hexagonal SLAM runs closer to the real one than the trajectory determined by hector SLAM. Furthermore, during both tests, hector SLAM got lost at some point while the hexagonal SLAM did not.

![Figure 10](image1.png)

(a) Ground truth position and position estimations for MIT Data Set—recording from 27 January 2012. (a) The whole test run. (b) Close up.
Figure 11. Ground truth position and position estimations for MIT Data Set—recording from 3 April 2012. (a) The whole test run. (b) Close up.

5. Conclusions and Future Works

In this paper, the algorithm for simultaneous localization and mapping directly on a hexagonal grid was presented. The algorithm generates a very accurate map, which can be used, for example, for path planning. Therefore, all advantages of the hexagonal grid can now be used in mobile robotics, which is the main achievement of this article. Additionally, it was shown that localization on the hexagonal grid is more accurate than on the square grid.

In the future, this algorithm will be improved by adding probabilistic aggregation of the map and information given by the odometry, and by applying Kalman or particle filter to estimate movement parameters. Moreover, loop closure will be added, and other methods of map access will be tested, including other types of interpolation or using more than three nearest points.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code and data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The author declares no conflict of interest.

References
1. Coxeter, H.S.M. *Introduction to Geometry*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1961.
2. Godfrey, D.A. A hexagonal feature around Saturn’s north pole. *Icarus* **1988**, *76*, 335–356.
3. Wallace, P.R. The band theory of graphite. *Phys. Rev.* **1947**, *71*, 622.
4. Curcio, C.A.; Sloan, K.R.; Kalina, R.E.; Hendrickson, A.E. Human photoreceptor topography. *J. Comp. Neurol.* **1990**, *292*, 497–523. [PubMed]
5. Middleton, L.; Sivaswamy, J. *Hexagonal Image Processing: A Practical Approach*; Springer Science & Business Media: Berlin, Germany, 2006.
6. Duszak, P.; Siemiątkowska, B.; Więckowski, R. Hexagonal Grid-Based Framework for Mobile Robot Navigation. *Remote Sens.* **2021**, *13*, 4216.
7. Duszak, P.; Siemiątkowska, B. The application of hexagonal grids in mobile robot Navigation. In Proceedings of the International Conference Mechatronics—Computing in Mechatronics, Warsaw, Poland, 16–18 September 2019; Springer: Cham, Switzerland, 2019; pp. 198–205.
8. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. May. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
9. Her, I. Geometric transformations on the hexagonal grid. *IEEE Trans. Image Process.* 1995, 4, 1213–1222.

10. Nagy, B. Isometric transformations of the dual of the hexagonal lattice. In Proceedings of the 6th International Symposium on Image and Signal Processing and Analysis, Salzburg, Austria, 16–18 September 2009; pp. 432–437.

11. Luczak, E.; Rosenfeld, A. Distance on a hexagonal grid. *IEEE Trans. Comput.* 1976, 25, 532–533.

12. Nagy, B. Nonmetrical distances on the hexagonal grid using neighborhood sequences. *Pattern Recognit. Image Anal.* 2007, 17, 183–190.

13. Kovács, G.; Nagy, B.; Vizvári, B. Weighted distances on the truncated hexagonal grid. *Pattern Recognit. Lett.* 2021, 152, 26–33.

14. Wüthrich, C.A.; Stucki, P. An algorithmic comparison between square-and hexagonal-based grids. *CVGIP: Graph. Model. Image Process.* 1991, 53, 324–339.

15. Yong-Kui, L. The generation of straight lines on hexagonal grids. In *Computer Graphics Forum*; Blackwell Science Ltd.: Edinburgh, UK, 1993; Volume 12, pp. 27–31.

16. Asharindavida, F.; Hundewale, N.; Aljahdali, S. Study on hexagonal grid in image processing. *Proc. ICIMK 2012*, 45, 282–288.

17. Coleman, S.A.; Scotney, B.W.; Gardiner, B. Design of Feature Extraction Operators for Use on Biologically Motivated Hexagonal Image Structures. In Proceedings of the MVA2009 IAPR Conference on Machine Vision Applications, Yokohama, Japan, linebreak 20–22 May 2009; pp. 178–181.

18. Middleton, L.; Sivaswamy, J. Edge detection in a hexagonal-image processing framework. *Image Vis. Comput.* 2001, 19, 1071–1081. [CrossRef]

19. Wu, Q.; He, X.; Hintz, T. March. Bi-lateral filtering based edge detection on hexagonal architecture. In Proceedings of the (iCASSP’05) IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia, PA, USA, 23 March 2005; Volume 2, pp. ii–713. [CrossRef]

20. Mostafa, K.; Chiang, J.Y.; Her, I. Edge-detection method using binary morphology on hexagonal images. *Imaging Sci. J.* 2015, 63, 168–173. [CrossRef]

21. Birdsong, J.B.; Rummelt, N.I. The hexagonal fast fourier transform. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 1809–1812.

22. Veni, S.; Narayanankutty, K.A. Vision-based hexagonal image processing using Hex-Gabor. *Signal Image Video Process.* 2014, 8, 317–326. [CrossRef]

23. Azam, M.; Anjum, M.A.; Javed, M.Y. Discrete cosine transform (DCT) based face recognition in hexagonal images. In Proceedings of the 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, 26–28 February 2010; Volume 2, pp. 474–479.

24. Schlosser, T.; Friedrich, M.; Kowerko, D. Hexagonal image processing in the context of machine learning: Conception of a biologically inspired hexagonal deep learning framework. In Proceedings of the 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 1866–1873.

25. Kerr, D.; Coleman, S.A.; McGinnity, T.M.; Wu, Q.; Clogenson, M. A novel approach to robot vision using a hexagonal grid and spiking neural networks. In Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–7.

26. Samadi, M.; Othman, M.F. Global path planning for autonomous mobile robot using genetic algorithm. In Proceedings of the 2013 International Conference on Signal-Image Technology & Internet-Based Systems, Kyoto, Japan, 2–5 December 2013; pp. 726–730.

27. Shao, X.; Zheng, R.; Wei, J.; Guo, D.; Yang, T.; Wang, B.; Zhao, Y. Path planning of mobile Robot based on improved ant colony algorithm based on Honeycomb grid. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Shenyang, China 22–24 January 2021; Volume 5, pp. 1358–1362.

28. Li, T.; Xia, M.; Chen, J.; Gao, S.; De Silva, C. A hexagonal grid-based sampling planner for aquatic environmental monitoring using unmanned surface vehicles. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 3683–3688. [CrossRef]

29. Leonard, J.J.; Durrant-Whyte, H.F. Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robot. Autom.* 1991, 7, 376–382. [CrossRef]

30. Griset, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Robot.* 2007, 23, 34–46. [CrossRef]

31. Kohlbrecher, S.; Von Stryk, O.; Meyer, J.; Klingauf, U. A flexible and scalable SLAM system with full 3D motion estimation. In Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, 1–5 November 2011; pp. 155–160.

32. Panchpor, A.A.; Shue, S.; Conrad, J.M. A survey of methods for mobile robot localization and mapping in dynamic indoor environments. In Proceedings of the 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES), Vijayawada, India, 2018, 4–5 January 2018; pp. 138–144.
36. Filipenko, M.; Afanasyev, I. Comparison of various slam systems for mobile robot in an indoor environment. In Proceedings of the 2018 International Conference on Intelligent Systems (IS), Funchal, Portugal, 25–27 September 2018; pp. 400–407.

37. Zhang, X.; Lu, G.; Fu, G.; Xu, D.; Liang, S. SLAM algorithm analysis of mobile robot based on lidar. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 4739–4745.

38. Zhang, X.; Lai, J.; Xu, D.; Li, H.; Fu, M. 2D LiDAR-based SLAM and path planning for indoor rescue using mobile robots. J. Adv. Transp. 2020, 2020, 8867937. [CrossRef]

39. Gao, X.; Zhang, T. Unsupervised learning to detect loops using deep neural networks for visual SLAM system. Auton. Robot. 2017, 41, 1–18. [CrossRef]

40. Zhang, S.; Lu, S.; He, R.; Bao, Z. Stereo Visual Odometry Pose Correction through Unsupervised Deep Learning. Sensors 2021, 21, 4735. [CrossRef]

41. Mazurek, P.; Hachaj, T. SLAM-OR: Simultaneous Localization, Mapping and Object Recognition Using Video Sensors Data in Open Environments from the Sparse Points Cloud. Sensors 2021, 21, 4734. [CrossRef] [PubMed]

42. Dinnissen, P.; Givigi, S.N.; Schwartz, H.M. Map merging of multi-robot slam using reinforcement learning. In Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, Korea, 14–17 October 2012; pp. 53–60.

43. Hoffmann, R.; Weikersdorfer, D.; Conradt, J. Autonomous indoor exploration with an event-based visual SLAM system. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 38–43.

44. Zhang, J.; Wang, X.; Xu, L.; Zhang, X. An Occupancy Information Grid Model for Path Planning of Intelligent Robots. ISPRS Int. J.-Geo-Inf. 2022, 11, 231. [CrossRef]

45. Snyder, W.E.; Qi, H.; Sander, W.A. Coordinate system for hexagonal pixels. In Proceedings of the Medical Imaging 1999: Image Processing International Society for Optics and Photonics, San Diego, CA, USA, 20–26 February 1999; Volume 3661, pp. 716–727.

46. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.

47. Fallon, M.; Johannsson, H.; Kaess, M.; Leonard, J.J. The mit stata center dataset. Int. J. Robot. Res. 2013, 32, 1695–1699. [CrossRef]